Федеральное агентство по образованию

Тульский государственный университет

КАФЕДРА ТЕХНОЛОГИИ МАШИНОСТРОЕНИЯ

Илюхин А.Ю., доцент, к.т.н.

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ

для бакалавров по направлению 150900 – «Технология, оборудование и автоматизация машиностроительных производств»
и студентов специальностей:
151001 «Технология машиностроения»,
150401 «Проектирование технических и технологических комплексов»,

Тула 2005

Разработал:

кан.тех.наук., доц. А.Ю.Илюхин

Рассмотрено на заседании кафедры

Протокол № от "___" _____2005 г.

Зав.кафедрой _____ А.С.Ямников

Содержание

Лабораторная работа №1.
Системы счисления.

Введение

1. Перевод чисел из одной системы счисления в другую

Системой счисления называется совокупность символов, используемых для изображения чисел, т.е. кодирования числовой информации.

Системы счисления делятся на позиционные и непозиционные.

В непозиционной системе счисления местоположение символа определяющего цифру (число) не оказывает влияние на размер числа. Примером такой системы является Римская система счисления. Символы используемые в Римской системе счисления отображения чисел:

I - 1, V - 5, X - 10, L - 50, C - 100

Правило записи чисел: значение числа определяется суммой всех значений символов, расположенных правее максимального числа за вычетом значений символов, расположенных левее данного символа.

Примеры:

III (3), IV (4), XXII (22), XLI (41), LXXXIII (83)

Количество цифр применяемых в позиционной системе счисления называется основанием системы счисления p . Местоположение символа в числе называется разрядом, каждый разряд имеет свой вес.

В любой системе счисления число можно представить

$A_n A_{n-1} \dots A_2 A_1 A_0, A_{-1} A_{-2} \dots A_{-m} = A_n * p^n + A_{n-1} * p^{n-1} + \dots + A_2 * p^2 + A_1 * p^1 + A_0 * p^0 + A_{-1} * p^{-1} + A_{-2} * p^{-2} + \dots + A_{-m} * p^{-m}$

Например:

$345{,}16_{(10)} = 3 * 10^2 + 4 * 10^1 + 5 * 10^0 + 1 * 10^{-1} * 6 * 10^{-2}$,

где (10) - основание десятичной системы счисления.

В вычислительной технике при кодировании информации широко используются двоичная, восьмеричная и шестнадцатиричная системы счисления, которые представлены в таблице 1.

Чтобы перевести число из одной системы счисления в другую необходимо разделить его на основание той системы в которую оно переводится, полученный остаток будет младшим разрядом числа в новой системе счисления, частное от деления делится на основание, остаток - следующий разряд и так далее, деление продолжается до тех пор пока не

получится частное меньше основания системы в которую мы переводим - это будет старший разряд число в новой системе счисления.

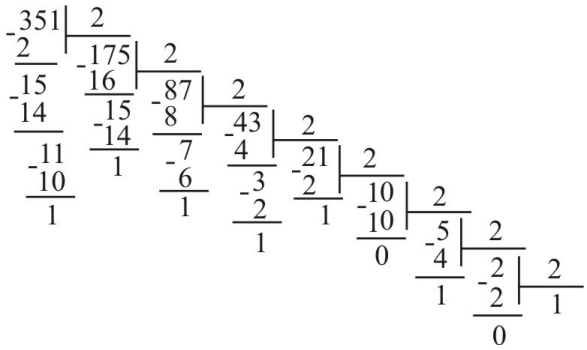
Таблица 1

Система счисления			
двоичная	восмиричная	десятичная	шестнадцатиричная
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10

Например, перевести число 351 из десятичной системы счисления в шестнадцатиричную и двоичную:

```
-351 | 16
 32  |-----
----  -21 | 16
 -31   16 |----
  16  ---   1
  ---   5
  15(F)
```

$351_{(10)} = 15F_{(16)}$



$351_{(10)} = 101011111_{(2)}$

Для перевода чисел из двоичной системы счисления в восьмеричную, шестнадцатиричную и наоборот можно воспользоваться следующим правилом: группа цифр из 3-х, 4-х двоичной системы заменяется на соответствующую цифру для восьмеричной, шестнадцатиричной системы и наоборот в соответствии с таблицей 2.

Например:

$110\ 111\ 100\ 001_{(2)} = 6741_{(8)}$

$2305_{(8)} = 010\ 011\ 000\ 101_{(2)}$

$1111\ 0000\ 1011_{(2)} = F0B_{(16)}$

$577_{(16)} = 0101\ 0111\ 0111_{(2)}$

Таблица 2

Система счисления			
двоичная	восьмиричная	двоичная	шестнадцатиричная
000	0	0000	0
001	1	0001	1
010	2	0010	2
011	3	0011	3
100	4	0100	4
101	5	0101	5
110	6	0110	6
111	7	0111	7
-	-	1000	8
-	-	1001	9
-	-	1010	A
-	-	1011	B
-	-	1100	C
-	-	1101	D
-	-	1110	E
-	-	1111	F

2.*Описание практической части работы:*
2.1. *Цели лабораторной работы:*
2.2. *Постановка задачи:*
2.3. *Порядок выполнения работы:*
2.3.1. Ознакомиться с теоретической частью.
2.3.2. Получить задание у преподавателя.
2.3.3. Выполнить работу.
2.3.4. Оформить отчет:
2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.
2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.
3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.
4. *Решение поставленной задачи*:
   4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.
   4.2. *Описание логической структуры программы (алгоритм решения)* содержит:
   - краткое описание схемы программы,
   - алгоритм решения (по ГОСТ) - рисунок,
   - краткое описание используемых операторов языка программирования (при необходимости).

   4.3. *Описание программы* содержит:
   - название файла, его размер,
   - текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

   4.4. *Результат работы программы:*
   - значения, полученные в результате выполнения программы
   - анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Дайте определение системы счисления ?
2. Какие бывают системы счисления ?
3. Как записываются числа в позиционной системе счисления ?
4. Что называется основанием системы счисления ?

5. Как переводится заданное число из одной системы счисления в другую ?
6. Какое число больше $100_{(4)}$ или $4_{(100)}$?
7. Какое число больше $20_{(7)}$ или $30_{(5)}$?
8. Перевести в десятичную систему число $38_{(5)}$?
9. Перевести число $222_{(8)}$ в шестнадцатиричную систему счисления ?
10. Какое число больше $100_{(8)}$ или XCLII ?

Таблица задания:

	Система счисления			
nn	Двоичная	Восьмиричная	Десятичная	16-ирична
1	1111000111001010	131532	17523	6953
2	1110100000101011	165624	14915	A4C2
3	1011011111011110	153645	32133	4334
4	1000000011011000	114424	10251	A136
5	1001011110101110	177141	58516	9CA9
6	1000011001101111	133474	44736	5BA8
7	1101000000000100	153522	48255	6F89
8	1000101100111110	166367	25414	6663
9	1001110010000101	167265	14521	371A
10	1011010100000110	121020	19816	A625
11	1010110000001010	105150	45554	0499
12	1011000100111000	117621	27846	9D56
13	1000000010001110	152340	33457	2824
14	1011001010101111	157612	18633	1935
15	1101000000110011	164442	61627	2A71
16	1110011000001001	113341	48017	CADA
17	1000111110101010	155565	19827	CCD9
18	1110110101001000	147545	29673	4EEA
19	1011011000101101	133364	29544	AB65
20	1000001100001010	102162l	40274	1541
21	1000001101010100	112643	30299	AAA1
22	1011101010110101	111136	32455	19D2
23	1011110111100100	126342	43673	CF1F
24	1101001111111111	136722	35186	D098
25	1111000000011110	110504	15165	D5B3
26	1010001101110110	123311	33907	AE66
27	1010110110100001	156125	37856	786B
28	1001000000110001	131642	20644	B4FA
29	1001000110101110	115264	61033	BEBD
30	1011101111001000	130275	60055	112D

Лабораторная работа №2
Разработка алгоритмов с разветвляющейся структурой.

1. Правила выполнения изображения схем алгоритмов (ГОСТ 19.701-90) (ИСО 5807-85).

Алгоритм - конечная последовательность точно определенных действий, приводящих к однозначному решению поставленной задачи.

Алгоритм должен обладать такими свойствами как:

- массовость (универсальность);
- определенность (детерминированность);
- правильность (адекватность);
- поэтапность (дискретность).

Алгоритмы могут быть заданы:

- словесно, с помощью слов и предложений естественного языка;
- таблично, в форме таблиц и расчетных формул;
- графически, с помощью специальных символов - блоков.

Описание алгоритмов с помощью блок-схем - наиболее наглядный и распространенный способ задания алгоритмов.

Условные обозначения и правила выполнения изображения схем алгоритмов изложены в ГОСТ 19.701-90 (ИСО 5807-85).

Стандарт не распространяется на форму записей и обозначений, помещаемых внутри символов или рядом с ними и служащих для уточнения выполняемых ими функций.

Требования стандарта являются обязательными. Схемы алгоритмов состоят из имеющих заданное значение символов, краткого пояснительного текста и соединяющих линий. Схемы могут использоваться на различных уровнях детализации, причем число уровней зависит от размеров и сложности задачи обработки данных. Уровень детализации должен быть таким, чтобы различные части и взаимосвязь между ними были понятны в целом.

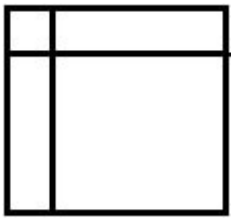
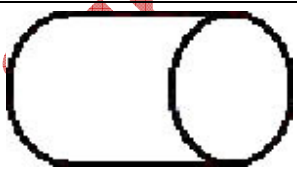
В стандарте используются следующие понятия:

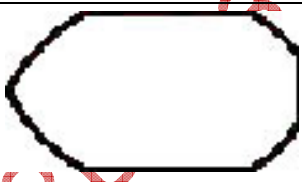
1) основной символ - символ, используемый в тех случаях, когда точный тип (вид) процесса или носителя данных неизвестен или отсутствует необходимость в описании фактического носителя данных;

2) специфический символ - символ, используемый в тех случаях, когда известен точный тип процесса или носителя данных или когда необходимо описать фактический носитель данных;

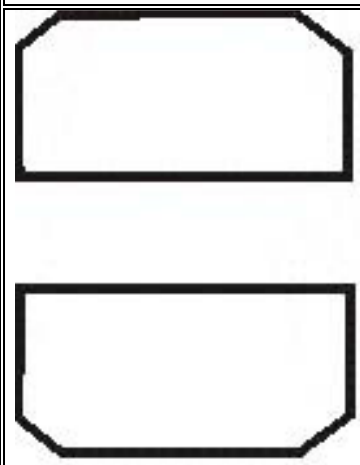
3) схема - графическое представление определения, анализа или метода решения задачи, в котором используются символы для отображения операций, данных, потока, оборудования и т. д.

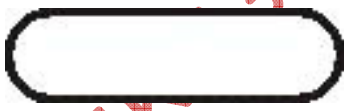
ОПИСАНИЕ СИМВОЛОВ

	1. Символы данных
	1.1. Основные символы данных
	1.1.1. Данные. Символ отображает данные, носитель данных не определен.
	1.1.2. Запоминаемые данные. Символ отображает хранимые данные в виде, пригодном для обработки, носитель данных не определен.
	1.2. Специфические символы данных
	1.2.1. Оперативное запоминающее устройство. Символ отображает данные, хранящиеся в оперативном запоминающем устройстве.
	1.2.2. Запоминающее устройство с последовательным доступом. Символ отображает данные, хранящиеся в запоминающем устройстве с последовательным доступом (магнитная лента, кассета с магнитной лентой, магнитофонная кассета).
	1.2.3. Запоминающее устройство с прямым доступом. Символ отображает данные, хранящиеся в запоминающем устройстве с прямым доступом (магнитный диск, магнитный барабан, гибкий магнитный диск).

	1.2.4. Документ. Символ отображает данные, представленные на носителе в удобочитаемой форме (машинограмма, документ для оптического или магнитного считывания, микрофильм, рулон ленты с итоговыми данными, бланки ввода данных).
	1.2.5. Ручной ввод. Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом).
	1.2.6. Карта. Символ отображает данные, представленные на носителе в виде карты (перфокарты, магнитные карты, карты со считываемыми метками, карты со сканируемыми метками).
	1.2.7. Бумажная лента. Символ отображает данные, представленные на носителе в виде бумажной ленты.
	1.2.8. Дисплей. Символ отображает данные, представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран для визуального наблюдения, индикаторы ввода информации).
	2. Символы процесса
	2.1. Основные символы процесса

	2.1.1. Процесс. Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).
	2.2. Специфические символы процесса
	2.2.1. Предопределенный процесс. Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).
	2.2.2. Ручная операция. Символ отображает любой процесс, выполняемый человеком.
	2.2.3. Подготовка. Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последовательную функцию (установка переключателя, модификация индексного регистра или инициализация программы).

	2.2.4. Решение. Символ отображает решение или функцию переключаемого типа, имеющую один вход и ряд альтернативных выходов, один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.
	2.2.5. Параллельные действия. Символ отображает синхронизацию двух или более параллельных операций.
	2.2.6. Граница цикла. Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или конце в зависимости от расположения операции, проверяющей условие.
	3. Символы линий
	3.1. Основной символ линий
	3.1.1. Линия. Символ отображает поток данных или управления.
	3.2. Специфические символы линий
	3.2.1. Передача управления. Символ отображает непосредственную передачу управления от одного процесса к другому, иногда с возможностью прямого возвращения

	к инициирующему процессу после того, как инициированный процесс завершит свои функции. Тип передачи управления должен быть назван внутри символа (например, запрос, вызов, событие).
	3.2.2. Канал связи. Символ отображает передачу данных по каналу связи.
	3.2.3. Пунктирная линия. Символ отображает альтернативную связь между двумя или более символами. Кроме того, символ используют для обведения аннотированного участка.
	4. Специальные символы
	4.1. Соединитель. Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линий и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.
	4.2. Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).
	4.3. Комментарий. Символ используют для добавления описательных комментариев или пояснительных записей в целях

	объяснения или примечаний.
● ● ●	4.4. Пропуск. Символ (три точки) используют схемах для отображения пропуска символа или группы символов, в которых не определены ни тип, ни число символов. Символ используют только в символах линий или между ними. Он применим равным образом в схемах, изображающих общие решения с неизвестным числом повторений.

Правила применения символов:

1) Символ предназначен для графической идентификации функции, которую он отображает, независимо от текста внутри этого символа.

2) Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

3) Формы символов, установленные настоящим стандартом, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов. Символы должны быть, по возможности, одного размера.

4) Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

5) Минимальное количество текста, необходимо для понимания функции данного символа, следует помещать внутри данного символа. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока.

6) Если объем текста, помещаемого внутрь символа, превышает его размеры, следует использовать символ комментария.

7) В схемах может использоваться идентификатор символов. Это связанный с данным символом идентификатор, который определяет символ для использования в справочных целях в других элементах документации (например, в листинге программы). Идентификатор символа должен располагаться слева над символом.

8) В качестве первого и последнего символа алгоритма должен быть использован символ указателя конца.

Правила выполнения соединений:

1) Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева на право и сверху вниз считаются стандартным. В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используют стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

2) В схемах следует избегать пересечение линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

3) Две или более входящие линии могут объединяться одну исходящую линию. Если две или более линий объединяются в одну линию, место объединения должно быть смещено.

4) Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

5) При необходимости линии в схемах следует разрывать для избежания излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва - внутренним соединителем.

6) Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей.

ПРИМЕНЕНИЕ СИМВОЛОВ

Символ	Наименование символа	1	2	3	4	5
Символы данных						
Основные	Данные	+	+	+	+	+
	Запоминаемые данные	+	-	+	+	+
	Специфические ОЗУ	+	-	+	+	+
	ЗУ с послед. выборкой	+	-	+	+	+
	ЗУ с прямым доступом	+	-	+	+	+
	Документ	+	-	+	+	+
	Ручной ввод	+	-	+	+	+
	Карта	+	-	+	+	+
	Бумажная лента	+	-	+	+	+
	Дисплей	+	-	+	+	+

Символы процесса						
Основные	Процесс	+	+	+	+	+
Специфические	Предопределенный процесс	-	+	+	+	-
	Ручная операция	+	-	+	+	-
	Подготовка	+	+	+	+	-
	Решение	-	+	+	-	-
	Параллельные действия	-	+	+	+	-
	Граница цикла	-	+	+	-	-
Символы линий						
Основные	Линия	+	+	+	+	+
Специфические	Передача управления	-	-	-	+	-
	Канал связи	+	-	+	+	+
	Пунктирная линия	+	+	+	+	+
Специальные символы	Соединитель	+	+	+	+	+
	Терминатор	+	+	+	-	-
	Комментарий	+	+	+	+	+
	Пропуск	+	+	+	+	+

Примечание. Знак "+" указывает, что символ используют в данной схеме, знак "-" - не используют.

1 - Схема данных;
2 - Схема программы;
3 - Схема работа системы;
4 - Схема взаимодействия программ;
5 - Схема ресурсов системы;
ОЗУ - оперативное запоминающее устройство;
ЗУ - запоминающее устройство.

Примеры построения алгоритмов

Алгоритмы бывают: линейные, разветвляющиеся, циклические. Линейный алгоритм не содержит логических условий, имеет одну ветвь обработки и изображается линейной последовательностью связанных друг с другом блоков. Разветвляющийся алгоритм содержит одно или несколько логических

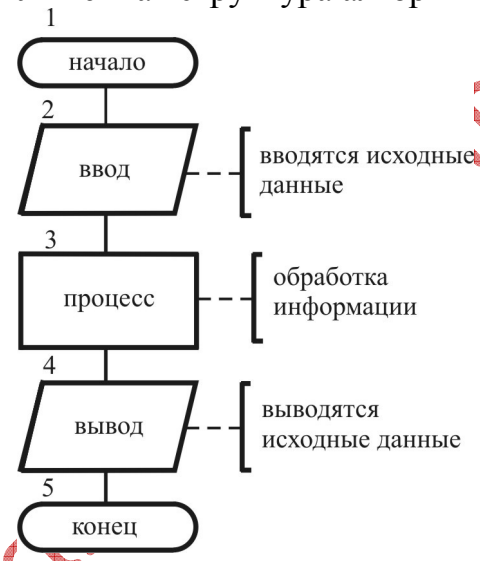
условий и имеет несколько ветвей обработки. Разветвляющиеся алгоритмы могут иметь несколько структур:

- неполная альтернатива, обработка производится при выполнении условия в противном случае обработка не производится;
- полная альтернатива, обработка производится при выполнении условия по ветви 1, в противном случае по ветви 2;
- конструкция выбора, обработка производится при выполнении одного из нескольких различных условий по соответствующей ему ветви.

Блок - Решение имеет один вход и несколько выходов, которые следует показывать:

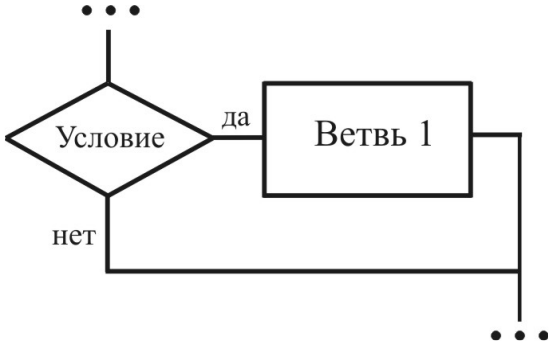
1) несколькими линиями от данного символа к другим символам;
2) одной линией от данного символа, которая затем разветвляется в соответствующее число линий.
3) каждый выход из символа должен сопровождаться соответствующими значениями условий, чтобы показать логический путь, который он представляет, с тем, чтобы эти соответствующие ссылки были идентифицированы.

Линейная структура алгоритма



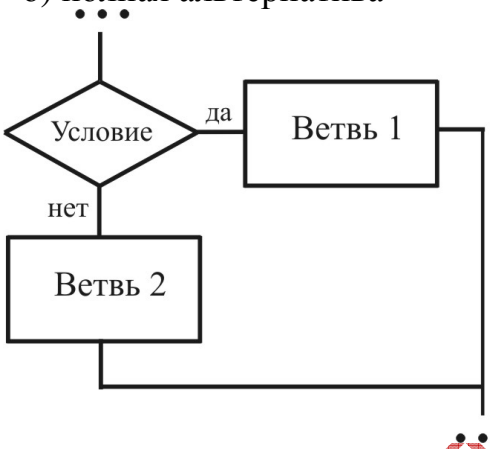
Разветвляющиеся структуры алгоритмов

а) неполная альтернатива



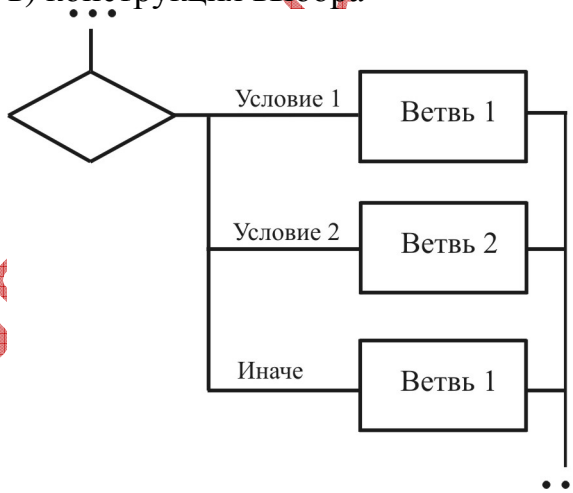
Если условие выполняется выполнить обработку информации по ветви 1.

б) полная альтернатива



Если условие выполняется выполнить обработку информации по ветви 1, иначе по ветви 2.

в) конструкция выбора



Если выполняется условие 1, то выполняется обработка по ветви 1, если выполняется условие 2, то выполняется обработка по ветви 2, если выполняется

условие 3, то выполняется обработка по ветви 3, иначе выполняется обработка по ветви 4.

Символы, рекомендованных к использованию в данной работе

Данные. Символ отображает данные, носитель данных не определен.

Процесс. Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).

Решение. Символ отображает решение или функцию переключаемого типа, имеющую один вход и ряд альтернативных выходов, один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.

Линия. Символ отображает поток данных или управления.

Соединитель. Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линий и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).

Комментарий. Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний.

2. Описание практической части работы

2.1. *Цели лабораторной работы*: Ознакомиться с понятием алгоритм, его свойствами и способами представления. Изучить основные положения ГОСТ 19.701-90 (ИСО 5807-85), связанные с изображением схем программ (графическим представлением алгоритма). Изучить построение линейных и разветвляющихся алгоритмов.

2.2. *Постановка задачи:* В соответствии с номером варианта (табл.1, 2) найти значение функции, заданной одним или несколькими математическими выражениями и для дискретной функции, заданной на нескольких интервалах.

Разработать и описать два алготитма: с линейной и разветвляющейся структурой.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы*

1 Дайте определение алгоритма ?

2 Назовите свойства алгоритмов?

3 Каким образом можно описать алгоритм решения задачи ?

4 Чем характеризуется линейная структура алгоритма ?

5 Как определяется разветвляющаяся структура алгоритма ?

6 Чем характеризуется полная и неполная альтернатива ?

7 В каких случаях используется конструкция выбора ?

8. Может ли разветвляющаяся структура иметь ветвь, направленную к началу программы ?

9. Если в алгоритме два условия (блока решения) стоят в одной ветви, где заканчивается первое и второе условия ?

10. Сколько условий можно записать в одном блоке решения ?

Таблица 1.

Задания для составления линейного алгоритма

n	Функция y(x)	n	Функция y(x)
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	**16**	$y=\ln\frac{\sqrt[3]{\lvert A^2+ctg^2 x\rvert}}{A+\ln^2 x};A=6x^x$
2	$y=\left(\frac{\lvert x^x-\sqrt{x}\rvert}{1+\sin^3 x}\right)$	**17**	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^2 x}{x^2+1}$	**18**	$y=0.5\ln(x+2)\sqrt[3]{\lvert 4-x^2\rvert}$
4	$y=\ln\frac{\lvert 2+tg^2 x\rvert}{x^{\sin x}}$	**19**	$y=0.6e^{x^2}-\frac{x-\cos^2 x}{\ln x}$
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	**20**	$y=2.8-\frac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$
6	$y=\frac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	**21**	$y=5x^x\ln^2(3+e^x)$
7	$y=\frac{\sqrt{x\sin^2 x}-1}{x+a^x};a=\sqrt{\pi}$	**22**	$y=\frac{\sqrt[3]{x}\ln 2x}{\sqrt{\lvert A^2-A-1\rvert}};A=e^{\sin x}$
8	$y=\frac{x^2-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^3$	**23**	$y=\frac{e^{2x-2}-\sqrt[3]{\lvert x-10\rvert}}{\ln^2 x}$
9	$y=\frac{\sqrt{\lvert\sin^2 x+ctgx\rvert}}{\ln x}$	**24**	$y=\frac{\lvert 3-tg^2 x\rvert}{B\ln x-B^2+1};B=e^{2(x+1)}$
10	$y=\frac{A+2ctg^2 x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	**25**	$y=\frac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$
11	$y=\frac{B^2+\sin^2(0.1x)}{B^3+2\sqrt{B}-1};B=10.5^{\ln x}$	**26**	$y=\left(\frac{x^x+\cos^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$
12	$y=\frac{\lvert 1-2tg^{2.2}x\rvert}{e^{x^2}-\ln x}$	**27**	$y=\frac{\sqrt{\lvert\cos 1.5x+tgx\rvert}}{\ln x}$

13	$y=\frac{\sqrt[3]{\lvert A^2+\ln x\rvert}}{x};A=e^{2x}$	**28**	$y=\frac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$
14	$y=\lvert A^3-10A^2+6\rvert;A=e^{x^2tgx}$	**29**	$y=\ln(x+2)\sqrt[3]{\lvert 2-x^2\rvert}$
15	$y=\sqrt[4]{\ln^2 x+C^2+C};C=2.2^{\ln x}$	**30**	$y=\frac{tgx-a^3}{a\lvert 1-x\rvert};a=\sqrt{\pi}$

Таблица 2.

Задания для составления разветвляющегося алгоритма

n	задание	n	задание
1	$y=\begin{cases}(x+1)^4+1, & x<-1\\ 1, & -1\le x<1\\ 1-(x-1)^4, & 1\le x\end{cases}$	2	$y=\begin{cases}11-7\sqrt{\lvert x-1\rvert}, & x<-3\\ x, & -3\le x<3\\ 7\sqrt{x+1}+11, & x\ge 3\end{cases}$
3	$y=\begin{cases}e^{x-1}, & x\le 1\\ lg(x+2), & x>1\end{cases}$	4	$y=\begin{cases}\ln x, & x\ge 3\\ e^x, & 2\le x<3\\ x, & x<2\end{cases}$
5	$y=\begin{cases}sin^2(x+0.1), & x\le 0.2\\ 1, & 0.2\le x\le 0.3\\ cos^2(x+0.1), & x>0.3\end{cases}$	6	$y=\begin{cases}(x+1)^4, & x<-1\\ 1-cos(\pi x), & -1\le x<1\\ -(x-1)^2, & x\ge 1\end{cases}$
7	$y=\begin{cases}1, & x<0\\ 2+cos(\pi x), & 0\le x<0.4\\ 1, & x\ge 0.4\end{cases}$	8	$y=\begin{cases}1+\cos x, & x<0.2\\ 1, & 0.2\le x<0.4\\ -1, & x\ge 0.4\end{cases}$
9	$y=\begin{cases}0, & x\le 1\\ x\ln x, & 1<x<3\\ x^x, & x\ge 3\end{cases}$	10	$y=\begin{cases}4x^2+2x-8, & x<3\\ 1, & 3\le x<3.5\\ x^3-x+10, & x\ge 3.5\end{cases}$
11	$y=\begin{cases}1-2x^2, & x\le 0\\ 1, & 0<x<0.5\\ 1+(x-0.5)^4, & x\ge 0.5\end{cases}$	12	$y=\begin{cases}0, & x\le 1\\ 1+\cos\pi x, & -1<x<0\\ 1.5, & x\ge 0\end{cases}$

13	$y=\begin{cases}cos0.5(x+1), если & x\le -1\\ 0, & -1<x<1\\ sin0.5\pi x, & x\ge 1\end{cases}$	14	$y=\begin{cases}cos0.5(x+1), если & x\le -1\\ 0, & -1<x<1\\ sin0.5\pi x, & x\ge 1\end{cases}$
15	$y=\begin{cases}x+0.6, если & x\le -1\\ x^3, если & -1<x<1\\ x-0.6, если & x\ge 1\end{cases}$	16	$y=\begin{cases}\sin^3(0.5+x), если & x<0.5\\ 1, если & 0.5\le x\le 1\\ x^2-1, если & x>1\end{cases}$
17	$y=\begin{cases}\sin^2(x+0.1), если x<0.2\\ 1, если 0.2\le x\le 0.3\\ \cos^2(x+0.1), если x>0.3\end{cases}$	18	$y=\begin{cases}4x^2+2x-8, если & x\le 1\\ 1, если & -1<x\le 1\\ x^3-x+10, если & x>1\end{cases}$
19	$y=\begin{cases}1, если & x<2\\ 2-(x-2)^2, если & 2\le x<3\\ (x-4)^2, если & x\ge 3\end{cases}$	20	$y=\begin{cases}x+0.8, если & x\le -1\\ x^3, если & -1<x\le 1\\ 2\sqrt{x-1}, если & x>1\end{cases}$
21	$y=\begin{cases}\lvert x-1\rvert, если & x<1\\ x^2, если & x\ge 1\end{cases}$	22	$y=\begin{cases}e^{x-1}, если & x\le 1\\ lg(x+2), если & x>1\end{cases}$
23	$y=\begin{cases}e^{x+1}, если & x\le 2\\ 5, если & x>2\end{cases}$	24	$y=\begin{cases}\sin^2 x, если & x\le 0.5\pi\\ 1, если & 0.5\pi<x<\pi\\ 0, если & x\ge\pi\end{cases}$
25	$y=\begin{cases}\lvert x-1\rvert, если & x<1\\ x+e^{2x}, если & x\ge 1\end{cases}$	26	$y=\begin{cases}sin0.5(x+1), если & x\ge 0\\ 0, если & x\le -2\\ \sin\pi x, если & x\ge 1\end{cases}$
27	$y=\begin{cases}3\sqrt{\lvert x+1\rvert}+5, если & x<3\\ x & 3\le x<1\\ 3.5+3\sqrt{x+1}, если & x\ge 1\end{cases}$	28	$y=\begin{cases}(x+1)^4, если & x<-1\\ 1-cos(\pi x), если & -1\le x<1\\ -(x-1)^2, если & x\ge 1\end{cases}$
29	$y=\begin{cases}4x^2-2x+8, если & x<3\\ 1, если & 3\le x<3.5\\ x^3+x+10, если & x\ge 3.5\end{cases}$	30	$y=\begin{cases}x+0.8, если & x\le -1\\ x^3, если & -1<x\le 1\\ 2\sqrt{x-1}, если & x>1\end{cases}$

Лабораторная работа №3
Разработка алгоритмов с циклической структурой.

1. Правила выполнения изображения схем алгоритмов (ГОСТ 19.701-90) (ИСО 5807-85).

Алгоритм - конечная последовательность точно определенных действий, приводящих к однозначному решению поставленной задачи. Алгоритм должен обладать такими свойствами как:

- массовость (универсальность);
- определенность (детерминированность);
- правильность (адекватность);
- поэтапность (дискретность).

Алгоритмы могут быть заданы:

- словесно, с помощью слов и предложений естественного языка;
- таблично, в форме таблиц и расчетных формул;
- графически, с помощью специальных символов - блоков.

Описание алгоритмов с помощью блок-схем - наиболее наглядный и распространенный способ задания алгоритмов.

Условные обозначения и правила выполнения изображения схем алгоритмов изложены в ГОСТ 19.701-90 (ИСО 5807-85).

Стандарт не распространяется на форму записей и обозначений, помещаемых внутри символов или рядом с ними и служащих для уточнения выполняемых ими функций.

Требования стандарта являются обязательными. Схемы алгоритмов состоят из имеющих заданное значение символов, краткого пояснительного текста и соединяющих линий. Схемы могут использоваться на различных уровнях детализации, причем число уровней зависит от размеров и сложности задачи обработки данных. Уровень детализации должен быть таким, чтобы различные части и взаимосвязь между ними были понятны в целом.

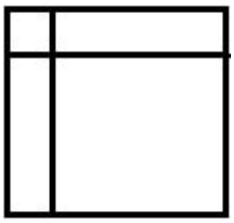
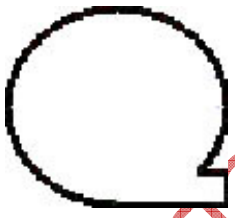
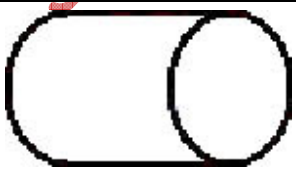
В стандарте используются следующие понятия:

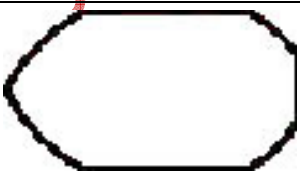
1) основной символ - символ, используемый в тех случаях, когда точный тип (вид) процесса или носителя данных неизвестен или отсутствует необходимость в описании фактического носителя данных;

2) специфический символ - символ, используемый в тех случаях, когда известен точный тип процесса или носителя данных или когда необходимо описать фактический носитель данных;

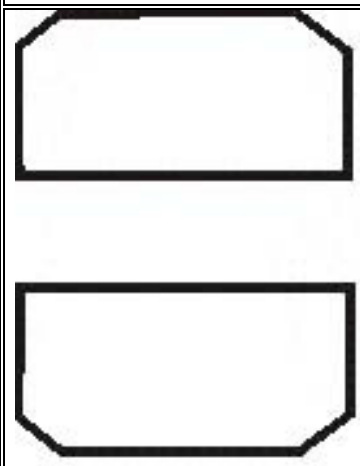
3) схема - графическое представление определения, анализа или метода решения задачи, в котором используются символы для отображения операций, данных, потока, оборудования и т. д.

ОПИСАНИЕ СИМВОЛОВ

	1. Символы данных
	1.1. Основные символы данных
	1.1.1. Данные. Символ отображает данные, носитель данных не определен.
	1.1.2. Запоминаемые данные. Символ отображает хранимые данные в виде, пригодном для обработки, носитель данных не определен.
	1.2. Специфические символы данных
	1.2.1. Оперативное запоминающее устройство. Символ отображает данные, хранящиеся в оперативном запоминающем устройстве.
	1.2.2. Запоминающее устройство с последовательным доступом. Символ отображает данные, хранящиеся в запоминающем устройстве с последовательным доступом (магнитная лента, кассета с магнитной лентой, магнитофонная кассета).
	1.2.3. Запоминающее устройство с прямым доступом. Символ отображает данные, хранящиеся в запоминающем устройстве с прямым доступом (магнитный диск, магнитный барабан, гибкий

	магнитный диск).
	1.2.4. Документ. Символ отображает данные, представленные на носителе в удобочитаемой форме (машинограмма, документ для оптического или магнитного считывания, микрофильм, рулон ленты с итоговыми данными, бланки ввода данных).
	1.2.5. Ручной ввод. Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом).
	1.2.6. Карта. Символ отображает данные, представленные на носителе в виде карты (перфокарты, магнитные карты, карты со считываемыми метками, карты со сканируемыми метками).
	1.2.7. Бумажная лента. Символ отображает данные, представленные на носителе в виде бумажной ленты.
	1.2.8. Дисплей. Символ отображает данные, представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран для визуального наблюдения, индикаторы ввода информации).

	2. Символы процесса
	2.1. Основные символы процесса
	2.1.1. Процесс. Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).
	2.2. Специфические символы процесса
	2.2.1. Предопределенный процесс. Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).
	2.2.2. Ручная операция. Символ отображает любой процесс, выполняемый человеком.
	2.2.3. Подготовка. Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последовательную функцию (установка переключателя, модификация индексного регистра или инициализация программы).

	2.2.4. Решение. Символ отображает решение или функцию переключаемого типа, имеющую один вход и ряд альтернативных выходов, один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.
	2.2.5. Параллельные действия. Символ отображает синхронизацию двух или более параллельных операций.
	2.2.6. Граница цикла. Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или конце в зависимости от расположения операции, проверяющей условие.
	3. Символы линий
	3.1. Основной символ линий
	3.1.1. Линия. Символ отображает поток данных или управления.
	3.2. Специфические символы линий

	3.2.1. Передача управления. Символ отображает непосредственную передачу управления от одного процесса к другому, иногда с возможностью прямого возвращения к инициирующему процессу после того, как инициированный процесс завершит свои функции. Тип передачи управления должен быть назван внутри символа (например, запрос, вызов, событие).
	3.2.2. Канал связи. Символ отображает передачу данных по каналу связи.
	3.2.3. Пунктирная линия. Символ отображает альтернативную связь между двумя или более символами. Кроме того, символ используют для обведения аннотированного участка.
	4. Специальные символы
	4.1. Соединитель. Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линий и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.
	4.2. Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец

	схемы программы, внешнее использование и источник или пункт назначения данных).
	4.3. Комментарий. Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний.
	4.4. Пропуск. Символ (три точки) используют схемах для отображения пропуска символа или группы символов, в которых не определены ни тип, ни число символов. Символ используют только в символах линий или между ними. Он применим равным образом в схемах, изображающих общие решения с неизвестным числом повторений.

Правила применения символов:

1) Символ предназначен для графической идентификации функции, которую он отображает, независимо от текста внутри этого символа.

2) Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

3) Формы символов, установленные настоящим стандартом, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов. Символы должны быть, по возможности, одного размера.

4) Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

5) Минимальное количество текста, необходимо для понимания функции данного символа, следует помещать внутри данного символа. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока.

6) Если объем текста, помещаемого внутрь символа, превышает его размеры, следует использовать символ комментария.

7) В схемах может использоваться идентификатор символов. Это связанный с данным символом идентификатор, который определяет символ для использования в справочных целях в других элементах документации (например, в листинге программы). Идентификатор символа должен располагаться слева над символом.

8) В качестве первого и последнего символа алгоритма должен быть использован символ указателя конца.

Правила выполнения соединений:

1) Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева на право и сверху вниз считаются стандартным. В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используют стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

2) В схемах следует избегать пересечение линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

3) Две или более входящие линии могут объединяться одну исходящую линию. Если две или более линий объединяются в одну линию, место объединения должно быть смещено.

4) Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

5) При необходимости линии в схемах следует разрывать для избежания излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва - внутренним соединителем.

6) Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей.

ПРИМЕНЕНИЕ СИМВОЛОВ

Символ	Наименование символа	1	2	3	4	5
Символы данных						
Основные	Данные	+	+	+	+	+

	Запоминаемые данные	+	-	+	+	+
	Специфические ОЗУ	+	-	+	+	+
	ЗУ с послед. выборкой	+	-	+	+	+
	ЗУ с прямым доступом	+	-	+	+	+
	Документ	+	-	+	+	+
	Ручной ввод	+	-	+	+	+
	Карта	+	-	+	+	+
	Бумажная лента	+	-	+	+	+
	Дисплей	+	-	+	+	+
Символы процесса						
Основные	Процесс	+	+	+	+	+
Специфические	Предопределенный процесс	-	+	+	+	-
	Ручная операция	+	-	+	+	-
	Подготовка	+	+	+	+	-
	Решение	-	+	+	-	-
	Параллельные действия	-	+	+	+	-
	Граница цикла	-	+	+	-	-
Символы линий						
Основные	Линия	+	+	+	+	+
Специфические	Передача управления	-	-	-	+	-
	Канал связи	+	-	+	+	+
	Пунктирная линия	+	+	+	+	+
Специальные символы	Соединитель	+	+	+	+	+
	Терминатор	+	+	+	-	-
	Комментарий	+	+	+	+	+
	Пропуск	+	+	+	+	+

Примечание. Знак "+" указывает, что символ используют в данной схеме, знак "-" - не используют.

1 - Схема данных;

2 - Схема программы;

3 - Схема работа системы;

4 - Схема взаимодействия программ;
5 - Схема ресурсов системы;
ОЗУ - оперативное запоминающее устройство;
ЗУ - запоминающее устройство.

Примеры построения алгоритмов

Алгоритмы бывают: линейные, разветвляющиеся, циклические.

Символы, рекомендованных к использованию в данной работе

Данные. Символ отображает данные, носитель данных не определен.
Процесс. Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).
Подготовка. Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последовательную функцию (установка переключателя, модификация индексного регистра или инициализация программы).
Решение. Символ отображает решение или функцию переключаемого типа, имеющую один вход и ряд альтернативных выходов, один из которых может быть активизирован после вычисления условий, определенных внутри этого символа.
Граница цикла. Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или конце в зависимости от расположения операции, проверяющей условие.
Линия. Символ отображает поток данных или управления.
Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).
Комментарий. Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний.

Пропуск. Символ (три точки) используют схемах для отображения пропуска символа или группы символов, в которых не определены ни тип, ни число символов. Символ используют только в символах линий или между ними. Он применим равным образом в схемах, изображающих общие решения с неизвестным числом повторений.

Циклический алгоритм может представлен в виде следующих основных структур:

- цикл - ДО ;
- цикл - ПОКА ;
- цикл с параметром.

Цикл - ДО начинается с выполнения тела цикла, затем проверяется условие окончания цикла, таким образом тело цикла обязательно будет реализовано хотя бы один раз. Такую разновидность цикла еще называют циклом с постусловием. В стандартном виде цикл выполняется до тех пор пока условие не станет истинным.

Словесная запись соответствующего цикла может быть определена как: повторять тело цикла до выполнения заданного условия. Графически данная конструкция может быть представлена:

а) с использованием блока решение;

б) с использованием блоков начало и конец цикла

а) Цикл - ДО с блоком решения (устаревшая конструкция алгоритма)

б) Цикл - ДО с блоками начало и конец цикла

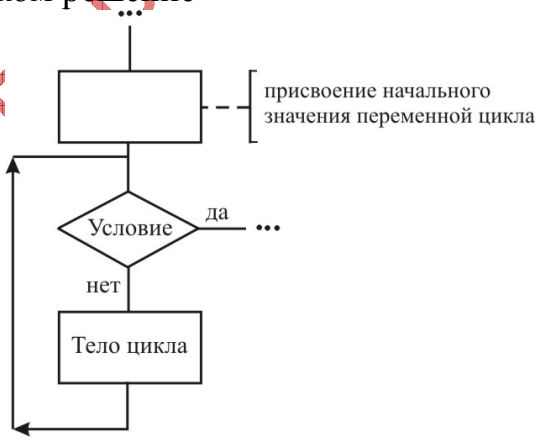
Цикл - ПОКА начинается с проверки условие окончания цикла, поэтому такую разновидность цикла называют еще циклом с предусловием. Стандартно цикл выполняется только в том случае, когда условие истинно. В частности, может оказать, что тело цикла не будет выполнено ни разу если с самого начала условие продолжения цикла не выполнялось.Словесная запись соответствующего цикла может быть определена как: пока выполняется заданное условие выполнять тело цикла.

Графически данная конструкция может быть представлена:

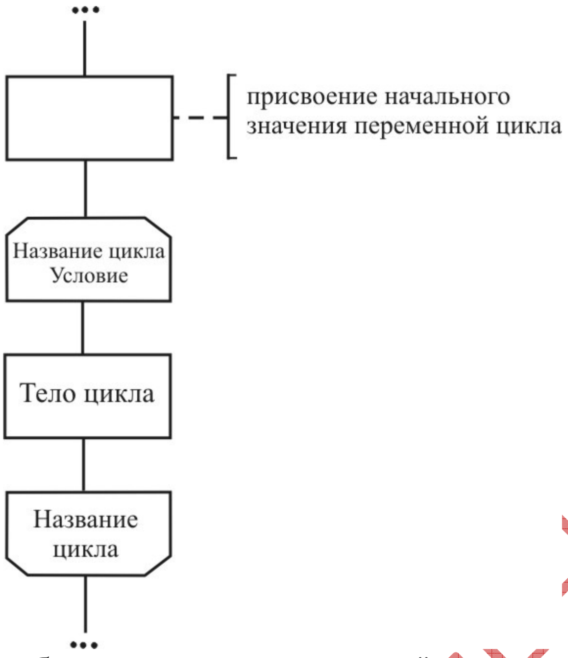
а) с использованием блока решение;

б) с использованием блоков начало и конец цикла.

а) Цикл - ПОКА с блоком решение



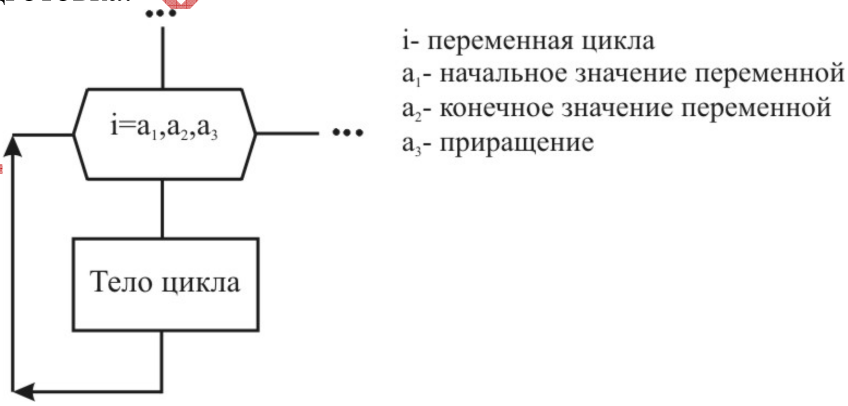
б) Цикл - ПОКА с блоками начало и конец цикла



Для того чтобы не происходило "зацикливание" (бесконечное повторение тела цикла), необходимо, чтобы в теле цикла осуществлялись преобразования, приводящие к изменению параметра входящего в условие завершения цикла. Цикл с параметром представляет собой такую управляющую структуру, которая используется в тех случаях, когда тело цикла выполняется при каждом значении некоторого параметра, изменяющегося в заданных пределах с заданным шагом, т.е. количество циклов заранее известно.

Словесная запись такой структуры может выглядеть так: для каждого параметра i , изменяющегося от A до B с шагом C, выполнять тело цикла.

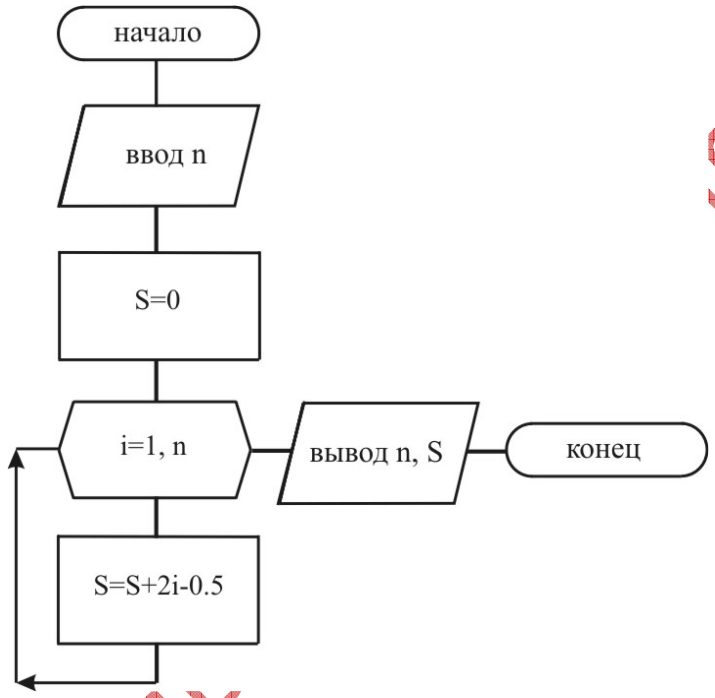
Графически данная конструкция может быть представлена с использованием символа подготовка:



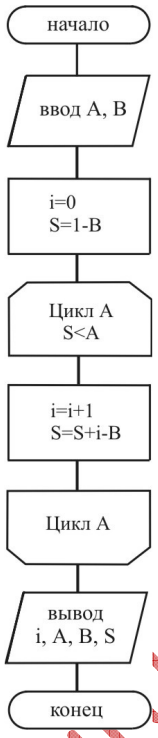
Рассмотренный цикл параметром еще называют арифметическим циклом, если шаг изменения параметра равен единице, то его можно не указывать.

Примеры построения алгоритмов:

Задача 1. Вычислить сумму n первых десяти членов прогрессии $K = 2 \times i - 0{,}5$, где K и i соответственно значение и номер члена прогрессии.



Задача 2. Определить количество первых членов прогрессии $K = i^2 - B$, сумма которых не превышает заданное число A, где K и i соответственно значение и номер члена прогрессии.



2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить построение циклических алгоритмов, решить задачу с помощью организации арифметических и логических циклов.

2.2. *Постановка задачи:* В соответствии с номером варианта найти значение функции, заданной одним или несколькими математическими выражениями.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.
2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.
3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.
4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:
- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:
- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*
- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы – отвечают на поставленную цель.

2.4. *Контрольные вопросы:*
1. Дайте определение алгоритма ?
2. Назовите свойства алгоритмов?
3. Каким образом можно описать алгоритм решения задачи ?
4. Чем характеризуется циклическая структура алгоритма ?
5. Каким образом отображается циклическая структура алгоритма на блок-схеме ?
6. Чем отличается цикл ДО от цикла ПОКА ?
7. Как изображается в схеме программы логический цикл ?
8. Какой из циклов эффективнее (быстрее выполняется в программе) логический или арифметический ?
9. Когда предпочтительнее использовать арифметический цикл в программе, написанной на языке Basic Microsoft ?
10. Что такое пустой цикл и зачем он бывает нужен в программе ?

Таблица

Задания для разработки циклических алгоритмов

n	Функция y(x)	X_{min}	X_{max}	ΔX
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5	0.02

2	$y=\left(\frac{\lvert x^{x}-\sqrt{x}\rvert}{1+\sin^{3}x}\right)$	0.4	0.8	0.04
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^{2}x}{x^{2}+1}$	0.1	0.6	0.05
4	$y=\ln\frac{\lvert 2+tg^{2}x\rvert}{x^{\sin x}}$	0.3	0.7	0.05
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	0.3	0.7	0.05
6	$y=\frac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.4	0.8	0.04
7	$y=\frac{\sqrt{x\sin^{2}x}-1}{x+a^{x}};a=\sqrt{\pi}$	0.25	0.2	0.02
8	$y=\frac{x^{2}-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^{3}$	0.1	0.6	0.05
9	$y=\frac{\sqrt{\lvert\sin^{2}x+ctgx\rvert}}{\ln x}$	0.3	0.7	0.05
10	$y=\frac{A+2ctg^{2}x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	0.1	0.6	0.05
11	$y=\frac{B^{2}+\sin^{2}(0.1x)}{B^{3}+2\sqrt{B}-1};B=10.5^{\ln x}$	0.1	0.6	0.05
12	$y=\frac{\lvert 1-2tg^{2.2}x\rvert}{e^{x^{2}}-\ln x}$	0.25	0.2	0.02
13	$y=\frac{\sqrt[3]{\lvert A^{2}+\ln x\rvert}}{x};A=e^{2x}$	0.3	0.7	0.05
14	$y=\lvert A^{3}-10A^{2}+6\rvert;A=e^{x^{2}tgx}$	0.25	0.2	0.02
15	$y=\sqrt[4]{\ln^{2}x+C^{2}+C};C=2.2^{\ln x}$	0.4	0.8	0.04
16	$y=\ln\frac{\sqrt[3]{\lvert A^{2}+ctg^{2}x\rvert}}{A+\ln^{2}x};A=6x^{x}$	0.1	0.6	0.05

17	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$	0.3	0.7	0.05
18	$y=0.5\ln(x+2)\sqrt[3]{\lvert 4-x^2\rvert}$	0.4	0.8	0.04
19	$y=0.6e^{x^2}-\frac{x-\cos^2 x}{\ln x}$	0.1	0.5	0.02
20	$y=2.8-\frac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$	0.25	0.2	0.02
21	$y=5x^x\ln^2(3+e^x)$	0.3	0.7	0.05
22	$y=\frac{\sqrt[3]{x}\ln 2x}{\sqrt{\lvert A^2-A-1\rvert}};A=e^{\sin}$	0.4	0.8	0.04
23	$y=\frac{e^{2x-2}-\sqrt[3]{\lvert x-10\rvert}}{\ln^2 x}$	0.25	0.2	0.02
24	$y=\frac{\lvert 3-tg^2 x\rvert}{B\ln x-B^2+1};B=e^{2(x+1)}$	0.4	0.8	0.04
25	$y=\frac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$	0.1	0.6	0.05
26	$y=\left(\frac{x^x+\cos^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5	0.02
27	$y=\frac{\sqrt{\lvert\cos 1.5x+tgx\rvert}}{\ln x}$	0.25	0.2	0.02
28	$y=\frac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6	0.05
29	$y=\ln(x+2)\sqrt[3]{\lvert 2-x^2\rvert}$	0.4	0.8	0.04
30	$y=\frac{tgx-a^3}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.3	0.7	0.05

Лабораторная работа №4
Знакомство с персональной ЭВМ, MS DOS, с оболочкой NC .

1.Оболочка Norton Commander

Norton Commander (NC) обеспечивает выполнение следующих функций:

- редактирование файлов;
- упрощение выполнение конкретного набора команд MS DOS.

Достоинством Norton Commander является то, что в нем у вас остается возможность выполнения любых команд MS DOS (кроме PATH и SET).

ВЫЗОВ Norton Commader

[d:] [path] NC
либо [d:] [path] NCSMALL,
где d - диск, path - путь доступа.

На экране появляются панели Norton Commander, внизу экрана -меню функциональных клавиш NC (максимальное количество панелей - 2,одна - активная).

ИСПОЛЬЗОВАНИЕ ФУНКЦИОНАЛЬНЫХ КЛАВИШ

Клавиша	Назначение
F1	Помощь (Help)
F2	Вызов созданного пользователем меню
F3	Просмотр (View) указанного файла
F4	Редактирование (Edit) указанного файла
F5	Копирование (Copy) выбранной группы файлов
F6	Переименование или перемещение (Ren/Mov) выбранной группы файлов, переименование каталога
F7	Создание (Mkdir) каталога
F8	Удаление (Delete) выбранной группы файлов или каталога
F9	Удаление (Delete) выбранной группы файлов или каталога
F10	Выход (Quit) в режим команд MS-DOS
Shift-F3	Работает в любом режиме так, как F3 работает в режиме без панелей
Shift-F4	Работает в любом режиме так, как F4 работает в

	режиме без панелей
Shift-F5	Работает в любом режиме так, как F5 работает в режиме без панелей
Shift-F6	Работает в любом режиме так, как F6 работает в режиме без панелей
Shift-F7	Работает в любом режиме так, как F7 работает в режиме без панелей
Shift-F8	Работает в любом режиме так, как F8 работает в режиме без панелей
Shift-F9	Сохранить настройку в файле NC.INI

Инструкция по Norton Commander

Alt-F1	Выбрать диск (левая панель)
Alt-F2	Выбрать диск (правая панель)
Alt-F3	Просмотр (View) указанного файла, но для файлов типов dbf и wk? не вызываются программы dbview.exe и 123view.exe
Alt-F4	Редактирование (Edit) указанного файла альтернативным редактором
Alt-F5	Не задействована
Alt-F6	Не задействована
Alt-F7	Поиск файла(ов) по всем каталогам диска
Alt-F8	Выбор команды из стека сохраненных команд
Alt-F9	Переход в 43-строчный режим (для EGA,VGA)
Alt-F10	Создать окно с деревом каталогов активной панели (NCD-Tree)
Ctrl-F1	Зажечь/погасить левую панель
Ctrl-F2	Зажечь/погасить правую панель

Если вы выполняете команду из меню NC, то прервать работу можно с помощью клавиш F10 или ESC.

УКАЗАНИЕ И ИСПОЛНЕНИЕ ФАЙЛОВ

В данной конструкции встречаются термины - "указанный файл" и" выбранный файл или группа файлов". "Указанный файл" в конкретный момент времени может быть только один и от остальных файлов он визуально отличается (выделен фоновой полосой). "Выбранный файл или группа

файлов" визуально отличаются от остальных файлов (выделено цветом, яркостью или вертикальной яркой полоской справа). Исполняется всегда указанный файл. Для работы в NC используют клавиатуру, "мышь" либо и то и другое.

При использовании клавиатуры:
/УКАЗАТЬ ФАЙЛ/ - использовать клавиши направлений;
/ИСПОЛНИТЬ/ - нажать клавишу ENTER.
При использовании "мыши":
/УКАЗАТЬ ФАЙЛ/ - нажать левую кнопку "мыши";
/ИСПОЛНИТЬ/ - дважды быстро нажать левую кнопку.

ВЫБОР И ОТМЕНА ВЫБОРА ФАЙЛОВ

С использованием клавиатуры выбор и отмена группы файлов производится двумя способами:

1. Нажать серую клавишу (на цифровой клавиатуре) "+" (для выбора) или "-" (для отмены выбора), после чего в диалоговом "окне" указать конкретное имя файла (группы файлов),используя при необходимости символы * и ?;
2. Указать файл. Нажать клавишу INSERT (если файл уже выбран, то эти действия приведут к отмене выбора).

При использовании "мыши":
/ВЫБРАТЬ ФАЙЛ/ - нажать правую кнопку "мыши", повторное нажатие приведет к отмене выбора.

Над выбранными файлами могут производиться групповые операции копирования, удаления, пересылки и переименования.

ПАНЕЛИ Norton Commander

Активна только одна панель. Настройка панелей производится в меню NC (F9) по вашему желанию.

Комбинации клавиш	Назначение
TAB или Ctrl+I	Переключение активной панели
Ctrl+O	Удаление панелей с экрана
Ctrl+P	Удаление неактивной панели
Ctrl+U	Обмен панелей местами
Ctrl+L	Просмотр статуса активной панели число и суммарный объём файлов текущего каталога, ёмкость ОЗУ, текущего диска, свободное место в них

Повторное использование указанных комбинаций клавиш имеет обратное действие.

РАБОТА С КАТАЛОГАМИ В ПАНЕЛЯХ Norton Commander.

Войти в указанный каталог: указать его и нажать ENTER или исполнить, используя манипулятор.

Выход в родительский каталог: указать файл(две точ.) и нажать ENTER или исполнить, используя манипулятор, или нажать Ctrl+PgUp без указания каталога.

Повторное чтение каталога - Ctrl+R (выбранные группы файлов при этом не сохраняются), возможен переход на другой диск.

Комбинации клавиш	Назначение
Ctrl+\	Переход в корневой каталог

ПРИМЕЧАНИЕ: используя NC легко удалять каталоги файлов.

Процедура удаления состоит в следующем:

- войти в каталог;
- произвести выбор всех файлов каталога (см.4)
- выполнить F8 (Delete);
- выйти в родительский каталог (при этом каталог, который вы хотите удалить указан полосой);
- выполнить F8 (Delete)

ОПИСАНИЕ КОМАНД Norton Commander.

Команды View (просмотреть) и Edit (редактировать) из меню NC выполняются с указанным файлом.

Команды COPY(копировать), REN/ MOV(заменить имя / переместить),DELETE (удалить)работают либо с указанным файлом, либо с выбранной группой файлов. При этом появляется соответствующее диалоговое "окно". Для команд COPY и REN/ MOV указывают путь и имя файла. Для команды DELETE, подтверждают удаление, выбрав "delete", и отменяют, выбрав "cancel". Возможно удаление файлов с атрибутом только чтение", при этом запрашивается дополнительное подтверждение.

Возможен ввод из командной строки любой команды MS-DOS, при этом введенные команды помещаются в стек и могут быть в последствии извлечены

из него в область ввода. При редактировании команды в области ввода используются приведенные в следующей таблице клавиши.

Клавиши редактирования командной строки и вспомогательные клавиши

Комбинации клавиш	Назначение
Ctrl+B	Удалить/зажечь меню клавиш
Ctrl+E	Показать в области ввода предыдущую команду из стека
Ctrl+X	Показать в области ввода следующую команду из стека
Ctrl+ J	Поместить в область ввода наименование указанного файла (курсор должен находиться в конце области ввода и предшествоваться пробелом)
Ctrl+ D	Перемещение курсора на символ вправо
Ctrl+ S	Перемещение курсора на символ влево
Ctrl+ → Ctrl+ F	Перемещение курсора на слово вправо
Ctrl+ ← Ctrl+ A	Перемещение курсора на слово влево
Ctrl-Home	Перемещение курсора в начало строки
Ctrl-End	Перемещение курсора в конец строки

ПРОСМОТР(View) УКАЗАННОГО файла.

Укажите файл.
Выполните View (F3) из NC меню.

Клавиши, используемые в режиме просмотра файла

Комбинации клавиш	Назначение
F7	Поиск(Search) цепочки, указанной в диалоговом "окне" (регистр не имеет значения).
Shift-F7	Повторить последний поиск

↑	Экран перемещается вверх по тексту
↓	Экран перемещается вниз по тексту
→	Экран перемещается вверх по тексту на страницу
←	Экран перемещается вниз по тексту на страницу
→	Текст на экране перемещается влево
←	Текст на экране перемещается вправо
Ctrl+	Текст на экране перемещается влево на 40 символов
Ctrl+ ←	Текст на экране перемещается вправо на 40 символов
Home	Просмотр начала файла
End	Просмотр конца файла
F10 или Esc	Выход(Quit) в NC

- Для просмотра файла используйте клавиши PgUp и PgDn;
- Для поиска указанной цепочки используйте функциональную клавишу F7. В диалоговом "окне" наберите требуемую цепочку (регистр не имеет значения). В результате поиска первая найденная строка с указанной цепочкой перемещается в начало экрана.

РЕДАКТИРОВАНИЕ (Edit) УКАЗАННОГО ФАЙЛА

Укажите файл.
Выполните Edit (F4 или Alt + F4, в зависимости от настройки NC) из NC меню.

Клавиши, используемые в режиме редактирования файла.

Клавиши	Назначение
F1	Помощь(Help) [выход Ctrl+Q или ESC]
F2	Запись(Save) файла на диск
F7	Поиск(Search) цепочки, указанной в диалоговом "окне" (регистр не имеет значения)
F10 или Esc	Выход(Quit) в NC с запросом о записи файла на диск

Shift-F2	Записать файл на диск с другим именем
Shift-F7	Повторить последний поиск
Shift-F10	Выход(Quit) в NC с записью файла

Перемещение курсора в Edit

Чтобы переместить курсор на:	Выполните:
символ влево	Ctrl+S или ←
символ вправо	Ctrl+D или →
слово влево	Ctrl+A или Ctrl+ ←
слово вправо	Ctrl+F или Ctrl+ →
строку вверх	Ctrl+E или ↑
строку вниз	Ctrl+X или ↓
предыдущую страницу	Ctrl+R или PgUp
следующую страницу	Ctrl+C или PgDn
в конец строки	<End>
в начало строки	<Home>
в конец текста	Ctrl + <End>
в начало текста	Ctrl + <Home>

ПРИМЕЧАНИЕ: Чтобы вставить строку, нажмите клавишу ENTER в конце строки, после которой производите вставку.

Удаление слов, символов, строк в Edit и в командной строке

Чтобы удалить:	Выполните:
строку	Ctrl+Y
все от курсора до конца строки	Ctrl+K
символ, на который указывает курсор	Delete или Ctrl+G
символ слева от курсора	Backspace или Ctrl+H
слово слева от курсора	Ctrl+W или Ctrl+Backspace
слово справа от курсора	Ctrl+T

НАСТРОЙКА Norton Commander

Меню настройки Norton Commander вызывается нажатием клавиши F9 или клавиш Ctrl+N или с помощью "мыши" (см. следующий раздел). Оно расположено в верхней строке экрана и содержит следующие опции: "Left", "Files", "Commands", "Options", "Right". Для выбора опции необходимо либо нажать клавишу с первой буквой опции, либо подвести клавишами с горизонтальными стрелками к опции курсор и нажать на ввод, либо использовать "мышь". когда опция выбрана, под ней открывается вертикальное подменю, выбор из которого производится так же, как из главного меню, с тем только отличием, что курсор перемещается клавишами с вертикальными стрелками. Некоторые из опций этих подменю в свою очередь открывают подменю или же окна ввода.

Опции "Left" и "Right" управляют состоянием соответственно левой и правой панелей. Их подменю содержат три группы опций:

- первая включает опции задающие тип панели: "Brief", "Full", "Info", "Tree" и "On/Off". Панель типа "Brief" содержит только имена файлов, "Full" - имена, размер и дату создания (коррекции) файлов, "Info" - информацию о версии Norton Commander, общем и свободном объеме ОЗУ, общем и свободном пространстве на текущем диске, числе и суммарном объеме файлов текущего каталога, "Tree" - содержит дерево каталогов текущего диска и позволяет быстро путешествовать по нему. Опция "On/Off" зажигает/гасит панель (аналогично Ctrl+F1/Ctrl+F2).

- вторая включает опции, задающие способ сортировки файлов в панели:

"Name", "eXt", "tiMe", "Size", "Unsorted" - по именам, расширениям, дате/времени создания, размеру, не сортируя, соответственно.

- третья включает одну опцию, задающую считывание диска (аналогично

Alt+F1/Alt+F2).

Подменю опции "Files" содержит опции по своему действию полностью аналогичные функциональным клавишам F1 - F8, F10.

Подменю опции "Commands" содержит три группы опций. Опции первой группы по своему действию полностью аналогичны функциональным клавишам Alt+F7 - Alt+F10. Из опций второй группы две эквивалентны комбинациям клавиш Ctrl+U, Ctrl+O, а третья - "**Compare directory**" - осуществляет сравнение оглавлений левой и правой панелей, выбирая в каждом из них те файлы, которые в другом либо отсутствуют либо имеют более раннюю дату создания (коррекции). Третья содержит две опции - "Menu file edit" и "**eXtension file edit**" - реализующие наиболее мощные и удобные функции Norton Commander.

"**Menu file edit**" позволяет пользователю создавать или изменять собственные меню (файлы с именами nc.mnu), вызываемые нажатием клавиши F2. При этом в каждом каталоге может быть создано локальное меню. Там, где локальных меню нет, используется главное меню, расположенное в корневом каталоге диска, с которого берется COMMAND.COM. В принципе меню может быть создано любым текстовым редактором, но режим "Menu file edit" удобен тем, что в нем в нижней части экрана находится описание структуры меню.

Структура меню такова:

```
' Комментарий
M: Имя первой строки меню
Первая команда
.............
Последняя команда
' Комментарий
N: Имя второй строки меню
Первая команда
.............
Последняя команда
' Комментарий
Z: Имя последней строки меню
Первая команда
.............
Последняя команда
' Комментарий
```

Здесь:

Комментарий - любая последовательность символов, Norton Commander его игнорирует.

M,N,Z - клавиши, выбирающие соответствующую строку меню (функциональные клавиши обозначаются F1, ..., F10).

Имя строки меню - текст, появляющийся в меню у обозначения соответствующей клавиши.

Первая команда, ..., последняя команда - команды MS-DOS, выполняемые при выборе строки меню, ими могут быть любые команды MS-DOS. Следует иметь ввиду, что Первая команда должна начинаться в той же колонке, что и Имя строки меню.

"eXtension file edit" - позволяет создать в каталоге из которого был запущен Norton Commander файл расширений nc.ext, указывающий Norton Commander что делать с файлом при "исполнении", в зависимости от расширения имени

файла. В принципе файл расширений может быть создан любым текстовым редактором, но режим "eXtension file edit" удобен тем, что в нем в нижней части экрана находится описание структуры файла
расширений.

Эта структура такова:

ex1: Команда Параметры

.....

exn: Команда Параметры

Здесь:

ex1, ...,exn - расширения имен файлов (допустимо использование в них символов шаблона * и ?).

Команда - команда MS-DOS, выполняемая при "исполнении" файлов с соответствующим расширением.

Параметры - параметры выполняемой команды, символ ! имеет следующие специальные значения: ! - имя "исполняемого" файла (без расширения), !.! - имя "исполняемого" файла (с расширением), !\ - текущий путь, !: - текущий диск.

Следует иметь ввиду, что число строк файла nc.ext не должно превышать 15 , так как шестнадцатая строка обрабатываеся не полностью, а последующие просто игнорируются.

Подменю опции "Options" содержит опции "Colors...", "Auto menus", "Path prompt", "Key bars", "Full screen", "Mini status", "Ins moves down", "cLock", "Editor...", задающие режимы работы Norton Commander.

Опции "Auto menus", "Path prompt", "Key bars", "Full screen", "Mini status", "Ins moves down", "cLock" работают как двоичные переключатели, включая и выключая:

"**Auto menus**" - автоматическое появление пользовательского меню при старте Norton Commander и после завершения выполнения выбранной строки меню;

"**Path prompt**" - высвечивание в приглашении текущего пути;

"**Key bars**" - высвечивание в нижней строке экрана назначения функциональных клавиш;

"**Full screen**" - полноэкранный режим панелей;

"**Mini status**" наличие строки статуса (имя, размер, дата создания/ коррекции) "указанного" файла в нижней строке панелей;

"**Ins moves down**" - сдвиг курсора на следующую строку панели при нажатии клавиши Ins для выбора/отмены файлов;

"**cLock**" - высвечивание часов в правом верхнем углу экрана.

Опция "**Colors**..." открывает подменю с тремя режимами работы экрана: "B&W" (черно-белый), "Color"(цветной), "Lap-top" (для переносных

компьютеров). Следует заметить, что последний режим удобен для всех типов экранов.

Опция "**Editor**..." открывает подменю с двумя опциями - "Built-in" и "External". Выбор опции из этого подменю указывает Norton Commander какой редактор использовать для редактирования "указанного" файла по нажатию клавиши F4 - внутренний ("Built-in") или внешний ("External"). если выбирается "External", то открывается окно для ввода его имени (включая диск и путь). Заметим, что какой бы из них не был выбран, другой остается доступен через нажатие Alt+F4.

Использование "мыши" при работе с Norton Commander

При работе с Norton Commander "мышь" может использоваться пятью различными способами:

- в панелях для указания, выбора, отмены и исполнения файлов, прокрутки содержимого панелей;
- в командах View и Edit для позиционирования курсора и прокрутки текста в окне;
- вместо функциональных клавиш;
- в окне запроса для подтверждения или отказа от действия;
- в режиме настройки Norton Commander.

В панели для:

- указания файла - подвести к нему курсор "мыши" и нажать левую кнопку "мыши";
- исполнения файла - подвести к нему курсор "мыши" и быстро нажать левую кнопку "мыши" дважды;
- выбора файла - подвести к нему курсор "мыши" и нажать правую кнопку "мыши";
- отмены выбранного файла - подвести к нему курсор "мыши" и нажать правую кнопку "мыши";
- прокрутки файлов - подвести к началу (концу) панели курсор "мыши" и нажать левую кнопку "мыши", если вместо левой нажать правую кнопку - прокручиваемые файлы будут выбираться.

В командах View и Edit для:

- позиционирования курсора подвести к нужному месту курсор "мыши" и нажать левую кнопку "мыши";
- прокрутки текста - подвести к краю экрана курсор "мыши" и нажать левую кнопку "мыши".

Вместо функциональных клавиш - подвести курсор "мыши" к подсказке в нижней строке экрана и нажать левую кнопку "мыши", нажатие правой кнопки "мыши" эквивалентно нажатию функциональной клавиши с Shift.

В ответ на запрос Norton Commander на подтверждение действия нажатие правой кнопки "мыши" подтверждает действие, левой - отменяет. На двухкнопочной "мыши" действие отменяется одновременным нажатием

кнопок. Можно также выбрать подтверждение или отмену установив курсор

"мыши" на соответствующую подсказку в окне запроса и дважды быстро нажав левую кнопку "мыши".

Для настройки Norton Commander необходимо либо поместить курсор в поле подсказки клавиши F9, либо в верхнюю строку экрана и нажать левую кнопку "мыши" - появится меню настройки, затем подвести курсор "мыши" к нужной опции и нажать левую кнопку "мыши" для подтверждения или правую для отмены.

Устройство персонального компьютера (ПК)

Все персональные ПК состоят из четырех основных частей: системного блока, монитора, клавиатуры и мыши. Кроме того ПК может включать ряд дополнительных устройств: блок бесперебойного питания, акустические системы (колонки), принтер, сканер, и т.д.

Системный блок состоит из корпуса, в котором размещаются: материнская плата, блок питания, устройства для записи и чтения информации (дисководы, CD-ROM и т.д.).

Клавишное устройство является основным устройством ввода информации, оно обеспечивает диалоговое общение пользователя с ПЭВМ. Клавишное устройство выполняет следующие функции:

- ввод команд пользователя, обеспечивающих доступ к ресурсам ПЭВМ в различных режимах;
- запись, корректировку и отладку программ пользователя;
- ввод данных и команд в процессе решения задач на ПЭВМ. Клавишное устройство включает в себя клавиатуру и электронный блок кодирования символов клавиатуры. Клавиатура состоит из клавиш, которые можно разбить на следующие группы:

1) алфавитно-цифровые и знаковые клавиши;
2) функциональные клавиши;
3) служебные клавиши для управления перемещения курсором, для управления редактированием текстов, смены и фиксации регистров, модификации кодов клавиш.

Монитор относится к классу внешних устройств оперативного вывода данных на экран электронно-лучевой трубки (ЭЛТ). В ПЭВМ символьный дисплей осуществляет:

- вывод содержимого текстовых файлов, например, исходных модулей программ на языке высокого уровня (Бейсик, Паскаль и т.д.);

- информационное взаимодействие с пользователем при диалоговой обработке данных.

Мышь - это самое распространенное сегодня устройства для дистанционного управления графическими изображениями на экране. Набирать какие-либо команды не нужно, достаточно при работе в программе указать мышкой нужную операцию меню или иконку в окне на экране, а затем щелкнуть кнопкой. Вот и все, что требуется, а остальное сделает программа.

Печатающее устройство (принтер) предназначен для вывода результатов обработки информации на бумажный бланк, т.е. для документального оформления итоговых данных.

Сканер позволяет оптическим путем вводить черно-белую или цветную печатную графическую информацию с листа бумаги. Отсканировав рисунок и сохранив его в виде файла на диске, можно затем вставить его изображение в любое место в документе с помощью программы текстового процессора.

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить основные команды и функции, выполняемые в среде MS DOS с помощью оболочки Norton Commander..

2.2. *Постановка задачи:* Ознакомиться с размещением файлов на жестком магнитном диске и в соответствии с заданием выполнить несколько команд, связанных с просмотром каталогов и файлов. Отразить указанные действия в отчете.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.
2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.
3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.
4. *Решение поставленной задачи*: Отразить последовательность команд и действий, необходимых для выполнения заданных функций.

Выводы - отвечают на поставленную цель и отражают результаты проделанной работы.

2.4. *Контрольные вопросы:*

1.Как в Norton Commander запускаются команды MS DOS ?
2. Укажите назначение операционной системы ?
3. Какие действия позволяет совершать пользователю операционная система?
4. Для чего предназначена программа NC.exe ?
5. Что такое дерево каталогов ?
6. Назначение команды REM ?
7. Что такое панель и сколько их в NC ?
8. Какие устройства ввода используются для выполнения команд в NC ?
9. Для чего предназначена клавиша F1 в NC ?
10. Как узнать на какой версии Вы работаете ?

Таблица

Задания

n	задание	n	задание
1	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt). Удалить созданный файл и каталог.	16	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутрь него скопировать любой файл из другого каталога, просмотреть содержимое этого файла. Удалить созданный файл и каталог.
2	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутрь него скопировать любой файл из другого каталога, отредактировать содержимое файла. Удалить созданный файл и каталог.	17	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутрь него перенести любой файл из другого каталога, просмотреть содержимое этого файла. Перенести его назад. Удалить созданный каталог.
3	Зайти в рабочий каталог, в нем создать подкаталог с номером группы	18	Зайти в рабочий каталог на двух панелях, выбрать два подкаталога и

	(Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt). Переименовать этот файл (в Readme.txt) Удалить созданный файл и каталог.		сравнить их содержимое, произвести сортировку файлов по имени.
4	Зайти в рабочий каталог на двух панелях, выбрать два подкаталога и сравнить их содержимое, произвести сортировку файлов по дате создания. Просмотреть содержимое последнего в списке файла.	19	Зайти в рабочий каталог, выбрать подкаталог и произвести сортировку файлов по расширению. Просмотреть содержимое файла с расширением .bas.
5	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt), установить на него атрибут «только чтение». Удалить созданный файл и каталог.	20	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt), установить на него атрибут «архивный». Удалить созданный файл и каталог.
6	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), просмотреть дерево каталогов на диске, занести в протокол путь к созданному подкаталогу.	21	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), поменять панели местами, погасить левую панель.
7	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), погасить	22	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), сделать левую

	панели, включить левую панель. Отсортировать файлы по размеру. Просмотреть последний файл. Если файл просмотреть не удастся, обосновать это.		панель информационной, выбрать средний режим панели файлов.
8	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.rrr). Раскрасить название файла в красный цвет. Удалить созданный файл и каталог.	23	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), зайти в рабочий каталог на двух панелях, выбрать два подкаталога, на левой панели выбрать средний режим панели файлов, чем он отличается от краткого и полного.
9	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), зайти в рабочий каталог на двух панелях, выбрать два подкаталога, на левой панели выбрать широкий режим панели файлов, чем он отличается от среднего и полного.	24	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.rrr). Раскрасить название файла в желтый цвет. На левой панели перейти на другой диск. Удалить созданный файл (Gxxxxxx.rrr).
10	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), зайти в рабочий каталог на двух панелях, выбрать два подкаталога, на левой панели выбрать	25	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), зайти в рабочий каталог на двух панелях, выбрать два подкаталога, на левой

	краткий режим панели файлов, чем он отличается от среднего и полного.		панели выбрать полный режим панели файлов, чем он отличается от альтернативного полного.
11	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутрь него копировать любой файл из другого каталога с расширением .txt, просмотреть содержимое этого файла. Переименовать данный файл. Удалить созданный каталог.	26	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), зайти в рабочий каталог на двух панелях, выбрать два подкаталога, на левой панели выбрать детальный режим панели файлов, описать его особенности
12	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри создать файл с расширением .ftr, внутри него написать сведения о лабораторной работе, на другой панели войти в этот же каталог, выбрать режим быстрого просмотра. Какие сведения дает данный ражим.	27	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt). На левой панели сменить диск. Поменять панели местами. Удалить созданный файл и каталог.
13	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt). Просмотреть историю папок. Какую информацию	28	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутрь него перенести любой файл из другого каталога, просмотреть содержимое этого файла. Перенести его назад. Удалить созданный каталог.

	она дает.		
14	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt). Выйти из своей папки. При помощи «Поиска папки» найти свою папку на диске. Зарисовать ее местоположение.	29	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), погасить панели, включить левую панель. Отсортировать файлы по размеру. Просмотреть последний файл. Если файл просмотреть не удастся, обосновать это.
15	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt), установить на него атрибут «только чтение». Удалить созданный файл и каталог.	30	Зайти в рабочий каталог, в нем создать подкаталог с номером группы (Gxxxxxx), внутри него создать подкаталог с именем temp, внутри которого создать файл с именем (Gxxxxxx.txt), установить на него атрибут «архивный». Удалить созданный файл и каталог.

Лабораторная работа №5
Организация вычислений на алгоритмическом языке QB.

1. Запуск среды программирования QuickBasic 4.xx и 7.xx.

Для запуска среды QuickBasic 4.xx необходимо найти на диске исполняемый файл qb.exe или qbx.exe. После этого необходимо набрать на клавиатуре команду < qb [/ключи] > (или < qbx [/ключи] >)и нажать клавишу <ENTER>.

[ ]	необязательный параметр
/b	режим работы с черно-белым монитором
/h	режим максимально возможного разрешения экрана для используемого оборудования
/ah	позволяет динамическим массивам, переменным и константам быть больше 64 К
/c:<размер буфера>	устанавливает размер буфера для обмена данными
/cmd <строка>	используется как установка для функции COMMAND$ (должен быть последним в командной строке DOS)
/G	установка режима максимально возможной скорости для графики (обычно для мониторов типа A CGA)
/l<имя библиотеки>	загружается указанная библиотека
/mdf	указывается системе на преобразование чисел из формата IEEE в двоичный формат Microsoft
/run <исходный файл>	загрузка, компиляция и запуск на счет исходного файла
<исходный файл>	загрузка исходного файла

Допускается использовать любые комбинации ключей и в любом порядке за исключением команды:/cmd < строка > она должна следовать последней. После запуска программы на дисплее появляется следующее изображение (рис. 1).

Рис.1.

На данном рисунке можно выделить 3 зоны:

1- зона главного меню;

2- зона (окно) текстового редактора (Untitled);
3- зона (окно) непосредственного выполнения (Immediate).
В зоне меню имеются следующие ключевые слова:
File команды работы с файлами;
Edit команды работы с редактором;
View команды просмотра;
Search команды поиска, поиска и замены фрагментов текста;
Run команды выполнения программы;
Debug команды облегчающие отладку;
Calls команды вызова процедур;
Help команды помощи пользователю.

Организация диалога в среде программирования QuickBASIC

В среде программирования QuickBASIC используется оконная технология организации диалога, рассчитанная на использование в качестве устройств ввода клавиатуры и специального манипулятора мышь. Маркер курсора клавиатуры или мыши позиционируется в одном из окон и нажимается клавиша <ENTER> или соответствующая кнопка мыши для выполнения выбранного действия связанного с этим окном.

Рассмотрим подробнее технологию организации диалога с использованием клавиатуры. Для выбора какого либо пункта меню необходимо нажать клавишу <ALT> фон слова File изменится и среда ожидает выбора конкретного раздела меню. Управляя курсором клавишами стрелки выбрать нужный раздел, после этого необходимо нажать клавишу <ENTER>. Под выбранным пунктом меню появится подменю. Управляя курсором клавишами < стрелки > выбрать нужный раздела подменю ,после этого необходимо нажать клавишу <ENTER>. В случае необходимости отмены исполнения выбранного пункта меню достаточно нажать клавишу <ESC>. Рассмотрим подробнее каждый из этих пунктов.

Работа с файлами

Подменю состоит из следующих команд:

New program	создание новой программы при выборе данного пункта меню появляется диалоговое окно
Open program...	вызов ранее созданной и сохраненной программы при выборе данного пункта меню появляется диалоговое окно
Merge...	выбраны файл вставляется в текущий ниже места положения курсора при выборе данного

	пункта меню появляется диалоговое окно
Save	запись текущего модуля в текущий катало с именем по умолчанию
Save As ..	запись текущего модуля с выбором каталога и заданием имени при выборе данного пункта меню появляется диалоговое окно
Save All	запись всех модулей находящихся в памяти сохранением информации в специализированном файле. Файлы помещаются в текущий каталог с именами по умолчанию.
Create File	создание нового файла с указанием его типа (в виде отдельного файла, включаемого файла, текстового документа) при выборе данного пункта меню появляется диалоговое окно
Load File	команда аналогичная Open program с добавлением возможности указания типа файла (в виде отдельного файла, включаемого файла, текстового документа) при выборе данного пункта меню появляется диалоговое окно
Unload File	вывод каталога файлов находящихся в памяти для выбора файлов и удаления их из памяти при выборе данного пункта меню появляется диалоговое окно
Print	вывод на печатающее устройство текста программы при выборе данного пункта меню появляется диалоговое окно
Dos Shell	возможность кратковременного выхода в DOS без выгрузки системы QuickBasic. Для возвращения надо набрать EXIT и нажать клавишу <ENTER>
Exit	выход из системы QuickBasic, если имеются не сохраненные изменемия текстов в файлах система запросит разрешение сохранить их перед выходом

Редактирование

Undo	<ALT>+<BACK SPASE>	восстановление первоначального вида редактируемой строки

Cut	<SHIFT>+<DEL>	удаление выбранного фрагмента текста с сохранением его в буфере
Copy	<CTRL>+<INS>	копирование выбранного фрагмента текста в буфер
Paste	<SHIFT>+<INS>	копирование содержимого буфера в текущий текст. Место вставки определяется текущим положением курсора.
Clear	<DEL>	удаление выбранного фрагмента текста без сохранения его в буфере, если такового нет то стирается символ над маркером
New Sub...		страницы с новой процедурой
New FUNCTION...		страницы с новой процедурой-функцией
Sintax Checking		включение/отключение автоматической проверки правильности написания слов qb

Команды просмотра

SUBs...	<F2>	вывод каталога модулей и страниц с возможностью выхода для редактирования в любую страницу любого модуля. Имеется возможность передвижения выбранной страницы из одного модуля в другой или ее удаления при выборе данного пункта меню появляется диалоговое окно
Next SUBs	<SHIFT>+<F2>	переход на следующую страницу в каталоге
Split		разделение окна редактирования на две части: верхнюю и нижнюю. Переход осуществляется нажатием клавиш <F6> / <SHIFT>+<F6>
Next Statement		после прерывания программы устанавливает маркер на следующий оператор за последним выполненным
Output Screen	<F4>	просмотр полученной

		информации. Повторение команды все возвращает в исходное состояние
Include File		при позиционировании маркера на строке с командой $INCLUDE '< имя файла >' происходит загрузка в память данного файла, как отдельного модуля и позиционирование маркера в его тексте. Если файл был загружен ранее, то происходит просто позиционирование маркера в его тексте.
Include Lines		включает режим показа содержимого включаемых файлов по месту положения команд $INCLUDE. В данном режиме редактирование текста невозможно
Options		выбор параметров экрана с возможностью их сохранения

Поиск, поиск и замена фрагментов текста

Find		режим поиска с заданием фрагмента текста при выборе данного пункта меню появляется диалоговое окно
Selected Text	<CTRL>+<\>	режим поиска заранее выбранного текста
Repeat Last Find	<F3>	режим повтора поиска с параметрами заданными в предыдущих командах
Change...		режим поиска и замены с параметрами заданными в предыдущих командах при выборе данного пункта меню появляется диалоговое окно
Label		поиск алфавитно-цифровой метки при выборе данного пункта меню появляется диалоговое окно
Run		команды выполнения программы

Start	<SHIFT>+<F5>	запуск программы из главного модуля
Restart		запуск программы из главного модуля на пошаговое исполнение. Движение по шагам осуществляется при нажатии на клавишу <F8>
Continue	<F5>	продолжение работы программы запущенной командой Start с места ее останова
Modify Comands..		устанавливает строку возвращаемую COMMAND$ функцией
Make EXE File...		создание выполняемого файла с именем главного модуля
Make Library...		преобразование главного модуля вместе с процедурами в файлы QUICK LIBRARY
Set Main Module		режим объявления любого из модулей расположенных в памяти главным
Debug		команды облегчающие отладку
Add Watch		внесение выражений и имен переменных в окно контроля при выборе данного пункта меню появляется диалоговое окно
Watchpoint		внесение выражений и имен переменных приостанавливающих выполнение программ при достижении условия TRUE или nonTRUE
Delete Watch		выборочное удаление контрольных точек при выборе данного пункта меню появляется диалоговое окно
Delete All Watch		удаление всех контрольных точек

Trase On		отслеживание операторов при запуске программы
Histore On		включение и отключение режима за поминания последних 20 выполненных строк
Toggle Breakpoint	<F9>	включение или отключение выделенной точки прерывания
Clear All Breakpoint		отключение всех выделенных точек прерывания
Set Next Statement		пропуск ряда операторов до указанного
Calls		команда вызова процедур
Help		команды помощи пользователю
General	<F1>	основная помощь при выборе данного пункта меню появляется диалоговое окно
Topic	<SHIFT>+<F1>	помощь для правильного написания оператора при выборе данного пункта меню появляется диалоговое окно
Close Help		выход из подменю помощи

Порядок записи арифметических операций

Порядок выполнения арифметических операций прежде всего определяется скобкамию. При их отсутствии операции выполняются согласно приоритету. При равнозначности приоритетов они выполняются слева направо.

Все операции должны быть указаны явно. Символы операций не могут следовать друг за другом. Этого правила целесообразно придерживаться, используя для отделения соседних операций скобки.

Тем не менее в языке Microsoft Basic допускаются следующие последовательности знаков операций: *-, *+, ^+, ^-, где (+) и (-) – операции присвоения знака числу.

Функциональные операции

ABS – абсолютное значение числа;

ATN – арктангенс числа;

CDBL – преобразование числа к удвоенной точности (8 байт);

CINT – преобразование вещественного числа или выражения в целое путем округления;

COS – косинус угла;

CSNG – преобазование числа к обычной точности (4 байт);

EXP – экспонента числа;

FIX – целая часть дробного числа;

INT – наибольшее целое число, меньшее или равное аргументу;

LOG – натуральный логарифм числа;

RND – случайное число между 0 и 1 с равномерным законом распределения;

SGN – знак числа;

SIN – синус угла;

SQR – квадратный корень числа;

TAN – тангенс угла.

Вычисление триганометрических функций

arcsin x: ARCSIN (x) = ATN (x/SQR(1-x*x));

arccos x: ARCCOS (x) = 1.570796 – ATN (x/SQR(1-x*x));

arcctg x: ARCCOT (x) = 1.57096 – ATN (x).

Оператор комментария - REM

Назначение: Включение в программу поясняющих записей-комментариев.

Синтаксис:

REM <комментарий>

‘ <комментарий>

Оператор присваивания - LET

Назначение: Присваивание, переменной значения некоторого числового или строкового выражения.

Синтаксис:

[LET] <переменная> = <выражение>

Оператор ввода данных с клавиатуры - INPUT

Назначение: Создает условия для ввода данных с клавиатуры в процессе выполнения программы.

Синтаксис:

INPUT[;]["<строка приглашения>" {; | ,}] <список переменных>

Аргумент	*Описание*
;	Точка с запятой после ключевого слова INPUT предписывает курсору оставаться на той же самой

	строке после нажатия клавиши ENTER
<строка приглашения>	Текстовая константа или текстовая переменная, заключённая в кавычки и выводимая на экран в качестве приглашения
;	Точка с запятой после строки приглашения выводит на экран вопросительный знак.
,	Запятая отменяет вывод вопросительного знака после строки приглашения
<список переменных>	Список разделенных запятыми переменных, которым присваиваются вводимые значения

В ответ на приглашение пользователь вводит данные в соответствии со списком переменных.

При несоответствии числа или типа вводимых данных числу и типу переменных списка выдается следующее сообщение об ошибке: Redo from start - повторить сначала. Присваивание входных значений переменным не производится до тех пор, пока не будут введены все данные в соответствии со списком переменных. До нажатия клавиши ENTER допускается внесение исправлений в набираемую строку ввода.

Редактирующие комбинации клавиш, предназначенные для перемещения курсора, удаления и вставки символов текста во входную строку, описаны в табл. 9.1.

Примечание: знак "+" указывает на одновременное нажатие двух клавиш.

Таблица 9.1.

Редактирующие комбинации клавиш

Клавиши	Действия
CTRL+\ или RIGHT	Перемещение курсора на один символ вправо
CTRL+] или LEFT	Перемещение курсора на один символ влево
CTRL+F или CTRL+RIGHT	Перемещение курсора на одно слово вправо
CTRL+B или CTRL+LEFT	Перемещение курсора на одно слово влево
CTRL+K или HOME	Перемещение курсора в начало вводимой строки
CTRL+N или END	Перемещение курсора в конец вводимой строки

CTRL+R или INS	Переключение режимов вставки и замены. В режиме вставки по мере ввода новых символов символы над курсором и справа от него сдвигаются вправо; в режиме замены просто заменяются
CTRL+I или TAB	Перемещает курсор к ближайшей позиции табуляции. В режиме вставки символы над курсором и справа от него сдвигаются вправо
DEL	Удаление символа над курсором CTRL+H или BACKSPACE Удаление символа слева от курсора. При достижении курсором начала строки удаляются символы над курсором
CTRL+E или CTRL+END	Удаление символов от курсора до конца строки
CTRL+U или ESC	Удаление всей строки независимо от положения курсора;
CTRL+M или RETURN	Запись входной строки в память
CTRL+T	Переключение режима отображения наименования функциональной клавиши в нижней части экрана
CTRL+BREAK или CTRL+C	Отказ от ввода данных и принудительное завершение программы

Оператор вывода данных на терминал - PRINT

Назначение: Вывод данных на экран.

Синтаксис:

PRINT [<список выражений>] [{ , | ;}]

Если аргумент <список выражений> опущен, то на экран выводится пустая строка. При наличии <списка выражений> значения выражений выводятся на экран. Выражения в списке могут быть числовыми или строковыми. Строковые константы должны быть заключены в кавычки. За выводимыми числами всегда следует пробел; положительным числам всегда предшествует пробел, а отрицательным - знак минус.

Расположение каждого выводимого на экран элемента определяется знаками препинания, используемыми для разделения элементов списка. BASIC разделяет строку на зоны по 14 символов. Запятая в списке выражений

определяет вывод каждого очередного значения с начала следующей зоны. Переменные, разделенные точкой с запятой, печатаются непосредственно друг за другом. Один или несколько пробелов или символов табуляции между выражениями действуют аналогично точке с запятой.

2. Описание практической части работы*:*

2.1. *Цель работы:* Изучить основные команды и функции, выполняемые в среде qbx.exe (qb.exe), последовательность составления программ с линейной структурой и запуска программ в среде.

2.2 *Постановка задачи:* В среде программирования QB набрать несколько строк на русском языке, используя операторы REM (номер группы, фамилию разработчика, номер и название лабораторной работы); в соответствии с вариантом осуществить ввод переменной, рассчитать значение функции и вывести его на экран монитора; осуществить запись программы в файл (имя файла должно содержать не более 8 символов на английском языке).

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы:

- порядок запуска среды программирования;
- последовательность набора текста программы;
- последовательность записи файла на диск.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования: REM, INPUT, LET, PRINT, STOP, END (и других при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер;
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Укажите последовательности действий при выполнении арифметических выражений в Basic Microsoft?
2. Как обозначается функция вычисления квадратного корня в языке Basic Microsoft? (SQR)
3. Какие скобки применяются при записи арифметических выражений в Basic Microsoft?
4. Как записывается в языке Basic Microsoft e^x?
5. Как можно вычислить arcsin (x) в Basic Microsoft?
6. Какое действие имеет более высокий приоритет при выполнении операций: возведение в степень (^) или умножение (*)?
7. Какое действие имеет более высокий приоритет при выполнении операций: сложение(+) или деление (/)?
8. Какое действие имеет более высокий приоритет при выполнении операций: деление(/) или целочисленное деление(\)?
9. Какое значение возвращает функция MOD, укажите результат: A%=10 MOD3
10. Можно ли в арифметическом выражении использовать функцию RND (x)?

Таблица

Варианты заданий

n варианта	Функция y(x)	n варианта	Функция y(x)
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	16	$y=\ln\frac{\sqrt[3]{\lvert A^2+ctg^2 x\rvert}}{A+\ln^2 x};A=6x^x$
2	$y=\left(\frac{\lvert x^x-\sqrt{x}\rvert}{1+\sin^3 x}\right)$	17	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^2 x}{x^2+1}$	18	$y=0.5\ln(x+2)\sqrt[3]{\lvert 4-x^2\rvert}$

4	$y=\ln\dfrac{\lvert 2+tg^2x\rvert}{x^{\sin x}}$	19	$y=0.6e^{x^2}-\dfrac{x-\cos^2 x}{\ln x}$
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	20	$y=2.8-\dfrac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$
6	$y=\dfrac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	21	$y=5x^x\ln^2\left(3+e^x\right)$
7	$y=\dfrac{\sqrt{x\sin^2 x}-1}{x+a^x};a=\sqrt{\pi}$	22	$y=\dfrac{\sqrt[3]{x}\ln 2x}{\sqrt{\lvert A^2-A-1\rvert}};A=e^{\sin x}$
8	$y=\dfrac{x^2-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^3$	23	$y=\dfrac{e^{2x-2}-\sqrt[3]{\lvert x-10\rvert}}{\ln^2 x}$
9	$y=\dfrac{\sqrt{\lvert\sin^2 x+ctgx\rvert}}{\ln x}$	24	$y=\dfrac{\lvert 3-tg^2x\rvert}{B\ln x-B^2+1};B=e^{2(x+1)}$
10	$y=\dfrac{A+2ctg^2x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	25	$y=\dfrac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$
11	$y=\dfrac{B^2+\sin^2(0.1x)}{B^3+2\sqrt{B}-1};B=10.5^{\ln x}$	26	$y=\left(\dfrac{x^x+\cos^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$
12	$y=\dfrac{\lvert 1-2tg^{2.2}x\rvert}{e^{x^2}-\ln x}$	27	$y=\dfrac{\sqrt{\lvert\cos 1.5x+tgx\rvert}}{\ln x}$
13	$y=\dfrac{\sqrt[3]{\lvert A^2+\ln x\rvert}}{x};A=e^{2x}$	28	$y=\dfrac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$
14	$y=\lvert A^3-10A^2+6\rvert;A=e^{x^2tgx}$	29	$y=\ln(x+2)\sqrt[3]{\lvert 2-x^2\rvert}$
15	$y=\sqrt[4]{\ln^2 x+C^2+C};C=2.2^{\ln x}$	30	$y=\dfrac{tgx-a^3}{a\lvert 1-x\rvert};a=\sqrt{\pi}$

Лабораторная работа №6
Организация программ с разветвляющейся структурой.

1. Описание операторов, применяющихся при организации программ с разветвляющейся структурой.

Оператор условного перехода - IF...THEN...ELSE

Назначение: Позволяет передавать управление программой в зависимости от результата проверки условия. Имеет две разновидности.

Синтаксис:

Первая форма представляет собой оператор, записываемый в одну строку:

IF <условие> THEN <выражение 1> [ELSE <выражение 2>]

Аргумент	**Описание**
<условие>	Логическое выражение, принимающее значение TRUE (истина) <- ненулевое значение, - или FALSE (ложь) - нулевое значение;
<выражение 1>	Вычисляется, если условие принимает значение TRUE, и пропускается в противном случае;
<выражение 2>	Вычисляется, если условие принимает значение FALSE.

Если ELSE-часть отсутствует, а <условие> имеет значение FALSE, управление передается следующему оператору.

Вторая (блоковая) форма оператора условного перехода имеет вид:

IF <условие 1> THEN
[<блок 1>]
[ELSEIF <условие 2> THEN
[<блок 2>]]
. . . . . .
[ELSE
[<блок N>]]
END IF

Аргумент	*Описание*
<условие 1>	Логическое выражение, принимающее значение TRUE (ненулевое) или FALSE (нулевое) (i = 1...N).
<блок 1>, …<блок N>	Последовательность операторов, занимающих одну или несколько строк.
ELSEIF	Проверка дополнительного условия (<условие 2>).
ELSE	Определяет блок операторов, получающих управление в случае, если ни одно из вышеперечисленных условий не

	выполнится.

Оператор выбора - SELECT CASE

Назначение: Выбор и выполнение одного из нескольких блоков в зависимости от значения ключевого слова.

Синтаксис:

SELECT CASE <ключевое слово>
CASE [<ключ 1>]
[операторный блок 1>]
[CASE [<ключ 2>]
[<операторный блок 2>]]
[CASE [<ключ N-1>]
[операторный блок N-1>]]
[CASE ELSE
[<операторный блок N>]]
END SELECT

Аргумент	*Описание*
<ключевое слово>	**Любое числовое или строковое выражение**
<Операторный блок 1>, …<операторный блок N>	Операторный блок - это любое число операторов на одной или более строках

Синтаксис аргумента <ключ 1>:

В качестве ключа (I = 1...N) используется одна из следующих форм:

1. <выражение> [, <выражение>...]
2. <выражение> TO <выражение>...
3. IS <операция отношения> <выражение> [, ...]

<выражение>	Любое числовое или строковое выражение. Тип выражения должен соответствовать типу текущего ключевого слова
<операция отношения>	< меньше <= меньше или равно > больше >= больше или равно <> не равно = равно

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить последовательность написания программ с разветвляющейся структурой на алгоритмическом языке (Basic) и структуру линейного и блочного оператора IF.

2.2. *Постановка задачи:* В соответствии с заданием лабораторной работы №2 разработать программы с линейной и разветвляющейся структурой для нахождения заданной функции.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования: IF (строчная и блочная формы) (при необходимости SELECT CASE).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы без комментариев (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Каким блоком изображается в алгоритме <условие>?

2. В чем отличие разветвляющихся структур с полной и неполной альтернативой выбора?

3. Назначение конструкции выбора, реализуемой оператором SELECT CASE?

4. Укажите синтаксис линейного оператора IF?

5. Укажите синтаксис блочного оператора IF?

6. Можно ли выйти из блочного оператора IF (из одного из выполняемых блоков) с помощью оператора GOTO?

7. Укажите виды условий, используемых в операторе IF?

8. Какое действие выполняется в операторе IF: IF A NOT B THEN PRINT ‘A’ ELSE PRINT ‘B’, если A=B?

9. Можно ли использовать оператор IF для возврата на начало программы?

10. Как изображается конструкция выбора в схемах программы?

Таблица

Задания для написания программы с разветвляющимся алгоритмом

n	задание	n	задание
1	$y=\begin{cases}(x+1)^4+1, & x<-1\\ 1, & -1\le x<1\\ 1-(x-1)^4, & 1\le x\end{cases}$	2	$y=\begin{cases}11-7\sqrt{\lvert x-1\rvert}, & x<-3\\ x, & -3\le x<3\\ 7\sqrt{x+1}+11, & x\ge 3\end{cases}$
3	$y=\begin{cases}e^{x-1}, & x\le 1\\ lg(x+2), & x>1\end{cases}$	4	$y=\begin{cases}\ln x, & x\ge 3\\ e^x, & 2\le x<3\\ x, & x<2\end{cases}$
5	$y=\begin{cases}\sin^2(x+0.1), & x<0.2\\ 1, & 0.2\le x\le 0.3\\ \cos^2(x+0.1), & x>0.3\end{cases}$	6	$y=\begin{cases}(x+1)^4, & x<-1\\ 1-\cos(\pi x), & -1\le x<1\\ -(x-1)^2, & x\ge 1\end{cases}$
7	$y=\begin{cases}1, & x<0\\ 2+\cos(\pi x), & 0\le x<0.4\\ 1, & x\ge 0.4\end{cases}$	8	$y=\begin{cases}1+\cos x, & x<0.2\\ 1, & 0.2\le x<0.4\\ -1, & x\ge 0.4\end{cases}$
9	$y=\begin{cases}0, & x\le 1\\ x\ln x, & 1<x<3\\ x^x, & x\ge 3\end{cases}$	10	$y=\begin{cases}4x^2+2x-8, & x<3\\ 1, & 3\le x<3.5\\ x^3-x+10, & x\ge 3.5\end{cases}$

11	$y=\begin{cases}1-2x^2, & x\le 0\\ 1, & 0<x<0.5\\ 1+(x-0.5)^4, & x\ge 0.5\end{cases}$	12	$y=\begin{cases}0, & x\le 1\\ 1+\cos \pi x, & -1<x<0\\ 1.5, & x\ge 0\end{cases}$
13	$y=\begin{cases}\cos 0.5(x+1), & x\le -1\\ 0, & -1<x<1\\ \sin 0.5\pi x, & x\ge 1\end{cases}$	14	$y=\begin{cases}\cos 0.5(x+1), & x\le -1\\ 0, & -1<x<1\\ \sin 0.5\pi x, & x\ge 1\end{cases}$
15	$y=\begin{cases}x+0.6, & x\le -1\\ x^3, & -1<x<1\\ x-0.6, & x\ge 1\end{cases}$	16	$y=\begin{cases}\sin^3(0.5+x), & x<0.5\\ 1, & 0.5\le x\le 1\\ x^2-1, & x>1\end{cases}$
17	$y=\begin{cases}\sin^2(x+0.1), & x<0.2\\ 1, & 0.2\le x\le 0.3\\ \cos^2(x+0.1), & x>0.3\end{cases}$	18	$y=\begin{cases}4x^2+2x-8, & x\le 1\\ 1, & -1<x\le 1\\ x^3-x+10, & x>1\end{cases}$
19	$y=\begin{cases}1, & x<2\\ 2-(x-2)^2, & 2\le x<3\\ (x-4)^2, & x\ge 3\end{cases}$	20	$y=\begin{cases}x+0.8, & x\le -1\\ x^3, & -1<x\le 1\\ 2\sqrt{x}-1, & x>1\end{cases}$
21	$y=\begin{cases}\lvert x-1\rvert, & x<1\\ x^2, & x\ge 1\end{cases}$	22	$y=\begin{cases}e^{x-1}, & x\le 1\\ \lg(x+2), & x>1\end{cases}$
23	$y=\begin{cases}e^{x+1}, & x\le 2\\ 5, & x>2\end{cases}$	24	$y=\begin{cases}\sin^2 x, & x\le 0.5\pi\\ 1, & 0.5\pi<x<\pi\\ 0, & x\ge \pi\end{cases}$
25	$y=\begin{cases}\lvert x-1\rvert, & x<1\\ x+e^{2x}, & x\ge 1\end{cases}$	26	$y=\begin{cases}\sin 0.5(x+1), & x\ge 0\\ 0, & x\le -2\\ \sin \pi x, & x\ge 1\end{cases}$
27	$y=\begin{cases}3\sqrt{\lvert x+1\rvert}+5, & x<3\\ x, & 3\le x<1\\ 3.5+3\sqrt{x+1}, & x\ge 1\end{cases}$	28	$y=\begin{cases}(x+1)^4, & x<-1\\ 1-\cos(\pi x), & -1\le x<1\\ -(x-1)^2, & x\ge 1\end{cases}$
29	$y=\begin{cases}4x^2-2x+8, & x<3\\ 1, & 3\le x<3.5\\ x^3+x+10, & x\ge 3.5\end{cases}$	30	$y=\begin{cases}x+0.8, & x\le -1\\ x^3, & -1<x\le 1\\ 2\sqrt{x}-1, & x>1\end{cases}$

Лабораторная работа №7
Организация программ с циклической структурой.

1. Описание операторов, используемых для организации программ с циклической структурой

Оператор цикла FOR … NEXT

Назначение: Применяется для циклического выполнения определенной группы операторов заданное число раз.

Синтаксис: FOR <счетчик> = <начальное значение> TO
<конечное значение> [STEP <приращение>]
NEXT [<счетчик 1>[, <счетчик 2>...]

Аргумент	*Описание*
<счетчик>	Внутренняя числовая переменная счетчика циклов. Переменная не может быть элементом записи или элементом массива
<начальное значение>	Начальное значение счетчика циклов
<конечное значение>	Конечное значение счетчика циклов
<приращение>	Приращение счетчика циклов; может иметь отрицательное значение

Оператор цикла - DO...LOOP

Назначение: Оператор обеспечивает циклическое выполнение группы операторов, пока <условие> в конструкции WHILE истинно (TRUE) или до тех пор, пока <условие> в конструкции UNTIL не станет истинным. Имеет две разновидности, в зависимости от того, проверяются ли условия в начале или конце цикла.

Синтаксис:

1. DO
[<блок операторов>]
LOOP [[WHILE | UNTIL) <условие>]
2. DO [(WHILE | UNTIL) <условие>]
[<блок операторов>]
LOOP

Аргумент	*Описание*
<блок операторов>	Один или несколько операторов языка BASIC, которые будут циклически выполняться ;
<условие>	Любое выражение, принимающее значение либо TRUE (не нуль), либо FALSE (нуль).

Оператор цикла – WHILE…WEND

Назначение: Выполнение совокупности операторов пока условие истинно.

Синтаксис:

WHILE <условие>

[<операторы>]

WEND

Если условие истинно, то выполняются все операторы до ключевого слова WEND. Затем происходит возврат на проверку условия. Если оно по-прежнему истинно, то процесс продолжается. Если ложно, то управление передается следующему за WEND оператору.

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить построение циклических программ с использованием оператора FOR и DO.

2.2. *Постановка задачи:* В соответствии со схемами программ лабораторной работы №3 рассчитать значения функции с равномерно изменяющимся аргументом.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования: FOR, DO (при необходимости WHILE…WEND).

4.3. *Описание программы* содержит:

- название файла, его размер,

- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Можно ли выйти из цикла FOR до его завершения и если можно то как?
2. В каких случаях следует использовать цикл FOR, а каких DO... LOOP?
3. Чем отличаются логические циклы «Do» и «Пока»?
4. Можно ли выйти из цикла DO до его полного выполнения по дополнительному условию оператором GOTO<метка>?
5. Можно ли закончить цикл FOR до завершения цикла DO, если цикл DO находится внутри цикла FOR?
6. Сколько конструкций имеет цикл DO в языке Basic Microsoft?
7. Назначение логического цикла WHILE.... WEND?
8. Какова максимальная глубина вложения циклов в BASIC?
9. Как изображаются логические циклы в схемах программ?
10. Как изображаются арифметические циклы в схемах программ?

Таблица

Задания для написания программы с циклическим алгоритмом

n	Функция y(x)	X_{min}	X_{max}	ΔX
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5	0.02
2	$y=\left(\frac{\lvert x^x-\sqrt{x}\rvert}{1+\sin^3 x}\right)$	0.4	0.8	0.04
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^2 x}{x^2+1}$	0.1	0.6	0.05
4	$y=\ln\frac{\lvert 2+tg^2 x\rvert}{x^{\sin x}}$	0.3	0.7	0.05
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	0.3	0.7	0.05
6	$y=\frac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.4	0.8	0.04
7	$y=\frac{\sqrt{x\sin^2 x}-1}{x+a^x};a=\sqrt{\pi}$	0.25	0.2	0.02

8	$y=\frac{x^2-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6	0.05
9	$y=\frac{\sqrt{\lvert\sin^2 x+ctgx\rvert}}{\ln x}$	0.3	0.7	0.05
10	$y=\frac{A+2ctg^2 x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	0.1	0.6	0.05
11	$y=\frac{B^2+\sin^2(0.1x)}{B^3+2\sqrt{B}-1};B=10.5^{\ln x}$	0.1	0.6	0.05
12	$y=\frac{\lvert 1-2tg^{2.2}x\rvert}{e^{x^2}-\ln x}$	0.25	0.2	0.02
13	$y=\frac{\sqrt[3]{\lvert A^2+\ln x\rvert}}{x};A=e^{2x}$	0.3	0.7	0.05
14	$y=\lvert A^3-10A^2+6\rvert;A=e^{x^2tgx}$	0.25	0.2	0.02
15	$y=\sqrt[4]{\ln^2 x+C^2+C};C=2.2^{\ln x}$	0.4	0.8	0.04
16	$y=\ln\frac{\sqrt[3]{\lvert A^2+ctg^2x\rvert}}{A+\ln^2 x};A=6x^x$	0.1	0.6	0.05
17	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$	0.3	0.7	0.05
18	$y=0.5\ln(x+2)\sqrt[3]{\lvert 4-x^2\rvert}$	0.4	0.8	0.04
19	$y=0.6e^{x^2}-\frac{x-\cos^2 x}{\ln x}$	0.1	0.5	0.02
20	$y=2.8-\frac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$	0.25	0.2	0.02
21	$y=5x^x\ln^2(3+e^x)$	0.3	0.7	0.05
22	$y=\frac{\sqrt[3]{x}\ln 2x}{\sqrt{\lvert A^2-A-1\rvert}};A=e^{\sin x}$	0.4	0.8	0.04
23	$y=\frac{e^{2x-2}-\sqrt[3]{\lvert x-10\rvert}}{\ln^2 x}$	0.25	0.2	0.02
24	$y=\frac{\lvert 3-tg^2x\rvert}{B\ln x-B^2+1};B=e^{2(x+1)}$	0.4	0.8	0.04
25	$y=\frac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$	0.1	0.6	0.05
26	$y=\left(\frac{x^x+\cos^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5	0.02

27	$y=\frac{\sqrt{\lvert\cos 1.5x+tgx\rvert}}{\ln x}$	0.25	0.2	0.02
28	$y=\frac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6	0.05
29	$y=\ln(x+2)\sqrt[3]{\lvert 2-x^2\rvert}$	0.4	0.8	0.04
30	$y=\frac{tgx-a^3}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.3	0.7	0.05

Лабораторная работа №8
Организация работы с массивами.

1. Описание операторов, используемых для организации работы с массивами

Оператор объявления переменных - DIM

Назначение: Объявление переменных и резервирование для них памяти.

Синтаксис: DIM [SHARED] <переменная> [(<индексы>) [AS <тип>] [, <переменная> [(<индексы>)] [AS <тип>]]…

Аргумент	*Описание*
SHARED	Атрибут, указывающий на совместное использование переменных всеми процедурами модуля
<переменная>	Имя простой или индексной переменной
<индекс>	Размерность переменной с индексом; максимальное число индексов – 8
<тип>	Указатель типа описываемой переменной; тип выбирается из следующего списка: INTEGER, LONG, SINGLE, DOUBLE, STRING или <имя структуры>

Запись индекса в общем виде :

[<нижняя граница> TO] <верхняя граница> [, [<нижняя граница> TO] <верхняя граница>]…

Оператор изменения начала отсчета индексов -OPTION BASE

Назначение: Установка нижней границы индексов массивов.

Синтаксис: OPTION BASE <нижняя граница>

Нижняя граница индекса равна либо 0, либо 1. По умолчанию, устанавливается значение 0; для установки значения 1 необходимо выполнить оператор OPTION BASE 1.

Использование оператора OPTION BASE не является обязательным

Оператор инициализации массива - ERASE

Назначение: Инициализация элементов статических массивов; освобождение памяти под динамическими массивами.

Синтаксис: ERASE <имя массива>[, <имя массива>. .]

Аргументы <имя массива> являются именами массивов, которые необходимо переопределить. Оператор ERASE дает различный результат для массивов, описанных в метакомандах $STATIC и $DYNAMIC. Оператор ERASE устанавливает элементы статических массивов равными нулю в случае числовых массивов или заполняет их пустыми строками в случае строковых массивов. Если массив представляет собой массив записей, то оператор ERASE

устанавливает нулевые значения для элементов каждой записи, включая строковые элементы.

Применение оператора ERASE к динамическим массивам освобождает занимаемую ими память.

Оператор переопределения параметров массивов - REDIM

Назначение: Переопределение параметров массивов, объявленных метакомандой $DYNAMIC.

Синтаксис: REDIM [SHARED] <переменная> (<список индексов>) [AS <тип>] [, <переменная> (<список индексов>) [AS <тип>]]...

Аргумент	Описание
SHARED	Необязательный атрибут SHARED обеспечивает доступность описанных переменных во всех процедурах модуля, может использоваться в операторе REDIM только в программе на уровне модуля
<переменная>	Имя переменной языка Microsoft BASIC
<список индексов>	Перечень индексов с указанием их границ: определяет размерность массива. Правила записи индексов описаны ниже
AS <тип>	Объявляет тип элементов массива: INTEGER, LONG, SINGLE, DOUBLE, STRING, или тип, определяемый пользователем (структура)

Аргумент <список индексов> в операторах REDIM имеет следующий формат:[<нижняя граница> TO] <верхняя граница> [, [<нижняя граница> TO] <верхняя граница>]...

Наличие ключевого слова TO указывает на то, что устанавливаются пределы, как на верхнюю границу индекса массива, так и на нижнюю. Аргументы <нижняя граница> и <верхняя граница> являются числовыми выражениями.

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Закрепить навыки по разработке циклических и разветвляющихся структур алгоритмов на примере обработки массивов.

2.2. *Постановка задачи:* В соответствии с вариантом задания произвести обработку одномерного массива, состоящего из более десяти элементов.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования: DIM, OPTION BASE (при необходимости ERASE, REDIM).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы – отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Допускается ли совпадение имени простой переменной с именем массива?

2. Сколько элементов в массиве, если он объявлен в операторе DIM a (10)?

3. Какой массив занимает больше места в памяти ЭВМ: a или в, если эти массивы заданы следующим образом: DIM a%(15), DIM в #(10).

4. Могут ли в одном массиве находиться числовые и символьные данные?

5. Сколько элементов массива резервирует среда программирования qb.exe, если массив не объявлен оператором DIM?

6. Сколько циклов надо выполнить, чтобы найти минимальное (или максимальное) значение одномерного числового массива A()?

7. Сколько циклов надо выполнить, чтобы рассортировать одномерный массив A()?

8. Можно ли использовать оператор REDIM до использования оператора DIM?

9. Можно ли присваивать элементам числового массива символьные значения?

10. Можно ли после работы программы освободить память занимаемую массивом?

11. Чем отличаются динамические и статические массивы в языке Basic Microsoft?

12. Можно ли при обозначении элементов массива использовать отрицательные числа?

Таблица

Задания

n	задание	n	задание
1	Найти минимальное значение в одномерном массиве A()	16	Найти номера элементов массива A(), значения которых совпадают с заданным
2	Найти максимальное значение в одномерном массиве A()	17	Найти номера элементов массива A(), значения которых меньше заданного.
3	Найти минимальное значение в двухмерном числовом массиве A()	18	Найти номера элементов массива A(), значения которых больше заданного.
4	Найти максимальное значение в двухмерном числовом массиве A()	19	В массиве A() найти все числа меньше заданного значения
5	Сформировать массив B(), состоящий из положительных элементов массива A()	20	В массиве A() найти все числа больше заданного значения
6	Сформировать массив B(), состоящий из отрицательных элементов массива A()	21	В массиве A() найти все числа, находящиеся в диапазоне от a до в
7	Сформировать массив B(), состоящий из четных чисел массива A()	22	В массиве A() найти количество элементов, сумма которых меньше заданного

8	Сформировать массив B(), состоящий из нечетных чисел массива A()	23	В массиве A() найти количество элементов, сумма которых больше заданного
9	Найти сумму положительных элементов массива A()	24	В массиве A() найти все суммы соседних пар элементов
10	Найти сумму отрицательных элементов массива A()	25	Сформировать массив A() из элементов, являющихся средним значением соседних пар чисел
11	Найти среднее значение всех элементов массива A()	26	Сортировать одномерный числовой массив A() по убыванию методом минимального элемента
12	Найти номер минимального элемента в массиве A()	27	Определить количество одинаковых элементов в целочисленном массиве A()
13	Найти номер максимального элемента в массиве A()	28	Определить сколько раз в массиве A() встречается заданное число
14	Найти количество минимальных элементов в массиве A()	29	Найти номера элементов массива A(), значения которых равны заданному числу.
15	Найти количество макси-мальных элементов в массиве A()	30	Найти сколько элементов массива A() встречаются в массиве B().

Лабораторная работа №9
Работа с различными типами данных.

1. Типы данных, используемые в языке QuickBASIC

Программа, написанная на любом алгоритмическом языке, представляет собой последовательность операций выполняемых над некоторыми данными. По сути дела, один алгоритмический язык отличается от другого множеством допустимых данных и наборов операций над ними.

Основными данными языка QuickBASIC являются числовые и символьные данные, присутствующие в программах в виде констант и переменных.

Константы - это предварительно определенные величины, которые не изменяются в процессе выполнения программы.

Переменные - это элемент языка программирования, имеющий имя и тип.

Константы используемые в Бейсике можно разделить на две группы:

- литерные константы, представляющие собой последовательность знаков (литер) и выражаемые в виде чисел и строк;

- именованные константы, переменные особого рода, значения которых не могут быть изменены в программе.

Числовые константы могут быть целыми или вещественными с фиксированной и плавающей точкой со знаком, причем указание знака "+" не обязательно.

Типы числовых констант и их представление сведены в таблицу 1

Таблица 1.

Тип констант характеристика	Десятичная	Шестнадцатиричная	Восьмиричная
Целый			
Литеры	0-9	0-9, A-F (a-f)	0-7
Число байт	2	2	2
Диапазон	от -32768 до 32767	от &H0 до &HFFFF	от &O0 до &O177777
Маркер типа	%	%	%
Целый удвоенной точности			
Литеры	0-9	0-9, A-F (a-f)	0-7
Число байт	4	4	4
Диапазон	от -2147483648 до	от &H0& до &HFFFFFFFF&	от &O0 до &O37777777777&

	2147483647		
Маркер типа	&	&	&
Вещественный с фиксированной точкой обычной точности			
Литеры	0 - 9 , (.)		
Число байт	4		
Диапазон	от до		
Маркер типа	!		
Вещественный с фиксированной точкой двойной точности			
Литеры	0 - 9 , (.)		
Число байт	8		
Диапазон	от до		
Маркер типа	#		
Вещественный с плавающей точкой обычной точности			
Литеры	0 - 9 , (.) , E		
Число байт	4		
Диапазон	от -3.37E+38 до 3.37E+38		
Маркер типа	!		
Вещественный с плавающей точкой двойной точности			
Литеры	0 - 9 , (.) , D		
Число байт	8		
Диапазон	от -1.67D+308 до 1.67D+308		
Маркер типа	#		

Строковые константы - это последовательность не более 32767 литер кода ASCII (за исключением символа (") и кодов управления, заключенных в кавычки .

Оператор объявления констант - CONST

Назначение: Объявление символьных констант, используемых вместо численных или символьных значений.

Синтаксис: CONST <имя константы> = <выражение 1>

[, <имя константы> = <выражение 2>]...

Аргумент	*Описание*
<имя константы>	Определяется по правилам формирования имен переменных (до 40 символов). Можно добавлять к имени маркеры (%,&,!,#,$), задающие тип и не являющиеся частью имени.
<выражение 1>	Выражение может включать символы, другие константы или любые операции, за исключением операции возведения в степень 0. В составе выражения нельзя использовать конкатенацию строк, определяемые пользователем переменные и функции, а также встроенные функции (1=1, 2,...).

Оператор задания списка констант - DATA

Назначение: Содержит числовые и строковые данные для оператора READ.

Синтаксис: DATA <константа 1> [, <константа 2>]...,

где <константа i> - числовая или строковая константа (i=1, 2, ...).

Оператор объявления типа переменных - DEF

Назначение: Устанавливает тип данных для переменных и для функций, определяемых операторами DEF FN и FUNCTION**.**

Синтаксис:

DEFINT <интервал букв> [, <интервал букв>]...

DEFSNG <интервал букв> [, <интервал букв>]…

DEFDBL <интервал букв> [, <интервал букв>]...

DEFLNG <интервал букв> [, <интервал букв>]...

DEFSTR <интервал букв> [, <интервал букв>]...,

где <интервал букв> = <буква 1>- <буква 2>, <буква 3>, <буква 4>.

Оператор объявления общих переменных – COMMON

Назначение: Определяет глобальные переменные, совместно используемые в различных модулях или программах.

Синтаксис: COMMON [SHARED] [/<имя блока>/] <список переменных>

Аргумент	*Описание*
SHARED	Атрибут, указывающий на совместное использование переменных всеми процедурами модуля
<имя блока>	Имя (до 40 символов), объединяющее группу переменных по определенному признаку. Такие группы

	часто называют COMMON -блоками
<список переменных>	Список переменных, совместно используемых в модулях или связанных программах.

Общий вид списка переменных:

<имя переменной>[( )] [AS <тип>]

[, <имя переменной>[( )] [AS <тип>]…

<имя переменной> - Обычное имя переменной или массива в языке BASIC

<тип> - Один из следующих типов: INTEGER, LONG, SINGLE, DOUBLE, STRING или <имя структуры>

Оператор объявления локальных переменных - STATIC

Назначение: Обеспечивает локализацию простых переменных и массивов внутри функций, процедур-функций и процедур (DEF FN, FUNCTION или SUB) и сохранение **их** значений между вызовами процедур.

Синтаксис: STATIC <список переменных>

где **<**список переменных>:

<имя переменной> [( )] [AS <тип>]

[, <имя переменной> [( )] [AS <тип>]]...

Аргумент	*Описание*
<имя переменной>	Имя переменной или массива в языке BASIC
<тип>	Один из следующих типов: INTEGER, LONG, SINGLE, DOUBLE, STRING или <имя структуры>

Оператор объявления глобальных переменных - SHARED

Назначение: Обеспечивает процедурам SUB и FUNCTION доступ к переменным головного модуля без передачи **их в** качестве параметров процедуры.

Синтаксис: SHARED <имя переменной>[( )] [AS <тип>]

[, <имя переменной> [( )] [AS <тип>]...

Аргумент	*Описание*
<имя переменной>	Имя переменной или массива; за именем массива должны следовать "()"
<тип>	Один из следующих типов: INTEGER, LONG, SINGLE, DOUBLE, STRING или <имя структуры>

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить способы задания типов данных в Quick Basic Qbx.exe и их использование при организации вычислений.

2.2. *Постановка задачи*

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер;
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы занести в таблицу:

n	целое		действительное	
	одинарной точности	двойной точности	одинарной точности	двойной точности
…	…	…	…	…

- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Укажите какое логическое значение примет условие A# =B% TRYE или FALSE, если A# =2, B% =2?
2. Какое число занимает больше места в памяти ЭВМ: целое одинарное точности или действительное одинарной точности?
3. Какое число занимает больше места в памяти ЭВМ: целое двойной точности или действительное одинарной точности?
4. Какое число занимает больше места в памяти ЭВМ: целое двойной точности или действительное удвоенной точности?
5. Можно ли складывать числа разных типов и какой результат при этом получится?
6. Назовите способы задания типов данных, используемых в Basic Microsoft?
7. Какого типа будет число D, если в программе указано: DEFINT A-F?
8. Какого типа будет число i, если в программе не использован оператор DEF?
9. Можно ли в Basic Microsoft использовать смешанный тип данных (состоящий из нескольких типов),
10. Назначение оператора TYPE?

Таблица

Задания

n	задание	n	задание
1	Вычислить максимальное значение (n-c)!, c=const	16	Вычислить максимальное значение sin n* n^c, c=7
2	Вычислить максимальное значение c^n , c=const	17	Вычислить максимальное значение суммы элементов ряда 1.5^{i+2} , где i изменяется от 1 до n, для заданного значения n.
3	Вычислить максимальное значение tg x	18	Вычислить максимальное значение
4	Вычислить максимальное значение cos n* c^n, c=const	19	Вычислить максимальное значение суммы элементов ряда 2^i , где i изменяется от 1 до n, для заданного значения n.
5	Вычислить максимальное	20	Вычислить максимальное

	значение суммы элементов ряда $i+3^{0.5i}$, где i изменяется от 1 до n, для заданного значения n.		значение n!
6	Вычислить максимальное значение суммы элементов ряда $i+3^{2i}$, где i изменяется от 1 до n, для заданного значения n.	21	Вычислить максимальное значение суммы элементов ряда $1.5i^{i+1}$, где i изменяется от 1 до n, для заданного значения n.
7	Вычислить максимальное значение суммы элементов ряда $\frac{2.5^{i-1}}{3^i}$, где i изменяется от 1 до n, для заданного значения n.	22	Вычислить максимальное значение суммы элементов ряда $\frac{2^i+0.5^{i+1}}{2^i}$, где i изменяется от 1 до n, для заданного значения n.
8	Вычислить максимальное значение $\frac{n!}{c^n}$, c=const	23	Вычислить максимальное значение суммы элементов ряда $\frac{3^{0.5i}}{2^i}$, где i изменяется от 1 до n, для заданного значения n.
9	Вычислить максимальное значение суммы элементов ряда $\frac{2^i+0.5}{3^{0.5i}}$, где i изменяется от 1 до n, для заданного значения n.	24	Вычислить максимальное значение суммы элементов ряда $\frac{2^{i-1}}{0.5^{i+1}}$, где i изменяется от 1 до n, для заданного значения n.
10	Вычислить максимальное значение суммы элементов ряда 2^{i+1}, где i изменяется от 1 до n, для заданного значения n.	25	Вычислить максимальное значение суммы элементов ряда $\frac{0.5^{i+1}+2^{2i-1}}{3^i}$, где i изменяется от 1 до n, для заданного значения n.
11	Вычислить максимальное значение суммы элементов ряда $2^i+3^{0.5i}$, где i изменяется от 1 до n, для заданного значения n.	26	Вычислить максимальное значение $\frac{(n+c)!}{(n-c)!}$, c=1, 2, 3

12	Вычислить максимальное значение суммы элементов ряда $\frac{2^i}{3^{0.5i}}$, где i изменяется от 1 до n, для заданного значения n.	27	Вычислить максимальное значение суммы элементов ряда $\frac{i+1+2^{i-1}}{0.5^{i+1}}$, где i изменяется от 1 до n, для заданного значения n.
13	Вычислить максимальное значение суммы элементов ряда $\frac{0.5^{i+1.5}}{i^{0.5i}}$, где i изменяется от 1 до n, для заданного значения n.	28	Вычислить максимальное значение суммы элементов ряда $\frac{i+1+2^{2i+1}}{2^{i+1.5}}$, где i изменяется от 1 до n, для заданного значения n.
14	Вычислить максимальное значение суммы элементов ряда $\frac{0.5^{i-1}}{2^{0.5i+1}}$, где i изменяется от 1 до n, для заданного значения n.	29	Вычислить максимальное значение суммы элементов ряда $\frac{i^{i+1}+0.5^{i+1}}{3^{i-1}}$, где i изменяется от 1 до n, для заданного значения n.
15	Вычислить максимальное значение суммы элементов ряда $\frac{i+1}{3^{0.5i}}$, где i изменяется от 1 до n, для заданного значения n.	30	Вычислить максимальное значение суммы элементов ряда $\frac{0.5^{i-1}+2i+1}{i-2}$, где i изменяется от 1 до n, для заданного значения n.

Лабораторная работа №10
Обработка символьной информации.

1. Обработка текстовой информации.

Символьные константы, переменные и массивы:

Символьные константы, используемые в качестве литеральных констант в тексте исходной программы, представляют собой последовательность произвольных знаков алфавита, окаймляемую двойными кавычками.

Версия Quick BASIC допускает употребление именованных констант символьного типа :

```
CONST  RUS$ = "Россия"
```

Идентификаторы текстовых переменных распознаются либо по последнему символу имени ($), либо по первой букве, если эта буква была указана в списке оператора DEFSTR :

```
DEFSTR   D-F, Q
```

Quick BASIC позволяет вводить объекты символьного типа с помощью описателя AS STRING , который может встретиться в одном из декларативных операторов: COMMON, DECLARE, DEF, DIM, FUNCTION, SHARED, STATIC, SUB, TYPE,

REDIM.

Например : DIM fio(40) AS STRING, address AS STRING *40

```
TYPE  anketa
fam     AS STRING *20
name    AS STRING *10
otech   AS STRING *10
datar   AS STRING *10
END TYPE
```

В операторах DIM и REDIM имеется возможность объявления не только символьных массивов, но и скалярных переменных. Указание о длине символьного объекта после описателя AS STRING причисляет его к разряду статических.

```
DEFSTR   D-F, Q
DIM gruppa AS STRING *4
```

Максимальная длина текстовой переменной не может превышать 32767 символов.

Ввод строки символов

При помощи оператора LINE INPUT программа может выдать запрос на ввод символьного значения с клавиатуры или из последовательного файла данных.

LINE INPUT [;] ["текст";] <имя переменной>
LINE INPUT#n, <имя переменной>

<имя переменной> - имя единственной переменной, в которую водятся все набираемые символы до появления управляющего кода "возврат каретки". Среди них могут быть и пробелы, и запятые.

Оператор LINE INPUT# не выдает приглашение ко вводу в виде вопросительного знака, но все остальные атрибуты (точка с запятой вначале, текстовая подсказка) имеют тот же смысл, что и в операторе INPUT.

Ввод данных с клавиатуры по операторам INPUT и LINE INPUT сопровождается отображением поступающих значений на экране дисплея.

Операции, применяемые к символьным переменным

В символьном выражении операции производятся над символьными операндами, представляющими собой символьные константы, переменные, элементы символьных массивов, а также функций обработки
символьных данных и другие символьные выражения.

В символьных выражениях может использоваться операция конкатенации (сцепления), которая обозначается знаком плюс:

A$= "Quick"
B$="BASIC"
C$=A$+B$

В результате выполнения операции значение С равно "QuickBASIC"

Операция конкатенации сводится к приписыванию значения очередного "слагаемого" в хвост к предыдущей строке.

В качестве операндов символьного выражения наряду с текстовыми константами и переменными могут выступать стандартные или нестандартные функции символьного типа.

Функция MID$, используемая для выделения внутренней подстроки, может применяться и как оператор для замены внутренней подстроки новым значением. В этом случае в качестве первого аргумента может выступать только идентификатор символьной переменной :

A$ = "Turbo BASIC"
MID$ (A$,1,5) = "Quick"

Длина заменяемой подстроки при таком присвоении не меняется. Если заменяющее значение имеет меньшую длину, то недостающие символы справа

дополняются пробелами. В противном случае используются первые n символов замещающего выражения.

Кроме стандартных функций, перечисленных в качестве операндов символьных выражений могут выступать служебные переменные текстового типа, а также символьные функции пользователя.

Стандартные процедуры обработки текстовой информации:

Функция INSTR

Назначение: Осуществляет поиск первого вхождения одной строки в другую строку и возвращает позицию начала вхождения найденной подстроки.

Синтаксис: INSTR ([начало,] строка 1,строка2)

Необязательный аргумент *начало* устанавливает позицию начала поиска в диапазоне от 1 до 32767. По умолчанию начальная позиция равна 1. Если величина этого аргумента выйдет за допустимый диапазон, то появится сообщение «Illegal Function Call» (недопустимый вызов функции).

Аргумент *строка1* является строкой для поиска. Аргумент *строка2* является искомой подстрокой. Оба аргумента могут быть строковыми переменными, строковыми выражениями или литералами.

Функция INSTR возвращает значение 0, если: аргумент *начало* превышает длину строки1; *строка1* — нулевая (пустая) строка; *строка2* не найдена.

Если *строка2* - пустая строка, то функция возвращает значение начальной позиции для поиска.

Функция LEFT$

Назначение: Возвращает подстроку, содержащую указанное число символов в левой части заданной строки.

Синтаксис: LEFT$(x$,n)

Аргумент x$ — исходная строка для выделения подстроки. Аргумент n задает число символов искомой подстроки и должен находиться в диапазоне от 1 до 32767. Если n превышает длину строки, то возвращается исходная строка x$. Если n равно 0, то возвращается нулевая (пустая) строка.

Функция RIGHT$

Назначение: Возвращает заданное число крайних правых символов исходной строки.

Синтаксис: RIGHT$ (.строка, число)

Аргумент *число* задает число символов, выделяемых в правой части исходной строки, заданной аргументом *строка.*

Если число символов больше длины или равно длине исходной строки, то функция возвращает всю строку. Если число выделяемых символов равно 0, то возвращается пустая строка.

Функция LEN

Назначение: Возвращает длину строки в байтах.

Синтаксис: LEN(x$)

Функция LTRIM$

Назначение: Возвращает копию строки с удаленными лидирующими пробелами.

Синтаксис: LTRIM$(строка)

Аргумент *строка* — исходное строковое выражение. Аналогичную операцию с хвостовыми пробелами выполняет функция RTRIM$.

Функция MID$

Назначение: Возвращает подстроку заданной длины исходной строки, начиная с указанного символа.

Синтаксис: MID$(строка, начало[, длина])

Аргумент *строка* представляет собой исходное строковое выражение. Аргументы *начало* и *длина* устанавливают начало выделения и длину искомой подстроки соответственно. Эти аргументы должны иметь целый тип **и** находиться в диапазоне от 1 (начало) или от 0 (длина) до 32767.

Если аргумент *длина* опущен или с начала выделения подстроки до конца исходной строки осталось менее символов, чем это установлено аргументом *длина,* то возвращаются все символы исходной строки, начиная с указанного.

Если аргумент *начало* превышает длину исходной строки, то результатом вызова функции будет нулевая (пустая) строка.

При установке нулевой длины подстроки функция возвращает также нулевую подстроку.

В случае выхода числовых аргументов за пределы указанных диапазонов произойдет ошибка «Illegal function call» (недопустимый вызов функции).

Оператор MID$

Назначение: Заменяет символы одной строки символами другой строки.

Синтаксис: MID$(переменная, начало[,длина])=выражение

Параметр *переменная* - строковая переменная, которая должна быть изменена. Параметр *выражение* - строковое выражение, замещающее часть строковой переменной. Параметры *начало* - числовые выражения целого типа, указывающие начало замещения и число заменяемых символов в строковой переменной.

Результатом выполнения оператора является замещение указанного числа символов строковой переменной, начиная с заданной позиции, символами строкового выражения.

Если необязательный параметр *длина* опущен, то используются все символы строкового выражения. Замещение символов всегда происходит в пределах длины исходной строковой переменной.

Функция STR$

Назначение: Возвращает строковое представление указанного аргумента.

Синтаксис: STR$ (выражение)

Аргумент *выражение* является числовым выражением целого типа. Если аргумент положителен, то функция возвращает строку с лидирующим пробелом.

Функция VAL

Назначение: Возвращает числовое представление строки.

Синтаксис: VAL (строка)

Функция VAL ликвидирует лидирующие пробелы, символы табуляции и перевода строки в аргументе *строка.*

Функция VAL является обратной по назначению функции STR$. Если первый символ аргумента — не числовой, то функция VAL возвращает 0.

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Ознакомиться с основными функциями и операторами обработки символьной информации, используемыми в языке Quick Basic.

2.2. *Постановка задачи:* В соответствии с вариантом в заданной строке определить количество слов, все слова записать в одномерный массив.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.2. Выполнить работу.

2.3.3. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Что такое символьная информация ?
2. Какие символы при работе на ЭВМ Вы знаете ?
3. Что такое основная кодовая страница ?
4. Какие действия можно производить с символьными переменными?
5. Можно ли сложить две символьные переменные и что произойдет если слагаемые поменять местами ?
6. Как определить длину символьной переменной произвольной длины ?
7. Что является разделителем при вводе символьных переменных с клавиатуры ?
8. Можно ли определить встречается ли в переменной искомый набор символов ?
9. Может ли символьная переменная не содержать ни одного символа?
10. Какова длина переменной, не содержащей ни одного символа ?.

Таблица

Задания

n	задание	n	задание
1	В заданном предложении определить количество слов	16	В заданном предложении определить встречается ли в нем заданное слово
2	В заданном предложении добавить пробелы, чтобы получить требуемую длину строки	17	Переписать заданное предложение в обратной последовательности
3	В заданном предложении определить количество раз встречается заданная буква	18	В заданном предложении к каждому слову подписать в конце специальный знак
4	В заданном предложении заменить одно слово на другое	19	В заданном предложении определить все встречающиеся цифры
5	В заданном предложении заменить один символ на другой	20	В заданном предложении определить все встречающиеся знаки препинания
6	В заданном предложении удалить лишние пробелы (оставив между словами один пробел)	21	В заданном предложении удалить символы, заключенные в круглые скобки
7	В заданном предложении заменить прописные буквы на строчные	22	В заданном предложении поменять местами слова, заключенные в круглые скобки
8	Определить сколько символов (букв) встречается в заданном предложении	23	В заданном предложении определить встречаются ли числа в квадратных скобках
9	В заданном предложении определить все слова меньше заданной длины	24	Определить сколько раз встречается в предложении заданное сочетание букв
10	В заданном предложении поменять местами слова	25	В заданном предложении определить все слова, начинающиеся и

			заканчивающиеся на одну и ту же букву
11	В заданном предложении к каждому слову подписать спереди специальный знак	26	В заданном предложении определить все слова, начинающиеся и заканчивающиеся на заданные буквы
12	В заданном предложении каждое слово написать с заглавной буквы	27	В заданном предложении определить все слова, имеющие заданную длину
13	В заданном наборе символов определить количество предложений	28	В заданном предложении определить все слова больше заданной длины
14	В заданном предложении определить встречаются ли числа	29	В заданном предложении определить все числа в квадратных скобках
15	В заданном предложении определить все встречающиеся числа	30	В заданном предложении определить встречается ли знаки арифметических действий

Лабораторная работа №11.
Организация ввода исходных данных.

1. Операторы, используемые для организации ввода исходных данных

Оператор задания спмска констант – DATA

Назначение: Содержит числовые и строковые данные для оператора

Синтаксис: DATA <константа 1> [, <константа 2>]…,

где <константа i> - числовая или строковая константа (i=1, 2…)

Оператор ввода данных с клавиатуры - INPUT

Назначение: Создает условия для ввода данных с клавиатуры в процессе выполнения программы.

Синтаксис: INPUT[;]["<строка приглашения>" {; | ,}] <список переменных>

Аргумент	*Описание*
;	Точка с запятой после ключевого слова INPUT предписывает курсору оставаться на той же самой строке после нажатия клавиши ENTER
<строка приглашения>	Текстовая константа или текстовая переменная, заключённая в кавычки и выводимая на экран в качестве приглашения
;	Точка с запятой после строки приглашения выводит на экран вопросительный знак.
,	Запятая отменяет вывод вопросительного знака после строки приглашения
<список переменных>	Список разделенных запятыми переменных, которым присваиваются вводимые значения

В ответ на приглашение пользователь вводит данные в соответствии со списком переменных.

При несоответствии числа или типа вводимых данных числу и типу переменных списка выдается следующее сообщение об ошибке: Redo from start - повторить сначала. Присваивание входных значений переменным не производится до тех пор, пока не будут введены все данные в соответствии со списком переменных. До нажатия клавиши ENTER допускается внесение исправлений в набираемую строку ввода.

Редактирующие комбинации клавиш, предназначенные для перемещения курсора, удаления и вставки символов текста во входную строку, описаны в табл. 1.

Таблица 1.

Редактирующие комбинации клавиш

Клавиши	*Действие*
CTRL+\ или RIGHT	Перемещение курсора на один символ вправо
CTRL+] или LEFT	Перемещение курсора на один символ влево
CTRL+F или CTRL+RIGHT	Перемещение курсора на одно слово вправо
CTRL+B или CTRL+LEFT	Перемещение курсора на одно слово влево
CTRL+K или HOME	Перемещение курсора в начало вводимой строки
CTRL+N или END	Перемещение курсора в конец вводимой строки
CTRL+R или INS	Переключение режимов вставки и замены. В режиме вставки по мере ввода новых символов символы над курсором и справа от него сдвигаются вправо; в режиме замены просто заменяются
CTRL+I или TAB	Перемещает курсор к ближайшей позиции табуляции. В режиме вставки символы над курсором и справа от него сдвигаются вправо
DEL	Удаление символа над курсором
CTRL+H или BACKSPACE	Удаление символа слева от курсора. При достижении курсором начала строки удаляются символы над курсором
CTRL+E или CTRL+END	Удаление символов от курсора до конца строки;
CTRL+U или ESC	Удаление всей строки независимо от положения курсора
CTRL+M или RETURN	Запись входной строки в память

CTRL+T	Переключение режима отображения наименования функциональной клавиши в нижней части экрана
CTRL+BREAK или CTRL+C	Отказ от ввода данных и принудительное завершение программы

Примечание: знак "+" указывает на одновременное нажатие двух клавиш.

Оператор ввода данных из файла - INPUT #

Назначение: Считывание элементов данных с устройства последовательного доступа или из файла и присваивание их переменным

Синтаксис:

INPUT# <номер файла>, <список переменных>

<Номер файла> соответствует файлу, уже открытому для чтения.

<Список переменных> содержит имена переменных, которым присваиваются считываемые из файла значения. Тип считываемых элементов данных должен соответствовать типам переменных списка.

В отличие от оператора INPUT оператор INPUT# не выводит на экран вопросительный знак.

Элементы данных в файле должны быть записаны таким образом, как если бы они вводились в ответ на приглашение оператора INPUT. Для числовых значений начальные пробелы, символы «возврат каретки» и «перевод строки» игнорируются. Первый символ, не являющийся пробелом, символом «возврат каретки» и «перевод строки», рассматривается как начало числа. Число завершается пробелом, запятой или символами «возврат каретки», «перевод строки».

Если BASIC осуществляет поиск строкового элемента данных в последовательном файле, он также игнорирует начальные пробелы и символы «возврат каретки», «перевод строки». Если при вводе очередного числового или строкового элемента данных достигнут конец файла, то ввод прекращается.

Оператор считывания входных констант - READ

Назначение: Считывание данных из оператора DATA и присваивание их переменным.

Синтаксис: READ <список переменных>

Аргумент <список переменных> представляет собой последовательность переменных языка BASIC, разделенных запятыми. Оператор READ всегда используется совместно с операторами DATA. Оператор READ устанавливает

однозначное соответствие между <списком переменных> и данными, содержащимися в операторе DATA. Эти переменные могут быть как числовыми, так и строковыми. Попытка присвоить строковое значение числовой переменной вызывает синтаксическую ошибку на этапе выполнения. Считывание числового значения в строковую переменную не порождает ошибки и формирует ее значение как строку цифр.

Считываемые в переменные целого типа значения округляются перед присваиванием. Если считываемое значение выходит за допустимый диапазон изменения переменной, то возникает ошибка выполнения.

При считывании строковых значений в строковые переменные фиксированной длины лишние символы отбрасываются справа. Если строковые значения короче строковых переменных, то они выравниваются по левой границе, а оставшиеся позиции заполняются пробелами.

В операторе READ можно использовать только отдельные элементы записей.

Отдельный оператор READ может иметь доступ к одному или нескольким операторам DATA, или несколько операторов READ могут использовать один и тот же оператор DATA. Если число переменных в <списке переменных> оператора READ превышает количество значений в операторах DATA, то появляется сообщение об ошибке: Out of DATA - ошибка в данных.

Если число переменных в операторе READ меньше числа элементов одного или нескольких операторов DATA, то следующий оператор READ начнет чтение данных с первого непрочитанного элемента в операторах DATA. Если операторов READ больше нет, то лишние данные не используются.

Для того чтобы заново прочитать операторы DATA, используется оператор RESTORE.

Оператор ввода строки символов – LINE INPUT

Назначение: Ввод строки длиной до 256 символов в строковую переменную без использования разделителей.

Синтаксис: LINE INPUT[,] ["<строка приглашения>",] <строковая переменная>

Аргумент <строка приглашения> является строковой константой и выводится на экран в качестве подсказки. Вопросительный знак в конце <строки приглашения> не выводится. Все символы, введенные в ответ на приглашение присваиваются, <строковой переменной>.

Точка с запятой, указанная непосредственно после оператора LINE INPUT, оставляет курсор на прежней строке после нажатия клавиши ENTER.

Оператор LINE- INPUT использует те же редактирующие комбинации клавиш, что и оператор INPUT.

Оператор установки флажка считывания в начало входного потока - RESTORE

Назначение: Установка флажка считывания на выбранный оператор DATA.

Синтаксис: RESTORE [[<номер строки> или <метка строки>}]

После выполнения оператора RESTORE без указания <номера строки> или <метки строки> следующий оператор READ будет считывать первый элемент из первого оператора DATA программы.

Если <номер строки> или <метка строки> заданы, то следующий оператор READ обратится к первому элементу выбранного оператора DATA, причем номер или метка строки должны относиться к программе уровня модуля. Отметим, что в среде QuickBASIC операторы DATA автоматически переносятся в программу уровня модуля.

Оператор открытия файла или устройства ввода-вывода -OPEN

Назначение: Позволяет определить файл или устройство для ввода или вывода данных.

Синтаксис:

1. OPEN <файл> [FOR <тип организации 1>]
[ACCESS <режим доступа>] [<статус доступа>]
AS [#]<номер файла> [LEN = <длина записи>]

2. OPEN <тип организации 2>,[#]<номер файла>, <файл> [, <длина записи>]

Аргумент <файл> - строковое выражение, которое содержит либо зарезервированное слово, определяющее устройство, либо имя файла или маршрут к файлу, аналогично маршруту, определяемому в среде DOS.

Аргумент <тип организации 1> определяет способ организации данных в файле и указывает направление передачи данных.

Аргумент <тип организации 1>	*Описание*
OUTPUT	Последовательный файл вывода
INPUT	Последовательный файл ввода
APPEND	Последовательный расширяемый файл вывода. Указатель позиции в файле устанавливается на конец файла, а указатель номера записи - на последнюю запись. Операторы PRINT # и WRITE # будут записывать данные в конец файла

RANDOM	Файл произвольного доступа. Этот режим устанавливается по умолчанию. Если не указана опция ACCESS в этом режиме, то при выполнении оператора OPEN, осуществляются три попытки открыть файл. Попытки установить доступ к файлу выполняются в следующем порядке: 1) чтение/запись; 2) только запись; 3) только чтение
BINARY	Двоичный файл. Для чтения и записи информации в любой байт файла используются операторы GET и PUT. Если не определена опция ACCESS, то осуществляются три попытки открыть файл; они следуют в том же порядке, как и для файла произвольного доступа

Аргумент <режим доступа> - выражение, определяющее тип операции, выполняемой над открываемым файлом.

Аргумент <режим доступа>	*Описание*
READ	Файл открывается только для чтения;
WRITE	Файл открывается только для записи;
READ WRITE	Файл открывается как для чтения, так и для записи. Этот режим возможен только для файлов произвольного доступа, двоичных файлов и файлов, открываемых для работы с опцией APPEND;

Опция <статус доступа> используется в многозадачном режиме для ограничения доступа других процессов (задач) к открытому файлу. Тип защиты может быть следующим:

Опция <статус доступа>	*Описание*
Отсутствует (По умолчанию)	Если статус доступа не указан, файл может быть открыт для чтения и записи любое число раз в этом процессе, но другим процессам запрещен доступ до тех пор, пока файл открыт

SHARED	Любой процесс на любой машине может читать из файла или записывать в него
LOCK READ	Другим процессам запрещается чтение из файла. Этот статус допустим только тогда, когда нет других процессов с режимом доступа READ
LOCK WRITE	Другим процессам запрещается записывать в файл. Данный статус допускается только при условии, что нет процессов у которых уже установлен режим доступа WRITE
LOCK READ WRITE	Другим процессам запрещается как читать из файла, так и записывать в него. Этот статус допустим при отсутствии процессов, имеющих режимы доступа READ или WRITE, а также если ранее не были установлены статусы LOCK READ или LOCK WRITE

Аргумент <номер файла> - числовое выражение целого типа, значение которого должно быть в диапазоне от 1 до 255. Когда выполняется оператор, номер ассоциируется с самим открываемым файлом.

Аргумент <длина записи> - числовое выражение целого типа.

Вторая синтиксическая форма оператора OPEN

Аргумент <тип организации 2> - строковое выражение, единственный символ которого должен быть одним из следующих:

Аргумент <тип организации 2>	*Описание*
O	Последовательный выводной файл
I	Последовательный вводной файл
R	Файл произвольного доступа для ввода-вывода
B	Двоичный файл
A	Последовательный расширяемый файл вывода. Указатель позиции в файле устанавливается на конец файла, а указатель номера записи - на последнюю запись. Операторы PRINT # и WRITE # будут записывать данные в конец файла.

Оператор закрытия файла или устройства ввода-вывода - CLOSE

Назначение: Завершение работы с файлами или устройствами ввода-вывода.

Синтаксис: CLOSE [[#]<номер файла>[, [#]<номер файла>]...]

Аргумент <номер файла> - номер, под которым был открыт файл. В операторе нет аргумента, с помощью которого было бы возможно сразу закрыть все открытые файлы и устройства.

Оператор CLOSE выполняет функцию, противоположную оператору OPEN.

Оператор закрытия всех файлов - RESET

Назначение: Закрывает все дисковые файлы.

Синтаксис: RESET

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Организовать ввод данных с клавиатуры и файла и вывод результатов на экран.

2.2. *Постановка задачи:* При решении задачи в соответствии с индивидуальным заданием организовать ввод исходных данных с клавиатуры, файла и тела программы.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,

- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,

- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы

- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4.*Контрольные вопросы:*

1. Сколько видов ввода исходных данных Вы знаете?
2. Укажите назначение и синтаксис оператора INPUT?
3. Назовите операторы ввода данных с клавиатуры?
4. Назовите операторы ввода данных с файла?
5. Организация и назначение ввода с тела программы?
6. Сколько файлов можно открыть для ввода информации?
7. Назовите особенности ввода информации с файла по сравнению с вводом с клавиатуры?
8. Назовите отличительную особенность операторов CLOSE и RESET?
9. Укажите последовательность ввода данных с файла?
10. Как происходит восстановление данных при повторном считывании с оператора DATA?

Таблица

Задания

n	задание	n	задание
1	Найти минимальное значение в одномерном массиве A()	16	Найти номера элементов массива A(), значения которых совпадают с заданным
2	Найти максимальное значение в одномерном массиве A()	17	Найти номера элементов массива A(), значения которых меньше заданного.
3	Найти минимальное значение в двухмерном числовом массиве A()	18	Найти номера элементов массива A(), значения которых больше заданного.
4	Найти максимальное значение в двухмерном числовом массиве A()	19	В массиве A() найти все числа меньше заданного значения

5	Сформировать массив B(), состоящий из положительных элементов массива A()	20	В массиве A() найти все числа больше заданного значения
6	Сформировать массив B(), состоящий из отрицательных элементов массива A()	21	В массиве A() найти все числа, находящиеся в диапазоне от a до в
7	Сформировать массив B(), состоящий из четных чисел массива A()	22	В массиве A() найти количество элементов, сумма которых меньше заданного
8	Сформировать массив B(), состоящий из нечетных чисел массива A()	23	В массиве A() найти количество элементов, сумма которых больше заданного
9	Найти сумму положительных элементов массива A()	24	В массиве A() найти все суммы соседних пар элементов
10	Найти сумму отрицательных элементов массива A()	25	Сформировать массив A() из элементов, являющихся средним значением соседних пар чисел
11	Найти среднее значение всех элементов массива A()	26	Сортировать одномерный числовой массив A() по убыванию методом минимального элемента
12	Найти номер минимального элемента в массиве A()	27	Определить количество одинаковых элементов в целочисленном массиве A()
13	Найти номер максимального элемента в массиве A()	28	Определить сколько раз в массиве A() встречается заданное число
14	Найти количество минимальных элементов в массиве A()	29	Найти номера элементов массива A(), значения которых равны заданному числу.
15	Найти количество макси-мальных элементов в массиве A()	30	Найти сколько элементов массива A() встречаются в массиве B().

Лабораторная работа №12.
Организация вывода информации на дисплей и печатающее устройство.

1. Операторы, используемые для организации вывода информации на дисплей и печатающее устройство

Оператор вывода данных на терминал - PRINT

Назначение: Вывод данных на экран.

Синтаксис: PRINT [<список выражений>] [{, | ;}]

Если аргумент <список выражений> опущен, то на экран выводится пустая строка. При наличии <списка выражений> значения выражений выводятся на экран. Выражения в списке могут быть числовыми или строковыми. Строковые константы должны быть заключены в кавычки. За выводимыми числами всегда следует пробел; положительным числам всегда предшествует пробел, а отрицательным - знак минус.

Расположение каждого выводимого на экран элемента определяется знаками препинания, используемыми для разделения элементов списка. BASIC разделяет строку на зоны по 14 символов. Запятая в списке выражений определяет вывод каждого очередного значения с начала следующей зоны. Переменные, разделенные точкой с запятой, печатаются непосредственно друг за другом. Один или несколько пробелов или символов табуляции между выражениями действуют аналогично точке с запятой.

Оператор вывода данных на терминал в заданном формате - PRINT USING

Назначение: Вывод строк и чисел в заданном формате в соответствии с шаблоном.

Синтаксис: PRINT USING <шаблон>, <список выражений> [{ , | ;}]

Аргумент <шаблон> представляет собой символьную константу или переменную, содержащую специальные форматирующие символы. Эти форматирующие символы определяют поля для вывода и формат печатаемых строк и чисел.

Аргумент <список выражений> содержит строковые и числовые выражения, разделенные точкой с запятой.

Вывод строк символов: При выводе строк с помощью оператора PRINT USING можно использовать один из следующих форматирующих символов:

Символ	*Описание*
!	Выводит только первый символ заданной строки
\ \	Выводит 2+n символов строки, где n - число пробелов между двумя символами (обратные слеши). Если обратные слеши не

	разделены пробелами, то печатается два символа. Если строка длиннее задаваемого поля, то избыточные символы игнорируются. Если поле длиннее строки, то строка выравнивается по левой границе поля, а свободные позиции заполняются пробелами
&	Определяет символьное поле переменной длины. В поле, описанном знаком &, строка выводится без преобразования

Вывод чисел: При выводе чисел с помощью оператора PRINT USING для форматирования полей используются следующие символы:

Символ	*Описание*
#	Указывает цифровую позицию, которая заполняется при выводе. Если число содержит десятичных знаков меньше, чем число заданных позиций, то оно выравнивается по правой границе поля, а незадействованные позиции заполняются пробелами
.	Задает местоположение десятичной точки; слева располагается целая часть числа, а справа дробная. Если в шаблоне указано, что десятичной точке предшествует цифра, то эта цифра всегда выводится. Числа при необходимости округляются
+	Включает режим вывода знака числа (плюса или минуса) перед числом, если символ указан в шаблоне первым, или после числа, если символ указан в шаблоне последним
-	Задает знаковую позицию числа и может быть только последним символов в шаблоне. При выводе отрицательного числа в эту позицию помещается знак минус, а при выводе положительного числа – пробел
**	Включает режим заполнения начальных пробелов звездочками. Двойная звездочка, кроме того, резервирует позиции для двух дополнительных цифр
$$	Выводит знак денежной единицы непосредственно перед форматируемым числом. Символы $$ резервируют две дополнительные цифровые позиции, одна из которых используется под знак денежной единицы
**$	Объединяет действие символов * и $$. Начальные пробелы заполняются звездочками, а перед числом выводится знак денежной единицы. Символы **$ резервируют три дополнительных цифровых позиции, одна из которых

	используется под знак денежной единицы. При выводе отрицательных чисел знак минус появляется перед знаком денежной единицы
,	Если запятая указана слева от десятичной точки, то она включает режим вывода запятой перед каждой третьей цифрой слева от десятичной точки. Если запятая указана в конце шаблона, то она является разделителем и резервирует дополнительную цифровую позицию. Действие запятой не распространяется на экспоненциальный формат (^^^^или^^^^)
^ ^^^	Задает экспоненциальный формат. Можно также использовать пять символов (""-) для вывода очень больших чисел в формате E+xxx. Десятичная точка может занимать любое положение. Значащие цифры выравниваются по левой границе, а после них указывается экспоненциальный порядок. При отсутствии в шаблоне знака числа положительному числу предшествует пробел, отрицательному - знак минус
_	Символ подчеркивания указывает, что символ, следующий за ним в шаблоне должен быть помещен в поле вывода без изменений. Для вывода самого символа подчеркивания его необходимо указать в форматной строке дважды (_)

Если длина выводимого числа превышает длину заданного числового поля в шаблоне, то перед числом печатается знак %. Подобная ситуация может возникнуть при округлении. Если поле вывода числа содержит более 24 позиций, то появляется сообщение об ошибке: Illegal function call - неверный вызов функции

Оператор вывода данных с символами-разделителями на экран монитора - WRITE

Назначение: Вывод данных на экран монитора.

Синтаксис: WRITE [<список выражений>]

Если <список выражений> опущен, то на экран выводится пустая строка. Если <список выражений> задан, то значения выражений выводятся на экран дисплея. Выражения в списке могут быть числовыми и строковыми и должны быть разделены запятыми. При выводе на экран каждый печатаемый элемент отделяется от предыдущего запятой.

Операторы вывода данных на принтер -LPRINT, LPRINT USING

Назначение: Вывод данных на принтер LPT1.

Синтаксис 1: LPRINT [<список выражений>][{ ; | ,}]

Синтаксис 2: LPRINT USING <шаблон>, <список выражений>[{ ; | ,}]

Эти операторы по своему действию аналогичны операторам PRINT и PRINT USING и отличаются от них только тем, что информация выводится на печатающее устройство. Оператор LPRINT предполагает использование принтера с длиной строки 80 символов. Эта характеристика может быть изменена оператором WIDTH LPRINT.

Оператор задания длины строки при выводе на принтер - WIDTH LPRINT

Назначение: Задает или изменяет длину строки при выводе информации на печать с помощью принтера.

Синтаксис: WIDTH LPRINT <длина строки>

Аргумент <длина строки> может принимать значения от 1 до 255. Вывод данных на устройство печати с использованием стандартного файла аналогично выводу информации на монитор. Если в программе, подготовленной для вывода информации на экран монитора, оператор PRINT заменить на LPRINT, программа будет выполнена правильно и вся информация, предназначенная для вывода на монитор, будет передана на принтер.

Оператор вывода данных на терминал - PRINT

Назначение: Вывод данных на экран.

Синтаксис: PRINT [<список выражений>] [{, | ;}]

Если аргумент <список выражений> опущен, то на экран выводится пустая строка. При наличии <списка выражений> значения выражений выводятся на экран. Выражения в списке могут быть числовыми или строковыми. Строковые константы должны быть заключены в кавычки. За выводимыми числами всегда следует пробел; положительным числам всегда предшествует пробел, а отрицательным - знак минус.

Расположение каждого выводимого на экран элемента определяется знаками препинания, используемыми для разделения элементов списка. BASIC разделяет строку на зоны по 14 символов. Запятая в списке выражений определяет вывод каждого очередного значения с начала следующей зоны. Переменные, разделенные точкой с запятой, печатаются непосредственно друг за другом. Один или несколько пробелов или символов табуляции между выражениями действуют аналогично точке с запятой.

Оператор вывода данных на терминал в заданном формате - PRINT USING

Назначение: Вывод строк и чисел в заданном формате в соответствии с шаблоном.

Синтаксис: PRINT USING <шаблон>, <список выражений> [{ , | ;}]

Аргумент <шаблон> представляет собой символьную константу или переменную, содержащую специальные форматирующие символы. Эти форматирующие символы определяют поля для вывода и формат печатаемых строк и чисел.

Аргумент <список выражений> содержит строковые и числовые выражения, разделенные точкой с запятой.

Вывод строк символов: При выводе строк с помощью оператора PRINT USING можно использовать один из следующих форматирующих символов:

Символ	*Описание*
!	Выводит только первый символ заданной строки
\ \	Выводит 2+n символов строки, где n - число пробелов между двумя символами (обратные слеши). Если обратные слеши не разделены пробелами, то печатается два символа. Если строка длиннее задаваемого поля, то избыточные символы игнорируются. Если поле длиннее строки, то строка выравнивается по левой границе поля, а свободные позиции заполняются пробелами
&	Определяет символьное поле переменной длины. В поле, описанном знаком &, строка выводится без преобразования

Вывод чисел: При выводе чисел с помощью оператора PRINT USING для форматирования полей используются следующие символы:

Символ	*Описание*
#	Указывает цифровую позицию, которая заполняется при выводе. Если число содержит десятичных знаков меньше, чем число заданных позиций, то оно выравнивается по правой границе поля, а незадействованные позиции заполняются пробелами
.	Задает местоположение десятичной точки; слева располагается целая часть числа, а справа дробная. Если в шаблоне указано, что десятичной точке предшествует цифра, то эта цифра всегда выводится. Числа при необходимости округляются
+	Включает режим вывода знака числа (плюса или минуса) перед числом, если символ указан в шаблоне первым, или после числа, если символ указан в шаблоне последним
-	Задает знаковую позицию числа и может быть только

	последним символов в шаблоне. При выводе отрицательного числа в эту позицию помещается знак минус, а при выводе положительного числа - пробел
* *	Включает режим заполнения начальных пробелов звездочками. Двойная звездочка, кроме того, резервирует позиции для двух дополнительных цифр
$ $	Выводит знак денежной единицы непосредственно перед форматируемым числом. Символы $$ резервируют две дополнительные цифровые позиции, одна из которых используется под знак денежной единицы
* *$	Объединяет действие символов * и $$. Начальные пробелы заполняются звездочками, а перед числом выводится знак денежной единицы. Символы **$ резервируют три дополнительных цифровых позиции, одна из которых используется под знак денежной единицы. При выводе отрицательных чисел знак минус появляется перед знаком денежной единицы
,	Если запятая указана слева от десятичной точки, то она включает режим вывода запятой перед каждой третьей цифрой слева от десятичной точки. Если запятая указана в конце шаблона, то она является разделителем и резервирует дополнительную цифровую позицию. Действие запятой не распространяется на экспоненциальный формат (^^^^или^^^^^)
^ ^^^	Задает экспоненциальный формат. Можно также использовать пять символов (""-) для вывода очень больших чисел в формате E+xxx. Десятичная точка может занимать любое положение. Значащие цифры выравниваются по левой границе, а после них указывается экспоненциальный порядок. При отсутствии в шаблоне знака числа положительному числу предшествует пробел, отрицательному - знак минус
_	Символ подчеркивания указывает, что символ, следующий за ним в шаблоне должен быть помещен в поле вывода без изменений. Для вывода самого символа подчеркивания его необходимо указать в форматной строке дважды (_)

Если длина выводимого числа превышает длину заданного числового поля в шаблоне, то перед числом печатается знак %. Подобная ситуация может возникнуть при округлении. Если поле вывода числа содержит более 24

позиций, то появляется сообщение об ошибке: Illegal function call - неверный вызов функции

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Организовать ввод данных с клавиатуры и файла и вывод результатов на экран монитора и в файл (принтер).

2.2. *Постановка задачи:* При решении задачи в соответствии с индивидуальным заданием организовать вывод исходных данных и результатов расчета на экран монитора, печатающее устройство и в файл.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Какими операторами осуществляется вывод информации на экран монитора?
2. Какими операторами осуществляется вывод информации на печатающее устройство?
3. Можно ли с помощью одних и тех же операторов вывести информацию на печать и экран?
4. Можно ли с помощью оператора PRINT выводить данные в разные зоны экрана?
5. Чем отличается форматный вывод информации от бесформатного?
6. Чем ограничивается количество выводимой информации на экран монитора?
7. Чем ограничивается количество выводимой информации на печатающее устройство?
8. Что произойдет с программой если принтер не включен?
9. Могут ли одни и теже данные при выводе на экран и печать иметь различный вид?
10. Что произойдет если при выводе по формату выводимое число окажется больше, чем под него отведено позиций?

Таблица

Задания

n	задание	n	задание
1	Найти минимальное значение в одномерном массиве A()	16	Найти номера элементов массива A(), значения которых совпадают с заданным
2	Найти максимальное значение в одномерном массиве A()	17	Найти номера элементов массива A(), значения которых меньше заданного.
3	Найти минимальное значение в двухмерном числовом массиве A()	18	Найти номера элементов массива A(), значения которых больше заданного.
4	Найти максимальное значение в двухмерном числовом массиве A()	19	В массиве A() найти все числа меньше заданного значения
5	Сформировать массив B(), состоящий из положительных элементов массива A()	20	В массиве A() найти все числа больше заданного значения
6	Сформировать массив B(),	21	В массиве A() найти все числа,

	состоящий из отрицательных элементов массива A()		находящиеся в диапазоне от a до в
7	Сформировать массив B(), состоящий из четных чисел массива A()	22	В массиве A() найти количество элементов, сумма которых меньше заданного
8	Сформировать массив B(), состоящий из нечетных чисел массива A()	23	В массиве A() найти количество элементов, сумма которых больше заданного
9	Найти сумму положительных элементов массива A()	24	В массиве A() найти все суммы соседних пар элементов
10	Найти сумму отрицательных элементов массива A()	25	Сформировать массив A() из элементов, являющихся средним значением соседних пар чисел
11	Найти среднее значение всех элементов массива A()	26	Сортировать одномерный числовой массив A() по убыванию методом минимального элемента
12	Найти номер минимального элемента в массиве A()	27	Определить количество одинаковых элементов в целочисленном массиве A()
13	Найти номер максимального элемента в массиве A()	28	Определить сколько раз в массиве A() встречается заданное число
14	Найти количество минимальных элементов в массиве A()	29	Найти номера элементов массива A(), значения которых равны заданному числу.
15	Найти количество макси-мальных элементов в массиве A()	30	Найти сколько элементов массива A() встречаются в массиве B().

Лабораторная работа №13.
Работа с параметрами экрана в текстовых режимах.

1. Оператор задания поля для вывода информации на экран монитора - WIDTH

Назначение: Задает число строк и число позиций в строке при выводе данных на экран монитора.

Синтаксис: WIDTH <длина строки> [, <число строк>]

Аргумент <длина строки> позволяет установить ширину экрана, при выводе информации на монитор; при этом изменяется размер символов; допустимо только два возможных значения этого аргумента - 40 и 80, по умолчанию принимается значение 80.

Аргумент <число строк> может принимать значения 25, 30, 43, 50 или 60 строк, и это зависит от типа используемого адаптера (EGA, VGA, MCGA) и режима вывода на экран, заданного оператором SCREEN.

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить возможность изменения параметров экрана при выводе текстовой информации.

2.2. *Постановка задачи:* В соответствии с вариантом задания организовать вывод значений функции в виде таблицы с максимально возможным числом строк с использованием символов псевдографики, выделив миганием и цветом максимальное и минимальное значения функции.

2.3. *Порядок выполнения работы*:

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4.*Контрольные вопросы:*

1. Что такое текстовый режим ?
2. Какой оператор устанавливает текстовый вывод на экран монитора?
3. Сколько строк и столбцов стандартно принято в текстовом режиме в языке Basic Microsoft ?
4. Каким оператором можно поменять число строк, одновременно выводимых на экран ?
5. Сколько экранов можно использовать для вывода информации ?
6. Что такое активная страница ?
7. Можно ли с помощью оператора PRINT выводить информацию на экран и при этом не видеть текста ?
8. Что произойдет с выведенной информацией на экране монитора, если изменить параметры вывода на экран ?
9. Каким оператором можно очистить экран для вывода новой информации ?
10. Что произойдет если на экран монитора вывести строк больше, чем на нем размещается ?

Таблица

Задания

n	Функция y(x)	X_{min}	X_{max}	n	Функция y(x)	X_{min}	X_{max}
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5	16	$y=\ln\frac{\sqrt[3]{\lvert A^2+ctg^2 x\rvert}}{A+\ln^2 x};A=6x^x$	0.1	0.6

2	$y=\left(\frac{\lvert x^x-\sqrt{x}\rvert}{1+\sin^3 x}\right)$	0.4	0.8	17	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$	0.3	0.7
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^2 x}{x^2+1}$	0.1	0.6	18	$y=0.5\ln(x+2)\sqrt[3]{\lvert 4-x^2\rvert}$	0.4	0.8
4	$y=\ln\frac{\lvert 2+tg^2 x\rvert}{x^{\sin x}}$	0.3	0.7	19	$y=0.6e^{x^2}-\frac{x-\cos^2 x}{\ln x}$	0.1	0.5
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	0.3	0.7	20	$y=2.8-\frac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$	0.25	0.2
6	$y=\frac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.4	0.8	21	$y=5x^x\ln^2(3+e^x)$	0.3	0.7
7	$y=\frac{\sqrt{x\sin^2 x-1}}{x+a^x};a=\sqrt{\pi}$	0.25	0.2	22	$y=\frac{\sqrt[3]{x}\ln 2x}{\sqrt{\lvert A^2-A-1\rvert}};A=e^{\sin x}$	0.4	0.8
8	$y=\frac{x^2-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6	23	$y=\frac{e^{2x-2}-\sqrt[3]{\lvert x-10\rvert}}{\ln^2 x}$	0.25	0.2
9	$y=\frac{\sqrt{\lvert\sin^2 x+ctgx\rvert}}{\ln x}$	0.3	0.7	24	$y=\frac{\lvert 3-tg^2 x\rvert}{B\ln x-B^2+1};B=e^{2(x+1)}$	0.4	0.8
10	$y=\frac{A+2ctg^2 x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	0.1	0.6	25	$y=\frac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$	0.1	0.6
11	$y=\frac{B^2+\sin^2(0.1x)}{B^3+2\sqrt{B}-1};B=10.5^{\ln x}$	0.1	0.6	26	$y=\left(\frac{x^x+\cos^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5
12	$y=\frac{\lvert 1-2tg^{2.2}x\rvert}{e^{x^2}-\ln x}$	0.25	0.2	27	$y=\frac{\sqrt{\lvert\cos 1.5x+tgx\rvert}}{\ln x}$	0.25	0.2
13	$y=\frac{\sqrt[3]{\lvert A^2+\ln x\rvert}}{x};A=e^{2x}$	0.3	0.7	28	$y=\frac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6
14	$y=\lvert A^3-10A^2+6\rvert;A=e^{x^2tgx}$	0.25	0.2	29	$y=\ln(x+2)\sqrt[3]{\lvert 2-x^2\rvert}$	0.4	0.8
15	$y=\sqrt[4]{\ln^2 x+C^2+C};C=2.2^{\ln x}$	0.4	0.8	30	$y=\frac{tgx-a^3}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.3	0.7

Лабораторная работа №14.
Работа с параметрами библиотеки пользователя.

1. Операторы работы с библиотеками пользователя.

Оператор объявления BASIC-процедур - DECLARE (BASIC)

Назначение*: Объявляет ссылки к BASIC-процедурам и вызывает проверку типов аргументов.*

Синтаксис:

DECLARE {FUNCTION | SUB} **<**имя**>** [([<список параметров>])]

Аргумент	Описание
<имя**>**	Имя процедуры; оно ограничено 40 символами. Имя процедуры-функции FUNCTION может сопровождаться маркером типа (%, &, !, #, $) для возвращаемой величины
<список параметров**>**	Список параметров используется при вызове процедуры, определяет только количество и тип аргументов

Оператор объявления процедур на языках семейства Microsoft - DECLARE

Назначение: Объявляет вызываемые последовательности внешних процедур, написанных на других языках программирования семейства Microsoft.

Синтаксис 1:

DECLARE FUNCTION **<**имя**>** [CDECL] [ALIAS "< альтернативное имя>"] [([<список параметров>])]

Синтаксис 2

DECLARE SUB **<**имя**>** [CDECL] [ALIAS "<альтернативное имя>"] [([<список параметров>])]

Элемент	*Описание*
FUNCTION	Указывает, что внешняя процедура возвращает значение; может использоваться в составе арифметических выражений
SUB	Указывает, что внешняя процедура вызывается аналогично BASIC-процедуре
<имя**>**	Имя, используемое в BASIC-программе для вызова процедуры; длина до 40 символов; имя процедуры-функции может включать маркер типа (%, &, !, #, $) для

	.возвращаемой величины.
CDECL	Указывает, что процедура использует порядок размещения аргументов, свойственный языку C; ключевое слово CDECL указывает на передачу аргументов справа налево, а не слева направо, как принято в среде языка BASIC
ALIAS	Указывает, что процедура имеет другое имя в .OBJ-файле или библиотеке; если отсутствует ключ ALIAS, то имя процедуры отображается строчными буквами, маркер типа удаляется, а в начале имени добавляется символ подчеркивания, и именно это имя используется при поиске библиотек и внешних файлов; если за ключом CDECL следует ключ ALIAS, то используется указанное альтернативное имя.

Синтаксис аргумента <список параметров>
[{BYVAL SEG}] <переменная> [AS <тип>]
[, [{BYVAL SEG}] <переменная> [AS <тип>]] ...

Элемент	*Описание*
BYVAL	Указывает, что параметр передается значением, а не ссылкой; передача ссылкой принята по умолчанию; ключ можно применять только для числовых параметров типа INTEGER, LONG, SINGLE, DOUBLE; при использовании BYVAL текущий аргумент преобразуется к типу, указанному в операторе DECLARE, непосредственно перед передачей параметра
SEG	Указывает, что параметр передается адресом сегмента
<переменная>	Имя, допустимое для переменных языка BASIC; существенным является тип переменной; если переменная - массив, то в скобках можно указать его размерность (в частности, чтобы обеспечить совместимость с прежними версиями языка)
AS <тип>	Указывает тип переменной: INTEGER, LONG, SINGLE, DOUBLE, STRING, ANY - или имя структуры; можно указывать также и маркер типа (%, &, !, #, $) или принимать его по умолчанию; при объявлении внешних процедур, написанных на других языках, можно указать тип ANY, и это приведет к тому, что проверка типа не выполняется; нельзя

	использовать тип ANY для аргумента, передаваемого значением

Если не используются ключи BYVAL и SEG, то аргументы передаются смещением.

Оператор вызова BASIC-процедур - CALL

Назначение: Передает управление подпрограмме SUB на языке BASIC.

Синтаксис 1: CALL **<**имя> [(<список аргументов>)]

Синтаксис 2: <имя> [(<список аргументов>)]

Аргумент	*Описание*
<имя>	Имя ограничено длиной в 40 символов. Имя должно быть объявлено в операторе SUB, если процедура размещается в этом же модуле
<список аргументов>	Переменные или константы, передаваемые в процедуру. Аргументы в списке отделяются за-- пятыми. Аргументы, передаваемые ссылкой, могут быть изменены при выполнении процедуры

Если <список аргументов> включает переменную массива, то массив указывается именем, за которым следуют пустые скобки.

2. Описание практической части работы*:*

2.1. *Цели лабораторной работы:* Изучить возможность подключения библиотек пользователя, работающих в среде qbx.exe.

2.2. *Постановка задачи:* В соответствии с вариантом задания рассчитать значение функции на интервале x от 0 до 360 град., используя возможность подключения функций из библиотеки пользователя.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Какие программы помещают в библиотеки ?
2. Есть ли в языке Basic Microsoft стандартные библиотеки и что в них содержится ?
3. Для чего нужна библиотека пользователя ?
4. Можно ли в Basic Microsoft использовать одновременно несколько библиотек ?
5. Как вызываются функции и подпрограммы из библиотеки пользователя ?
6. Может ли имя переменной совпадать с именем процедуры, размещенной в подключенной библиотеки ?
7. Каким оператором вызывается подпрограмма из библиотеки ?
8. Может ли библиотечный файл быть в виде текстового файла, написанного на языке Basic Microsoft и как его подключить ?
9. Что содержит файл с расширением .мак ?
10. можно ли объединить несколько библиотек в одну ?

Таблица

Задания

n	Функция y(x)	n	Функция y(x)
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	16	$y=\ln\frac{\sqrt[3]{\lvert A^2+ctg^2 x\rvert}}{A+\ln^2 x};A=6x^x$

2	$y=\left(\frac{\left\|x^{x}-\sqrt{x}\right\|}{1+\sin^{3}x}\right)$	17	$y=\sqrt{x+e^{x}}-\cos x+\sqrt[3]{x}$
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^{2}x}{x^{2}+1}$	18	$y=\cos x\sqrt[3]{\left\|4-x^{2}\right\|}$
4	$y=\ln\frac{\left\|2+tg^{2}x\right\|}{x^{\sin x}}$	19	$y=0.6e^{x^{2}}-\frac{x-\cos^{2}x}{\ln x}$
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	20	$y=2.8-\frac{F}{\sqrt[3]{F^{2}+1}};F=e^{\sin^{2}x}$
6	$y=\frac{2.5tgx-a^{2.1}}{a\left\|1-x\right\|};a=\sqrt{\pi}$	21	$y=5x^{x}\cos x\left(3+e^{x}\right)$
7	$y=\frac{\sqrt{x\sin^{2}x-1}}{x+a^{x}};a=\sqrt{\pi}$	22	$y=\frac{\sqrt[3]{x}\cos 2x}{\sqrt{\left\|A^{2}-A-1\right\|}};A=e^{\sin x}$
8	$y=\frac{x^{2}-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^{3}$	23	$y=\frac{\sin x-\sqrt[3]{\left\|x-10\right\|}}{\ln^{2}x}$
9	$y=\frac{\sqrt{\left\|\sin^{2}x+ctgx\right\|}}{\ln x}$	24	$y=\frac{\left\|3-tg^{2}x\right\|}{B\ln x-B^{2}+1};B=e^{2(x+1)}$
10	$y=\frac{A+2ctg^{2}x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	25	$y=\frac{\sqrt{x}-\sin^{2}x}{x+2^{x+1}}$
11	$y=\frac{B^{2}+\sin^{2}(0.1x)}{B^{3}+2\sqrt{B}-1};B=10.5^{\ln x}$	26	$y=\left(\frac{x^{x}+\cos^{2}x}{\left\|2-x\right\|}\right)^{\frac{1}{3}}$
12	$y=\frac{\left\|1-2tg^{2.2}x\right\|}{e^{x^{2}}-\ln x}$	27	$y=\frac{\sqrt{\left\|\cos 1.5x+tgx\right\|}}{\ln x}$
13	$y=\frac{\sqrt[3]{\left\|A^{2}+\sin x\right\|}}{x};A=e^{2x}$	28	$y=\frac{x^{3}-4x+6}{ax+\sin x\times\cos x};a=\pi^{3}$
14	$y=\left\|A^{3}-10A^{2}+6\right\|;A=e^{x^{2}tgx}$	29	$y=\cos(x+2)\sqrt[3]{\left\|2-x^{2}\right\|}$
15	$y=\sqrt[4]{\cos x+C^{2}+C};C=2.2^{\ln x}$	30	$y=\frac{tgx-a^{3}}{a\left\|1-x\right\|};a=\sqrt{\pi}$

Лабораторная работа №15.
Работа с файлами в среде BASIC MICROSOFT.

1. Операторы, применяемые в среде BASIC MICROSOFT.

Оператор открытия файла или устройства ввода-вывода -OPEN

Назначение: Позволяет определить файл или устройство для ввода или вывода данных.

Синтаксис:

2. OPEN <файл> [FOR <тип организации 1>]
[ACCESS <режим доступа>] [<статус доступа>]
AS [#]<номер файла> [LEN = <длина записи>]

2. OPEN <тип организации 2>,[#]<номер файла>, <файл> [, <длина записи>]

Аргумент <файл> - строковое выражение, которое содержит либо зарезервированное слово, определяющее устройство, либо имя файла или маршрут к файлу, аналогично маршруту, определяемому в среде DOS.

Аргумент <тип организации 1> определяет способ организации данных в файле и указывает направление передачи данных.

Аргумент <тип организации 1>	*Описание*
OUTPUT	Последовательный файл вывода
INPUT	Последовательный файл ввода
APPEND	Последовательный расширяемый файл вывода. Указатель позиции в файле устанавливается на конец файла, а указатель номера записи - на последнюю запись. Операторы PRINT # и WRITE # будут записывать данные в конец файла
RANDOM	Файл произвольного доступа. Этот режим устанавливается по умолчанию. Если не указана опция ACCESS в этом режиме, то при выполнении оператора OPEN, осуществляются три попытки открыть файл. Попытки установить доступ к файлу выполняются в следующем порядке: 1) чтение/запись; 2) только запись; 3) только чтение
BINARY	Двоичный файл. Для чтения и записи

	информации в любой байт файла используются операторы GET и PUT. Если не определена опция ACCESS, то осуществляются три попытки открыть файл; они следуют в том же порядке, как и для файла произвольного доступа

Аргумент <режим доступа> - выражение, определяющее тип операции, выполняемой над открываемым файлом.

Аргумент <режим доступа>	*Описание*
READ	Файл открывается только для чтения;
WRITE	Файл открывается только для записи;
READ WRITE	Файл открывается как для чтения, так и для записи. Этот режим возможен только для файлов произвольного доступа, двоичных файлов и файлов, открываемых для работы с опцией APPEND;

Опция <статус доступа> используется в многозадачном режиме для ограничения доступа других процессов (задач) к открытому файлу. Тип защиты может быть следующим:

Опция <статус доступа>	*Описание*
Отсутствует (По умолчанию)	Если статус доступа не указан, файл может быть открыт для чтения и записи любое число раз в этом процессе, но другим процессам запрещен доступ до тех пор, пока файл открыт
SHARED	Любой процесс на любой машине может читать из файла или записывать в него
LOCK READ	Другим процессам запрещается чтение из файла. Этот статус допустим только тогда, когда нет других процессов с режимом доступа READ
LOCK WRITE	Другим процессам запрещается записывать в файл. Данный статус допускается только при условии, что нет процессов у которых уже установлен режим доступа WRITE
LOCK READ WRITE	Другим процессам запрещается как читать из файла, так и записывать в него. Этот статус допустим

	при отсутствии процессов, имеющих режимы доступа READ или WRITE, а также если ранее не были установлены статусы LOCK READ или LOCK WRITE

Аргумент <номер файла> - числовое выражение целого типа, значение которого должно быть в диапазоне от 1 до 255. Когда выполняется оператор, номер ассоциируется с самим открываемым файлом.

Аргумент <длина записи> - числовое выражение целого типа.

Вторая синтиксическая форма оператора OPEN

Аргумент <тип организации 2> - строковое выражение, единственный символ которого должен быть одним из следующих:

Аргумент <тип организации 2>	*Описание*
O	Последовательный выводной файл
I	Последовательный вводной файл
R	Файл произвольного доступа для ввода-вывода
B	Двоичный файл
A	Последовательный расширяемый файл вывода. Указатель позиции в файле устанавливается на конец файла, а указатель номера записи - на последнюю запись. Операторы PRINT # и WRITE # будут записывать данные в конец файла.

Операторы вывода данных в файл -PRINT #, PRINT USING

Назначение: Запись данных в последовательный файл.

Синтаксис: PRINT #<номер файла>, [USING <шаблон>;] <список выражений>[(, | ;)]

Аргумент <номер файла> соответствует номеру, присвоенному при открытии файла. Аргумент <шаблон> состоит из форматирующих символов введенных при описании оператора PRINT USING. Выражения из <списка выражений> могут быть числовыми или строковыми, и их значения предназначены для записи в файл. Если <список выражений> отсутствует, то оператор PRINT # помещает в фаю пустую строку.

Оператор PRINT # записывает данные в файл точно так же, как оператор PRINT выводит данные на экран.

Оператор записи данных в последовательный файл WRITE #

Назначение: Запись данных в последовательный файл.

Синтаксис: WRITE #<номер файла>,<список выражений>

Неформатированная запись данных в файл с разделителем - запятая.
<номер файла> - номер файла, открытого оператором OPEN
<список выражений> - <выражение>{,|;}<выражение>{,|;} ...

Оператор закрытия файла или устройства ввода-вывода - CLOSE

Назначение: Завершение работы с файлами или устройствами ввода-вывода.

Синтаксис: CLOSE [[#]<номер файла> [, [#]<номер файла>]...]

Аргумент <номер файла> - номер, под которым был открыт файл. В операторе нет аргумента, с помощью которого было бы возможно сразу закрыть все открытые файлы и устройства.

Оператор закрытия всех файлов - RESET

Назначение: Закрывает все дисковые файлы.

Синтаксис: RESET

Функция определения длины файла - LOF

Назначение: Определяет длину указанного файла в байтах.

Синтаксис: LOF(<Номер файла>)

Аргумент <номер файла> должен соответствовать присвоенному в операторе OPEN. Выдается длина файла любого типа организации. Функцию LOF нельзя использовать при работе с устройствами SCRN, KYBD, CONS и LPTn. Если устройство открывается как файл с помощью оператора OPEN COM, то функция LOF выдает число свободных байтов в его выходном буфере.

Функция определения текущей позиции файла – LOC

Назначение: Определяет текущую позицию файл.

Синтаксис: LOC(<номер файла>)

Аргумент <номер файла> должен соответствовать присвоенному в операторе OPEN. При работе с файлами произвольного доступа функция LOC выдает номер последней записи, прочитанной или записанной в файл. При работе с файлами последовательного доступа LOC выдает номер последнего записанного или прочитанного 128-байтного блока. При работе с двоичными файлами LOC выдает позицию последнего прочитанного или записанного байта.

При работе с портами ввода-вывода (COM:) функция LOC выдает число символов во входной очереди, подлежащих считыванию. Получаемое значение зависит от того, для какого режима открыто устройство - текстового или двоичного. В текстовом режиме процедуры нижнего уровня прекращают постановку символов в очередь, как только встречается символ конца файла.

Сам этот символ в очередь не включается, и его нельзя считать. В двоичном режиме символ конца файла игнорируется, и можно считать файл целиком. Функцию LOC нельзя применять при работе с устройствами SCRN, KYBD и LPTn.

Функция проверки признака конца файла - EOF

Назначение: Проверяет условие конца файла.

Синтаксис: EOF(<номер файла>)

Функция EOF возвращает значение -1 (TRUE), если обнаруживается признак конца последовательного файла (символ с кодом 26). Функцию EOF используют для проверки этого признака при вводе данных. Это дает возможность избежать появления сообщений об ошибке при попытке чтения из закончившегося файла.

Если функция EOF применяется при работе с двоичными файлами или файлами произвольного доступа, она выдает значение "TRUE", когда последний оператор GET не обеспечивает считывания полной записи. Это происходит, когда файл считан целиком.

Функцию нельзя применять при работе с устройствами SCRN, KYBD, CONS и LPTn.

Если EOF используется при работе с устройством COM, условие окончания файла зависит от режима, для которого открыто устройство (текстовый или двоичный). В текстовом режиме функция EOF выдает значение О (FALSE) до тех пор, пока не встретится код 26, после чего до закрытия устройства она выдает значение 1 (TRUE). В двоичном режиме значение "TRUE" (-1) выдается, когда входная очередь пусто (LOC(n) = 0). Если очередь не пуста, EOF генерирует значение "FALSE" (0).

Оператор вывода списка имен файлов - FILES

Назначение: Выводит на экран имена файлов, расположенных в текущем или указанном каталоге.

Синтаксис: FILES [<файл>]

Аргумент <файл> - строковая переменная или константа, содержащая имя файла или маршрут к файлу. В аргументе можно указывать и имя дисковода. Если аргумент <файл> опущен, оператор выдает поименный список всех файлов текущего каталога. В аргументе можно использовать символы-заменители: знаки вопроса (?) или звездочки С) (так же как в команде D1R среды DOS). Знак вопроса соответствует любому символу в имени файла или расширении. Звездочка определяет один или более символов, подставляемых вместо нее.

Оператор изменения имени файла - NAME

Назначение: Изменение имени файла на диске.

Синтаксис: NAME <старое имя> AS <новое имя>

Аргументы <старое имя> и <новое имя> - строковые выражения, содержащие имена файлов, а если требуется, то и маршруты к файлам. Оператор может работать только с одним дисководом.

Оператор записи управляющей строки в драйвер - IOCTL

Назначение: Передает строку с управляющими данными в драйвер устройства.

Синтаксис: IOCTL [#]<номер файла>, <строка>

Аргумент <номер файла> - номер, который был присвоен устройству при его открытии. Аргумент <строка> содержит команду, посылаемую устройству. Команда определяется спецификой драйвера устройства. Длина строки управляющих данных не более 32767 байт.

Оператор работает только тогда, когда все следующие условия будут удовлетворены:

- Драйвер устройства установлен.
- Драйвер устройства в состоянии поддерживать работу со строками управляющих данных, посылаемых оператором IOCTL. Можно проверить возможность такой поддержки с помощью функции DOS - &H44, используя прерывания &H21 и системную процедуру CALL INTERRUPT.
- Выполнен оператор OPEN для данного устройства, и в данный момент оно открыто.

Оператор пересылки содержимого области памяти в файл или на устройство - BSAVE

Назначение: Пересылает содержимое области памяти в файл или на выходное устройство. При этом запись производится побайтно и называется обр. памяти, этот файл может быть использован оператором BLOAD для считывания и загрузки в оперативную память.

Синтаксис: BSAVE <файл>, <смещение>, <длина>

Аргумент	Описание
<файл>	Строковое выражение, содержащее имя файла или устройства. Оператор поддерживает л>с выходные устройства, кроме консоли (SCRICONS:)
<смещение>	Задает начальный адрес сохраняемой области памяти
<длина>	Задает число байт сохраняемой памяти. Аргумент -

	числовое выражение, приведена целому типу без знака, и его значение ограничено диапазоном О - 65535

Оператор загрузки образа памяти из файла или устройства - BLOAD

Назначение: Загружает в оперативную память содержимое файла, сохраненное оператором BSAVE, из файла или устройства ввода.

Синтаксис: BLOAD <файл>, [<смещение>]

Аргумент	Описание
<файл>	Строковое выражение, содержащее спецификацию файла. Оператор поддерживает любые устройства ввода, кроме клавиатуры (KYBD:)
<смещение>	Смещение адреса начала загрузки

Оператор установки начальной позиции в файле - SEEK

Назначение: Установка позиции в файле для последующего чтения или записи.

Синтаксис: SEEK [#]<номер файла>, <позиция>

Аргумент <номер файла> - целое число, которое использовалось в операторе OPEN при открытии файла.

Аргумент <позиция> - числовое выражение, указывающее, в каком месте файла осуществлять последующее чтение или запись. Значение аргумента в интервале от 1 до 231 - 1. Для файлов прямого доступа аргумент <позиция> задает номер записи в файле.

Операторы захвата и освобождения файла - LOCK, UNLOCK

Назначение: Осуществляют захват и освобождение всего или части открытого файла для обеспечения доступа к нему нескольких процессов.

Синтаксис: LOCK [#]<номер файла>[,{<запись>|[<начало>] TO <конец>)]

……..

UNLOCK [#]<номер файла>[,[<запись>|[<начало>] TO <конец>}]

Эти операторы применяются в сетевом режиме, когда несколько процессов требуют доступа к одному файлу. Операторы имеют следующие аргументы:

Аргумент	Описание
<номер файла>	Номер, соответствующий моменту открытия файла

<запись>	Номер защищаемой записи или байта; любой номер в интервале от 1 до 231 - 1. Длина записи не более 32767 байт
<начало>	Номер первой защищаемой записи или байта
<конец>	Sk>Мер последней защищаемой записи или байта

Операторы LOCK и UNLOCK всегда используются совместно.

Оператор выделения памяти (для файлов произвольного доступа) - FIELD

Назначение*: Выделяет память под переменные в буфере файла произвольного доступа.*

Синтаксис: FIELD [#]<номер файла>, <длина поля> AS <имя переменной> . . .

Аргумент	Описание
<номер файла>	Номер файла при его открытии
<длина поля>	Ширина поля записи в файле
<имя переменной>	Имя строковой переменной, которая содержит или считанные данные, или данные, присвоенные переменной, для записи в файл

Операторы записи информации в файл и считывания из файла - PUT, GET

Назначение: Записывают содержимое переменной или буфера (при произвольном доступе) в файл на диске.

Синтаксис:

PUT [#]<номер файла> [, [<номер записи>] [, <переменная>]]

GET [#]<номер файла> [, [<номер записи>] [, <переменная>]]

Аргумент	Описание
<номер файла>	Номер файла при его открытии
<номер записи>	Для файлов произвольного доступа определяет номер записываемой записи, а для двоичных файлов - номер байта, с которого начинается запись. Начальная запись или байт размещается в файле под номером 1. Если аргумент опущен, то в файл помещается та запись (байт), которая является следующей после последнего выполненного оператора PUT (GET), или та запись (байт), на которую установлен указатель последним оператором SEEK. Наибольшее возможное

	значение аргумента 231-1
<переменная>	Переменная, содержащая выходные данные для записи в файл. Оператор записывает столько байт в файл, сколько байт отведено под переменную. Если используется переменная, то не требуется применять функции MKI$, MKL$, MKS$ или MKD$ для преобразования числовых полей, предназначенных для записи. В этом случае можно не использовать оператор FIELD. Для файлов произвольного доступа можно использовать любую переменную, длина которой меньше или равна длине записи. Обычно тип и длина переменной определяются в соответствии с полем записи. Для двоичных файлов можно использовать любую переменную. Длина записи не более 32767 байт

Оператор изменения текущего каталога – CHDIR

Назначение: Изменяет имя текущего каталога и имя дисковода.

Синтаксис: CHDIR <маршрут>

Аргумент <маршрут> - строковое выражение, которое определяет имена дисковода и каталога в следующем виде:

[<имя дисковода>[/]<имя каталога> [<имя каталога>]...

Строковое выражение длиной не более 64 символов определяет <маршрут>.

Оператор создания нового каталога - MKDIR

Назначение: Создает новый каталог.

Синтаксис: MKDIR <маршрут>

Аргумент <маршрут> - строковое выражение, задающее имя создаваемого каталога, длиной не более 128 символов. Оператор MKDIR аналогичен команде MKDIR дисковой операционной системы, однако в языке BASIC недопустимо сокращение MD, принятое в DOS.

2. Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить возможности работы с файлами среде BASIC MICROSOFT при решении заданной задачи.

2.2. *Постановка задачи:* В соответствии с заданием организовать создание новой директории, внутри которой создать временные файлы для значений функции, вычисленных с разным шагом. В программе организовать

просмотр файлов с помощью оператора LINE INPUT# с целью выбора наиболее удачного расчета. После окончания работы удалить временные файлы, сохранив выбранный, перенеся его в рабочую директорию. Удалить рабочую директорию.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Что такое файл?

2. Можно ли открыть один и тот же файл одновременно для считывания и записи?

3. Какие файлы данных Вы знаете?

4. Чем отличаются бинарный файл и файл последовательного доступа?

5. Можно ли дописать информацию в имеющийся файл?

6. Можно ли удалить файл из программы, написанной на языке Basic Microsoft?

7. Каким образом можно создать и удалить каталог из программы, написанной на языке Basic Microsoft?

8. Что такое путь к файлу?

9. Что такое расширение файла?

10. Сколько символов может иметь файл, создаваемый в среде Basic Microsoft?

Таблица

Задания

n	Функция y(x)	X_{min}	X_{max}
1	$y=\left(\frac{x^x+\sin^2 x}{\lvert 2-x\rvert}\right)^{\frac{1}{3}}$	0.1	0.5
2	$y=\left(\frac{\lvert x^x-\sqrt{x}\rvert}{1+\sin^3 x}\right)$	0.4	0.8
3	$y=\frac{1-x^{2.6}}{F^{1.5}-\sin x};F=\frac{2ctg^2 x}{x^2+1}$	0.1	0.6
4	$y=\ln\frac{\lvert 2+tg^2 x\rvert}{x^{\sin x}}$	0.3	0.7
5	$y=e^{\frac{x\sin x}{x+\cos x}}+tg^{2.2}x$	0.3	0.7
6	$y=\frac{2.5tgx-a^{2.1}}{a\lvert 1-x\rvert};a=\sqrt{\pi}$	0.4	0.8
7	$y=\frac{\sqrt{x\sin^2 x-1}}{x+a^x};a=\sqrt{\pi}$	0.25	0.2
8	$y=\frac{x^2-1.8x+2.6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6
9	$y=\frac{\sqrt{\lvert\sin^2 x+ctgx\rvert}}{\ln x}$	0.3	0.7
10	$y=\frac{A+2ctg^2 x}{\sqrt{A+tgx}};A=\pi^{\sin x}$	0.1	0.6
11	$y=\frac{B^2+\sin^2(0.1x)}{B^3+2\sqrt{B}-1};B=10.5^{\ln x}$	0.1	0.6

12	$y=\frac{\left\|1-2tg^{2.2}x\right\|}{e^{x^2}-\ln x}$	0.25	0.2
13	$y=\frac{\sqrt[3]{\left\|A^2+\ln x\right\|}}{x};A=e^{2x}$	0.3	0.7
14	$y=\left\|A^3-10A^2+6\right\|;A=e^{x^2tgx}$	0.25	0.2
15	$y=\sqrt[4]{\ln^2 x+C^2+C};C=2.2^{\ln x}$	0.4	0.8
16	$y=\ln\frac{\sqrt[3]{\left\|A^2+ctg^2x\right\|}}{A+\ln^2 x};A=6x^x$	0.1	0.6
17	$y=\sqrt{x+e^x}-\ln^2 x+\sqrt[3]{x}$	0.3	0.7
18	$y=0.5\ln(x+2)\sqrt[3]{\left\|4-x^2\right\|}$	0.4	0.8
19	$y=0.6e^{x^2}-\frac{x-\cos^2 x}{\ln x}$	0.1	0.5
20	$y=2.8-\frac{F}{\sqrt[3]{F^2+1}};F=e^{\ln^2 x}$	0.25	0.2
21	$y=5x^x\ln^2(3+e^x)$	0.3	0.7
22	$y=\frac{\sqrt[3]{x}\ln 2x}{\sqrt{\left\|A^2-A-1\right\|}};A=e^{\sin x}$	0.4	0.8
23	$y=\frac{e^{2x-2}-\sqrt[3]{\left\|x-10\right\|}}{\ln^2 x}$	0.25	0.2
24	$y=\frac{\left\|3-tg^2x\right\|}{B\ln x-B^2+1};B=e^{2(x+1)}$	0.4	0.8
25	$y=\frac{\sqrt{x}-\sin^2 x}{x+2^{x+1}}$	0.1	0.6
26	$y=\left(\frac{x^x+\cos^2 x}{\left\|2-x\right\|}\right)^{\frac{1}{3}}$	0.1	0.5
27	$y=\frac{\sqrt{\left\|\cos 1.5x+tgx\right\|}}{\ln x}$	0.25	0.2
28	$y=\frac{x^3-4x+6}{ax+\sin x\times\cos x};a=\pi^3$	0.1	0.6
29	$y=\ln(x+2)\sqrt[3]{\left\|2-x^2\right\|}$	0.4	0.8

30	$y = \frac{tgx - a^3}{a\lvert 1 - x\rvert}; a = \sqrt{\pi}$	0.3	0.7

Лабораторная работа №16
Создание исполняемых файлов и библиотек пользователя.

1. Создание исполняемых файлов и библиотек пользователя

В среде Basic Microsoft исполняемые файлы создаются при помощи меню «Run» «Make EXE File».

Меню «Run»

Меню «Make EXE File»

Библиотеки пользователя создаются при помощи меню «Make Library».

Меню «Make Library»

2.Описание практической части работы:

2.1. *Цели лабораторной работы*: Ознакомиться с принципом создание исполняемых файлов и библиотек пользователя.

2.2. *Постановка задачи:* Описать последовательность создания библиотек пользователя (своей библиотеки) и создать исполняемые файлы: работающий самостоятельно и с поддержкой библиотек среды программирования Basic Microsoft (BC7). Указать отличия между созданными файлами, дать рекомендации по их использованию.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Можно ли создать исполняемый файл в среде Basic Microsoft ?
2. Зачем нужно создавать исполняемые файла с поддержкой библиотеки ?
3. Какой файл занимает больше памяти на жестком диске откомпилированный с библиотекой или без нее ?
4. Какие файлы создаются при компилировании текстового файла, написанного на языке Basic Microsoft ?
5. Какие программы, входящие в пакет среды программирования Basic Microsoft версии 4.0, 4.5 и BC.7, нужны для создания исполняемого файла?
6. Что такое компилирование программы ?
7. Можно ли откомпилировать текстовый файл Basic Microsoft не из среды программирования ?
8. Может ли библиотечный файл быть в виде текстового файла, написанного на языке Basic Microsoft и как его подключить ?
9. Что содержит файл с расширением .мак ?
10. Можно ли объединить несколько библиотек в одну ?

Задание

Произвести компиляцию программы одной из ранее сделанных лабораторных работ (5…15).

Лабораторная работа №17
Построение графических примитивов в среде BASIC MICROSOFT

1. Синтаксис операторов для вывода графических примитивов в Qb.

Оператор отображения точки на экране монитора:
Назначение: Высвечивает на экране точку
Синтаксис:
PSET [STEP] (x,y) [,цвет] ,
где STEP - координаты x, y задаются относительно текущего положения курсора (в приращении), по умолчанию абсолютная система координат;
x,y - координаты точки растра;
цвет - цвет точки, по умолчанию цвет текущего переднего плана.

Оператор отображения точки на экране монитора:
Назначение: Высвечивает на экране точку
Синтаксис:
PRESET [STEP] (x,y) [,цвет],
где STEP - координаты x, y задаются относительно текущего положения курсора (в приращении), по умолчанию абсолютная система координат;
x, y - кординаты точки растра;
цвет - цвет точки, по умолчанию цвет фона.

Оператор отображения линии или прямоугольника :
Назначение: На экран выводится отрезок или прямоугольник
Синтаксис:
LINE [[STEP] (x1,y1)] - [STEP] (x2,y2) [, [цвет]
[, [B [F]] [, <стиль%>]]]
где STEP – атрибут, указывающий, что испотльзуется относительная форма задания координат;
x1, y1 и x2, y2 - координаты начала и конца линии или координаты противоположных углов прямоугольника;
цвет% - цвет линии, по умолчанию цвет переднего плана;
B или BF - построение прямоугольника соответственно без и с закрашиванием внутренней части;
стиль% - 16 битовое число служащее маской для построения прерывистых линий.

Оператор построения окружностей и эллипсов:

Назначение: Вычерчивание окружности с заданным радиусом или эллипса с заданным отношением полуосей относительно некоторого центра.

Синтаксис:

CIRCLE [STEP] (x, y), <радиус> [<цвет>] [, <начало>] [, [<конец>] [,<сжатие>]]]]

где STEP –аргумент указывает, что координаты x, y интерпретируются как относительные смещения от текущей позиции графического курсора;

x, y - кординаты центра окружности или эллипса;

<радиус> - радиус окружности или большая полуось эллипса;

<цвет> - цвет точки, по умолчанию цвет текущего переднего плана.

<начало> - координата начала дуги в радианах;

<конец> - координата конца дуги в радианах;

<сжатие> - отношение оси Y к ос X.

Функция определения координат или цвета пикселя - POINT

Назначение: Определяет номер цвета пикселя или выдает его координаты.

Синтаксис: POINT(x,y)

POINT(<число>)

Если в функции POINT задается пара координат x и y, возвращается номер цвета пикселя с этими координатами. В том случае, когда координаты выходят из допустимого диапазона, выдается значение -1.

Функция POINT с аргументом <число> позволяет получить текущую координату курсора (варианты значений аргумента перечислены ниже).

Значение	Возвращаемый результат
0	Текущая абсолютная координата x
1	Текущая абсолютная координата y
2	Текущая относительная координата x в системе координат, задаваемой оператором WINDOW. Если оператор WINDOW не выполнялся, то возвращается значение функции POINT (0)
3	Текущая относительная координата y. Если оператор WINDOW не выполнялся, то возвращается значение функции POINT (0).

Функция определения кода и байт-атрибута символа –SCREEN

Назначение: Выдает код ASCII или байт-атрибут символа, высвеченного в заданной позиции.

Синтаксис: SCREEN (<строка>, <столбец>[, <флажок>])

Аргумент	Описание
<строка>	Числовое выражение, задающее номер строки, в которой находится символ
<столбец>	Числовое выражение, задающее номер столбца, в котором находится символ.
<флажок>	Числовое выражение. Если его значение не равно нулю, выдается байт-атрибут символа. Если значение нулевое или <флажок> отсутствует, выдается код ASCII этого символа

Каждый символ представляется двумя байтами: первый - байт-атрибут - содержит информацию о цвете, а второй - код ASCII отображаемого символа. Байт-атрибут имеет следующую структуру:

Бит *Назначение*

X... .... Бит мерцания: символ мерцает, если бит равен 1

XXX .... Номер цвета фона

XXXX Номер цвета переднего плана. Старший бит - бит интенсивности свечения

2.Описание практической части работы*:*

2.1. *Цели лабораторной работы:* Ознакомиться с режимами графического вывода информации на экран монитора в среде программирования Basic Microsoft версии 4.0, 4.5, 7.1 и изучить возможность построения графических изображений.

2.2. *Постановка задачи:* В соответствии с вариантом задания построить на экране монитора заданное изображение, состоящее из отрезков прямых, дуг окружностей и точек. Построение выполнить в 9, 12 и 13 графических режимах. Фон и геометрические фигуры выполнить разными цветами

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Какие графические примитивы используются в Basic Microsoft?
2. Укажите назначение и синтаксис оператора CIRCLE?
3. Назовите отличие в работе оператора PSET и PRESET?
4. Можно ли построить с помощью оператора LINE прямоугольник?
5. Назначение оператора SCREEN?
6. Назначение оператора DRAW?
7. Как построить с помощью оператора CIRCLE сектор?
8. Произойдет ли ошибка если строящееся изображение (например, линия) выйдет за зону экрана?
9. Для чего используется оператор VIEW?
10. Назначение и синтаксис оператора WINDOW?

Таблица

Задания

n	задание	n	задание
1	В центре экрана построить равносторонний треугольник с вершиной, направленной вверх, внутрь вписать окружность, в центре которой построить точку.	16	В центре экрана построить квадрат с вершиной, направленной вверх, внутрь вписать окружность, в центре которой построить точку.
2	В центре экрана построить окружность, в которую правильный пятигранник, в вершинах которого поставить точки.	17	В центре экрана построить квадрат, его описать треугольником, в вершинах которого поставить точки.
3	В центре экрана построить правильный шестиугольник, описанный окружностью. Внутрь шестиугольника вписать еще одну окружность. В центре фигур построить точку.	18	В центре экрана построить равносторонний треугольник с вершиной, направленной вниз, внутрь вписать окружность, в центре которой построить точку.
4	В центре экрана построить пятиконечную звезду, описанную окружностью, в вершинах которой поставить точки.	19	В центре экрана построить окружность, на ней провести оси симметрии, в центре поставить точку.
5	В центре экрана построить квадрат, у которого верхнюю часть (симметричную относительно оси OX) заменить дугой окружности. В центре окружности поставить точку.	20	В центре экрана построить квадрат, к верхней грани которого достроить дугу окружности, проходящую через его вершины. Общие точки пометить другим цветом.
6	В центре экрана построить окружность, в верхнюю часть которой вписать равносторонний прямоугольный треугольник, а нижнюю часть описать	21	В центре экрана построить сегмент, соответствующий заданному центральному углу с осью симметрии, расположенной вертикально. В местах соединения прямой

	квадратом. В общих точках с окружностью поставить точки.		линии и дуги окружности поставить точки.
7	В центре экрана построить квадрат, внутрь вписать окружность, в центре которой построить точку.	22	В центре экрана построить окружность, в которую правильный шестигранник, в вершинах которого поставить точки.
8	В центре экрана построить равносторонний треугольник с вершиной, направленной вверх, к его левой грани достроить дугу окружности, в центре которой построить точку.	23	В центре экрана построить правильный шестиугольник, к его правой грани достроить дугу окружности. Точки контакта шестиугольника с дугой отметить другим цветом.
9	В центре экрана построить квадрат с вершиной, направленной вверх. К его нижней грани достроить дугу окружности, внутри которой построить точку.	24	В центре экрана построить квадрат, к верхней грани которого треугольник, проходящий через его вершины. Общие точки пометить другим цветом.
10	В центре экрана построить окружность, в которую вписать равносторонний треугольник с вершиной, направленной вверх, в центре которых построить точку.	25	В центре экрана построить квадрат, к нижней грани которого достроить дугу окружности, проходящую через его вершины. Общие точки пометить другим цветом.
11	В центре экрана построить равносторонний треугольник с вершиной, направленной вверх, его описать окружностью. Внутрь треугольника вписать окружность, в центре которой построить точку.	26	В центре экрана построить окружность, верхнюю часть описать квадратом, в нижнюю часть вписать равносторонний прямоугольный треугольник. В общих точках с окружностью поставить точки.
12	В центре экрана построить квадрат, к левой грани которого достроить дугу	27	В центре экрана построить окружность, в которую вписать равносторонний

	окружности, проходящую через его вершины. Общие точки пометить другим цветом.		треугольник с вершиной, направленной влево, в центре которых построить точку.
13	В центре экрана построить равносторонний треугольник с вершиной, направленной вверх, к его нижней грани достроить дугу окружности, в центре которой построить точку.	28	В центре экрана построить квадрат, к левой грани которого достроить треугольник, проходящий через его вершины. Общие точки пометить другим цветом.
14	В центре экрана построить окружность, в которую вписать квадрат, в центре которых построить точку.	29	В центре экрана построить пятиконечную звезду, описанную окружностью, в которой поставить точку.
15	В центре экрана построить окружность, в которую вписать равносторонний треугольник с вершиной, направленной влево. Внутрь треугольника вписать еще одну окружность. В центре фигур построить точку.	30	В центре экрана построить квадрат с вершиной, направленной вверх, его описать окружностью Внутрь квадрата вписать окружность, в центре которой построить точку.

Лабораторная работа №18
Организация обработки ошибок в процессе работы программы.

1. Обработка ошибок

Оператор формирования кода ошибки – ERROR

Назначение: Генерирует ситуацию возникновения ошибки с заданным кодом, а также позволяет пользователю определить свои коды ошибок.

Синтаксис ERROR <код ошибки>

Аргумент <код ошибки> должен быть целым числом в диапазоне от 1 до 255.

Для задания пользовательского кода ошибки рекомендуется использовать значение кода начиная с 200 и не совпадающее с имеющимися кодами или выходящее за пределы 255.

При выполнении оператора программа моделирует возникновение ошибки с соответствующим кодом и печатает сообщение об ошибке. Если аргумент функции ERROR не находит заданного кода, выдается сообщение:

Unprintable error (Неопознанная ошибка).

При наличии подпрограммы обработки ошибки (ON ERROR) происходит переход на эту подпрограмму.

Оператор установки среды обработки ошибки – ON ERROR

Назначение: При появлении ошибки передает управление в подпрограмму обработки ошибки.

Синтаксис ON ERROR GOTO [<номер строки> | <метка строки>]

Аргументы <номер строки> и <метка строки> определяют первую строку процедуры обработки ошибки. Эта строка должна обязательно принадлежать уровню модуля.

Если <номер строки> равен 0, то такой оператор запрещает обработку ошибок, а не определяет строку с номером 0 в качестве начальной строки процедуры. Возникающие после этого ошибки вызывают только печать сообщения об ошибке и прекращение выполнения программы. В том случае, когда обработка ошибок разрешена, возникающие ошибки вызывают переход на указанную процедуру обработки ошибок.

Выполнение оператора ON ERROR с номером 0 внутри подпрограммы обработки ошибок прекращает выполнение программы и выводит сообщение об ошибке прерывания. Это позволяет прекратить выполнение программы, если возникает ошибка, которая не может быть обработана указанной процедурой.

Необходимо иметь в виду, что так называемая процедура обработки ошибки не является подпрограммой, функцией DBF FN, процедурой SUB или

процедурой-функцией FUNCTION в обычном смысле. Это блок операторов, помеченный начальным номером строки или меткой.

В блоке обработки ошибок не допускается повторная обработка ошибок; ошибки, возникшие при работе блока, прекращают работу программы с выдачей сообщения об ошибке.

Функция определения кода ошибки - ERR

Назначение: Возвращает код ошибки.

Синтаксис: ERR

В случае возникновения ошибки функция ERR возвращает код ошибки. Поскольку функция ERR возвращает правильное значение только после возникновения ошибки, то эту функцию следует использовать в подпрограммах обработки ошибок для анализа ошибочных ситуаций.

Функция определения номера строки, связанной с ошибкой, - ERL

Назначение: Возвращает номер строки, в которой возникла ошибочная ситуация.

Синтаксис: ERL

В случае возникновения ошибки функция ERL возвращает номер строки, в которой возникла ошибка. Поскольку функция ERL возвращает правильное значение только после возникновения ошибки, то эту функцию следует использовать в подпрограммах обработки ошибок для анализа ошибочных ситуаций.

Функция ERL возвращает только номер строки, но не ее метку, которая расположена перед строкой, вызвавшей ошибку. Если программа не пронумерована, то функция ERL всегда возвращает 0.

2.Описание практической части работы*:*

2.1. *Цели лабораторной работы:* Изучить возможности корректировки ошибок, возникающих в процессе выполнения программы.

2.2. *Постановка задачи:* Разработать программу, содержащую блок обработки ошибок неправильных действий пользователя, обеспечивающий вывод информации на печать и в рабочий файл на диск А.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу:

2.3.3.1. Разработать алгоритм вывода k значений функции y=f(x) на интервале $[a,b]$ и алгоритм процедуры обработки ошибок.

2.3.3.2. Разработать программу, в которой установить начало обработки ошибок оператором ON ERROR GOTO $\langle$метка$\rangle$. $\langle$Метка$\rangle$ указывает на начало блока обработки ошибок.

2.3.3.3. Сгенерировать ошибку и определить код ошибки с помощью оператора PRINT в блоке обработки ошибок.

2.3.3.4. Произвести обработку ошибок оператором SELECT CASE ERR.

2.3.3.5. Блок обработки ошибок закончить оператором RESUME.

2.3.1. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
-текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

-значения, полученные в результате выполнения программы

-анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Зачем нужна обработка ошибок в программе ?

2. Можно ли избежать меток в программе, если в ней предусматривается обработка ошибок ?

3. Как определить какая ошибка произошла ?

4. Как определить где произошла ошибка ?
5. Как определить на каком устройстве произошла ошибка 7
6. Как сказывается на работе программы включение обработки ошибок ?
7. Назначение оператора RESUME при обработке ошибок ?
8. Сколько блоков обработки ошибок может быть в программе ?
9. Каким оператором осуществляется обработка ошибок ?
10. Можно ли на время отключить в программе обработку ошибок ?

Лабораторная работа №19
Организация прерываний в среде BASIC MICROSOFT.

1. Операторы организации прерываний в среде Basic Microsoft

Оператор установки и управления отображением функциональных клавиш – KEY(n), KEY LIST,

KEY {ON, OFF}

Назначение: Присваивает клавише строку символов и может отображать значения клавиш.

Синтаксис: KEY n, <строковое выражение>

KEY LIST

KEY ON

KEY OFF

Аргумент n определяет номер функциональной клавиши. Допустимыми являются номера с 1 по 10, а также 30 и 31 для функциональных клавиш F11 и F12 расширенной 101-клавишной клавиатуры. Аргумент <строковое выражение> - это текстовая строка, состоящая не более чем из 15 символов, которые выводятся в нижнюю строку экрана при нажатии на соответствующую функциональную клавишу. Если строка содержит более 15 символов, лишние игнорируются.

Управление отображением значений функциональных клавиш реализуется с помощью операторов KEY ON, KEY OFF и KEY LIST:

Оператор	*Действие*
KEY ON	На нижней строке экрана выводятся первые 6 символов строки, соответствующей каждой функциональной клавише
KEY OFF	С нижней строки экрана удаляются все обозначения функциональных клавиш, и она освобождается для выдачи программной информации. Значения клавиш остаются прежними
KEY LIST	На экран выводятся 15-символьиые значения всех функциональных клавиш

При включенном отслеживании событий ON KEY обращается к подпрограмме каждый при нажатии клавиши.

0	Все перечисленные здесь клавиши KEY(0) ON, KEY(0) OFF и KEY(0) STOP
1-10	Функциональные клавиши F1-F10.

11	Клавиша СТРЕЛКА ВВЕРХ.
12	Клавиша СТРЕЛКА ВЛЕВО.
13	Клавиша СТРЕЛКА ВПРАВО.
14	Клавиша СТРЕЛКА ВНИЗ
15-25	Определенные пользователем клавиши. Более подробно смотрите в «Объявление определенных пользователем клавиш».
30, 31	Функциональные клавиши F11 и F12.

KEY(n%) ON	Включает отслеживание событий для указанной клавиши
KEY(n%) OFF	Выключает отслеживание событий для клавиш
KEY(n%) STOP	Приостанавливает отслеживание событий. Событие обрабатывается при включении KEY ON
строка	Метка или номер первой строки подпрограммы отслеживания событий

Оператор конца процедуры обработки прерываний - RESUME

Назначение: Обеспечивает переход к продолжению выполнения программы после выполнения процедуры обработки прерываний.

Синтаксис:

RESUME [0]
RESUME NEXT
RESUME {<номер строки> | <метка строки>}

Существует несколько форм оператора RESUME.

Форма оператора	*Действие*
RESUME [0]	Переход на оператор, который вызвал прерывание
RESUME NEXT	Переход на оператор, следующий за оператором, который вызвал прерывание
RESUME <номер строки>	Переход на строку с указанным номером
RESUME <метка строки>	Переход на указанную метку

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить возможность управления процессом работы программы с использованием функциональных клавиш.

2.2. *Постановка задачи:* Разработать программу генерирования случайных чисел в заданном диапазоне, используя прерывание от функциональных клавиш.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Для чего можно использовать прерывание от нажатия клавиши ?
2. Для чего предназначены клавиши F1...F12 на клавиатуре ?

3. Каким оператором можно вывести список функциональных клавиш ?

4. В какой строке выводится строка-подсказка обозначения функциональных клавиш ?

5. Сколько функциональных клавиш выводит оператор ON KEY ?

6. Сколько функциональных клавиш содержит список, выводимый с помощью оператора LIST KEY ?

7. Можно ли приостановить прерывание от функциональных клавиш ?

8. Каким оператором можно погасить строку вывода названия функциональных клавиш ?

9. Можно ли вывести информацию оператором PRINT в строку, в которой расположена строка названия функциональных клавиш ?

10. Где располагается блок обработки прерываний от нажатия клавиш ?

Лабораторная работа №20
Использование библиотеки интерфейса для создания вертикального меню.

1. Описание вызываемой процедуры

DECLARE SUB menuSV0 (cen%, vid%, a$(), yp1%, xp1%, lpol%, otst%, l%, t$, ten%, cSimPol%, cFonPol%, cSimm%, cFonm%, cSimK%, cFonK%, cod%)
'

Подпрограмма формирования вертикального меню выбора:

cen% - центрировать =1, иначе не центрировать
vid% - 0 - статичное меню, 1 - исчезающее меню
a$(0) - заголовок пунктов меню
a$() - массив пунктов меню
xp1% - координата левого
yp1% - верхнего угла меню
lpol% - отступ поля до рамки 0,1,2,...
otst% - отступ от верха рамки до первого пункта меню
l% - отступ слева и справа от рамки до поля меню
t$ - тип рамки
cFonPol% - цвет фона поля меню
cSimPol% - цвет рамки и шапки меню
cFonm% - первоначальный цвет фона пункта меню
cSimm% - первоначальный цвет букв пункта меню
cFonk% - цвет фона курсора меню
cSimk% - цвет букв курсора меню
ten% - вид тени 0,1,2,3,4
cod% - входной параметр : номер пункта меню установки курсора
cod% - выходной параметр : номер выбранного пункта меню (0 - <Esc>)

Описание массива пунктов меню

DIM a$(5) Присвоение значений элементов пунктов меню

```
a$(0) = " menu: "
a$(1) = "1111"
a$(2) = "2222"
a$(3) = "3333"
a$(4) = "4444"
a$(5) = "5555"
```

Пример вызова процедуры:

```
CALL menuSV0(1, 1, a$(), 5, 10, 1, 0, 0, "1", 1, 14, 1, 14, 1, 15, 4, cod%)
PRINT cod%
```

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить возможность организации выбора действий в программе с использованием вертикального меню.

2.2. *Постановка задачи:* Используя вертикальное меню организовать:

- ввод одномерного массива - с клавиатуры, с файла;
- сортировку - по возрастанию, по убыванию, не сортировать;
- вывод - на экран, на печать, в файл.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Что такое интерфейс ?
2. Как выглядит вертикальное меню ?
3. Для каких целей можно использовать вертикальное меню ?
4. Сколько параметров задается при вызове вертикального меню ?
5. Какой параметр возвращается из подпрограммы при вызове вертикального меню ?
6. Можно ли изменить цвет фона меню ?
7. Можно ли сделать меню всплывающим и исчезающим и для каких целей его можно использовать ?
8. В каком режиме работает вертикальное меню ?
9. Для решения каких задач удобнее использовать систему вертикальных меню ?
10. Что нужно организовать в программе, чтобы вернуться к предыдущему меню ?

Таблица задания

номер варианта	задание	номер варианта	задание
1		16	
2		17	
3		18	
4		19	
5		20	
6		21	
7		22	
8		23	
9		24	
10		25	
11		26	
12		27	
13		28	
14		29	
15		30	

Лабораторная работа №21
Использование библиотеки интерфейса для создания горизонтального меню.

Теория:

DECLARE SUB menuSV0 (cen%, vid%, a$(), yp1%, xp1%, lpol%, otst%, l%, t$, ten%, cSimPol%, cFonPol%, cSimm%, cFonm%, cSimK%, cFonK%, cod%)

Описание вызываемой процедуры:

DECLARE SUB menuG (a$(), ym%, k$(), yk%, tip%, cod%)

Подпрограмма горизонтального меню выбора 03.06.94

a$() - массив пунктов меню

ym% - номер строки горизонтального меню

k$() - массив контекстной подсказки по каждому пункту меню

yk% - номер строки контекста

tip% - тип цветовой гаммы 0,1,2,3,4,5,6

cod% - выходной параметр : номер выбранного пункта меню (0 -ESC)

Описание массива пунктов меню

```
DIM a$(5), k$(5)
DIM a1$(5), a2$(5), a3$(5), a4$(5), a5$(5)
Присвоение значений элементов пунктов меню

a$(0) = " menu: "
a$(1) = "1111"
a$(2) = "2222"
a$(3) = "3333"
a$(4) = "4444"
a$(5) = "5555"

k$(0) = " Контекстная помощь "
k$(1) = "1111    ooooooooooooooooooooooooooooooooooooooooooo"
k$(2) = "2222        ьоооооооооооооооол"
k$(3) = "33пррррррррррр  ррррррррррр33"
k$(4) = "4нннннннннннннн пррррло енллллллллллллллллллл444"
k$(5) = "55нелллллл енл ле55"

a1$(0) = " menu1: "
a1$(1) = "gggggg1111"
a1$(2) = "ggggggggg2222"
```

```
a1$(3) = "ggggggggggg3333"
a1$(4) = "ggggggggggg4444"
a1$(5) = "ggggg5555"

a2$(0) = " menu2: "
a2$(1) = "ghnnffgnbfgg1111"
a2$(2) = "gggggnfgngggg2222"
a2$(3) = "gggggggggfdnfdngggg3333"
a2$(4) = "gggggdnbdfggggg4444"
a2$(5) = "ggggg5555"

a3$(0) = " menu3: "
a3$(1) = "dhfdnbfgnf"
a3$(2) = "ggggggggg2222"
a3$(3) = "gggrthrtggggggggg3333"
a3$(4) = "ggggggggggg4444"
a3$(5) = "ggggg5555"

a4$(0) = " menu5: "
a4$(1) = "ggggggg1111"
a4$(2) = "ggggggggg2222"
a4$(3) = "ggggerherherggggggg3333"
a4$(4) = "ggggggggggg4444"
a4$(5) = "ggggg5555"

a5$(0) = " menu5: "
a5$(1) = "ggggggg1111"
a5$(2) = "ggggggggg2222"
a5$(3) = "ggggggggggg3333"
a5$(4) = "ggggggggggg4444"
a5$(5) = "ggggg5555"
```

Пример вызова процедуры:

```
DO

CALL menuG(a$(), 1, k$(), 25, 1, cod%)

COLOR 0, 3
LOCATE 3, 10
```

```
PRINT "   Номер выбранного пункта меню  cod%="; cod%
LOCATE 4, 10
PRINT "   Для завершения программы нажмите ESC !!!"

SELECT CASE cod%

      CASE 1
COLOR 0, 3
CLS
LOCATE 12, 10
PRINT "   Пример вызова процедуры вертикального меню"

CALL menuSV0(1, 1, a1$(), 5, 10, 1, 0, 0, "1", 1, 14, 1, 14, 1, 15, 4, cod%)

      CASE 2
CALL menuSV0(1, 1, a2$(), 10, 20, 1, 0, 0, "1", 2, 14, 2, 14, 1, 15, 4, cod%)

      CASE 3
CALL menuSV0(0, 0, a3$(), 15, 30, 1, 0, 0, "1", 3, 14, 4, 14, 1, 15, 4, cod%)

      CASE 4
CALL menuSV0(1, 1, a4$(), 5, 50, 1, 0, 0, "1", 4, 14, 5, 14, 1, 15, 4, cod%)

      CASE 5
CALL menuSV0(0, 0, a5$(), 15, 10, 1, 0, 0, "1", 0, 14, 6, 14, 1, 15, 4, cod%)

      CASE ELSE

END SELECT

LOOP UNTIL cod% = 0

   LOCATE 25, 1
   PRINT "  Нажми любую клавишу !!!   ";

c$ = INPUT$(1)
```

2.Описание практической части работы:

2.1. *Цели лабораторной работы* Изучить возможность организации выбора необходимых для работы действий в программе с использованием системы основного горизонтального меню и вспомогательных вертикальных.

2.2. *Постановка задачи* Организовать обработку информации в соответствии с заданием, выданным преподавателем. В работе использовать интерфейс, состоящий из системы меню.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Что такое интерфейс ?
2. Как выглядит вертикальное меню ?
3. Для каких целей можно использовать вертикальное меню ?

4. Сколько параметров задается при вызове вертикального меню ?

5. Какой параметр возвращается из подпрограммы при вызове вертикального меню ?

6. Можно ли изменить цвет фона меню ?

7. Можно ли сделать меню всплывающим и исчезающим и для каких целей его можно использовать ?

8. В каком режиме работает вертикальное меню ?

9. Для решения каких задач удобнее использовать систему горизонтального меню ?

10. Что нужно организовать в программе, чтобы вернуться к предыдущему или горизонтальному меню?

Лабораторная работа №22
Запуск исполняемых файлов с ключом.

1. Функция вызова командной строки запуска программы - COMMAND$

Назначение: Возвращает командную строку, из которой осуществлялся запуск программы.

Синтаксис: COMMANDS

Функция возвращает всю командную строку, начиная с символа следующего после имени запускаемой программы. Командная строка может содержать список параметров, разделенных пробелами или символами табуляции. Функция пересылает все значимые пробелы из командной строки и преобразует строчные буквы (a-z) в прописные (A-Z). COMMAND$ можно использовать и в программах, запускаемых в среде QuickBASIC. В этом случае необходимо при старте QuickBASIC указать параметр cmd или изменить содержимое командной строки, вызвав подменю Modify COMMANDS из меню RUN.

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить возможность организации работы программы в нескольких режимах в зависимости от заданного параметра (ключа).

2.2. *Постановка задачи* Разработать программу и создать исполняемый файл , позволяющий работать в нескольких режимах в соответствии с заданным ключом. В случае неправильного ввода ключа, вывести на экран монитора полный список правильных ключей.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1. Что такое исполняемый файл ?
2. Зачем нужен ключ при запуске программы ?
3. Сколько может быть ключей в программе ?
4. Каким оператором можно определить в программе значение ключа?
5. Как задать значение ключа в среде программирования Basic Microsoft ?
6. Каким образом программа реагирует на ввод ключа ?
7. Что происходит с программой если она вводится с ключом, а ключ в ней не предусмотрен ?
8. Что происходит с программой если ввести неправильный ключ ?
9. Какими операторами обрабатываются ключи в программе ?
10. Может ли ключ быть числовым значением ?

Лабораторная работа №23
Получение и обработка растровых изображений в редакторе Photo Shop .

1. Советы по работе с Adobe Photoshop:

Увеличить или уменьшить размер выделенной области можно при помощи команды **Expand / Contract (Расширить / Сжать)** из меню **Selection (Выделение).**

Нажав клавиши **Ctrl+Del** Вы легко зальете ее фоновым цветом, а вот нажатие **Alt+Del** приводит к заливке основным цветом.

Но если при этом Вам вдруг захочется нажать еще и клавишу **Shift**, то заливка произойдет в режиме **Preserve Transparency (Сохранить прозрачность)**. Уверяю Вас - с клавиатуры делать такие операции гораздо быстрее и удобней.

Если же к Вам закралось подозрение, что иллюстрация страдает цветовыми сдвигами, тогда воспользуйтесь пипеткой из окна **Levels** и укажите ею точку изображения, которая, по Вашему мнению, должна быть нейтрально серой.

Кстати, в большинстве диалоговых окон нажатие клавиши **Alt** приводит к замене кнопки **Cancel** на кнопку **Reset**. Таким образом, Вы легко вернетесь к первоначальным настройкам, не закрывая окно.

Когда Вы выделяете круглую область, то с клавишей **Alt** она строится от центра. С клавишей **Shift** у Вас получится круг. Одновременное **нажатие Shift+Alt** позволяет построить абсолютно круглое выделение с заданным центром. И это очень даже приятно.

Когда отсканированный образец слегка перекошен, не огорчайтесь. Просто постройте линию с шириной ноль пикселов вдоль верхнего края скана и посмотрите какой угол будет отображен в палитре **Info**. Затем в диалоговом окне **Numerical Transform (Трансформирование \ Числовые значения)** поставьте полученное значение угла и конечно же, не забудьте нажать на всенезабвеннейшую кнопочку **OK**!

Я понимаю, что курсоры должны быть разные, но не спешите каждый раз исправлять настройки редактора (**Preferences**), просто иногда нажимайте клавишу **Caps Lock** и курсор всенепременно будет чуток к Вашему настроению.

Нажав клавиши **Ctrl+Alt** и щелкнув мышью на каком-нибудь из каналов, можно вычесть его содержимое из текущей выделенной области.

При построении контуров, пожалуйста, не забывайте об опции **Rubber Band (Гибкое соединение)**. Она позволит Вам делать Ваши кривые более эластичными.

Если Ваше творение готово для печати, не забудьте проверить его на самый-самый черный цвет. Параметры для него в режиме **CMYK** следующие: **65C, 55M, 50Y, 95K**.

Опять же, если Вы готовите (я не побоюсь этого слова) картину для репродуцирования в печатных органах и если в ней Вы используете градиентные заливки, то оные стоит делать только в режиме **CMYK**.

Попробуйте и все поймете.

Когда Вы создаете новый слой, - по нажатию на пиктограмме **New Layer (Новый слой)**, попридержите клавишу **Alt**. И Вы легко и сразу сможете обозвать его по-своему.

Двойной щелчок почти по всем кнопкам на панели инструментов вызывает проявление панели с их параметрами. Вышеуказанное действие над кнопочкой **Zoom** позволяет увидеть картинку «без прикрас», то есть при увеличении один к одному. А вот инструмент **Hand Tool (Рука)** разворачивает изображение на все окно.

Если вдруг Вы забудете размеры своего детища, то нажмите левую кнопку мыши в информационном поле в нижней части основного экрана редактора и одновременно давите на одну из клавиш-модификаторов **(Ctrl или Alt)**. Я думаю, Вы сразу все вспомните!

А вот разновидности инструмента **Pen Tool (Перо)** и использование клавиш-модификаторов, как то **Alt, Ctrl и Shift** , Вы наглядно можете уяснить себе из следующей таблички:

Перо:
Ctrl - стрелка
Alt - угол
Ctrl+Alt - выделение всех точек
Стрелка:
Alt - выделение всех точек
Ctrl+Alt - угол

Запомните навсегда - нажатая клавиша **Shift** при работе с выделяющими инструментами приводит к объединению выделений, а вот при работе с каналами и слоями я бы порекомендовал держать и клавишу **Ctrl**.

Свободного пространства на диске у Вас должно быть минимум **в пять раз больше**, чем размеры изображений, которые Вы Обычно редактируете.

А впрочем, давайте не будем о грустном, давайте о легком. Вот, например, кисти и другие рисующие инструменты. Вам достаточно нажать на любую цифровую клавишу и непрозрачность Вашего инструмента в процентах будет определена. Но не забудьте, цифра ноль устанавливает 100-процентную непрозрачность.

Если Вам вознадобится распечатать плоды своего творчества, то первоначально откройте диалоговое окно **Image Size (Размер изображения)**, отключите режим **Resample Image (Интерполяция)** и установите значение разрешения для Вашей картинки. Затем снова включите режим **Resample Image** и введите желаемые размеры печатного оттиска.

Не часто, но бывает и так, что Вам необходимо не уменьшить, а увеличить изображение. Чтобы не потерять резкость, откройте окно **General Preferences** и замените режим интерполяции на **Nearest Neighbour**. Главное - это потом не забудьте вернуть прежние настройки.

Как Вы знаете, фильтр **Lens Flare (Блик)** на пустом слое не работает, но есть неплохой выход. Для этого залейте новый слой черным цветом и примените к нему фильтр **Lens Flare**. Затем установите для этого слоя режим наложения **Screen (Осветление)**. После этого Вы легко сможете подобрать положение, размер и интенсивность своего блика.

Если компоновка и количество окон палитр на экране Вас не устраивает, Вы легко можете поправить положение, перетаскивая палитры из одного окна в другое или просто удаляя с экрана кнопочкой **Close (закрыть)**.

Если при нажатой клавише **Alt** вы нажмете кнопку **Loads Path as a selection** в нижней части палитры **Path**, то кроме обычного превращения контура в выделение Вы сможете произвести операции по объединению, вычитанию и так далее нового выделения и уже существующего на экране.

Для экономии памяти компьютера, а также своего времени чаще пользуйтесь инструментом **Crop**. Кстати область кадрирования можно легко поворачивать.

Иногда пунктирная линия, обозначающая выделенную область, очень мешает увидеть некоторые детали изображения. Не беда. Одновременно нажмите клавиши **Ctrl+H**. Эта операция работает даже когда у Вас уже открыто какое-либо диалоговое окно или окно фильтра.

Если выделенную область Вам необходимо преобразовать в новый слой, то просто нажмите клавиши **Ctrl+J**.

Когда Вы пользуетесь фильтрами **Distortion (Искажение)**, большие деформации объектов получаются более качественными, если несколько раз применить один и тот же фильтр, но с меньшими искажениями. Нажатие клавиш **Ctrl+F** приводит к повторному действию фильтра с уже установленными параметрами.

Если у Вас появится желание создать собственную форму кисти, нет ничего проще! Выделите требуемый объект и выберите из меню палитры **Brush (Кисти)** команду **Define Brush (Определить кисть)**.

Для большинства функций программы Photoshop предусмотрены клавиатурные сокращения. Я не стану Вам их перечислять. Вы просто запустите

Photoshop и выберите **Keyboard (Клавиатура)** в меню **Help (Помощь)**. И еще хочу дать один важный совет - как можно больше используйте клавиатуру. Работа с ее помощью идет гораздо быстрее, чем с мышкой.

И еще о клавиатуре. С помощью курсорных клавиш Вы легко можете перемещать слои или объекты изображения. То же самое происходит, если у Вас выделено числовое поле в каком-либо диалоговом окне. Нажатие клавиши **Shift** увеличивает шаг смещения в **10** раз.

Во время клонирования при включенном параметре **Sample Merged (Совмещенные данные)** следует отключать корректирующие слои и включать после окончания этой операции, поскольку Вы дублируете эффект с корректирующим слоем. А это, скорее всего, не есть хорошо.

В одной из предыдущих лекций я говорил, а сейчас хочу повторить снова. Если какой-либо объект слоя Вам необходимо сильно уменьшить, то скопируйте объект в отдельный файл, уменьшите изображение до необходимых размеров, примените фильтр **Unsharp Mask** и только после этого верните объект на исходную картинку. Уверяю Вас, качество будет значительно выше!

Храните свои выделенные области как контуры. И места на диске практически не занимают и по именам легко узнаются и хорошо экспортируются в Adobe Illustrator.

Чтобы скопировать маску одного слоя на другой, нажмите клавишу **Ctrl** и щелкните на исходной маске, превратив ее в выделение. Затем перейдите на нужный слой и щелкните на пиктограмме **Add Layer Mask** в палитре **Layers**.

Если же в момент создания нового корректирующего слоя **(Adjustment Layer)** у Вас будет выделена какая-либо область, то она будет вставлена в маску нового корректирующего слоя.

Вы легко можете рисовать на корректирующем слое и таким образом маскировать те области, которые хотите предохранить от редактирования.

Для перевода изображений в градации серого цвета существует множество способов, но, пожалуй, наилучшим из них является следующий: сначала переведите изображение в режим **Lab Color**, затем в палитре **Channels (Каналы)** выделите канал **Lightness (Яркость)** и переведите изображение в режим **Grayscale**. На вопрос **Discard other channels? (Удалить остальные каналы?)** ответьте утвердительно.

Нажав клавишу **Alt** и щелкнув мышью на каком-либо слое, когда изменится курсор, Вы создадите так называемую макетную группу **(Clipping group)**. В результате маска прозрачности нижнего слоя в группе станет маской для верхнего и всех промежуточных слоев.

Вы знаете, именование слоев собственными именами является правилом хорошего тона. Заведите и Вы себе такую привычку. По крайней мере, тогда Вы сможете отменить отображение содержимого слоев в палитре **Layers**, что вдвое

увеличит количество отображаемых в палитре слоев и повысит быстродействие работы программы.

Палитра **Navigator (Навигатор)** также сможет ускорить Вашу работу с большими иллюстрациями, поскольку позволяет Вам быстро переходить от одного участка изображения к другому.

Я бы Вам посоветовал забыть о таких командах как **Scale, Rotate, Scew и Distort,** ибо все они доступны при использовании **команды Free Transform (свободная трансформация)**, да и вызывается она легко и быстро по клавишам **Ctrl+T**.

Если в работе Вы используете образец цвета **Pantone**, то при наведении пипеткой на этот цвет в палитрах **Color** или **Swatches** в заголовке окна отобразится номер по каталогу **Pantone**.

После применения какого-либо фильтра Вы можете ослабить его действие и изменить режим наложения, если выберете команду **Fade Filter (Ослабление)**.

Если на Вашем компьютере установлена специализированная программа управления шрифтами, например Adobe Type Manager, и если во время работы в Photoshop Вы изменили коллекцию шрифтов на своем компьютере, то Вы легко сможете обновить библиотеку шрифтов в Adobe Photoshop без перезагрузки программы. Для этого нужно нажать на клавишу **Shift** и щелкнуть мышью на инструменте **Text (текст)**.

Если Вам нужно объединить несколько слоев в один, то сначала свяжите их (значок цепочки в палитре **Layer**), а затем в этой же палитре либо выберите команду **Merge Linked (Сгруппировать связанные)**, либо просто нажмите клавиши **Ctrl+E**.

Очень часто активные экранные палитры мешают увидеть некоторые детали изображения. На этот случай имеется клавиша **Tab**. С помощью нее Вы можете убирать с экрана все палитры, когда Вам заблагорассудится.

Если же Вы воспользуетесь сочетание клавиш **Shif+Tab**, то на экране останется только панель инструментов, если, конечно, не считать самой иллюстрации.

Прервать надоевшую Вам операцию, когда на экран не выдается кнопка **Stop**, очень даже просто. Нажмите кнопочку **Esc**.

Когда ваше изображение изобилует деталями и слоями и Вы не помните, что и где находится, не отчаивайтесь, подведите курсор к интересующему Вас объекту, нажмите клавишу **Ctrl** и щелкните правой кнопкой мышки. Перед Вами появится меню, где перечисляются все слои, влияющие на изображение в данной точке. Щелчок на названии приводит к переходу на этот слой. Без посредства меню на слой можно перейти, если при вышеназванных условиях Вы держите еще и клавишу **Alt**.

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Изучить принципы построения и обработки растровых изображений в редакторе Photo Shop .

2.2. *Постановка задачи:* В соответствии с вариантом задания построить и обработать растровое изображение в редакторе Photo Shop.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

-значения, полученные в результате выполнения программы

-анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1.Что такое растровое изображение ?

2.Что такое атрибуты файла ?

3. Файлы с каким расширением можно обрабатывать в Fotoshop ?

4.От чего зависит размер файла графического изображения ?

5.Можно ли в программе Fotoshop получить из цветного изображения черно-белое и наоборот ?

6.Можно ли получить негатив цветного и черно-белого изображения ?

7.Можно ли объединить два изображения в одном ?

8.Что такое слой при построении изображения в Fotoshop ?

9.Сколько слоев можно использовать в редакторе Fotoshop ?

10.Если изображение состоит из нескольких слоев, в каком формате его можно сохранить ?

Лабораторная работа №24
Обработка текста с помощью текстового процессора Word.

1. *Средства обработки текстовой информации*

Средства данного класса являются одной из составляющих компонент автоматизации учрежденческой деятельности в самом широком смысле ее понимания, позволяя пользователю с максимальными удобствами создавать высококачественные документы различного назначения посредством, в первую очередь, ПК и соответствующего ПО. В настоящее время существует достаточное разнообразие подобных систем, начиная от простейших текстовых Редакторов, ориентированных на работу с простыми ПК, и кончая специализированными издательскими системами, обеспечивающими работу большого издательского коллектива с использованием режима телеобработки и других современных издательских технологий.

Однако средства обработки текстовой информации, несмотря на их весьма широкий спектр, обеспечивают следующие основные функции:

- создание и редактирование текстового документа, включая:

вставки, удаления, копирование, перемещение текста в документе, поиск и замену элементов документа, добавление в документ текстовой и графической информации;

- форматирование и распечатку документов с выбором размеров бумаги и форматов, а также с указанием числа копий и выводимой части документа;

- выравнивание документа и/или его отдельных частей по указанным границам с автоматической обработкой переносов строк;

- возможность создания документа по стандартному шаблону;

- использование различных шрифтов распространенных алфавитов и т.д.;

- *размещение* в документе таблиц, диаграмм, рисунков и т.д., а также ряд других функций, состав которых определяется *уровнем* и *назначением* конкретного ПС. В настоящем разделе кратко остановимся на текстовых процессорах, широко используемых на IBM-совместимых ПК для организации обработки различного рода текстовой информации, акцентируя внимание на наиболее популярных пакетах — *Ms Word 6.0* , *Word 7.0, Word 97* фирмы *Microsoft*

Элементы работы в среде пакета Ms Word 6.0

1. *Общие сведения.* Обработка ***текстовых*** документов — наиболее массовый вид прикладной деятельности на ЭВМ.

Пакет *Ms Word 6.0* (в дальнейшем просто *Word* или *пакет)* представляет собой относительно более ранних средств данного типа достаточно большую программную систему.

Перед использованием Word рекомендуется закрывать ненужные при работе с ним Windows-приложения и удалять из ОП резидентные программы, например Norton Commander.

2. *Главные окно и меню пакета Word.* После активации пиктограммы ***Word*** двойным щелчком мыши производится загрузка резидентной части пакета с выходом на его главное окно. Главное окно пакета содержит четыре основных поля:

- первая строка содержит имя пакета и имя активного в данный момент документа, а также кнопки для управления режимом визуализации окна (увеличения, уменьшения, сведения к пиктограмме), завершения работы с пакетом и др.;

- вторая строка содержит главное меню пакета (ГМП) и управляющие кнопки, аналогичные упомянутым; детальное их описание можно найти в литературе по пакету или Windows;

- третье поле представляет собой собственно рабочее окно текущего документа, через которое пользователь не только имеет возможность просматривать весь документ, но и проводить сам процесс создания/редактирования его;

- наконец, четвертая строка содержит справочную информацию по: текущему документу (номер просматриваемой страницы, общее число страниц, местоположение курсора), текущему времени, используемым функциям пакета и др.

Дополнительно к указанным рабочее окно документа обрамляется рамками, содержащими указатели, позволяющие посредством активации их мышью перемещать окно по текущему документу.

Главное меню пакета (ГМП) содержит 9 групп функций. Для выбора требуемой функции пакета активируется одинарным щелчком мыши соответствующая группа ГМП, раскрывая подокно, содержащее список функций.

Прежде всего, средства Help-группы предоставляют не только справочную информацию по всем возможностям пакета, но и предлагают демонстрационные примеры, позволяющие лучше уяснять основные возможности пакета и принципы их использования.

Группа *Window* содержит две секции: (1) функции *(New Window, Arrange All, Split)* реорганизации окон пакета, включая разделение окон на подокна и создание новых окон с одинаковыми документами; и (2) *список документов,* находящихся в рабочей области пакета и к которым можно легко обращаться, активируя их окна через соответствующие элементы данного списка. При необходимости перенести некоторый блок информации из неактивного документа открывается Window- подокно, в списке документов (вторая секция)

выбирается мышью нужный документ с его активацией и открытием окна и в нужном месте выбирается необходимый блок информации. Посредством Copy-функции *(Ctrl+C)* группы Edit выбранный блок копируется в *системный буфер обмена* (СБО; *Clipboard),* вновь через список документов Window- группы активируется предыдущее окно (документ) и по .Paste-функции (Ctrl+V) Edit-группы в нужное место документа копируется хранящаяся в СБО информация.

Средства File-группы ГМП разбиты на 6 секций по своему функциональному назначению. Первая секция содержит функции: создания нового (New; Ctrl+N), открытие существующего (Open; Ctrl+0) и закрытие текущего (Close) документа. При открытии существующего файла открывается Open-окно, поля и переключатели которого позволяют выбирать нужный файл из предлагаемого списка, обеспечивать поиск нужного файла, определять его формат по расширению имени и т.д. Вторая секция содержит функции сохранения: текущего документа с обновлением исходного файла (Save, Ctrl+S), текущего документа в новом файле и/или в новом формате (Save As) и всех открытых файлов и макросов (Save All). По функции Save As предоставляется возможность не только сохранять текущий документ согласно новому спецификатору файла, но и в нужном формате, определяемом списком из 23 допустимых, включая txt- и WordPerfet-форматы. Средства третьей секции группы обеспечивают: нахождение нужного файла по его спецификатору (find file); вывод краткой справки по текущему файлу, если она для него была определена (Summary Info), и определение стандартных установок для документа (Templates). Средства четвертой секции связаны с обеспечением вывода текущего документа на печать.

По функции ***Print Preview*** предоставляется возможность постранично просматривать файл, отслеживая его форматирование относительно заданных выходных параметров: размеры страницы и полей, расположение нумерации, сносок, типа принтера и др. По функции Page Setup открывается окно, поля которого позволяют определять размеры выходной страницы текущего документа при выводе на печать, поля и режим расположения документа на странице, нумерации страниц и т.д. По функции Print (Ctrl+F) открывается стандартное для Windows Print-окно, позволяющее определять режим непосредственного вывода документа на печать (весь документ или отдельные его страницы, тип принтера, количество копий и т.д.).

Средства *пятой* секции File-группы (Add Routing Slip, Send) обеспечивают отправку подготовленного документа Email-почтой удаленному адресату через системный Mail-сервер, а единственная Exit-функция шестой секции группы обеспечивает выход из среды пакета в Windows-среду; при этом запрашивается санкция на сохранение текущего документа (Yes), без сохранения (No) и отмену выхода (Cancel) из пакета.

3. ***Создание, редактирование и печать документов.***

После активации пиктограммы *Word* выходит на главное окно пакета, содержащее ГМП. Если до того пакет не использовался в требуемых пользователю условиях, необходима его предварительная настройка, основные элементы которой сводятся к следующему. Прежде всего, определяется используемый шрифт; из практических соображений с учетом сделанных предположений рекомендуется выбирать шрифт NTHetvetica/Cyrillic размера 11 и Normal-стиля. Делается это по рассмотренной выше Font-функции Format-группы ГМП, а по Default-кнопке данная установка сохраняется в пакетном файле normal.dot, делая ее глобальной для всего последующего использования пакета. Установка отображается, как правило, и в строке-меню окна пакета.

После этого определяется режим нумерации страниц создаваемого документа (если таковая требуется), что обеспечивают средства функции Page Numbers группы Insert. При этом стиль оформления нумерации можно определять посредством Style- функции Format-группы ГМП. Формат выходной страницы твердой копии документа определяется по функции Page Setup группы file ГМП, открывающей соответствующее окно с опциями и кнопками. По опции Paper Size выбирается формат А4 (210х297 мм), а по Margims-опции определяются размеры полей Top= Bottom= Left=Right=2.5 см и расстояние от нижнего края страницы нумерации — Footer=1.3 см. Сделанные установки по Default-кнопке определяются глобальными с сохранением их в пакетном файле normal.dot.

Несколько сложнее обстоит дело при необходимости импортировать объекты, подготавливаемые в среде приложений MS-DOS. В этом случае выполняется, например, следующая цепочка операций: текущий документ сводится к пиктограмме, производится отложенный выход из Widows в среду MS-DOS и запускается необходимое приложение, в среде которого создается необходимый объект. Полученный объект визуализируется на экране либо сохраняется в файле формата, допускающего последующее использование его в Word-среде. В первом случае содержимое экрана по PrintScreen- клавише копируется в СБО и производится возврат по Exit-команде в Windows-среду, а затем и в Word-среду. Данный подход наиболее приемлем для объектов графического типа. Во втором случае созданный файл с объектом импортируется посредством рассмотренных средств Insert-группы ГМП. В случае необходимости использования русскоязычных документов формата MS-DOS в среде Word оригинальной версии необходима предварительная конвертация их в формат Windows, что выполняет, например, утилита Salcombe.

2.Описание практической части работы*:*

2.1. *Цели лабораторной работы:* Изучить принципы обработки текста с помощью текстового процессора Word.

2.2. *Постановка задачи:* В соответствии с вариантом задания создать и обработать текстовый документ с помощью текстового процессора Word.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*

- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*

1.Как осуществить автоматическую правку текста ?

2.Как осуществить сортировку списка найденных файлов ?

3.Как осуществить привязку специальных свойств к содержимому текущего файла ?

4.Как изменить формат абзаца ?

5.Как в текстовом процессоре Word изменить угол дуги ,

6.Как в текстовом процессоре Word изменить цвет линии, дуги или обрамления ?

7.В чем заключается группировка графических объектов ?

8.Как в текстовом процессоре Word найти нужную запись в большой таблице ?

9. Как в текстовом процессоре Word создать оглавление ?

10. Как в текстовом процессоре Word произвести слияние основного документа с источником данных ?

Лабораторная работа №25
Создание реляционной базы данных в DBU.

1. Свойства баз данных

Обычно с базами данных работают две категории исполнителей.

Первая категория – *проектировщики.* Их задача состоит в разработке структуры таблиц базы данных и согласовании ее с заказчиком.

Вторая категория исполнителей, работающих с базами данных, – *пользователи.* Они получают исходную базу данных от проектировщиков и занимаются ее наполнением и обслуживанием.

Соответственно, система управления базами данных имеет два режима работы: *проектировочный* и *пользовательский.* Первый режим предназначен для создания или изменения структуры базы и создания ее объектов. Во втором режиме происходит использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

В мире существует множество систем управления базами данных. Несмотря на то что они могут по-разному работать с разными объектами и предоставляют пользователю различные функции и средства, большинство *СУБД* опираются на единый устоявшийся комплекс основных понятий. Это дает нам возможность рассмотреть одну систему и обобщить ее понятия, приемы и методы на весь класс *СУБД.*

База данных – это организованная структура, предназначенная для хранения информации.

Это утверждение легко пояснить, если, например, рассмотреть базу данных крупного банка. В ней есть все необходимые сведения о клиентах, об их адресах, кредитной истории, состоянии расчетных счетов, финансовых операциях и т. д. Доступ к этой базе имеется у достаточно большого количества сотрудников банка, но среди них вряд ли найдется такое лицо, которое имеет доступ *ко всей базе полностью* и при этом способно единолично вносить в нее произвольные изменения. Кроме данных, база содержит *методы, и средства,* позволяющие каждому из сотрудников оперировать только с теми данными, которые входят в его компетенцию. В результате взаимодействия данных, содержащихся в базе, с методами, доступными конкретным сотрудникам, образуется информация, которую они потребляют и на основании которой в пределах собственной компетенции производят ввод и редактирование данных.

С понятием *базы данных* тесно связано понятие *системы управления базой данных.* Это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации. Под *визуализацией информации* базы понимается отбор отображаемых данных в соответствии с заданным критерием,

их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

Структура простейшей базы данных

Если в базе нет никаких данных *(пустая база),* то это все равно полноценная база данных. Хотя данных в базе и нет, но информация в ней все-таки есть – это *структура базы.* Она определяет методы занесения данных и хранения их в базе.

Базы данных могут содержать различные объекты, но, основными объектами любой базы данных являются ее таблицы. Простейшая база данных имеет хотя бы одну таблицу. Соответственно, структура простейшей базы данных тождественна структуре ее таблицы.

Структуру двухмерной таблицы образуют столбцы и строки. Их аналогами в структуре простейшей базы данных являются *поля* и *записи.* Если записей в таблице пока нет, значит, ее структура образована только набором полей. Изменив состав полей базовой таблицы (или их свойства), мы изменяем структуру базы данных и, соответственно, получаем новую базу данных.

Свойства полей базы данных

Поля базы данных не только определяют структуру базы но и групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Ниже перечислены основные свойства полей таблиц баз данных на примере СУБД Microsoft Access.

- Имя поля – определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц).

- Тип поля – определяет тип данных, которые могут содержаться в данном поле.

- Размер поля – определяет предельную длину (в символах) данных, которые могут размещаться в данном поле.

- Формат поля – определяет способ форматирования данных в ячейках, принадлежащих полю.

- Маска ввода – определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных).

- Подпись – определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство Имя поля).

- Значение по умолчанию – то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных).

- Условие на значение – ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое

используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты).

- Сообщение об ошибке – текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство Условие на значение).

- Обязательное поле – свойство, определяющее обязательность заполнения данного поля при наполнении базы;

- Пустые строки – свойство, разрешающее ввод пустых строковых данных (от свойства Обязательное поле отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым).

- Индексированное поле – если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

Поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных. Так, например, список вышеуказанных свойств полей относится в основном к полям текстового типа. Поля других типов могут иметь или не иметь эти свойства, но могут добавлять к ним и свои. Например для данных, представляющих действительные числа, важным свойством является количество знаков после десятичной запятой. С другой стороны, для полей, используемых для хранения рисунков, звукозаписей, видеоклипов и других объектов *OLE,* большинство вышеуказанных свойств не имеют смысла.

Типы данных

Таблицы баз данных, как правило, допускают работу с достаточно большим количеством разных типов данных. Так, например, базы данных Microsoft Access работают со следующими типами данных.

- Текстовый – тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).

- Поле Мемо – специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он хранится в другом месте базы данных, а в поле хранится указатель на него, но для пользователя такое разделение заметно не всегда.

- Числовой – тип данных для хранения действительных чисел.

- Дата/время – тип данных для хранения календарных дат и текущего времени.

- Денежный – тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.

- Счетчик – специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование – для порядковой нумерации записей.

- Логический – тип для хранения логических данных (могут принимать только два значения, например Да или Нет).

Проектирование базы данных

Неоптимальные решения и прямые ошибки, заложенные на этапе проектирования, впоследствии очень трудно устраняются, поэтому этот этап является основополагающим.

Разработка структуры базы данных. Выяснив основную часть данных, которые заказчик потребляет или поставляет, можно приступать к созданию структуры базы, то есть структуры ее основных таблиц.

1. Работа начинается с составления генерального списка полей – он может насчитывать десятки и даже сотни позиций.

2. В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.

3. Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель – обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше – на одном рабочем месте.

Наметив столько таблиц, сколько подразделений охватывает база данных, приступают к дальнейшему делению таблиц. Критерием необходимости деления является факт множественного повтора данных в соседних записях. На рис. 6 показана таблица, у которой в поле Адрес наблюдается повтор данных. Это явное свидетельство того, что таблицу надо поделить на две взаимосвязанные таблицы.

1. В каждой из таблиц намечают *ключевое поле,* В качестве такового выбирают поле, данные в котором повторяться не могут. Например, для таблицы данных о студентах таким полем может служить индивидуальный шифр студента. Для таблицы, в которой содержатся расписания занятий, такого поля можно и не найти, но его можно создать искусственным комбинированием полей «Время занятия» и «Номер аудитории». Эта комбинация неповторима, так как в одной аудитории в одно и то же время не принято проводить два различных занятия.

Если в таблице вообще нет никаких полей, которые можно было бы использовать как ключевые, всегда можно ввести дополнительное поле типа Счетчик – оно не может содержать повторяющихся данных по определению.

2. С помощью карандаша и бумаги расчерчивают связи между таблицами. Такой чертеж называется *схемой данных.*

2.Описание практической части работы:

2.1. *Цели лабораторной работы:* Освоить навыки создания табличных баз данных в системе DBU и ей подобным.

2.2. *Постановка задачи:* Разработать табличную базу данных и создать индексный файл для сортированных данных по одному из ключей по своему усмотрению.

Разработать табличную базу данных и создать индексный файл для сортированных данных по одному из ключей по своему усмотрению.

2.3. *Порядок выполнения работы:*

2.3.1. Ознакомиться с теоретической частью.

2.3.2. Получить задание у преподавателя.

2.3.3. Выполнить работу.

2.3.4. Оформить отчет:

2.3.4.1. Содержание отчета:

1. *Цель работы* - краткая формулировка поставленной цели.

2. *Порядок выполнения* - определяются действия, необходимые для выполнения данной работы.

3. *Постановка задачи* - формулирование задачи в соответствии с индивидуальным заданием.

4. *Решение поставленной задачи*:

4.1. *Математическое описание решения поставленной задачи* содержит описание связей между параметрами с использованием принятых в математике обозначений.

4.2. *Описание логической структуры программы (алгоритм решения)* содержит:

- краткое описание схемы программы,
- алгоритм решения (по ГОСТ) - рисунок,
- краткое описание используемых операторов языка программирования (при необходимости).

4.3. *Описание программы* содержит:

- название файла, его размер,
- текст программы (или фрагмент для решения конкретной, наиболее важной части задания).

4.4. *Результат работы программы:*
- значения, полученные в результате выполнения программы
- анализ полученных результатов.

Выводы - отвечают на поставленную цель.

2.4. *Контрольные вопросы:*
1.Что такое реляционная база данных ?
2. Что такое индексный файл ?
3. Поясните понятие картеж ?
4. Поясните понятие запись ?
5. Какое множество называется доменом ?
6. Как удалить запись в системе DBU ?
7. Как вставить новую запись в систему DBU ?
8. Как добавить запись в имеющуюся базу данных ?
9. Можно ли упорядочить базу данных по значениям в нужном столбце?
10. Какие типы данных можно использовать в DBU ?