**Кафедра: Автоматика и Вычислительная Техника**

**ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА**

## Содержание

## Введение

## Теоретическая часть

## 1. Функция printf

## 2. Структура спецификаторов вывода

## 3. Функция scanf

## 4. Вопросы и ответы.

## 5. Обзор функций ввода-вывода

## Функция fgetc и макрокоманда getc

## Функция fgetchar и макрокоманда getchar

## Функции getch и getche

## Функция kbhit

## Функция ungetc

## Функция ungetch

## Функция fgets

## Функция gets

## Функция cgets

## Функция fputs

## Функция puts

## Функция cputs

## Другие функции серии ..printf

## Функции вывода на экран из conio.h

## Практические упражнения

## Лабораторные задания

## Библиографический список

**Введение**

Настоящие указания являются первой работой в серии, посвященной отдельным вопросам программирования на языке Си в оболочке ВС++2.0.

Любая достаточно сложная программа использует функции ввода-вывода данных для реализации дружественного интерфейса с пользователем.

В данных указаниях рассматриваются функции консоли и особенности их применения для обеспечения аккуратного ввода информации и упорядоченного вывода.

Ввод данных, вывод промежуточных и конечных результатов обычно выделяют в отдельные функции, что позволяет программисту тщательно обрабатывать эти данные, не затемняя основные действия программы.

Здесь не рассматривается графический ввод-вывод.

Часть указаний носит справочный характер. При этом уделяется внимание обзору как можно большего числа стандартных функций, так как их преимущественное использование делает программу более надежной и понятной.

Программы, написанные для практических и лабораторных задач, должны быть распечатаны и оформлены в соответствии со стандартными требованиями, предъявляемыми к программному обеспечению.

**Теоретическая часть**

## 

## 1. Функция printf

Предназначена для вывода переменного числа аргументов в стандартный поток вывода stdout. Перед выводом аргументы подвергаются форматированию. Возвращает число реально выведенных символов, включая управляющие символы.

Синтаксис: int printf(const char \*format [, argument, ...]);

Первый и обязательный аргумент format представляет собой строковую константу и содержит элементы двух видов:

1. Символы ASCII-таблицы, представленные их фактическим написанием (например, 1, \_ , пробел, Ф, символы псевдографики), символьными константами (например, \101 - ascii-код буквы А) или их мнемокодами.

Перечислим наиболее употребительные мнемокоды:

\n – перевод строки,

\r - возврат каретки,

\t – горизонтальная табуляция,

\v - вертикальная табуляция.

Эти символы выдаются на печать. Чтобы напечатать специальные символы \ и", перед ними надо поставить символ \.

2. Спецификаторы вывода, имеющие вид

% [flags] [width] [.prec] [F|N|h|l] type .

Каждый спецификатор начинается с % и заканчивается одним из символов type. Вслед за форматом идет перечень аргументов через запятую. Соответствие между аргументами и спецификаторами вывода осуществляется слева направо. При этом аргументов должно быть не больше спецификаторов. В противном случае, недостающие аргументы будут выбраны из стека и интерпретированы непредсказуемым образом.

## 2. Структура спецификаторов вывода

Таблица 1

Элемент type спецификатора

|  |  |
| --- | --- |
| Type | Формат вывода |
| d,i | десятичное целое со знаком |
| О | беззнаковое восьмеричное целое |
| U | беззнаковое десятичное целое |
| x,X | в функции printf = беззнаковое шестнадцатеричное целое; в функции scanf = шестнадцатеричное целое со знаком |
| f | вещественное число [-]dddd.ddd с фиксированной точкой |
| e | вещественное число [-]d.ddd e [+/-]ddd с плавающей точкой |
| g | формат е или f, выбираемый самой функцией в зависимости от точности |
| E,G | то же самое, что е, за исключением Е для экспоненты |
| с | символ, ascii-код которого содержится в аргументе |
| s | вывод строки, т.е. последовательности символов, которая оканчивается символом '\0' |
| **%** | символ *%* |
| P | адрес, содержащийся в аргументе-указателе. Для ближнего указателя выводится только смещение YYYY; для дальнего - сегментный адрес и смещение XXXX:YYYY |
| n | число реально выведенных до сих пор символов записывается в переменную, адрес которой указывается в соответствующем аргументе |

Таблица 2

Необязательный элемент [flag] спецификатора

|  |  |
| --- | --- |
| [fflag] | Что flag означает |
| отсутствует | выравнивание по правому краю; левый заполнитель: 0 или пробел (по умолчанию) |
| **-** | выравнивание по левому краю; справа добавляются пробелы |
| + | всегда указывать для числа его знак: + или - |
| пробел | Указывать знак только для отрицательных чисел |
| **#** | конвертировать, используя альтернативную Форму |

Таблица 3

Альтернативная форма для флага # при наличии указанного типа type

|  |  |
| --- | --- |
| c,s,d,i,u | добавить слева 0 для ненулевого аргумента |
| o | добавить слева 0 для ненулевого аргумента |
| x или X | добавить слева от аргумента 0х или 0Х |
| e,E,f | всегда использовать десятичную точку |
| g или G | то же, что е, Е или f, но без добавления нулей справа |

Таблица 4

Необязательный элемент [width] спецификатора

|  |  |
| --- | --- |
| [width] | Влияние на вывод |
| n | вывод не менее n символов, в качестве заполнителя – пробел |
| 0n | вывод не менее n символов, в качестве заполнителя - символ 0 |
| **\*** | следующий аргумент из списка задает ширину для текущего выводимого символа |

Таблица 5.

Необязательный элемент [.prec] спецификатора

|  |  |
| --- | --- |
| [.prec] | Влияние на вывод |
| отсутствует | точность по умолчанию |
| .0 | (d,i,o,u,x) Точность по умолчанию (e,E,f) Дробная часть отбрасывается, а десятичная точка не ставится |
| .n | длина дробной части не более n символов |
| **\*** | следующий аргумент из списка задает точность для текущего выводимого символа |

Таблица 6.

Необязательный модификатор [.prec]

|  |  |
| --- | --- |
| Модификатор | Как интерпретируется аргумент |
| F | p аргумент - дальний указатель |
| N | p аргумент - ближний указатель |
| l | d,i,o,u,x,X аргумент - long int |
| L | e,E,f,g,G аргумент - double (только для scanf) |
| L | e,E,f,g,G аргумент - long double |

**Пример 1.**

#include <stdio.h>

#include <math.h>

#include <conio.h>

# include <string.h>

void main()

{

clrscr();

printf("\nHello, World!");

printf("\nЗаголовки столбцов :\n\"Январь\"\t\t\" Другие\

месяцы ...\"");

printf("\nОшибка !\007"); //звуковой сигнал

int i= 1828;

printf("\nПолный адрес переменной i = %Fp, смещение =\

%Np", &i, &i);

printf("\nТолстой родился в %d году, а число е= % . 9 f " , i, М\_Е);

printf("\nВозможна ошибка %s: аргументов меньше, чем\

спецификаторов");

// разные системы счисления 33 = 0x21 = 041

printf("\n%i = %#х = %#0о", i, i, i);

char j=5, str[80] = "";

//строка с 5 повторяющимися символами –

while(j--)

strcat(str,"-");

//memset(str, '-', 5); // другой вариант

printf("\n%s", str);

int n=printf("\nДанный вызов printf вывел");

printf("%i символов, учитывая перевод строки (1 символ)", n);

if(!getch())

getch(); //задержка экрана

}

## 

## 3. Функция scanf

Предназначена для ввода переменного числа аргументов из стандартного потока ввода stdin. Перед вводом аргументы подвергаются форматированию. Возвращает число реально введенных аргументов (не символов!). Помещает вводимые данные по адресам, содержащимся в аргументах, т.е. аргументы передаются этой функции по адресу. Функция прекращает свою работу при первом неудачном вводе. Если не все данные введены успешно, то необходимо очистить буфер входного потока с помощью функции

fflush(stdin);

В противном случае при следующем вызове scanf будут вводиться не введенные ранее данные. Желательно очищать буфер и перед первым вызовом scanf. Синтаксис: int scanf(const char \*format [, ...]);

Первый и обязательный аргумент format представляет собой строковую константу и содержит только спецификаторы ввода и их разделители (см. таблицы для printf). Текст использовать нельзя. Уточнения к таблицам спецификаторов:

1. Для ввода в переменные типа double нужно использовать спецификатор %1f, а для ввода long double - %Lf.
2. В качестве разделителей спецификаторов можно использовать пробел или любой символ пунктуации. Выбранный разделитель должен разделять и вводимые данные. В противном случае будет введено только первое данное.
3. Функция scanf не проверяет для строк выход из диапазона, поэтому для строк в формате нужно указывать длину вводимой строки. Например, для строки char str[10] спецификатор должен иметь вид %10s. В противном случае если вводимая строка больше буфера, то буфер будет переполнен, что может вызвать "хорошо скрытую" ошибку.
4. Функция допускает возможность фильтрации вводимой информации. Ввод прекращается при первой встрече символа, отсутствующего в фильтре. Особенно эффективно при вводе данных из файла.

**Пример 2.**

char str[100];

scanf("%[A-Za-z]", str);// ввод до первого небуквенного символа

scanf("%[-+.0-9]", str);// ввод вещественного числа в строку

scanf("%[^-.0-9]", str);// ввод любых символов, пока не\

//встретится один из перечисленных после ^

**Пример 3.**

#include <stdio.h>

#include <math.h>

# include <conio.h>

#include <string.h>

void main()

{

char c; int i; long 1; float f; double d; long double 1d;

unsigned int ui;

int Age; char str[10]; char\*pc;

clrscr();

рrintf("\nВведите символ с=");

fflush(stdin); scanf("%c", &c);

printf("Введено с - %c", c);

printf("\nВведите через пробел целое и длинное целое"); fflush(stdin); int j=scanf("%d %ld", &i, &1);

printf("Введено %d аргументов:i = %i, 1 = %ld", j, i, 1 ) ;

printf("\nВведите беззнаковое целое");

fflush(stdin); scanf("%u", &ui);

printf("Введено ui = %u", ui);

printf("\nВведите через запятую вещ.числа float, double и\

long double\n");

fflush(stdin); scanf("%f,%lf,%Lf", & f, &d, &ld); //фиксир.\

//или плав. точка

printf("\Введено float = %g, double = %g, long double =\

%Lg",f,d,ld);

printf("\nВведите строку\n");

fflush(stdin); scanf("%10s", str); //фиксир. или плав. точка

printf("Введена строка %s", str);

printf("\nВведите адрес ячейки\n" );

fflush(stdin); scanf("%Fp", &pc); //фиксир. или плав. точка

printf("Введен адрес %Fp, содержимое ячейки по этому\

адресу %с", рс, \*рс );

do

{

fflush(stdin);

printf("\nВведите возраст:");

}

while( scanf("%d", &Age) != 1 || 0 > Age || Age > 100);

fflush(stdin);

printf("\nВозраст = %d", Age);

if(!getch())getch();

}

**Пример 4.**

Найти первое целое число в текстовом файле.

#include <stdio.h>

void main()

{

char str[1000];

int i;

FILE \*fp = fopen("ex.txt", "w+"); // Проверка опущена!

fputs("Год 1997 - число простое", fp);

rewind(fp);

fscanf(fp, "%[^0-9]%i", str, &i);

printf("\nПредыдущие символы :%s\nпeрвoe число = %i",

str, i);

}

## 

## 4. Вопросы и ответы

**Вопрос.** Как правильно "заморозить" экран?

**Ответ.** Некорректно использовать для этой цели вызов функции ввода символа с клавиатуры

getch();

Он нормально сработает, если "разморозить" экран нажатием клавиши, имеющей однобайтовый ascii-код. Это -такие клавиши, как Esc, Enter, Tab, буквы, цифры основной клавиатуры, и т.д.. Функция getch() считывает этот код, и программа продолжается. Но при нажатии некоторых других клавиш в буфер ввода-вывода с клавиатуры записываются два числа: 0 и расширенный код, совпадающий, как правило, со scan-кодом клавиши. Функция getch() считает 0, программа продолжится, а при последующем вызове getch() будет считан оставшийся, ненужный scan-код. Даже если getch() стоял в конце программы, то при повторном запуске программы буфер ввода-вывода с клавиатуры не будет очищен, getch() считает scan-код и программа не "заморозится". Таким образом, программа будет приостанавливаться через раз ! Правильнее будет вставить строку

if( !getch()) getch();

Другой вариант - ожидание в бесконечном цикле нажатия клавиши

while( !kbhit());

Он подходит только в конце программы, т.к. дальнейший вызов getch() считает случайно нажатую клавишу.

**Вопрос.** Как правильно использовать scanf при вводе?

**Ответ.** Допустим нужно ввести с клавиатуры возраст человека в переменную Age типа int.

do

{

fflush(stdin);

printf("\nВведите возраст:");

}

while(scanf("%d", &Age) != 1);

fflush(stdin);

printf("\nВозраст = %d", Age); //Отладочная проверка

В реальной программе желательно произвести проверку на осмысленность введенного значения. В данном случае, вероятно, 0 <= Age <= 100. Тогда условие в цикле while может принять вид

while( scanf("%d", & Age) != 1|| 0 > Age || Age > 100);

Получается довольно сложный код, но информация вводится в программу нечасто, а обрабатывается достаточно долго, поэтому стоит потратить усилия на обеспечение корректного ввода.

## 

## 5. Обзор функций ввода-вывода

Таблица 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя функции | С каким работает  потоком | | | Особенности |
| Файл | stdin/ stdout | Консоль |
| 1.Ввод символа | getc,  fgets | X  X | **-**  **-** | **-**  **-** | макро, эхо функция, эхо |
| getchar,  fgetchar | **-**  **-** | X  X | **-**  **-** | макро, Enter функция, Enter |
| getch  getche | **-**  **-** | **-**  **-** | X  X | без эха, без Enter эхо, без Enter |
| kbhit  ungetc  ungetch | **-**  X  **-** | **-**  **-**  **-** | X  **-**  X |  |
| 2.Вывод символа | putc  putchar  putch | X  **-**  **-** | **-**  X  **-** | **-**  **-**  X |  |
| З.Ввод строки | fgets  gets  cgets | X  **-**  **-** | **-**  X  **-** | **-**  **-**  X | с проверкой без проверки с проверкой |
| 4. Вывод строки | fputs puts cputs | X  **-**  **-** | **-**  X  **-** | **-**  **-**  X | без CR/LF  cCR/LF без CR/LF |
| 5.Форматный ввод | fscanf scanf cscanf sscanf | X  **-**  **-**  **-** | **-**  X  **-**  **-** | **-**  **-**  X  **-** | ввод из строки |
| 6.Форматный вывод | fprintf printf cprintf sprintf | X  **-**  **-**  **-** | **-**  X  **-**  **-** | **-**  **-**  X  **-** | Вывод в строку |

**Замечания:**

1. Под консолью понимается клавиатура при вводе и дисплей (или монитор) при выводе информации.
2. По умолчанию под стандартным вводом stdin понимается ввод с клавиатуры, под стандартным выводом stdout понимаетсявывод на монитор. Кроме них, существуют другие стандартные потоки:

* stderr - устройство стандартного вывода ошибок (монитор),
* stdaux - вспомогательное устройство (последовательный  
  com-порт)
* stdprn - принтер (параллельный 1tp-порт) Потоки stdin/stdout нельзя легально перенаправить программным способом в файл, т. к. они являются макроопределениями. Но перенаправление можно осу ществить с помощью команды DOS >. Например,

hello.exe > ex.txt

В этом случае все, что шло в поток stdout, пойдет в файл ex.txt.

Например,

#include <stdio.h>

#include <conio.h>

#include <mem.h>

void main()

{

clrscr();

FILE \*fp = fopen("exl.txt", "w+");

if(fp = NULL)

{

рerrоr("Ошибка при открытии файла"); // вывод в stderr return;

};

FILE \*pbuf=stdin; int с = getc(stdin);

fclose(fp);

}

### 

### Функция fgetc и макрокоманда getc

Прототипы:

int fgetc(FILE \*stream);

int getc(FILE \*stream);

Возвращает: символ, расширенный до int без продолжения знака. В случае ошибки или конца файла возвращает EOF ( = -1 = 0xFFFF).

Описание. Считывает очередной символ из входного потока и увеличивает указатель текущего положения (СР - current position) на 1. При вводе с клавиатуры (stream=stdin) выполняется после нажатия клавиши Enter. Макрокоманда getc полностью аналогична функции fgetc. В файле stdio.h определена, как

#define getc(f) ((--((f)->level) >= 0) ? (unsigned char)(\*(f)-->curp++) \ :\_fgetc (f))

Разберите синтаксис этого макроса !

Хотя getc и fgetc аналогичны, но лучше пользоваться макросом по следующей причине. Чтение из файла происходит блоками по 256b, 512b и т.д. После обработки одного блока в ОЗУ с диска считывается следующий блок. Макрос работает с текущим блоком напрямую, как видно из определения, и обращается к функции fgetc только после обработки этого блока. Функция fgetc также работает с блоком в ОЗУ, но каждое обращение к нему реализуется через вызов функции. Таким образом, использование getc увеличивает скорость за счет увеличения кода. Использование fgetc уменьшает код ценой уменьшения скорости.

### 

### Функция fgetchar и макрокоманда getchar

Прототипы: int fgetchar(void);

int getchar(void);

Возвращает : символ, расширенный до int без продолжения знака. В случае ошибки или конца файла возвращает EOF. Описание. Считывает очередной символ из стандартного входного потока и увеличивает СР на 1. Поток stdin, как и любой открытый файл, имеет буфер с размером по умолчанию 512b. Функция выполняется после нажатия клавиши Enter, после чего вводит 1 символ. Остальные введенные символы остаются в буфере, ожидая своей участи. Макрос getchar аналогичен функции fgetchar.

### 

### Функции getch и getche

Прототипы: int getch(void);

int getche(void);

Возвращают: символ, расширенный до int без продолжения знака.

Описание. Функция getch выводит символ на монитор, getche - не выводит. Функции имеют буфер на два символа: в случае нажатия

клавиши с расширенным кодом туда записывается 0 и scan-код нажатой клавиши. Не ожидают нажатия Enter. Буфер не очищается даже при повторном запуске программы.

### 

### Функция kbhit

Прототип: int kbhit(void);

Возвращает : 1, если к моменту вызова функции была нажата, но не обработана какая-нибудь клавиша; 0-в противном случае.

Описание. Функция оставляет коды нажатой клавиши в буфере ввода-вывода с клавиатуры, так что их можно прочитать, например, с помощью getch.

### 

### Функция ungetc

Прототип: int ungetc(int d,FILE\*stream);

Возвращает: символ d, посланный обратно во входной поток stream, и EOF в случае ошибки.

Описание. Выталкивает символ d обратно во входной поток stream. Вернуть можно только один символ. Следующее чтение из потока вернет символ d. Символ с расширенным кодом вернуть не удастся.

**Функция** **ungetch**

Прототип: int ungetch(int d);

Описание. Аналогична ungetc, но выталкивает символ d обратно в буфер клавиатуры.

### 

### Функция fgets

Прототип: char \*fgets(char \*target, int n, FILE \* stream); Описание. Вводит из stream не более n символов в строку по адресу target. Ввод заканчивается при встрече символа \n. Возвращает target при успешном вводе и NULL при встрече конца файла или в случае ошибки.

### Функция gets

Прототип: char \*gets(char \*target);

Описание. Вводит из stdin строку по адресу target. Число вводимых символов не ограничено. В остальном похожа на fgets.

### 

### Функция cgets

Прототип: char \*cgets(char \*target);

Описание. Читает строку из консоли. str[0] должно содержать максимальную длину вводимой строки. По окончании str[l] содержит число реально введенных символов. Введенная строка начинается с элемента str[2]. Для перехода на новую строку в формат нужно вставить два символа: \n\r. Возвращает начало введенной строки str[2].

### 

### Функция fputs

Прототип : int fputs(const char \*s, FILE \*stream);

Описание. Записывает строку в поток. Возвращает последний записанный символ.

**Функция puts**

Прототип: int puts(const char \*s);

Описание. Записывает строку в поток stdout. Возвращает последний записанный символ.

### 

### Функция cputs

Прототип: int cputs(const char \*s);

Описание. Записывает строку в текстовое окно на экране. Возвращает последний напечатанный символ.

### 

### Другие функции серии ..printf

Прототипы:

int fprintf(FILE \*stream, const char \*format[, argument,...]);

int cprintf(const char \*format[, argument,...]);

int sprintf(char \*buffer, const char \*format [, argument,...]);

Описание. Функции пишут соответственно в поток, в текстовое окно на экране и в строку. Возвращают число реально выведенных символов. Форматирование происходит так же, как для функции printf.

### 

### Функции вывода на экран из conio.h

Функция void clrscr(void) очищает экран.

Для помещения курсора в позицию (х, у) на экране в текстовом режиме используется функция void gotoxy(int х, int у).

Для определения координат курсора имеются функции int wherex() и int wherey().

По умолчанию текст имеет белый цвет на черном фоне. Чтобы изменить цвет текста нужно вызвать функцию

void textcolor(int newcolor),

где newcolor принимает значения из перечислимого типа enum COLORS { BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHTGRAY, DARKGRAY, LIGHTBLUE, LIGHTGREEN, LIGHTCYAN, LIGHTRED, LIGHTMAGENTA, YELLOW, WHITE };

Цвет можно задать его номером от 0 до 15. При этом цвет BLACK=0 всегда означает цвет фона. К цвету можно добавить мерцание BLINK. Например,

textcolor(BLUE | BLINK);

Каждый символ имеет свой фон, цвет которого можно изменить функцией

void textbackground(int newcolor).

Информацию на экране можно перемещать прямоугольными блоками с помощью функции

int movetext(int left, int top, int right, int bottom, int destleft, int desttop);

Размер текстового экрана можно изменить до размеров желаемого текстового окна:

void window(int left, int top, int right, int bottom);

При этом левый угол экрана имеет координаты (1,1). После перехода к текстовому окну система координат привязывается к этому окну. С текстовым окном работают функции: putch(), cputs, cprintf(), gotoxy(), wherex, wherey. Граница окна не рисуется, но текст при выходе из окна усекается. При достижении правой границы происходит автоматический переход на новую строку, а при достижении нижней границы - вертикальный скроллинг. Таким образом, текстовое окно обладает всеми свойствами текстового экрана. Текст может печататься на экране ярким или нормальным цветом. Для этого надо вызвать соответственно функции

void highvideo(void), void normvideo().

Эти установки влияют на все последующие вызовы функций, связанных с выводом текста на экран : putch(), cputs, cprintf().

**Вопрос.** Как узнать ascii-код определенной клавиши?

**Ответ.** Проще всего вставить в программу временный фрагмент типа

char d = getch();

d = getch();

Затем посмотреть в отладчике значения переменной d. Конечно, можно написать и более удобную программку.

#include <conio.h>

#include <stdio.h>

main()

{

char c, d;

clrscr();

printf("\nНажмите клавишу или комбинацию клавиш");

if(!(c = getch()))

{

d=getch();

printf("\nРасширенный код: 0, %i = '%c''', d, d);

}

else

printf(["\nAscci-код: %i = '%c''', с, с);

if( !getch())

getch();

}

**Вопрос.** Как напечатать на экране графическое представление управляющих символов.

**Ответ:** Печать управляющих символов реализует их функциональное назначение. Например, символ Tab, имеющий код '\t'=9, равносилен нескольким пробелам. Так действуют все функции вывода. Для графического представления подобных символов нужно поместить их код непосредственно в видеобуфер.

Для цветного графического адаптера видеобуфер начинается с абсолютного адреса 0хВ80001. На символ отводится 2 байта: первый - для asccii-кода, второй - для атрибутов. Этот метод называется отображением в память. Строки экрана располагаются в буфере последовательно.

#include <conio.h>

#include <stdio.h>

#include <dos.h>

void main()

{

char \*pc = (char \*)0xB80000001 + 2\*( 80\*(wherey()-l) +

wherex()-l);

pc[0] = '\t';

pc[1] = 7; // Обычный атрибут

//лучше использовать функцию записи по указанному

//адресу ОЗУ

//poke(0xB800, 2\*( 80\*(wherey()-l) + wherex() -l), 0x700 +

//'А');

// числа типа int записываются в ОЗУ наоборот : две

//старшие 16-е цифры - в младшем (правом) байте,

//младшие цифры - в старшем (левом) байте

if(!getch()) getch();

}

Практические упражнения

1. С клавиатуры вводится текстовая строка на русском языке. Найдите количество гласных и согласных букв. Используйте фильтр для ввода только текстовых символов.
2. Вычислите относительную погрешность правой формулы для вычисления производной некоторой простой функции y = f(х):

у' = (f(x0 + eps) - f(x0)) / eps .

Представьте результаты в табличной форме для приращений eps, логарифм которых пробегает от -1 до -14 с шагом 0.1. Таблицу запишите в файл. Вещественные числа имеют тип float. Найдите оптимальное приращение.

1. В файл записаны вперемежку текст и целые числа. Найдите сумму всех чисел.
2. Распечатайте на экране таблицу умножения для 16-х чисел.
3. Распечатайте на экране таблицу ASCII-кодов с графическим изображением управляющих символов.
4. Напишите программу, которая запрашивает пароль. Неверный пароль останавливает программу.

**Лабораторные задания**

1. Напишите программу, реализующую все возможности печатной машинки:

* с клавиатуры непосредственно на экран выводятся только символы, имеющиеся на печатной машинке;
* переход на новую строку по нажатию Enter;
* перемещения курсора по экрану с помощью стрелок;
* забой символа над курсором осуществляется пробелом.

1. Напишите программу, которая рисует в центре зеленого экрана прямоугольник размером с четверть экрана. Граница области – жёлтая, цвет заполнения бледно-синий, в центре крана – мерцающий текст красного цвета.
2. Напишите функцию, которая получает массив строк, рисует окно с границей из одинарной рамки и вставляет строки массива по одной в это окно. Размеры окна - минимально возможные.
3. Напишите функцию, реализующую меню с несколькими выборами, в котором выбираемый пункт высвечивается и перемещается с помощью стрелок.

Библиографический список

1. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. М.: Фин. и стат.,1992.
2. Керниган Б., Ритчи Д. Язык программирования Си. М.: Фин. и стат., 1992.
3. Керниган Б., Ритчи Д. Язык программирования Си. Задачи по курсу Си. М.: Фин. и стат., 1985.
4. Уинер Р. Язык Турбо Си. М.: Мир, 1991.
5. Хикс К. Си без проблем. Руководство пользователя. М.: Бином, 1997.