**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 2**

**по дисциплине «Организация ЭВМ и систем»**

Тема: **«ИССЛЕДОВАНИЕ ВИДЕОСИСТЕМЫ (ТЕКСТОВЫЙ РЕЖИМ)»**

|  |  |  |
| --- | --- | --- |
| Студенты |  |  |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2021

**Цель работы.**

Изучение работы с видеосистемой в текстовом режиме, освоение приемов использования цветовой палитры: измене­ние цвета символов и фона на всем экране и в отдельном окне.

**Задание (вариант №13)**

Написать программу, чтобы в окно с координатами **(10,12,70,23)** с шагами **0.3 секунд** и **1 строкой** выводилась строка при всех возможных комбинациях цвета фона и цвета символов. Строка содержит обозначение цвета фона и символа. Для каждой комбинации цветов в окне должны выводиться символьные обозначения цветов фона и символов. Цвет окна должен соответствовать цвету фона.

Дополнить программу скроллингом окна с **вверх**, используя функции прерывания 10h BIOS.

**Программная среда:**

Вся программа написана на языке программирования Turbo С++, компилировалась в DOS BOX.

**Краткие сведения о видеосистемах ПЭВМ, текстовом режиме их работы и функциях обслуживания текстового режима.**

Аппаратные средства для вывода информации на экран включают специальную электронную плату (видеоадаптер, либо адаптер дисплея, либо просто адаптер) и монитор (или просто экран).

В самом общем виде видеоадаптер состоит из двух основных частей: контроллера и видеопамяти (видеобуфера). Помимо этих обязательных узлов, наиболее совершенные видеоадаптеры имеют в своем составе например специализированные контроллеры быстрой манипуляции содержимым видеобуфера (так называемые контроллеры графики).

Основное назначение видеобуфера - хранение образа информации экрана.

Изображение на экране строится из небольших точек - так называемых пикселов (pixel - Picture Element).

Память, необходимая для хранения полного образа экрана, называется видеостраницей.

Управление параметрами видеосистемы может выполняться на двух уровнях:

1. на уровне портов видеоадаптера;
2. обращением к функциям BIOS.

Интегральной характеристикой особенностей работы адаптера является совокупность поддерживаемых им режимов. Поведение адаптера в том или ином режиме является фактическим стандартом и полностью характеризует все особенности адаптера, доступные для программиста средства управления адаптером и т.п.

При всем многообразии режимов работы видеоадаптеров их можно объединить в две группы: текстовые и графические. Переключение из текстового режима в графический и наоборот означает полное изменение логики работы видеоадаптера с видеобуфером.

Если видеоадаптер включен в текстовый режим, он рассматривает экран как совокупность так называемых текселов (texel - Text Element).

Каждому знакоместу экрана (текселу) в текстовом режиме соответствуют два байта памяти видеобуфера. Байт по четному адресу хранит ASCII-код символа, а следующий за ним байт по нечетному адресу кодирует особенности отображения символа на экране: цвет пикселов, из которых формируется очертание символа (Foreground Color), цвет всех остальных пикселов знакоместа или цвет фона символа (Background Color), мерцание символа и необходимость повышения яркости символа при отображении. Этот байт называется байтом атрибута.

Видеопамять адаптера при работе в текстовых режимах доступна непосредственно из программы.

Вывод на монитор содержимого видеобуфера происходит начиная с неко­торого начального адреса, называемого смещением до видеостраницы.

Видеоадаптер при работе в текстовом режиме периодически считывает содержимое ячеек видеобуфера и по коду символа и байту атрибута формирует пикселы, образующие в совокупности очертание символа и его фон. При этом байт символа служит индексом для входа в специальную таблицу - так называемую таблицу знакогенератора. Она содержит информацию, по которой видеоадаптер формирует пикселы для изображения того или иного символа.

Функции консольного ввода-вывода С++ помещены в файле <conio.h>, предназначены для облегчения работы по созданию простейшего оконного интерфейса.

window(int , int ,int , int ); - установка параметров активного текстового окна выполняет функция;

Фрейм окна С++ имеет следующую структуру:

struct text\_info

{ unsigned char

winright, winbottom; /\* столбец, строка правого нижнего угла \*/

attribute, normattr; /\* атрибуты окна\*/

currmode; /\* текущий режим работы видеоадаптера \*/

screenheight; /\* полная высота экрана \*/

screenwidth; /\* полная ширина экрана \*/

сurх, сurу; }; /\* строка, столбец текущей позиции курсора \*/

gettextinfo(struct text\_info \*t); - информация об активном окне

textcolor(), textbackground(), textattr() - управляют цветом символов

Видеоадаптеры всех типов аппаратно поддерживают курсор, который в текстовых режимах отображается на экране в виде одной или нескольких линий в пределах тексела. Курсор указывает на текущую позицию экрана (строку и столбец тексела), в которую будет записываться или из которой будет читаться средствами BIOS символ.

Программное прерывание 10h BIOS имеет в своем составе специальные функции для установки формы курсора, чтения и установки его координат.

Функция АН = 0lh задает высоту курсора. Регистр СН определяет номер верхней телевизионной линии, a CL - номер нижней линии при изображении курсора и т. д.

Среди функций консольного ввода-вывода С++ текущей позицией курсора в окне управляет функция gotoxy(int x, int y);.

Вывод информации на экран персонального компьютера может выполняться на трех уровнях:

1. на уровне MS-DOS с использованием функций прерывания 21h
2. на уровне BIOS с использованием функций прерывания 10h
3. непосредственным доступом к аппаратным средствам.

Вывод информации на уровне **MS-DOS** - мобильный, но самый медленный.

На уровне MS-DOS работают функции стандартного вывода, а их прототипы содержатся в файле <stdio.h>.

Вывод на уровне **BIOS** дает более широкие возможности по управлению экраном. Именно эти функции используются драйверами MS-DOS для вывода информации на экран.

Недостатком функций BIOS является невысокая скорость вывода, что особенно заметно при работе в графических режимах. На уровне BIOS работают функции консольного вывода, а их прототипы помещены в файле <conio.h>.

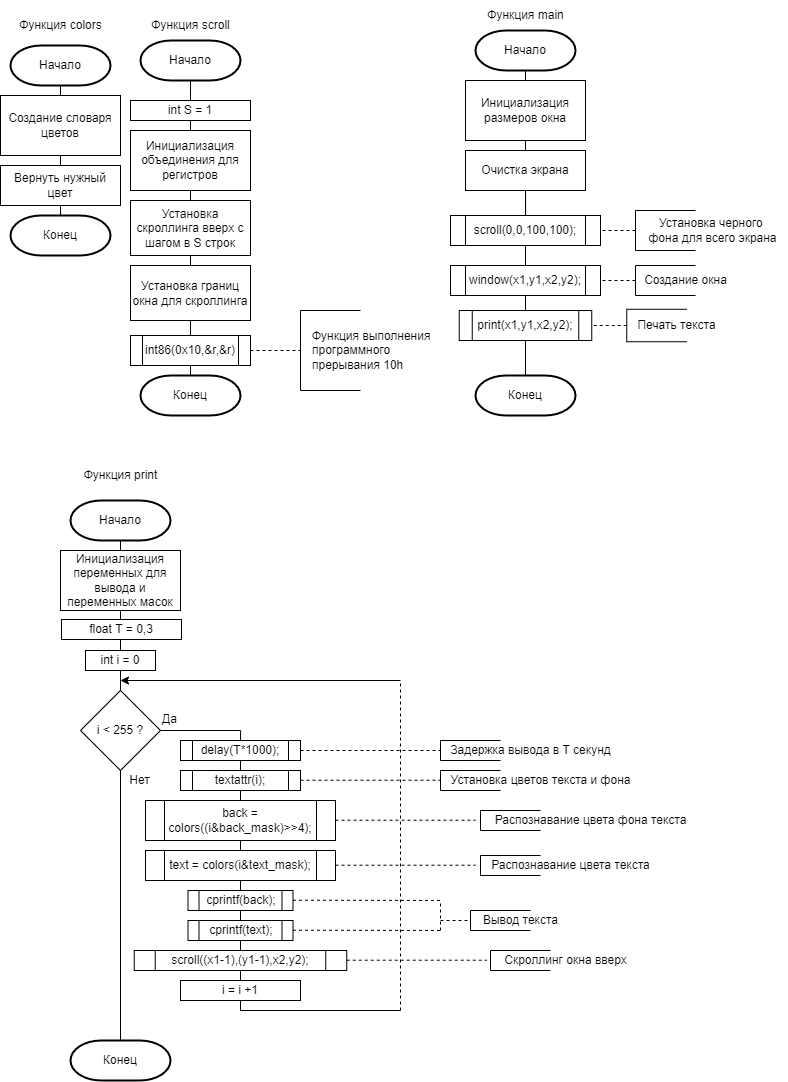
Функции АН = 06 и 07 прерывания 10h BIOS осуществляют так называемый скроллинг (прокрутку) окна экрана. При выполнении скроллинга на одну строку вверх вся информа­ция в окне перемещается на строку вверх. Внизу окна появляется чистая строка. При выполнении скроллинга на одну строку вниз содержимое окна сдвигается на строку вниз и вверху окна добавляется чистая строка. Значение регистра AL задает число строк, на которое выполняется скроллинг. Если AL=0, выполняется очистка окна. Значения в СН и CL определяют строку и столбец левого верхнего угла окна, в DH и DL -строку и столбец правого нижнего угла. Строки и столбцы нумеруются от 0. Значение в регистре ВН задает атрибут добавляемой чистой строки.

С++ включает большой набор функций ввода-вывода информации в окно экрана. Прототипы этих функций помещены в заголовочном файле <conio.h>. (clreol, clrscr, delline, insline,cprintf, cputs, movetext, putch, puttext, highvideo, lowvideo, normvideo, textattr, textcolor, textbackground и т. д.)

Функция textcolor(int newcolor) задает цвет символов, не затрагивая установленный цвет фона. Цвет может быть или числом, или формироваться из символических констант, значения которых определяет перечислимый тип COLORS.

Функция textbackground(int newcolor) задает цвет фона символов, не затрагивая установленный цвет символа. Цвет может быть или числом, или формироваться из символических констант, значения которых определяет перечислимый тип COLORS. Для цвета фона выбор ограничен значениями цветов 0-7. Если для цвета фона выбирается значение 8 - 15, то символы будут мерцать, так как бит мерцания установится в единицу, но цвет фона будет соответствовать значениям 0-7

**Схемы алгоритмов**



**Текст программы (Turbo C++ DOS)**

#include <dos.h>

#include <iostream.h>

#include <conio.h>

const char \*colors(int i)

{

const char \*vocabulary[] = {"BLACK", "BLUE", "GREEN", "CYAN",

"RED", "MAGENTA", "BROWN", "LIGHTGRAY", "DARKGRAY", "LIGHTBLUE",

"LIGHTGREEN", "LIGHTCYAN","LIGHTRED", "LIGHTMAGENTA","YELLOW","WHITE"};

return vocabulary[i];

}

void scroll(int lr,int lc,int rr, int rc)

{

int S = 1;

union REGS r;

r.h.ah = 6;

r.h.al = S;

r.h.ch = lc;

r.h.cl = lr;

r.h.dh = rc;

r.h.dl = rr;

int86(0x10,&r,&r);

}

void print(int x1,int y1,int x2,int y2)

{

const char \*text, \*back;

int text\_mask=15, back\_mask=112;

float T = 0.3;

for(int i=0;i<255;i++)

{

delay(T\*1000);

textattr(i);

back = colors((i&back\_mask)>>4);

text = colors(i&text\_mask);

cprintf(back);

cprintf(" ");

cprintf(text);

cprintf(" \r");

scroll((x1-1),(y1-1),x2,y2);

}

}

int main()

{

int x1 = 10, y1 = 12, x2 = 70, y2 = 23;

clrscr();

scroll(0,0,100,100);

window(x1,y1,x2,y2);

print(x1,y1,x2,y2);

return 0;

}