**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)

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2023

**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.

**Формулировка задания**

Написать программу, чтобы в окно с координатами (25, 8, 55, 18) с шагами Т (1,2 секунд) и S (2 строки) выводилась надпись на русском языке при всех возможных комбинациях цвета фона и цвета символов. Для каждой комбинации цветов в окне должны выводиться номера или символьные обозначения цветов фона и символов.

**Теоретическая информация**

Характеристикой особенностей работы адаптера является набор поддерживаемых им режимов. Поведение адаптера в каждом режиме устанавливает стандарт и полностью описывает все его особенности, включая доступные программисту средства управления адаптером и т.д.

Режимы работы видеоадаптеров могут быть объединены в две группы: текстовые и графические. Переключение между текстовым и графическим режимами полностью меняет логику работы видеоадаптера с видеобуфером.

В текстовом режиме видеоадаптер рассматривает экран как набор текстовых элементов (текселов), где каждый тексел соответствует двум байтам памяти видеобуфера. Первый байт содержит ASCII-код символа, а следующий за ним байт определяет атрибуты отображения символа на экране.

Переход в графический режим полностью изменяет логику работы видеоадаптера, позволяя управлять цветом каждого пиксела на экране. Количество строк и пикселов в каждой строке зависит от выбранного режима работы видеоадаптера, формируя матрицу пикселов.

Функции ввода-вывода консоли в C++ находятся в файле <conio.h> и предназначены для упрощения создания простых оконных интерфейсов.

Установку параметров активного текстового окна выполняет функция window(int, int, int, int), описывая активное текстовое окно через указание номеров столбца и строки левого верхнего угла, а также правого нижнего угла. Строки и столбцы нумеруются, начиная с 1.

Структура окна в C++ имеет следующий формат:

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

};

Информация об активном окне доступна через функцию gettextinfo(struct text\_info \*t), которая заполняет поля структуры данными об активном окне.

clrscr() очищает текстовое окно, причем цвет "заливки" окна соответствует значению переменной attribute в описании окна (структура text\_info). Функции управления цветом фона и символа описаны далее.

cputs(char \*str) выводит строку символов в текстовое окно, начиная с текущей позиции курсора. Функция аналогична стандартной библиотечной функции puts(), но выводит в пределах заданного окна и не добавляет символ '\n'. Возвращаемое значение - ASCII-код последнего выведенного символа.

textcolor(int newcolor) устанавливает цвет символов без изменения цвета фона. Цвет может быть числом или формироваться из символических констант, определенных перечислимым типом COLORS.

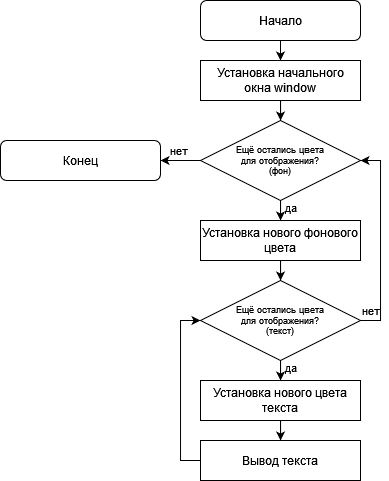
textbackground(int newcolor) устанавливает цвет фона символов без изменения цвета символа. Цвет также может быть числом или формироваться из символических констант типа COLORS. Для цвета фона доступны значения от 0 до 7. Если выбрано значение от 8 до 15, символы будут мерцать, так как бит мерцания установлен в единицу, но цвет фона будет соответствовать значениям от 0 до 7.

**Результаты запуска программы**

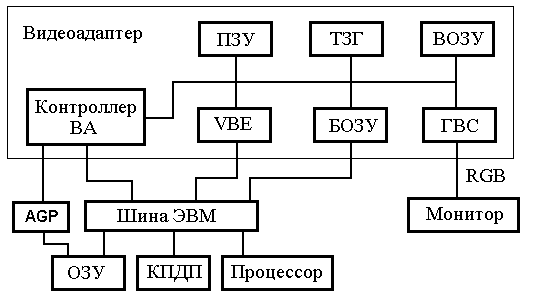




**Блок-схема**

****

**Структурная схема аппаратных средств**



**Текст программы**

#include <conio.h>

#include <dos.h>

#include <stdlib.h>

#include <locale.h>

int x1 = 25;

int y1 = 8;

int x2 = 55;

int y2 = 18;

int t = 1200;

int s = 1;

const int direction = 7; // 6 Up 7 Down

const char COLORS\_AS\_TEXT[][16] = {

    "Чёрный",

    "Синий",

    "Зелёный",

    "Бирюзовый",

    "Красный",

    "Пурпурный",

    "Коричневый",

    "Св. серый",

    "Тём. серый",

    "Св. синий",

    "Св. зелёный",

    "Св. бирюзовый",

    "Св. красный",

    "Св. пурпурный",

    "Жёлтый",

    "Белый"

};

void scroll(int direction, char l\_row, char l\_col, char r\_row, char r\_col, char attr) {

    union REGS r;

    if (direction)

    { r.h.al = 1; r.h.ah = direction; }

    else

    { r.h.al = 0; r.h.ah = 6; }

    r.h.ch = l\_row; r.h.cl = l\_col; r.h.dh = r\_row;

    r.h.dl = r\_col; r.h.bh = attr;

    int86(0x10,&r,&r);

}

void main()

{

    enum SCREEN\_COLORS {

    /\* Black         \*/ BLACK = 0,

    /\* Blue          \*/ BLUE = 1,

    /\* Green         \*/ GREEN = 2,

    /\* Cyan          \*/ CYAN = 3,

    /\* Red           \*/ RED = 4,

    /\* Magenta       \*/ MAGENTA = 5,

    /\* Brown         \*/ BROWN = 6,

    /\* Light Gray    \*/ LIGHTGRAY = 7,

    /\* Dark Gray     \*/ DARKGRAY = 8,

    /\* Light Blue    \*/ LIGHTBLUE = 9,

    /\* Light Green   \*/ LIGHTGREEN = 10,

    /\* Light Cyan    \*/ LIGHTCYAN = 11,

    /\* Light Red     \*/ LIGHTRED = 12,

    /\* Light Magenta \*/ LIGHTMAGENTA = 13,

    /\* Yellow        \*/ YELLOW = 14,

    /\* White         \*/ WHITE = 15

    };

    SCREEN\_COLORS background;

    SCREEN\_COLORS text;

    /\* Initialize window for textcolor(), textbackground(), textattr() \*/

    window(x1, y1, x2, y2);

    for (int i = 0; i < 16; i++) {

        background = SCREEN\_COLORS(i);

        /\* Set text background \*/

        textbackground(background);

        for (int j = 0; j < 16; j++) {

            text = SCREEN\_COLORS(j);

            /\* Set text foreground \*/

            textcolor(text);

            /\* Make empty line \*/

            for (int k = 1; k < s; k++) {

                insline();

                if (direction == 7)

                    cprintf("\n");

            }

            insline();

            cprintf("T: %s B: %d\r", COLORS\_AS\_TEXT[text], background);

            if (direction == 7)

                cprintf("\n");

            delay(t);

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

        }

    }

    getch();

}