**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. 3311 |  | Шарпинский Д. А. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2024

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

Видеосистемы ПЭВМ включают аппаратные средства для вывода информации на экран, такие как видеоадаптер и монитор. Видеоадаптер представляет собой специальную электронную плату, управляемую собственным микропроцессором, сравнимым по мощности с центральным процессором компьютера.

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 ряд дополнительных узлов, например, специализированные контроллеры быстрой манипуляции содержимым видеобуфера (так называемые контроллеры графики). Основное назначение видеобуфера — хранение образа информации экрана. Видеоадаптер формирует изображение на экране 25 и более раз в секунду, что создает иллюзию неподвижного изображения на экране монитора. Изображение на экране строится из небольших точек — так называемых пикселов (pixel — Picture Element). Число пикселов в строке и число самих строк различно для разных типов видеоадаптеров.

Память, необходимая для хранения полного образа экрана, называется видеостраницей. Часто общий объем видеопамяти намного превышает объем страницы. В этом случае появляется возможность хранить в видеобуфере не одну, а несколько страниц. Видеоадаптер способен выполнять переключение текущей видеостраницы.

Текстовый режим работы видеоадаптера 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. Это значит, что любая ячейка видеобуфера может быть прочитана программой так же, как и обычная ячейка оперативной памяти. И как в обычную ячейку памяти, в видеобуфер возможна запись значений из программы

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.

Переключение адаптера в один из графических режимов полностью изменяет логику работы аппаратуры видеосистемы. При работе в графическом режиме появляется возможность управлять цветом любой телевизионной точки экрана или пиксела. Число строк пикселов и число пикселов в каждой строке зависит от режима работы видеоадаптера. Таким образом, экран в графическом режиме представляет собой матрицу пикселов

Функции обслуживания текстового режима включают:

Задание формы курсора и его позиции на экране: Курсор указывает на текущую позицию экрана, в которую будет записываться или из которой будет читаться символ. Управление формой курсора позволяет изменять его внешний вид, что может быть полезно для различных режимов работы программы.

Выбор режима, видеостраницы и палитры: Позволяет переключать видеоадаптер между различными режимами работы (текстовые и графические), выбирать активную видеостраницу и управлять цветовой палитрой.

Управление цветом символов и фона: Позволяет изменять цвет символов и фона, что может быть использовано для выделения важной информации или создания визуальных эффектов.

Скроллинг и очистка окна и всего экрана: Позволяет перемещать содержимое окна или экрана вверх или вниз, а также очищать окно или экран, что может быть полезно для обновления отображаемой информации.

Вывод информации в окно экрана: Позволяет выводить текст и графику в заданное окно на экране, что может быть использовано для создания пользовательского интерфейса.

Эти функции обеспечивают гибкое управление видеосистемой и позволяют создавать интерактивные и визуально привлекательные приложения

**Задание на лабораторной работе**

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 11): Написать программу, в которой в окно с координатами (20, 10, 60, 20) с шагами 1,5 (секунд) и 2 (строки) выводится 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 номера цветов фона. Цвет окна должен соответствовать цвету фона.

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

**Алгоритмы и тексты отлаженных программ**

#include <conio.h>

#include <dos.h>

#define MAX\_WIDTH 80

#define MAX\_HEIGHT 25

#define BUFFER\_SIZE ((MAX\_WIDTH + 1) \* (MAX\_HEIGHT + 1) \* 2 + 1)

int x1 = 20, y1 = 10, x2 = 60, y2 = 20;

void fillEmptyCellsWithColor(int background\_color) {

    int x, y;

    char current\_char;

    int width = x2 - x1 + 1;

    int height = y2 - y1 + 1;

    char buffer[BUFFER\_SIZE] = {0};

    gettext(x1, y1, x2, y2, buffer);

    textbackground(background\_color);

    window(x1, y1, x2, y2);

    gotoxy(1, 1);

    for (y = 0; y < height; y++) {

        for (x = 0; x < width; x++) {

            current\_char = buffer[(y \* width + x) \* 2];

            if (current\_char == ' ') {

                gotoxy(x + 1, y + 1);

                putch(' ');

            }

        }

    }

}

void scrollWindow(int lines) {

    union REGS regs;

    regs.h.ah = 0x06;            // Scroll function (upwards)

    regs.h.al = lines;           // Number of lines to scroll

    regs.h.bh = 0x00;            // Attribute of blank lines (формирование атрибута из фона и текста)

    regs.h.ch = y1 - 1;          // Top-left row (adjusted for zero-based index)

    regs.h.cl = x1 - 1;          // Top-left column (adjusted for zero-based index)

    regs.h.dh = y2 - 1;          // Bottom-right row (adjusted for zero-based index)

    regs.h.dl = x2 - 1;          // Bottom-right column (adjusted for zero-based index)

    int86(0x10, &regs, &regs);   // BIOS interrupt 0x10 to perform the scroll

}

int main() {

    int text\_color, background\_color;

    // Clear the entire screen

    clrscr();

    // Set the window with coordinates (x1, y1) to (x2, y2)

    textbackground(0);

    textcolor(0);

    window(x1, y1, x2, y2);

    clrscr(); // Clear the newly defined window area

    for (text\_color = 0; text\_color < 16; text\_color++) {

        for (background\_color = #include <conio.h>

#include <dos.h>

int x1 = 20, y1 = 10, x2 = 60, y2 = 20;

void scrollWindow(int lines) {

    union REGS regs;

    regs.h.ah = 0x06;            // Scroll function (upwards)

    regs.h.al = lines;           // Number of lines to scroll

    regs.h.bh = 0x00;            // Attribute of blank lines (формирование атрибута из фона и текста)

    regs.h.ch = y1 - 1;          // Top-left row (adjusted for zero-based index)

    regs.h.cl = x1 - 1;          // Top-left column (adjusted for zero-based index)

    regs.h.dh = y2 - 1;          // Bottom-right row (adjusted for zero-based index)

    regs.h.dl = x2 - 1;          // Bottom-right column (adjusted for zero-based index)

    int86(0x10, &regs, &regs);   // BIOS interrupt 0x10 to perform the scroll

}

int main() {

    int text\_color = 0, background\_color = 0, count = 1;

    // Clear the entire screen

    clrscr();

    // Set the window with coordinates (x1, y1) to (x2, y2)

    textbackground(0);

    textcolor(0);

    window(x1, y1, x2, y2);

    gotoxy(1, 1);

    clrscr(); // Clear the newly defined window area

    for (background\_color = 0; background\_color < 16; background\_color++) {

        for (text\_color = 0; text\_color < 16; text\_color++) {

            textcolor(text\_color);            // Set text color

            textbackground(background\_color); // Set background color

            if (count == 12) {

                scrollWindow(2);

                gotoxy(1, 10);

                count = 10;

                getch();

            }

            // Print the line with the current color combination

            cprintf("Text: %d, Background: %d, count: %d", text\_color, background\_color, count);

            count++;

            gotoxy(1, count);

            // delay(1500);

            getch();

        }

    }

    getch(); // Wait for user before exiting

    return 0;

}0; background\_color < 16; background\_color++) {

            textcolor(text\_color);            // Set text color

            textbackground(background\_color); // Set background color

            // Print the line with the current color combination

            fillEmptyCellsWithColor(background\_color);

            cprintf("Text color: %d, Background color: %d\r\n\r\n", text\_color, background\_color);

            getch();

            if (background\_color % 5 == 0) {

              scrollWindow(5);

            }

        }

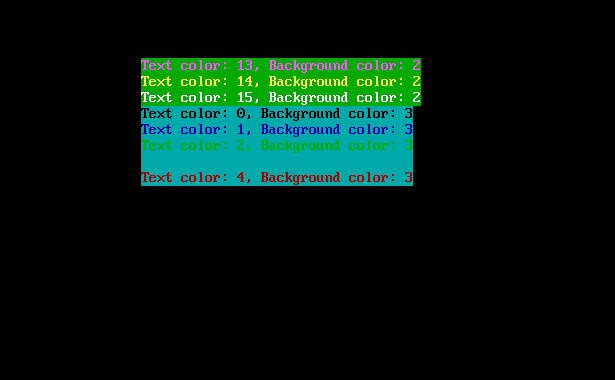
    }

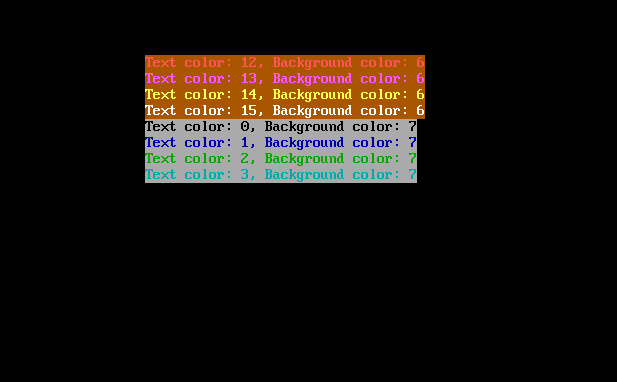
    getch(); // Wait for user before exiting

    return 0;

}

**Пример запуска программы**

****

****