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 _____ Баймухамедов Р. Р.

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 3): Написать программу, в которой в окно с

координатами (20,5,60,15) с шагами 0,3 (секунд) и 3 (строк) 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 41 // width of window
#define MAX_HEIGHT 11 // height of window
#define BUFFER_SIZE ((MAX_WIDTH + 1) * (MAX_HEIGHT + 1) * 2 + 1) // MW and MH + 1
just for reserve memory to border, *2 cause every texel contain 2 bytes, and +1
for end string symbol

void scrollWindow(int lines, int x1, int y1, int x2, int y2) {
    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
    regs.h.ch = y1 - 1;          // Top-left row (include zero index)
    regs.h.cl = x1 - 1;          // Top-left column
    regs.h.dh = y2 - 1;          // Bottom-right row
    regs.h.dl = x2 - 1;          // Bottom-right column

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

int main() {
    int text_color, background_color, string, x1 = 20, y1 = 5, x2 = 60, y2 = 15,
count = 1;
    clrscr(); // Clear the entire screen
    window(x1, y1, x2, y2); // Set the window with coordinates (x1, y1) to (x2,
y2)

    for (background_color = 0; background_color < 16; background_color++) {
        for (text_color = 0; text_color < 16; text_color++) {
            textcolor(text_color);            // Set color of symbols
            textbackground(background_color); // Set background color of symbols
            if(count == 10){
                scrollWindow(3, x1, y1, x2, y2);
                count = 7;
                gotoxy(1,count);
                delay(200);
            }
```

```
            cprintf("Text color: %d, Background color: %d", text_color,
background_color); // Print the line with the current color combination
            count++;
            gotoxy(1, count);
            delay(200);
        }
    }
    getch(); // Wait for user before exiting
    return 0;
}
```

Пример запуска программы