**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по лабораторной работе №2**

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

Тема: Исследование видеосистемы (текстовый режим)

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Соколовский В.Д. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2023

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

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. Режимы принято нумеровать, начиная с нуля. Чем совершеннее видеоадаптер, тем больше режимов он поддерживает. Как правило, более совершенные адаптеры полностью совместимы со своими предшественниками и с точки зрения прикладной программы отображает информацию точно так же, как и его предшественник.

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.

Задавая различные числовые значения байту атрибута в видеобуфере, можно управлять цветом символов и цветом фона, на котором эти символы отображаются. Например, если значение байта атрибута равно **112**, то выводится немерцающий символ черного цвета на сером фоне. Действительно, биты RGB цвета символа для данного кода атрибута равны нулю. Биты цвета фона равны 1, и на мониторе для точек фона будут смешиваться в необходимых пропорциях красный, синий и зеленый цвета. Для цветного видеоадаптера - это серый цвет. Повышение интенсивности цвета символа выполняется путем установки бита с номером 3 в 1. Светло-серый цвет - это белый цвет, поэтому на экране цветного монитора при работе видеоадаптера в текстовом режиме могут быть белые буквы, но не может быть белый фон. Например, символы, код атрибута которых в видеопамяти равен 15, будут отображаться белыми пикселами на черном фоне. В принципе, если задать цвета фона и символа одинаковыми, символы будут невидимыми, например красный символ на красном фоне (атрибут 0x44), что можно использовать в адаптерах, у которых мерцание символа с помощью бита 7 не реализовано.

Видеоадаптеры типов EGA и VGA имеют некоторые особенности использования бита интенсивности, которые будут рассмотрены несколько позже.

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. Адреса ячеек видеопамяти начинаются для разных типов адаптеров с разных границ, приведенных в табл. 2.1. Если адаптер работает в текстовых режимах "40 столбцов х 25 строк", то для хранения полного образа экрана (видеостраницы) требуется 25 х 40 х 2 = 2000 байт видеопамяти. В режимах "80 столбцов х 25 строк" видеостраница занимает уже 25 х 80 х 2 = 4000 байт. Минимальная конфигурация видеоадаптера CGA имеет обычно 16К байт видеопамяти, что позволяет хранить 8 страниц текста в режимах 0 или 1 и 4 страницы в режимах 2 или 3.

Вывод на монитор содержимого видеобуфера происходит начиная с некоторого начального адреса, называемого смещением до видеостраницы. Страница 0 имеет нулевое смещение. Страница 1 в режиме "80 строк х 25 столбцов" начинается с адреса, смещенного на 4096 байт (l000h) относительно начального адреса видеопамяти, страница 2 - со смещения 8192 байт (2000h) и т.д. Если изменить значение смещения, произойдет переключение страницы, т.е. на экране возникнет образ другой страницы видеопамяти. Иногда переключение видеостраниц в текстовом режиме используется для реализации динамических изображений.

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. Число строк и столбцов в одной ячейке таблицы различно для различных типов видеоадаптеров. Чем больше строк и столбцов использовано для символа, тем более качественно он изображается на экране.

Число знакомест в одной текстовой строке зависит от видеоадаптера и от режима его работы.

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.

Для определения окна с координатами (x1,у1,х2,у2) используется встроенная функция window, она определяет окно в консоли с координатами, которые были переданы как аргументы.

Для задержки вывода (шаг Т (секунд)) была использована функция delay(), которая в качестве аргумента принимает значение времени в миллисекундах. Для работы данной функции подключается библиотека "time.h". Для шага S (строк) после каждого вывода было использовано перевод строки cprintf(''\n\r'').

Для вывода надписи при всех возможных комбинациях цвета фона и цвета символов было использовано две конструкции switch-case (первая для фона, вторая для текста) и цикл for.

**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.

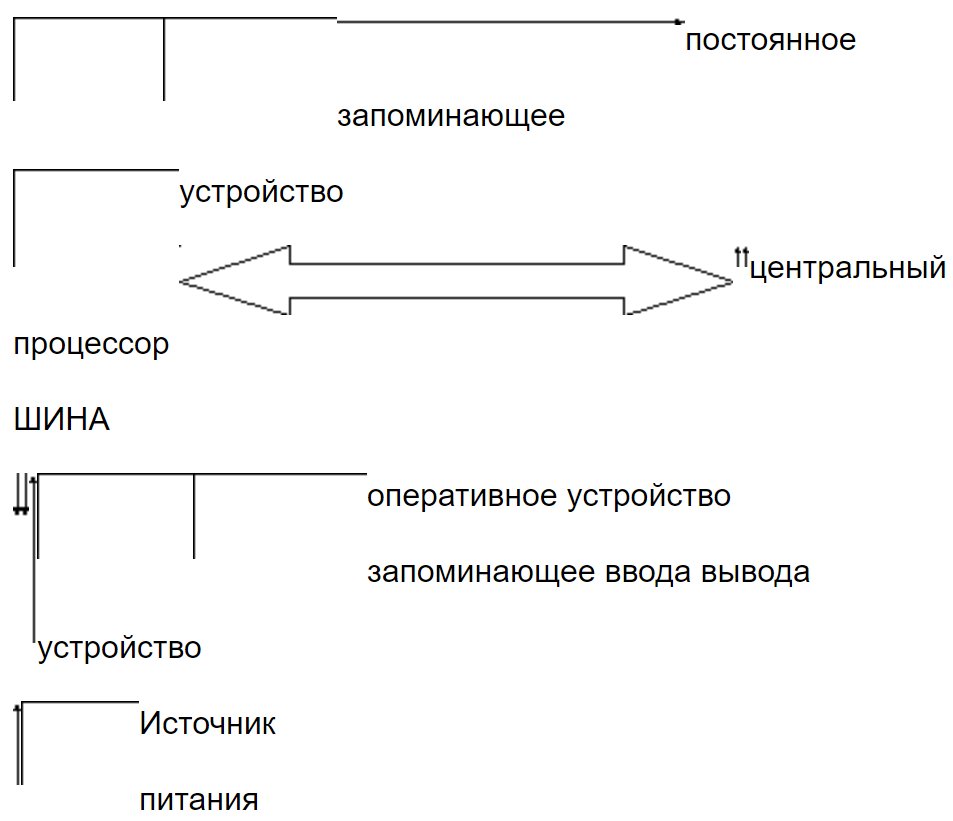
**Задание на лабораторную работу.**

1. Изменить программу, полученную на предыдущей работе таким образом, чтобы в окно с координатами (x1,у1,х2,у2) с шагами Т (секунд) и S (строк) выводилась надпись 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.

****

1. Организовать в окне вывод разноцветных сообщений со скроллингом окна.

**Структурная схема аппаратных средств**



**Пример запуска программы**

****

****

****

**Выводы.**

**В ходе данной лабораторной работы изучил работу с видеосистемой в текстовом режиме, освоил приемы использования цветной палитры: изменение цвета символов и фона на всем экране и в отдельном окне.**

Приложение А

рабочий код

#include <conio.h>

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void nums(int i){

switch (i){

case 0:

cputs("1");

break;

case 1:

cputs("2");

break;

case 2:

cputs("3");

break;

case 3:

cputs("4");

break;

case 4:

cputs("5");

break;

case 5:

cputs("6");

break;

case 6:

cputs("7");

break;

case 7:

cputs("8");

break;

case 8:

cputs("9");

break;

case 9:

cputs("10");

break;

case 10:

cputs("11");

break;

case 11:

cputs("12");

break;

case 12:

cputs("13");

break;

case 13:

cputs("14");

break;

case 14:

cputs("15");

break;

case 15:

cputs("16");

break;

default:

break;

}

}

void setColors(int i){

switch (i){

case 0:

cputs("1\r");

break;

case 1:

cputs("2\r");

break;

case 2:

cputs("3\r");

break;

case 3:

cputs("4\r");

break;

case 4:

cputs("5\r");

break;

case 5:

cputs("6\r");

break;

case 6:

cputs("7\r");

break;

case 7:

cputs("8\r");

break;

case 8:

cputs("9\r");

break;

case 9:

cputs("10\r");

break;

case 10:

cputs("11\r");

break;

case 11:

cputs("12\r");

break;

case 12:

cputs("13\r");

break;

case 13:

cputs("14\r");

break;

case 14:

cputs("15\r");

break;

case 15:

cputs("16\r");

break;

default:

break;

}

}

int main(){

textbackground(BLACK);

textcolor(WHITE);

clrscr();

window(20, 5, 60, 15);

int x = 1, y = 1;

gotoxy(x, y);

for(int i = 0; i <= 15; i++){

textbackground(i);

for(int j = 0; j <= 15; j++){

textcolor(j);

cputs("background: ");

nums(i);

cputs(" color: ");

setColors(j);

delay(500);

gotoxy(x,y--);

if(y < 1){

insline();

insline();

insline();

}

}

y += 3;

}

getch();

return 0;

}