**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. 8363 |  | Нерсисян А.С. |
| Преподаватель |  | Гречухин М.Н. |

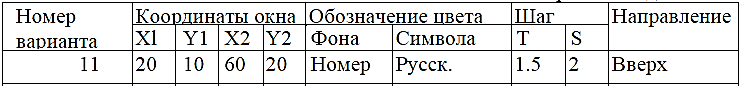
Санкт-Петербург

2019

**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. 
2. Организовать в окне вывод разноцветных сообщений со скроллингом окна.

**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 неко-торого 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.

**Алгоритмы**

Для получения двоичного представления числа типа long double (64) была использована конструкция union (объединение), в данном случае:

union type {

unsigned long long ll;

long double d;

}

что представляет собой объединение типов данных unsigned long long и long double в памяти (для двух переменных ll и d),(см. рис. 1), которое дает возможность записывать вещественное число в переменную типа long double и читать его как целый тип (unsigned long long).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *0* | *1* | *2* | *3* | *4* | *5* | *……………………….* | *60* | *61* | *62* | *63* |
| **1** | **0** | **0** | **1** | **1** | **1** | **…………………………** | **0** | **1** | **1** | **0** |

***unsigned long long long double***

У обеих типов используется одно и тоже место в памяти, т.е. если записать в **d** значение с плавающей точкой, скажем 3,1415…, то можно его взять как **unsigned long long**и сделать побитовые операции.

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 do while.

**Текст программы**

#include "conio.h"

#include "iostream.h"

#include "stdio.h"

#include "stdlib.h"

#include "dos.h"

#include "windows.h"

#include "time.h" // для delay();

union type

{

unsigned long long ll;

long double d;

};

void printLongDouble(long double x)

{

type m;

m.d = x;

for (int i = 16 \* sizeof(m.ll) - 1; i > -1; i--)

{

cprintf("%d", m.ll >> i & 1);

}

cprintf("\n\r");

}

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

{

union REGS r;

if(direction)

{

r.h.al = lines;

r.h.ah = direction;

}

else

{

r.h.al = lines;

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 GetPrintColor(unsigned char colors)

{

unsigned char bg = colors >> 4;

unsigned char text = colors & 15;

switch(bg)

{

case 0:

cprintf("1 ");

break;

case 1:

cprintf("2 ");

break;

case 2:

cprintf("3 ");

break;

case 3:

cprintf("4 ");

break;

case 4:

cprintf("5 ");

break;

case 5:

cprintf("6 ");

break;

case 6:

cprintf("7 ");

break;

case 7:

cprintf("8 ");

break;

case 8:

cprintf("9 ");

break;

case 9:

cprintf("10 ");

break;

case 10:

cprintf("11 ");

break;

case 11:

cprintf("12 ");

break;

case 12:

cprintf("13 ");

break;

case 13:

cprintf("14 ");

break;

case 14:

cprintf("15 ");

break;

case 15:

cprintf("16 ");

break;

}

switch(text)

{

case 0:

cprintf("Черный\r");

break;

case 1:

cprintf("Синий\r");

break;

case 2:

cprintf("Зеленый\r");

break;

case 3:

cprintf("Циановый\r");

break;

case 4:

cprintf("Красный\r");

break;

case 5:

cprintf("Пурпурный\r");

break;

case 6:

cprintf("Коричневый\r");

break;

case 7:

cprintf("Светло-серый\r");

break;

case 8:

cprintf("Темно-серый\r");

break;

case 9:

cprintf("Светло-голубой\r");

break;

case 10:

cprintf("Светло-зеленый\r");

break;

case 11:

cprintf("Светло-циановый\r");

break;

case 12:

cprintf("Светло-красный\r");

break;

case 13:

cprintf("Светло-пурный\r");

break;

case 14:

cprintf("Желтый\r");

break;

case 15:

cprintf("Белый\r");

break;

}

}

int main()

{

char t;

long double inputLongDouble;

scroll(0, 0, 0, 0, 25, 80, 7);

window(20, 10, 60, 20); // установка окна вывода в консоле

textattr(112); // устанавливает одновременно как цвета переднего плана, так и фона

scroll(0, 0, 9, 19, 19, 59, 7); // скроллинг снизу вверх

cprintf("\n\n\n\n\n\n\n\n\r");

do{

unsigned char i = 0;

do

{

textattr(i);

inputLongDouble = ((0.956358714)\*(rand() % 12501));

printLongDouble(inputLongDouble);

GetPrintColor(i);

i++;

delay(1500); // задержка в милисекундах

scroll(6, 2, 9, 19, 19, 59, i); // скроллинг

}while(i < 127); // т.к. 16 цветов фона + 16 цветов текста, всего 128 комбинациий, отсчет с 0я

cprintf("Что теперь? (y - снова сначала / n - выход): ");

do{

cscanf("%c", &t);

}while(t != 'y' && t != 'n');

}while(t == 'y');

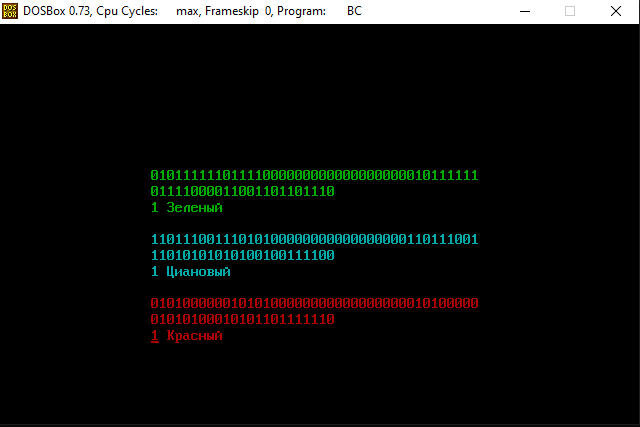
cprintf("\n\r ----- ВСЕ, 2-ЯЛАБА ГОТОВА! --------\n\r");

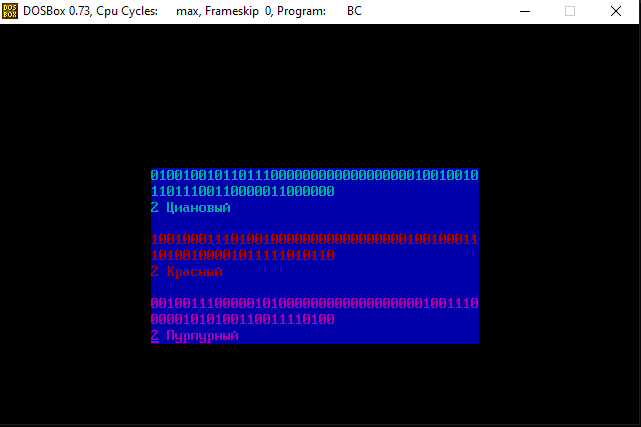
getch();

return 0;

}

**Примеры запуска программы**

****На рисунках демонстрирована работа программы. Программа выводит на экран внутреннее представление случайных чисел в двоичной системе счисления.

****

**Структурная схема аппаратных средств, используемых при выполнении программы с необходимой степенью детализации содержимого блоков.**

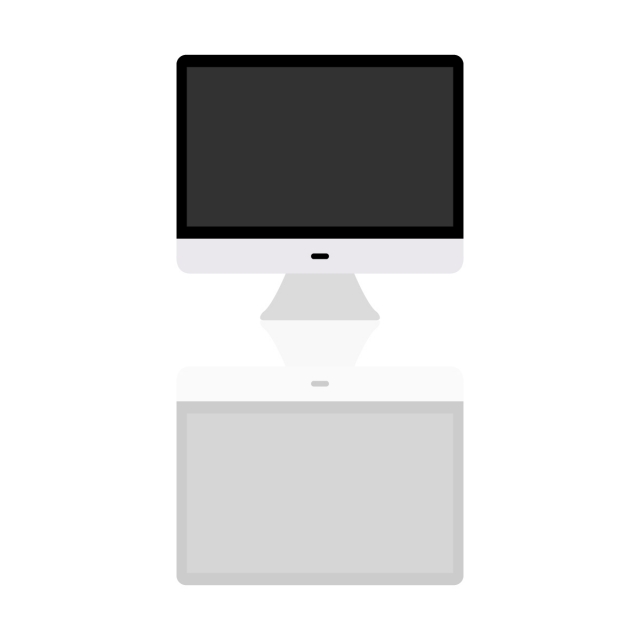
**ПЗУ**

**(RОM)**

**ОЗУ**

**(RAM)**

Контроллер



**ИП**

Монитор

Контроллер

**ЦП**

**(CPU)**

СИСТЕМНАЯ ШИНА