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)

Студентка гр. 3353 _____ Шинкарь К.Д.

Преподаватель _____ Гречухин М.Н.

Санкт-Петербург

2024

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: Написать программу, которая будет в окно с координатами (x1, y1, x2, y2) с шагами T (секунд) и S (строк) выводить 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. В моем случае (согласно 1 варианту) координаты окна (10, 5, 70, 15), цвет фона обозначается номером, цвет текста обозначается символами английского алфавита, временной шаг – 0,3 секунды, строки сдвигаются по одной вверх. В ходе лабораторной работы необходимо ознакомиться с организацией различных типов видеосистем, а также с различными типами отображения текстовой информации на экран монитора при использовании функций и стандартных библиотек C++.

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

В аппаратные средства для вывода информации на экран входят специальные электронные платы (это может быть видеоадаптер, либо адаптер дисплея, либо просто адаптер) и монитор (или же просто экран).

Видеоадаптеры – довольно сложные устройства, управляемые при помощи собственного микропроцессора, который сравним по мощности с центральным процессором компьютера.

Существует несколько стандартов, которым соответствуют все видеоадаптеры. Видеоадаптер состоит из двух основных частей: контроллера и видеопамяти (видеобуфера).

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 25 и более раз в секунду формирует изображение на экране. 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. Та видеостраница, которая постоянно "освежается" в данный момент, называется текущей. Видеоадаптер способен выполнять переключение текущей видеостраницы. Объем видеопамяти и число возможных страниц, зависит от конкретного адаптера.

Вывод информации на экран персонального компьютера может выполняться на трех уровнях:

1. на уровне MS-DOS с использованием функций прерывания 21h

2. на уровне BIOS с использованием функций прерывания 10h

3. непосредственным доступом к аппаратным средствам.

Вывод информации на уровне MS-DOS - мобильный, но самый медленный. Функции MS-DOS для вывода информации на экран вызывают драйвер консоли (выполняют вывод в специальный символьный файл CON). Если в системе инсталлирован специальный драйвер (например, ANSI.SYS), могут использоваться дополнительные средства по управлению экраном.

Суть расширенного управления состоит в передаче драйверу консоли специальных управляющих строк. Драйвер опознает начало управляющей строки по символу ASCII с кодом 27 (1Bh). Передаваемые на экран вслед за ним символы рассматриваются как параметры команды, которую выполняет драйвер. Таким образом, использование функций MS-DOS позволяет осуществить вывод через драйвер.

Другие достоинства функций MS-DOS - автоматическое позиционирование курсора и скроллинг экрана, реакция на нажатие комбинации клавиш Ctrl-Break. Недостатком является невозможность непосредственного управления курсором и атрибутом символов. На уровне MS-DOS работают функции стандартного вывода, а их прототипы содержатся в файле <stdio.h>.

Вывод на уровне BIOS дает более широкие возможности по управлению экраном. Именно эти функции используются драйверами MS-DOS для вывода информации на экран. Недостатком функций BIOS является невысокая скорость вывода. На уровне BIOS работают функции консольного вывода, а их прототипы помещены в файле <conio.h>.

Установку параметров активного текстового окна выполняет функция window(int , int ,int , int);. Она описывает активное текстовое окно: первая пара аргументов задает соответственно номера столбца и строки левого верхнего угла, вторая пара – правого нижнего угла. Функция clrscr() очищает все текстовое окно. Цвет "заливки" окна при очистке будет соответствовать

значению, установленному символической переменной attribute в описании окна (структурная переменная по шаблону text_info).

Функция textbackground(int newcolor) затрагивая установленный цвет символа. Цвет может быть или числом, или формироваться из символических констант, значения которых определяет перечислимый тип COLORS. Для цвета фона выбор ограничен значениями цветов 0-7. Если для цвета фона выбирается значение 8 - 15, то символы будут мерцать, так как бит мерцания установится в единицу, но цвет фона будет соответствовать значениям 0-7. Функция cprintf(const char *format,...) выполняет вывод информации с преобразованием по заданной форматной строке, на которую указывает format. Является аналогом функции стандартной библиотеки printf (), но выполняет вывод в пределах заданного окна. В отличие от printf () функция cprintf () иначе реагирует на специальный символ '\n': курсор переводится на новую строку, но не возвращается к левой границе окна. Поэтому для перевода курсора на начало новой строки текстового окна следует вывести последовательность символов CR-LF (0x0d, 0x0a). Остальные специальные символы воздействуют на курсор так же, как и в случае функций стандартного ввода-вывода. Функция возвращает число выведенных байтов, а не число обработанных полей, как это делает функция printf () .

Текст программы:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

void scroll(int direction, char leftrow, char leftcol, char rightrow, char rightcol, char attributes)
{
   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 = leftrow;
   r.h.cl = leftcol;
   r.h.dh = rightrow;
   r.h.dl = rightcol;
```

```
    r.h.bh = attributes;
    int86(0x10, &r, &r);
}

int main() {
    int i, j = 0;
    textbackground(0);
    clrscr() ;
    window(10, 5, 70, 15);
    textbackground(0);
    clrscr();
    _setcursortype(_NOCURSOR);
    for (i = 0; i < 8; i++) {
            textbackground(i);
            clrscr();
            for(j = 0; j <= 15; j++) {
               textcolor(j);
               gotoxy(1,10);
               cprintf("back: ");
               switch(i)
               {
                     case 0: cprintf("0");
               break;
                     case 1: cprintf("1");
               break;
                     case 2: cprintf("2");
               break;
                     case 3: cprintf("3");
               break;
                     case 4: cprintf("4");
               break;
                     case 5: cprintf("5");
               break;
                     case 6: cprintf("6");
               break;
                     case 7: cprintf("7");
               break;
               }
               switch(j)
               {
                     case 0: cprintf(" Text = BLACK");
               break;
                     case 1: cprintf(" Text = BLUE");
               break;
                     case 2: cprintf(" Text = GREEN");
               break;
                     case 3: cprintf(" Text = CYAN");
               break;
                     case 4: cprintf(" Text = RED");
               break;
                     case 5: cprintf(" Text = PINK");
               break;
                     case 6: cprintf(" Text = BROWN");
               break;
                     case 7: cprintf(" Text = LIGHTGRAY");
               break;
                     case 8: cprintf(" Text = DARKGRAY");
```

```
                break;
                        case 9: cprintf(" Text = LIGHTBLUE");
                break;
                        case 10: cprintf(" Text = LIGHTGREEN");
                break;
                        case 11: cprintf(" Text = LIGHTCYAN");
                break;
                        case 12: cprintf(" Text = LIGHTRED");
                break;
                        case 13: cprintf(" Text = LIGHTPINK");
                break;
                        case 14: cprintf(" Text = YELLOW");
                break;
                        case 15: cprintf(" Text = WHITE");
                break;
                }
            delay(300);
            gotoxy(wherex(), wherey()-1);
            scroll(6, 4, 9, 14, 69, i * 16 + j);
            scroll(6, 4, 9, 14, 69, i * 16 + j);
            }
    }
    return 0;
}
```

Результаты работы программы: