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. 1361 _____ Голубев Д. В.

Студентка гр. 1361 _____ Горбунова Д. А.

Преподаватель _____ Гречухин М. Н.

Санкт-Петербург

2022

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.

Формулировка задания

1. Изменить программу, полученную на предыдущей работе таким образом, чтобы в окно с координатами (15, 5, 65, 15) с шагами T (0,4 секунд) и S (2 строки) 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.

Теоретическая информация

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.

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. Каждому 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. Этот байт называется байтом атрибута.

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 консольного ввода-вывода C++ помещены в файле <conio.h>, предназначены для облегчения работы по созданию простейшего оконного интерфейса.

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. Строки и столбцы нумеруются, начиная от 1.

Фрейм окна C++ имеет следующую структуру:

```
struct text_info
{unsigned char
winright,  winbottom;   /* столбец, строка правого нижнего угла */
attribute, normattr;      /* атрибуты окна*/
currmode;                  /* текущий режим работы видеоадаптера */
screenheight;              /* полная высота экрана       */
screenwidth;               /* полная ширина экрана      */
curx, cury; };             /* строка, столбец текущей позиции курсора       */
```

Информация об активном окне доступна при выполнении функции

gettextinfo (struct text_info *t); При вызове эта функция заполняет поля структурной переменной описанной по шаблону text_info, указатель t на которую она получает.

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). Функции управления цветом фона и символа описаны далее.

Функция cputs(char *str) выводит строку символов в текстовое окно, начиная с текущей позиции курсора. На начало выводимой ASCII-строки указывает указатель str. Является аналогом функции стандартной библиотеки puts (), но выполняет вывод в пределах заданного окна и при выводе не

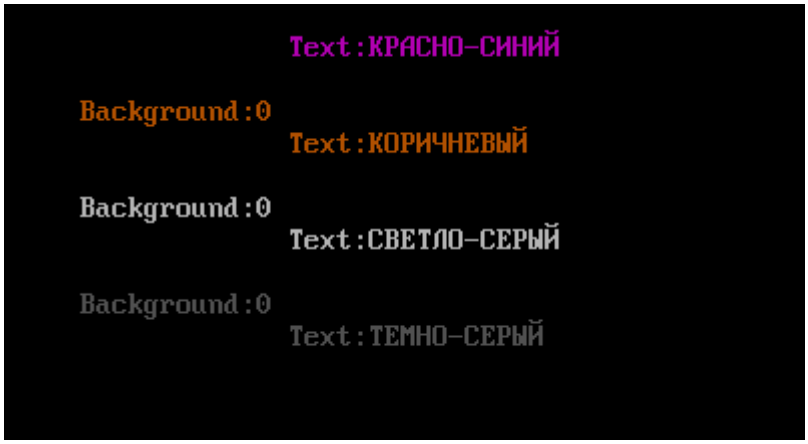
добавляет специальный символ '\n'. Реакция cputs() на специальный символ '\n' аналогична реакции cprintf(). Функция возвращает ASCII-код последнего выведенного на экран символа. В отличие от puts() в функции отсутствует возврат символа EOF. Другими словами, вывод происходит на экран в любом случае.

Функция textcolor(int newcolor) задает цвет символов, не затрагивая установленный цвет фона. Цвет может быть или числом, или формироваться из символических констант, значения которых определяет перечислимый тип COLORS.

Функция textbackground(int newcolor) задает цвет фона символов, не 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.

Результаты работы программы

После запуска программа создает окно необходимого размера и начинает цвет текста и цвет заднего фона. Также она выводит название цвета заднего фона и номер цвета текста (рисунок 1 и 2).



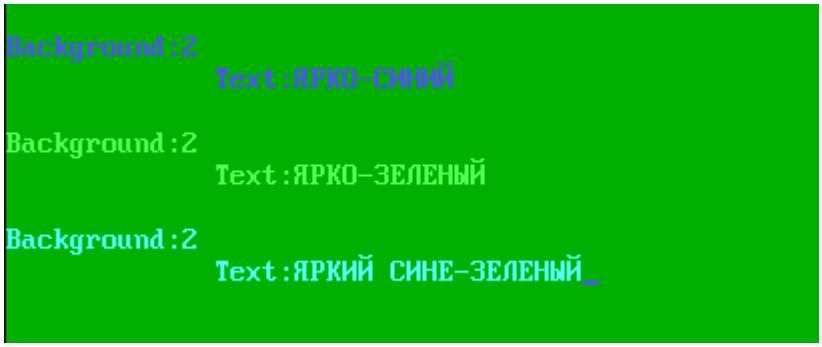
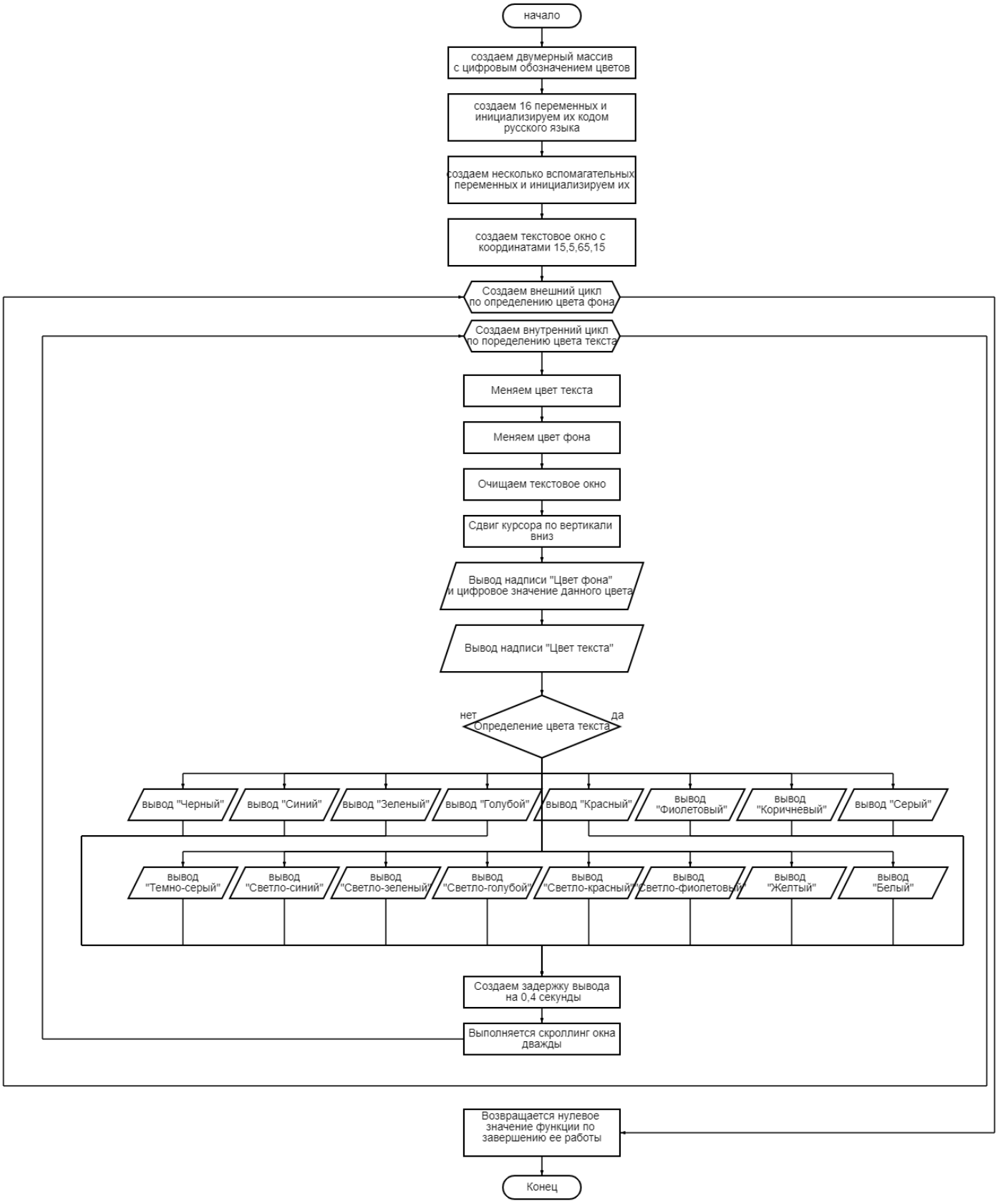
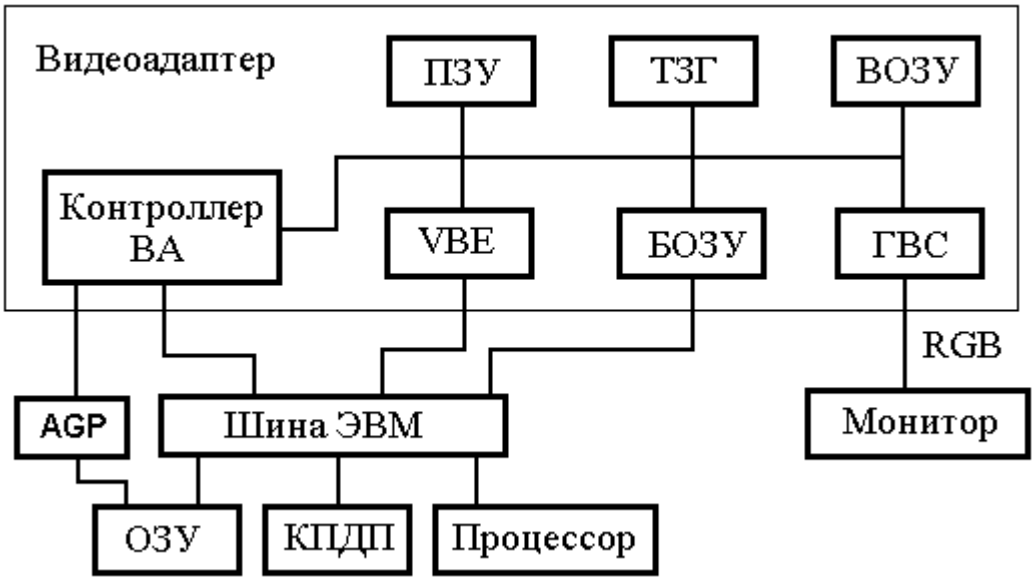


Рисунок 1, рисунок 2 — Результат работы программы

Приложение 1. Блок-схема алгоритма



Приложение 2. Структурная схема аппаратных средств



Приложение 3. Исходный код программы

```
#include <conio.h>
#include <dos.h>
void scroll(char x1, char y1, char x2, char y2, char attr)
{
union REGS r;
r.h.al=1;
r.h.ah=6;
r.h.ch = x1;
r.h.cl = y1;
r.h.dh = x2;
r.h.dl = y2;
r.h.bh = attr;
int86(0x10,&r,&r);
}

int main()
{
char colors_names[][16] =
{"0","1","2","3","4","5","6","7","8","9","10","11","12","13","14","15"};

char black_rus[] = "\227\205\220\215\233\211";
char blue_rus[] = "\221\210\215\210\211";
char green_rus[] = "\207\205\213\205\215\233\211";
char cyan_rus[] = "\221\210\215\205-\207\205\213\205\215\233\211";
char red_rus[] = "\212\220\200\221\215\233\211";
char magenta_rus[] = "\212\220\200\221\215\216-\221\210\215\210\211";
char brown_rus[] = "\212\216\220\210\227\215\205\202\233\211";
char lightgrey_rus[] = "\221\202\205\222\213\216-\221\205\220\233\211";
char darkgrey_rus[] = "\222\205\214\215\216-\221\205\220\233\211";
char lightblue_rus[] = "\237\220\212\216-\221\210\215\210\211";
char lightgreen_rus[] = "\237\220\212\216-\207\205\213\205\215\233\211";
char lightcyan_rus[] = "\237\220\212\210\211 \221\210\215\205-
\207\205\213\205\215\233\211";
char lightred_rus[] = "\237\220\212\216-\212\220\200\221\215\233\211";
char lightmagenta_rus[] = "\237\220\212\210\211 \212\220\200\221\215\216-
\221\210\215\210\211";
char yellow_rus[] = "\206\205\213\222\233\211";
char white_rus[] = "\201\205\213\233\211";

int xi=1,yi=1;
int atr_cnt=0;
window(15,5,65,15);
for(int i=0;i<8;i++)
```

```
{
       for(int j=0;j<16;j++)
       {
             textcolor(j);
             textbackground(i);
             clrscr();
             if (yi==5) yi=1;
        gotoxy(xi,yi);
        yi+=1;
             cprintf("Background:%s\n ", colors_names[i]);
             cprintf("Text:");
             switch(j)
        {
            case 0: cprintf("%s",black_rus); break;
            case 1: cprintf("%s",blue_rus); break;
            case 2: cprintf("%s",green_rus); break;
            case 3: cprintf("%s",cyan_rus); break;
            case 4: cprintf("%s",red_rus); break;
            case 5: cprintf("%s",magenta_rus); break;
            case 6: cprintf("%s",brown_rus); break;
            case 7: cprintf("%s",lightgrey_rus); break;
            case 8: cprintf("%s",darkgrey_rus); break;
            case 9: cprintf("%s",lightblue_rus); break;
            case 10: cprintf("%s",lightgreen_rus); break;
            case 11: cprintf("%s",lightcyan_rus); break;
            case 12: cprintf("%s",lightred_rus); break;
            case 13: cprintf("%s",lightmagenta_rus); break;
            case 14: cprintf("%s",yellow_rus); break;
            case 15: cprintf("%s",white_rus); break;

        }
       delay(400);
       for (int k = 0; k<2;k++) scroll(4,14,14,64,atr_cnt);
        atr_cnt++;
       }
}
return 0;
}
```