МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

Лабораторная работа №2
по дисциплине «Организация ЭВМ и систем»
ТЕМА: «ИССЛЕДОВАНИЕ ВИДЕОСИСТЕМЫ (ТЕКСТОВЫЙ РЕЖИМ)»

Студент гр. 1308, _____ Мельник Д. А.

Студент гр. 1308, _____ Лепов А. В.

Научный руководитель, _____ Костичев С. В.

Санкт-Петербург
2022

СОДЕРЖАНИЕ

1. КРАТКИЕ СВЕДЕНИЯ О ВИДЕОСИСТЕМАХ ПЭВМ

Аппаратные средства для вывода информации на экран включают видеоадаптер и монитор. Конструктивно видеоадаптеры – это сложные устройства, управляемые собственным микропроцессором, сравнимым по мощности с центральным процессором компьютера.

В самом общем виде видеоадаптер состоит из двух основных частей: контроллера и видеопамяти (видеобуфера). Видеоадаптер состоит из двух основных частей: контроллера и видеопамяти (видеобуфера). Основное назначение видеобуфера - хранение образа информации экрана. Изображение на экране строится из небольших точек - 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. Общий объем видеопамяти и число возможных страниц, зависит от конкретного адаптера.

Управление параметрами видеосистемы может выполняться на двух уровнях:

- на уровне портов видеоадаптера;
- обращением к функциям BIOS.

1.1. Текстовый режим работы видеосистем ПЭВМ

Режимы работы видеоадаптеров можно объединить в две группы:

- текстовые;
- графические.

Видеоадаптер, включённый в текстовый режим, рассматривает экран как совокупность 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 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. 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.

Вывод на монитор содержимого видеобуфера происходит, начиная с некоторого начального адреса, называемого смещением до видеостраницы. Если изменить значение смещения, произойдет переключение страницы, т.е. на экране возникнет образ другой страницы видеопамяти.

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.

1.2. Функции обслуживания текстового режима

- window() — устанавливает параметры активного текстового окна.
- clreol() — стирает в текстовом окне строку, на которую установлен курсор, начиная с текущей позиции курсора и до конца строки (до правой вертикальной границы окна).
- clrscr() — очищает все текстовое окно.
- delline() — стирает в текстовом окне всю строку текста, на которую установлен курсор.

- insline() — вставляет пустую строку в текущей позиции курсора со сдвигом всех остальных строк окна на одну строку вниз. При этом самая нижняя строка текста окна теряется.
- cprintf(const char *format, ...) — выполняет вывод информации с преобразованием по заданной форматной строке, на которую указывает format.
- cputs(char *str) — выводит строку символов в текстовое окно, начиная с текущей позиции курсора.
- movetext(int left, int top, int right, int bottom, int destleft, int desttop) — переносит окно, заданное координатами левого верхнего (left, top) и правого нижнего (right, bottom) углов, в другое место на экране, заданное координатами левого верхнего угла нового положения окна.
- putch(int ch) — выводит символ в текущей позиции текстового окна экрана.
- puttext(int left, int top, int right, int bottom, void *source) — выводит на экран текстовое окно, заданное координатами левого верхнего (left, top) и правого нижнего (right, bottom) углов.
- highvideo(void), lowvideo(void) и normvideo(void) — задают соответственно использование повышенной, пониженной и нормальной яркости для последующего вывода символов на экран.
- Функция textattr(int newattr) — устанавливает атрибут для функций, работающих с текстовыми окнами.
- textcolor(int newcolor) — задает цвет символов, не затрагивая установленный цвет фона.
- textbackground(int newcolor) — задает цвет фона символов, не затрагивая установленный цвет символа.

2. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

2.1. Общее задание

Написать программу, чтобы в окно с координатами (x1, y1, x2, y2) с шагами T (секунд) и S (строк) выводилась 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 или символьные обозначения цветов фона и символов Цвет окна должен соответствовать цвету фона.

Дополнить программу из п.1 скроллингом окна с направлением в соответствии с вариантом, используя функции прерывания 10h BIOS. Пример скроллинга приведен в методических указаниях.

Две отлаженные программы предъявить преподавателю.

2.2. Вариант задания (вариант 8)

Таблица. № 1. Вариант задания

Номер варианта	Координаты окна				Обозначение цвета		Шаг		Направление
	Xl	Y1	X2	Y2	Фона	Символа	T	S	
8	25	8	55	18	Англ.	Англ.	1.2	2	Вниз

3. РАСПРЕДЕЛЕНИЕ ОБЯЗАННОСТЕЙ В БРИГАДЕ

Мельник Д. А. – разработка алгоритма, построение блок-схем, написание отчёта, отладка программного кода;

Лепов А. В. – создание программного кода, отладка программного кода, построение блок-схем, написание отчёта.

4. СРЕДА РАЗРАБОТКИ

Компиляция программы была произведена в среде разработки TURBOC++.

5. БЛОК-СХЕМЫ АЛГОРИТМА

Ниже, на рис. 1-3 приведены блок-схемы алгоритма программы.

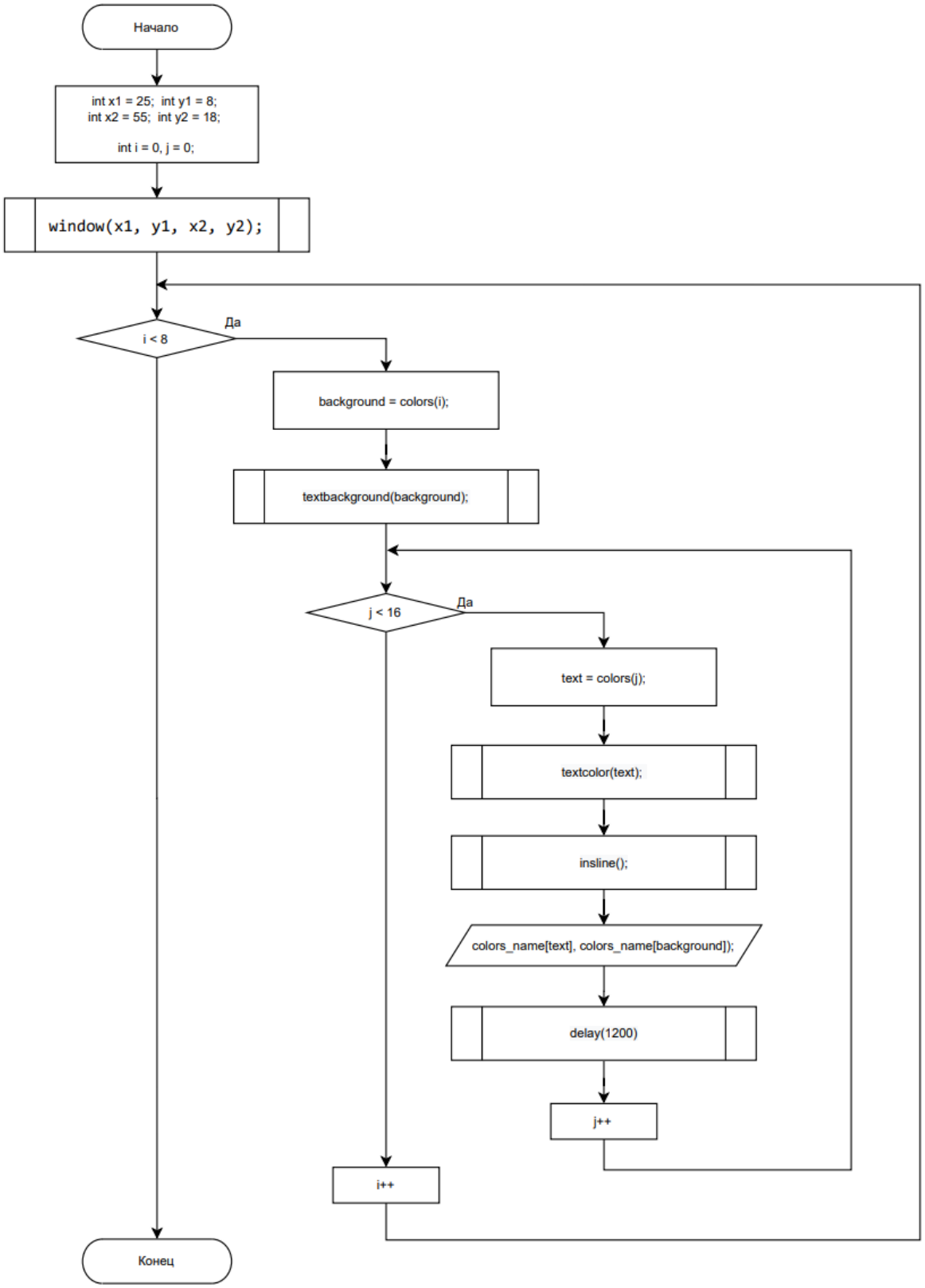

Рис. 1. Блок-схема алгоритма программы основного рабочего прототипа

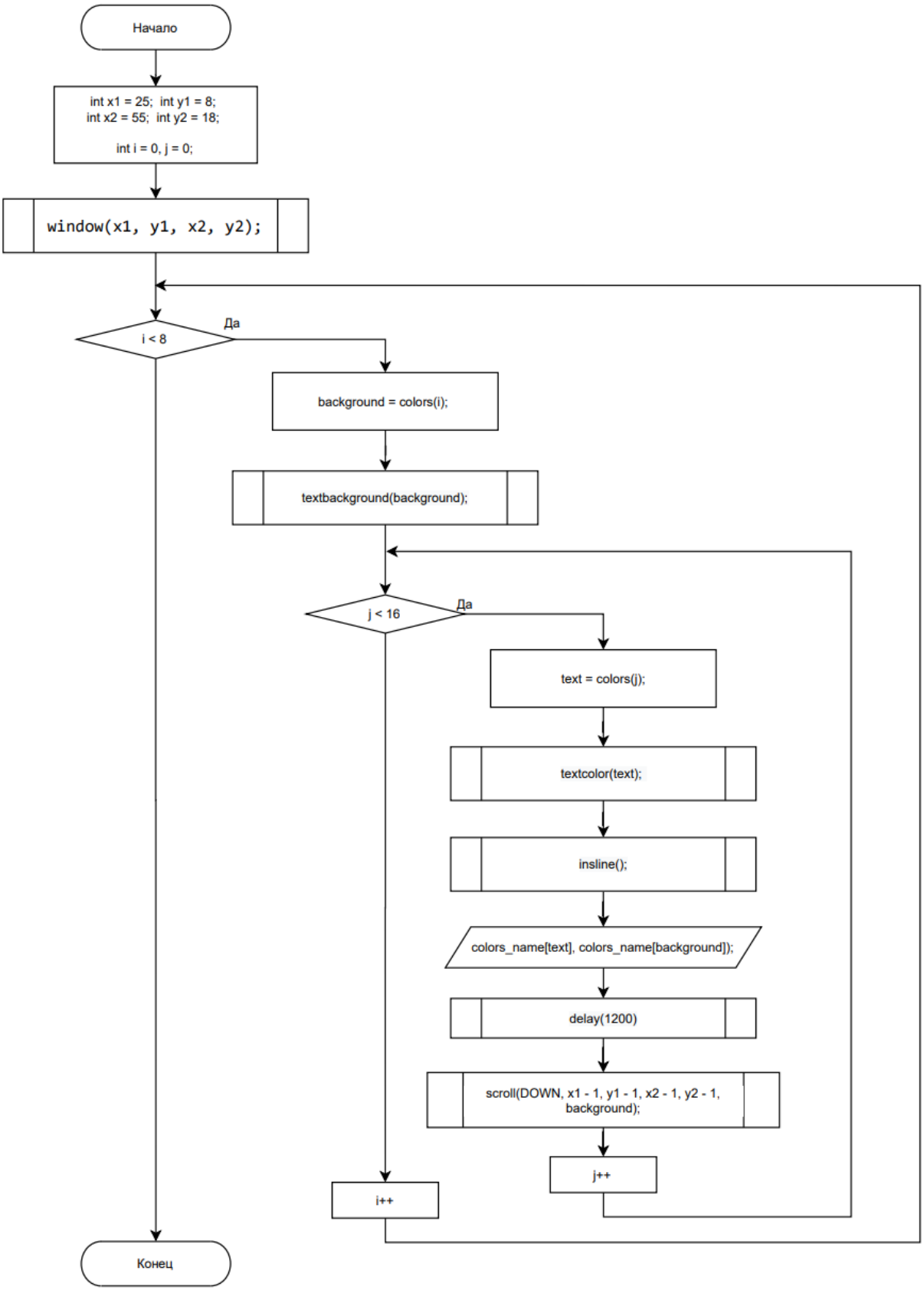

Рис. 2. Блок-схема алгоритма программы дополнения со скроллингом

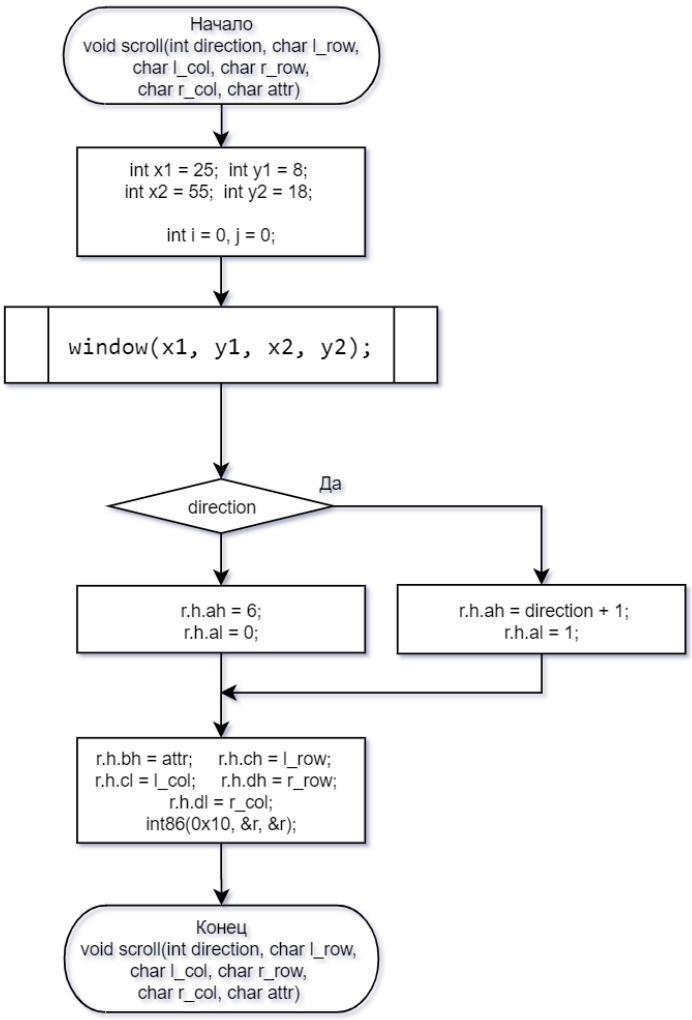

Рис. 3. Блок-схема алгоритма функции скроллинга

6. ПРИМЕРЫ КОМПИЛЯЦИИ ПРОГРАММЫ

Ниже, на рис. 4-7 приведены примеры запуска программы.

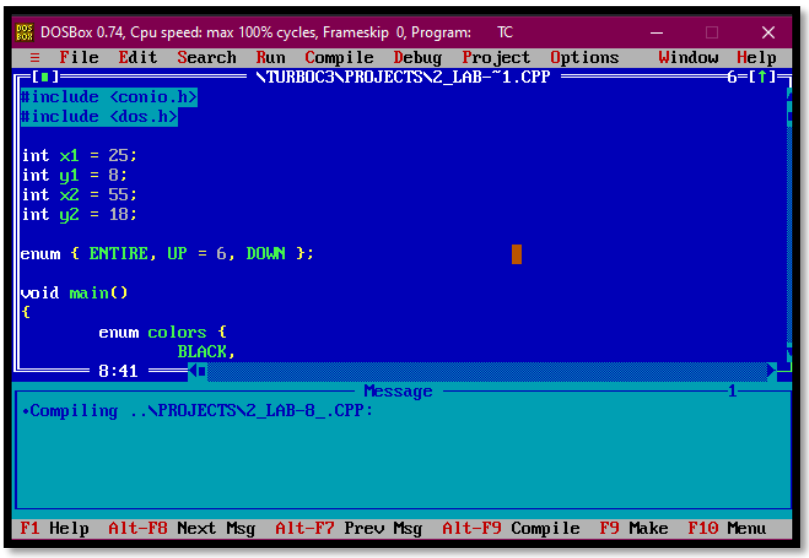

Рис. 4. Пример компиляции программного кода

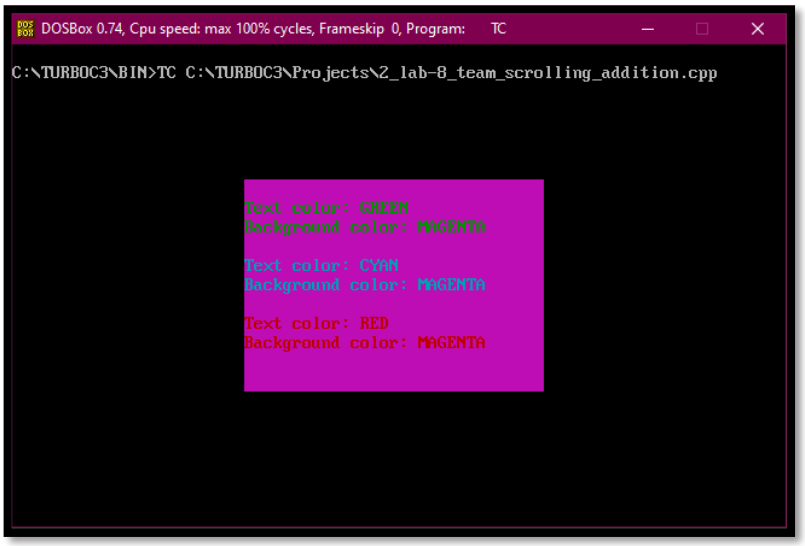

Рис. 5. Пример компиляции программного кода

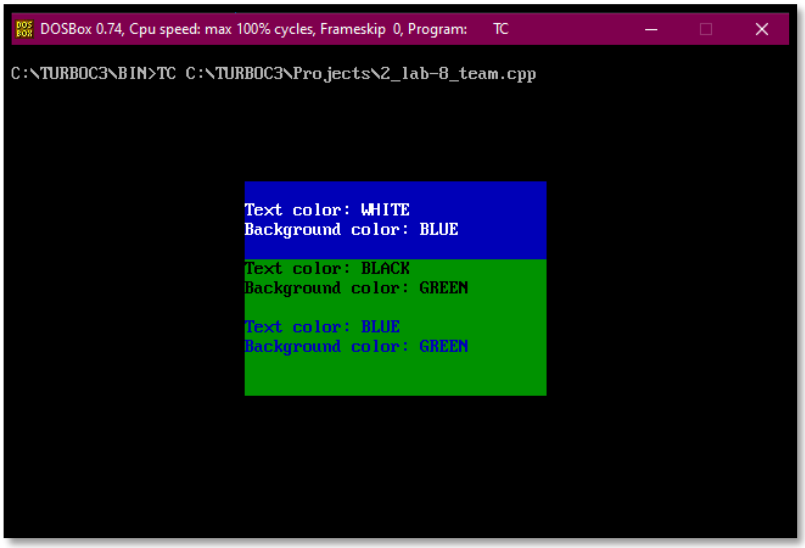

Рис. 6. Пример компиляции программного кода

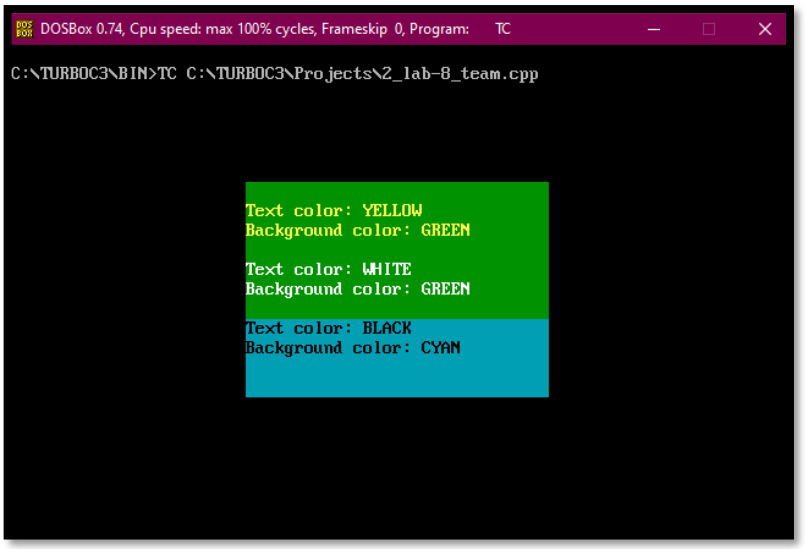

Рис. 7. Пример компиляции программного кода

7. ЛИСТИНГ ПРОГРАММНОГО КОДА

7.1. Программа № 1. Основной рабочий прототип

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//    ////////////////////////////////////////////////////////////////////    //
//    //                                                                //    //
//    //    Program for working with the text mode of the video system  //    //
//    //                                                                //    //
//    ////////////////////////////////////////////////////////////////////    //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

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

//////////////////////
//                  //
//    Coordinates   //
//                  //
//////////////////////
int x1 = 25;
int y1 = 8;
int x2 = 55;
int y2 = 18;

//////////////////////
//                  //
//    Main program  //
//                  //
//////////////////////
void main()
{
    enum colors {
        BLACK,
        BLUE,
        GREEN,
        CYAN,
        RED,
        MAGENTA,
        BROWN,
        LIGHTGRAY,
        DARKGRAY,
        LIGHTBLUE,
        LIGHTGREEN,
        LIGHTCYAN,
        LIGHTRED,
        LIGHTMAGENTA,
        YELLOW,
        WHITE
    };

    char colors_name[][16] = {
        "BLACK",
        "BLUE",
        "GREEN",
        "CYAN",
        "RED",
        "MAGENTA",
        "BROWN",
```

```
        "LIGHTGRAY",
        "DARKGRAY",
        "LIGHTBLUE",
        "LIGHTGREEN",
        "LIGHTCYAN",
        "LIGHTRED",
        "LIGHTMAGENTA",
        "YELLOW",
        "WHITE"
    };

    colors background;
    colors text;
    window(x1, y1, x2, y2);
    for (int i = 0; i < 8; i++)
    {
        background = colors(i);
        textbackground(background);
        for (int j = 0; j < 16; j++)
        {
            text = colors(j);
            textcolor(text);
            insline();
            cprintf("\r");
            cprintf("Text color: %s \r\nBackground color: %s\r\n\n",
colors_name[text], colors_name[background]);
            delay(1200); // time delay for 1.2 seconds
        }
    }
    getch();
    clrscr();
    getch();

}
```

7.2. Программа № 2. Дополнение со скроллингом

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//    ////////////////////////////////////////////////////////////////////    //
//    //                                                                //    //
//    //    Program for working with the text mode of the video system  //    //
//    //                                                                //    //
//    ////////////////////////////////////////////////////////////////////    //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

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

//////////////////////
//                  //
//    Coordinates   //
//                  //
//////////////////////
int x1 = 25;
int y1 = 8;
int x2 = 55;
int y2 = 18;

//////////////////////////////////
//                              //
```

```
//    Scrolling void function    //
//                               //
///////////////////////////////////
enum { ENTIRE, UP = 6, DOWN };
void scroll(int direction, char l_row, char l_col, char r_row, char r_col, char attr)
{
    union REGS r;
    if (direction)
    {
        r.h.ah = direction + 1;
        r.h.al = 1;
    }
    else
    {
        r.h.ah = 6;
        r.h.al = 0;
    }
    r.h.bh = attr;
    r.h.ch = l_row;
    r.h.cl = l_col;
    r.h.dh = r_row;
    r.h.dl = r_col;
    int86(0x10, &r, &r); // Interrupt character
}

////////////////////////
//                    //
//    Main program    //
//                    //
////////////////////////
void main()
{
    enum colors {
        BLACK,
        BLUE,
        GREEN,
        CYAN,
        RED,
        MAGENTA,
        BROWN,
        LIGHTGRAY,
        DARKGRAY,
        LIGHTBLUE,
        LIGHTGREEN,
        LIGHTCYAN,
        LIGHTRED,
        LIGHTMAGENTA,
        YELLOW,
        WHITE
    };

    char colors_name[][16] = {
        "BLACK",
        "BLUE",
        "GREEN",
        "CYAN",
        "RED",
        "MAGENTA",
        "BROWN",
        "LIGHTGRAY",
        "DARKGRAY",
        "LIGHTBLUE",
```

```
        "LIGHTGREEN",
        "LIGHTCYAN",
        "LIGHTRED",
        "LIGHTMAGENTA",
        "YELLOW",
        "WHITE"
    };

    colors background;
    colors text;
    window(x1, y1, x2, y2);
    for (int i = 0; i < 8; i++)
    {
        background = colors(i);
        textbackground(background);
        for (int j = 0; j < 16; j++)
        {
            text = colors(j);
            textcolor(text);
            insline();
            cprintf("\r");
            cprintf("Text color: %s \r\nBackground color: %s\r\n\n",
colors_name[text], colors_name[background]);
            delay(1200);  // time delay for 1.2 seconds
            scroll(DOWN, x1 - 1, y1 - 1, x2 - 1, y2 - 1, background);
        }
    }
    getch();
    clrscr();
    getch();
}
```