**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 3**

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

Тема: **«ИССЛЕДОВАНИЕ ВИДЕОСИСТЕМЫ (ГРАФИЧЕСКИЙ РЕЖИМ)»**

|  |  |  |
| --- | --- | --- |
| Студенты |  |  |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2022

**Цель работы**

Изучить работу с видеосистемой в графическом режиме, вывод графика заданной функции с масштабированием и разметкой осей

**Задание (вариант №13)**

1. Разработать программу для вывода на экран графика функции: Sin2(x)- Cos3(x) на [π/2;5π]

2. Произвести разметку осей и проставить истинные значения точек.

3. Найти максимальное значение функции на заданном интервале вывести в отдельное окно на экране вместе с графиком.

**Программная среда**

Вся программа написана на языке программирования Turbo С++, компилировалась в DOS BOX.

**Краткие сведения о видеосистемах ПЭВМ, графическом режиме их работы и функциях обслуживания графического режима.**

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.

Использование графики в языке С++ - это многошаговый процесс. Прежде всего необходимо определить тип видеоадаптера. Затем устанавливается подходящий режим его работы и выполняется инициализация графической системы в выбранном режиме. После этого становятся доступными для исполь­зования функции графической библиотеки graphics.h для построения основных графических примитивов, появляется возможность вывода текста с использованием различных шрифтов.

Использование библиотеки графики намного сокращает объем программирования для вывода основных графических примитивов. С++ "маскирует" многие технические детали управления оборудованием, о которых пользователь должен быть осведомлен при работе с видеоадаптером через порты или BIOS. Платой за эти удобства является значительное увеличение размера ЕХЕ-файлов. Весь код библиотеки графики разбивается на две части: немобильную, которая зависит от типа видеоадаптера и мобильную.

Немобильная часть представляет собой так называемый BGI-драйвер (BGI - Borland Graphics Interface). Драйвер является обработчиком прерывания 10h, который должен дополнить системный обработчик до того, как будут использоваться мобильные функции. Перед завершением программы таблица векторов прерывания восстанавливается.

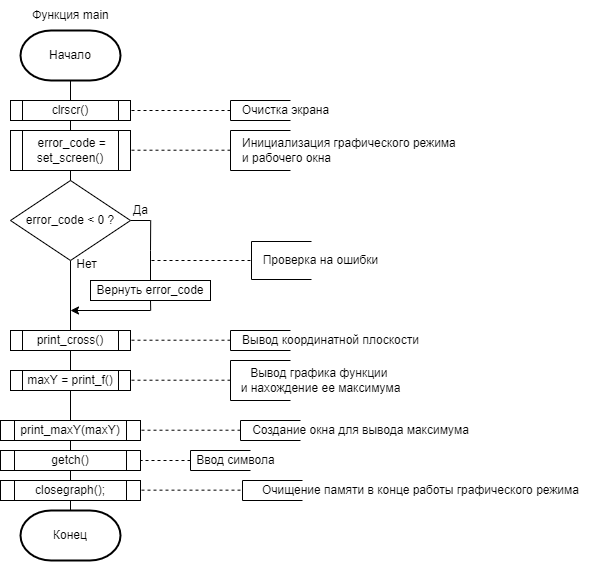
Основные функции, выполняемые BGI-драйвером, сводятся к установке и обновлению ряда внешних переменных, которые могут изменяться как функциями системного обработчика прерывания 10h (например, при переключении видеорежима, изменении регистров палитры и т.п.), так и мобильными функциями библиотеки графики. С++ включает целую коллекцию драйверов для каждого из типов адаптеров, хранимых обычно в отдельном поддиректории. Система графики является открытой для расширений, так как позволяет использовать и собственные BGI-драйверы.

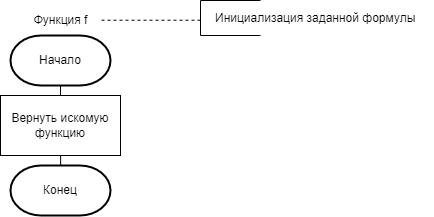
Совокупность внешних переменных библиотеки графики и особенностей поведения мобильных функций образует модель графики С++. Подробно эти элементы модели рассматриваются в следующих подразделах.

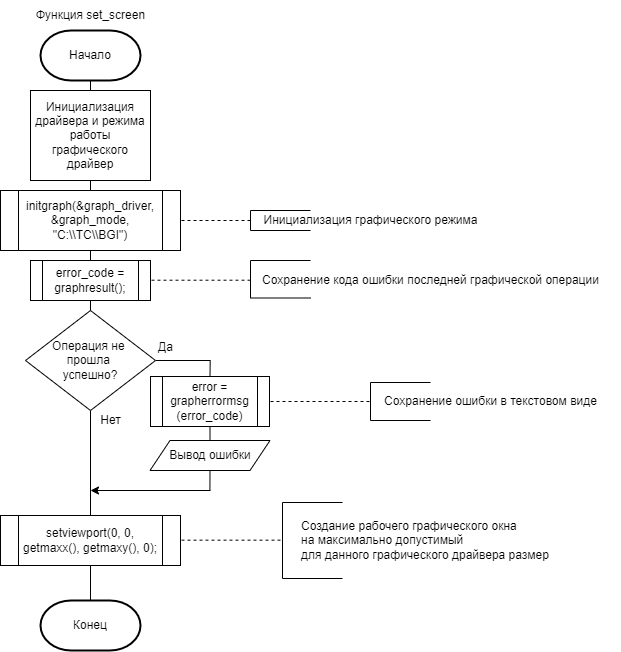
Прежде чем использовать функции графической библиотеки С++, необходимо инициализировать систему графики - загрузить соответствующий адаптеру или режиму BGI-драйвер, установить в начальные значения внешние переменные и константы, выбрать шрифт и т.д.

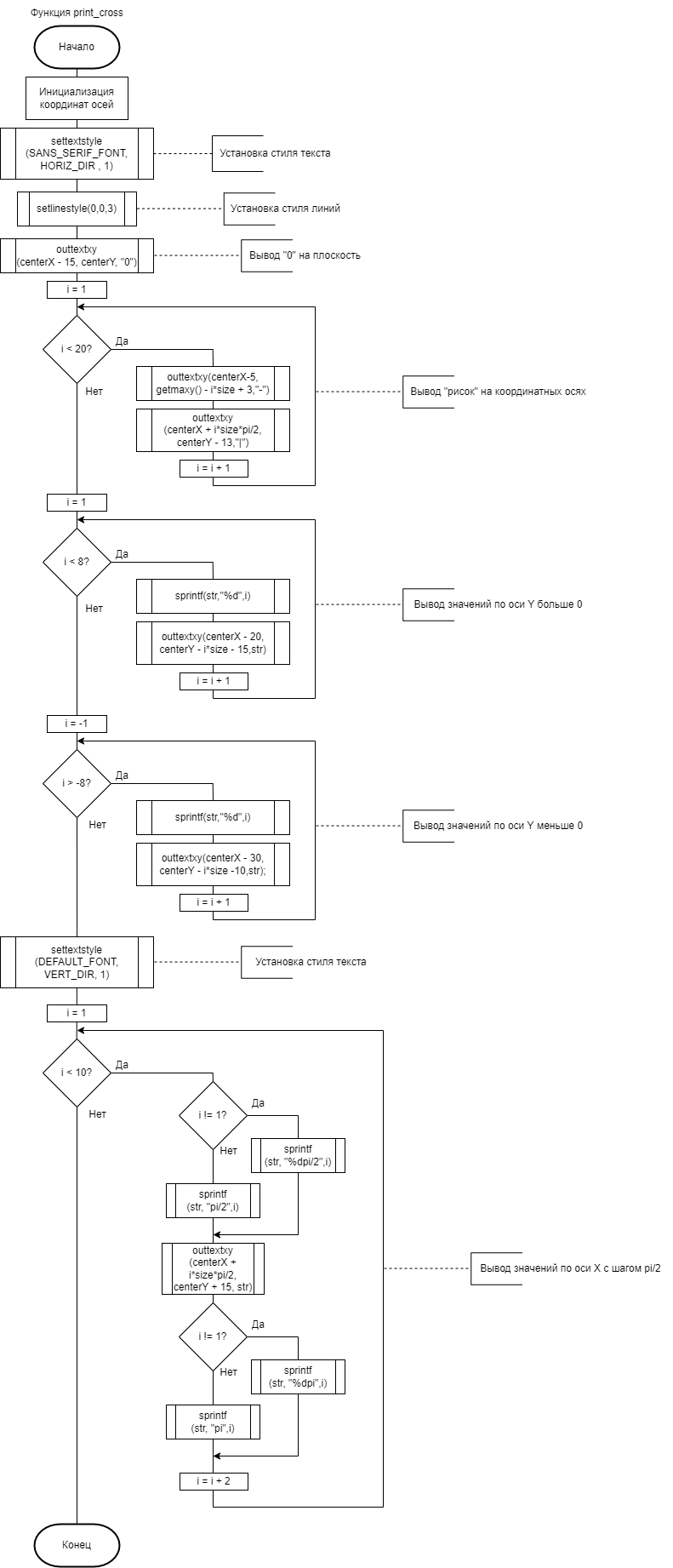
Графические режимы, поддерживаемые библиотекой графики, задаются символическими константами, описанными в заголовочном файле <graphics.h> в перечислимом типе graphics\_modes.

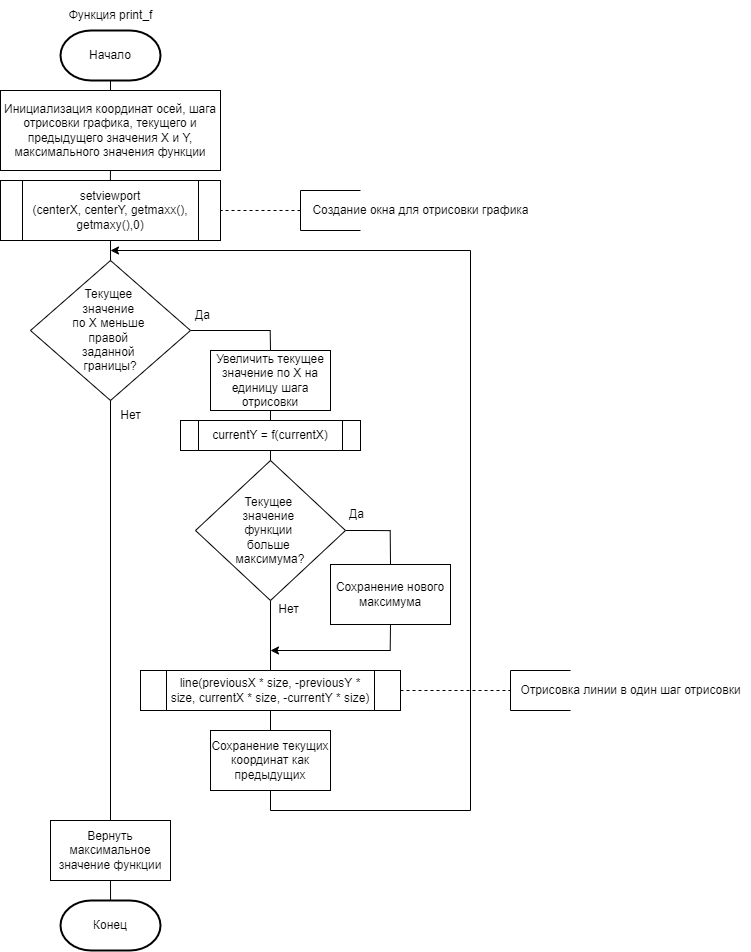
**Схемы алгоритмов**

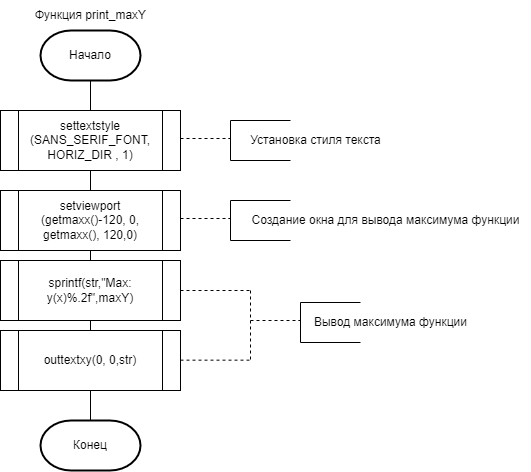












**Текст программы**

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<stdio.h>

#include<iostream.h>

const double pi = 3.1415926535;

const double a = pi / 2;

const double b = pi \* 5;

const double size = 32;

double f(double x)

{

return (pow(sin(x), 2) - pow(cos(x), 3));

}

int set\_screen()

{

int graph\_driver = DETECT;

int graph\_mode;

initgraph(&graph\_driver, &graph\_mode, "C:\\TC\\BGI");

int error\_code;

error\_code = graphresult();

if (error\_code != grOk)

{

char\* error = grapherrormsg(error\_code);

getch();

printf("%s\n", error);

return error\_code;

}

setviewport(0, 0, getmaxx(), getmaxy(), 0);

return 0;

}

void print\_cross()

{

const int centerX = getmaxx()/10, centerY = getmaxy()/2;

char \*str;

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR , 1);

setlinestyle(0,0,3);

line(centerX, 0, centerX, getmaxy());

line(0, centerY, getmaxx(), centerY);

outtextxy(centerX - 15, centerY, "0");

for(int i = 1; i < 20;i++)

{

outtextxy(centerX-5, getmaxy() - i\*size + 3,"-");

outtextxy(centerX + i\*size\*pi/2, centerY - 13,"|");

}

for(i = 1; i < 8; i++)

{

sprintf(str,"%d",i);

outtextxy(centerX - 20, centerY - i\*size - 15,str);

}

for(i = -1; i > -8; i--)

{

sprintf(str,"%d",i);

outtextxy(centerX - 30, centerY - i\*size -10,str);

}

settextstyle(DEFAULT\_FONT, VERT\_DIR, 1);

for(i = 1; i < 10; i=i+2)

{

if (i!=1) sprintf(str, "%dpi/2",i);

else sprintf(str, "pi/2",i);

outtextxy(centerX + i\*size\*pi/2, centerY + 15, str);

if (i!=1) sprintf(str, "%dpi",i);

else sprintf(str, "pi",i);

outtextxy(centerX + i\*size\*pi, centerY + 15, str);

}

}

float print\_f()

{

char \*str;

const int centerX = getmaxx()/10, centerY = getmaxy()/2;

double step = (b - a) / 50000;

double currentX = a, currentY = f(currentX);

double previousX = currentX, previousY = currentY;

float maxY = currentY;

setviewport(centerX, centerY, getmaxx(), getmaxy(),0);

while (currentX < b)

{

currentX += step;

currentY = f(currentX);

if (currentY > maxY) maxY = currentY;

line(previousX \* size, -previousY \* size, currentX \* size, -currentY \* size);

previousX = currentX;

previousY = currentY;

}

return maxY;

}

void print\_maxY(float maxY)

{

char \*str;

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR , 1);

setviewport(getmaxx()-120, 0, getmaxx(), 120,0);

sprintf(str,"Max: y(x)%.2f",maxY);

outtextxy(0, 0,str);

}

int main(void)

{

clrscr();

float maxY;

int error\_code = set\_screen();

if (error\_code<0) return error\_code;

print\_cross();

maxY = print\_f();

print\_maxY(maxY);

getch();

closegraph();

return 0;

}