**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 гр. 3353 |  | Карпенко А.Ю. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2024

**Цель работы:** изучение работы с видеосистемой в графическом режиме, вывод графика заданной функции с масштабированием и разметкой осей.

**Задание:** Разработать программу для вывода на экран графика заданной функции. Произвести разметку осей и проставить истинные значения точек. Найти максимальное значение функции на заданном интервале и вывести в отдельное окно на экране.

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

**Использование графики в C++**

Работа с графикой в языке C++ требует выполнения нескольких шагов. Основной задачей является определение типа видеоадаптера и установка подходящего графического режима. После этого производится инициализация графической системы, и можно начинать работу с функциями библиотеки **graphics.h**. Эта библиотека позволяет значительно упростить вывод графических примитивов, таких как линии, окружности, прямоугольники, эллипсы и другие фигуры, а также позволяет работать с текстом, используя различные шрифты.

**Инициализация графического режима**

Прежде всего, нужно выбрать и установить графический режим, который поддерживается вашим видеоадаптером. Для этого можно использовать следующие функции:

* **getgraphmode()** – возвращает текущий графический режим.
* **setgraphmode(int mode)** – устанавливает видеосистему в заданный графический режим.
* **getmaxx() и getmaxy()** – возвращают ширину и высоту экрана в пикселях для текущего режима.

После выполнения программы можно вернуть компьютер в текстовый режим с помощью функции **restorecrtmode()**.

**Основные графические функции**

1. **putpixel(int x, int y, int color)** – выводит пиксель с указанными координатами и цветом. Если необходимо "стереть" пиксель, можно вывести его с цветом фона.
2. **line(int x1, int y1, int x2, int y2)** – рисует отрезок прямой между точками (x1, y1) и (x2, y2). Используется текущий цвет и стиль.
3. **rectangle(int left, int top, int right, int bottom)** – выводит контур прямоугольника, используя координаты верхнего левого и нижнего правого углов.
4. **setcolor(int color)** – устанавливает цвет, который будет использоваться для всех графических операций.
5. **floodfill(int x, int y, int border)** – выполняет заливку области, ограниченной линией с цветом **border**, начиная с точки (x, y). Эта функция позволяет закрасить внутреннюю или внешнюю часть замкнутой области.
6. **setfillstyle(int pattern, int color)** – задает стиль и цвет для заливки фигур. Например, можно использовать предопределенные шаблоны заполнения.

**Преимущества и недостатки использования графической библиотеки**

Использование библиотеки **graphics.h** в C++ позволяет значительно сократить количество кода, необходимого для работы с графикой, так как она скрывает многие низкоуровневые детали управления видеоадаптером и графическим режимом. Однако это увеличивает размер исполняемых файлов, а также накладывает зависимость от конкретного типа видеоадаптера.

Графическая система C++ базируется на модели Borland Graphics Interface (BGI). BGI-драйвер отвечает за обработку прерываний и работу с графическими режимами, дополняя системный обработчик прерываний. Для каждого типа видеоадаптера в компиляторе предусмотрены отдельные драйверы, которые помогают определять и устанавливать графические режимы.

Текст программы

#include <conio.h>

#include <graphics.h>

#include <math.h>

#include <stdio.h>

#include <string.h>

#define M\_PI 3.14159265358979323846

int Xmax, Ymax, X0, X1, Y0, Y1;

float Rmax = 0.0;

void drawf(int pixelToXaxis, int pixelToYaxis) {

int startXcoordinate;//начальная точка графика на оси X.

int endXcoordinate; //конечная точка графика на оси X.

int N; //количество шагов для прорисовки графика.

double dx2; //шаг изменения по оси X.

double x; //текущая координата X для вычисления функции.

int i;

float R; //значение функции

startXcoordinate = X0 + (pixelToXaxis) \* 0.5; // Начало графика

endXcoordinate = X0 + (pixelToXaxis) \* 6; // Для диапазона от π/2 до 6π

N = endXcoordinate - startXcoordinate;

// Изменяем диапазон на от π/2 до 6π

dx2 = (6 \* M\_PI - (M\_PI/2)) / N; // Шаг для изменения X

x = M\_PI / 2; // Начало с π/2

for (i = 0; x <= 6 \* M\_PI; x += dx2, i++) {

R = pow(sin(x), 3.0) + pow(cos(x), 3.0); // Вычисление sin^3(x) + cos^3(x)

if (Rmax < R) Rmax = R;

putpixel(startXcoordinate + i, Y0 - 225 - (int)(R \* pixelToYaxis), RED);

// рисует пиксель на экране в координатах x, y, где X вычисляется на основе значения X в функции, а Y — на основе значения функции

//225: фиксированное смещение вверх от точки Y0, чтобы ось X не была на самом краю окна.

}

}

int main() {

char cRmax[10];

char str[24];

int driver, mode, graph\_error;

int zeroXaxis;

int pixelToXaxis;

int pixelToYaxis;

int i;

int j;

char kPi[10];

char kPj[10];

//Очищает экран, определяет драйвер графики, инициализирует графический режим.

clrscr();

driver = DETECT;

initgraph(&driver, &mode, "C://Turboc3//BGI");

graph\_error = graphresult();

//Если инициализация прошла успешно, она возвращает значение константы grOk.

//Если произошла ошибка, возвращается код ошибки, который записывается в переменную graph\_error.

//Проверка успешной инициализации графического режима. Если ошибка — выводится сообщение и программа завершается.

if (graph\_error != grOk) {

cprintf("ERROR WITH GRAPH %d", graph\_error);

getchar(); //выходит сообщение об ошибке с кодом. далее нажми, чтобы продолжить

return 255;

}

//Получение максимальных размеров окна графики и установка начальных координат осей.

Xmax = getmaxx(); // возвращает максимальное значение координаты по оси X (ширину графического окна в пикселях).

Ymax = getmaxy(); //возвращает максимальное значение координаты по оси Y (высоту графического окна в пикселях).

X0 = 20; //левый край графической области на 20 пикселей от левого края экрана.

Y0 = Ymax - 19; //координату по оси Y, выше нижнего края экрана на 19

X1 = Xmax - 19; // на 19 пикселей левее от правого края экрана

Y1 = 10; //задаёт верхнюю границу графической области на 10 пикселей ниже верхнего края экрана

//Устанавливается стиль линии, цвет, рисуется прямоугольник, который заливается чёрным цветом внутри.

setlinestyle(0, 1, 3); // вид линии (стиль - сплошная, шаблон - игнорируется, толщина - 3 пикселя)

setcolor(WHITE); //функция устанавливает текущий цвет рисования

rectangle(2, 0, Xmax - 2, Ymax - 30); //Прямоугольник рисуется в области, которая охватывает почти весь экран, оставляя небольшой отступ (2 пикселя слева и сверху и 30 пикселей снизу).

setfillstyle(SOLID\_FILL, BLACK); //устанавливает стиль заливки для замкнутых фигур (сплошная заливка, черная)

floodfill(20, 20, WHITE); //заполняет замкнутую область, начиная с точки (x, y), до границы указанного цвета.

setlinestyle(0, 1, 1); //снова меняем стиль линии

//Устанавливается нулевая ось X, рисуются оси X и Y.

zeroXaxis = Y0 - 225; //это переменная, которая определяет положение оси X на графике

line(X0, Y0 - 15, X0, Y1 - 20); // рисуется вертикальная линия (ось Y),

line(X0, zeroXaxis, X1, zeroXaxis); // рисует горизонтальную линию, которая будет представлять ось X.

//Рисуются подписи на осях и заголовок графика.

outtextxy(X0 + 40, Y1 + 40, "sin^3(x)+ cos^3(x)"); //функция для вывода текста text на графическое окно в координатах (x, y).

outtextxy(X1 + 3, zeroXaxis + 5, "x"); //обозначение оси х

outtextxy(X0 + 3, Y1 + 3, "y"); // обозначение оси у

settextstyle(SMALL\_FONT, HORIZ\_DIR, 4); //функция, задающая стиль шрифта для текста (тип шрифта, направление вывода текста, размер шрифта)

//Определяется масштабирование по осям X и Y и вызывается функция для рисования графика.

pixelToXaxis = (X1 - X0) / 10;

pixelToYaxis = 20;

drawf(pixelToXaxis, pixelToYaxis);

// Разметка оси X

for (i = 1; i <= 7; ++i) {

line(X0 + pixelToXaxis \* i, zeroXaxis, X0 + pixelToXaxis \* i, zeroXaxis - 5);

sprintf(kPi, "%dPI", i); // Метки для π, 2π, 3π и так далее

outtextxy(X0 + pixelToXaxis \* i - 3, zeroXaxis - 20, kPi);

}

//Рисуются метки на оси Y.

for (j = 1; j <= 10; ++j) {

line(X0, zeroXaxis - pixelToYaxis \* j, X0 + 10, zeroXaxis - pixelToYaxis \* j);

sprintf(kPj, "%d", j);

if (j != 10) {

outtextxy(X0 - 10, zeroXaxis - pixelToYaxis \* j - 3, kPj);

} else {

outtextxy(X0 - 15, zeroXaxis - pixelToYaxis \* j - 3, kPj);

}

}

//Вычисляется и отображается максимальное значение функции

sprintf(cRmax, "%f", Rmax); //записывает в строковую переменную cRmax значение переменной Rmax

strcpy(str, "MAX(f(x)) = "); // копирует текст "MAX(f(x)) = " в строковую переменную str

strcat(str, cRmax); //выполняет "склейку" двух строк.

//рисование рамки, заливку цветом и вывод текста на график.

setlinestyle(0, 1, 3);

setcolor(RED);

rectangle(X0 + 25, Ymax - 25, X0 + 200, Ymax);

//setfillstyle(SOLID\_FILL, BLACK);

floodfill(X0 + 30, Ymax - 10, BLACK);

setcolor(WHITE);

outtextxy(X0 + 30, Ymax - 20, str);

getchar(); // ждет нажатия от пользователя

closegraph(); //Закрывает графический режим и освобождает ресурсы, выделенные для работы с графикой.

return 0;

}

**Вывод**

В ходе лабораторной работы мы разработали программу, которая выводит на экран график заданной функции. Также мы произвели разметку осей и проставили истинные значения точек. Максимальное значение функции на заданном интервале — 1,000000. Это значение выводится в отдельное окно на экране.

