**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)»**

**Вариант 2**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3354 |  | Григорьев В.С. |
| Преподаватель |  | Анисимов А.В. |

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Функция | Диапазон аргумента | |
| Начало | Конец |
| 2 | Sin3(x/2)+Sqrt(x) | 3π/2 | 16π |

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, текстовом режиме их работы и функциях обслуживания текстового режима.**

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: отрезков прямых линий, окружностей, эллипсов, прямоугольников, секторов, дуг и т.д., появляется возможность вывода текста с использованием различных шрифтов.

С++ поддерживает фиксированное число драйверов, каждый из которых, в свою очередь, поддерживает ряд режимов. Как тип драйвера, так и режим могут быть заданы числом или символической константой.

Две функции позволяют определить ширину и высоту экрана в пикселах для текущего видеорежима: getmaxx() и getmaxy().

Выводит контур прямоугольника, заданного координатами левого верхнего (left, top) и правого нижнего (right, bottom) углов. Координаты углов задаются относительно координат левого верхнего угла текущего графического окна. Контур выводится линией текущего цвета и стиля. Цвет контура устанавливается функцией setcolor(). Стиль линии задается функцией setlinestyle().

Заполнение цветом прямоугольной области выполняется с помощью функций setfillstyle и floodfill.

Выбирает один из предопределенных стилей заполнения. Значение pattern идентифицирует стиль. Аргумент color задает цвет, используемый для пикселов по заданному шаблону. Задав стиль, перейдем к самому заполнению, выполняемому функцией floodfill():

Заполняет текущим стилем область экрана, ограниченную непрерывной линией с цветом border, начиная с точки с координатами (х, у). Функция заполняет область либо внутри замкнутой линии, либо вне ее. Это зависит от положения начальной точки: если она лежит внутри области, заполняется внутренняя область; если точка лежит вне замкнутой области, заполняется внешняя область; если точка лежит точно на линии цвета border, заполнение не производится. Заполнение начинается с начальной точки и продолжается во всех направлениях, пока не встретится пиксел с цветом border. Цвет border должен отличаться от цвета заполнения, в противном случае будет заполнен весь экран.

Построение графика функции выполняется с помощью функции drawf():

На вход подаем количество точек по оси X, для которых будет рассчитано значение функции.

Переменная dx отвечает за шаг аргумента x, шаг рассчитывается в соответствии с диапазоном аргумента, представленном в задании. Далее создан цикл, осуществляющий поиск максимума значения функции на представленном диапазоне аргумента x. После того, как максимум найден запускаем цикл отрисовки заданной функции: рассчитываем значение в каждой точке диапазона, и устанавливаем найденную точку на экран с помощью функции putpixel():

Определяет, лежит ли пиксел с координатами (х, у) в текущем графическом окне, и, если лежит, выводит на экран пиксел, код цвета которого равен pixelcolor. В противном случае цвет пиксела не изменяется. Используя функцию putpixel(), можно "стереть" пиксел, если вывести его с кодом цвета фона. Для того, чтобы пиксел лежал в созданном окне необходимо нормировать координату значения функции, привести ее к целочисленному значению, а потом привести эту координату к соответствующим параметрами окна.

**Текст программы**

#include <graphics.h>

#include <conio.h>

#include <math.h>

#include <stdio.h>

const float PI = 3.14159265358979323846;

const float START = 0; // Начало графика

const float END = 18 \* PI; // 18π

const int NUMBER\_OF\_STEPS = 200;

const int AXIS\_OFFSET = 300; // Смещение осей влево

// Функция для отображения

float myFunction(float x) {

return pow(sin(x / 2), 3) + sqrt(x);

}

// Функция для установки осей

void putAxis() {

int maxX = getmaxx();

int maxY = getmaxy();

int midX = maxX / 2 - AXIS\_OFFSET; // Сдвинуть ось X влево

int midY = maxY / 2;

// Рисуем оси

line(0, midY, maxX, midY); // Горизонтальная ось

line(midX, 0, midX, maxY); // Вертикальная ось

// Оси X

for (int i = 0; i <= 18; ++i) {

float xValue = i \* (END / 18);

int xCoord = midX + (xValue - 0) \* (maxX / (END - 0)); // Масштабирование

line(xCoord, midY - 5, xCoord, midY + 5); // Метка

char label[10];

sprintf(label, "%.1f", xValue);

outtextxy(xCoord - 15, midY + 10, label);

}

// Оси Y

for (int j = -10; j <= 10; ++j) {

int yCoord = midY - j \* (maxY / 20);

line(midX - 5, yCoord, midX + 5, yCoord); // Метка

char label[10];

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

outtextxy(midX + 10, yCoord - 10, label);

}

}

// Функция для отрисовки графика

void drawGraph() {

int maxX = getmaxx();

int maxY = getmaxy();

int midX = maxX / 2 - AXIS\_OFFSET; // Сдвинуть ось X влево

int midY = maxY / 2;

float step = (END - START) / NUMBER\_OF\_STEPS;

for (int i = 0; i < NUMBER\_OF\_STEPS - 1; ++i) {

float x1 = START + i \* step;

float y1 = myFunction(x1);

float x2 = START + (i + 1) \* step;

float y2 = myFunction(x2);

// Преобразование координат

int xCoord1 = midX + (x1 - 0) \* (maxX / (END - 0)); // Масштабирование по оси X

int yCoord1 = midY - (y1 \* (maxY / 40)); // Масштабирование по оси Y

int xCoord2 = midX + (x2 - 0) \* (maxX / (END - 0)); // Масштабирование по оси X

int yCoord2 = midY - (y2 \* (maxY / 40)); // Масштабирование по оси Y

// Проверка на выход за пределы окна

if (yCoord1 >= 0 && yCoord1 <= maxY && yCoord2 >= 0 && yCoord2 <= maxY) {

line(xCoord1, yCoord1, xCoord2, yCoord2);

}

}

}

int main() {

int grDriver = DETECT, grMode;

initgraph(&grDriver, &grMode, "C:\\TurboC3\\BGI");

putAxis();

drawGraph();

getch();

closegraph();

return 0;

}

**Пример запуска программы**

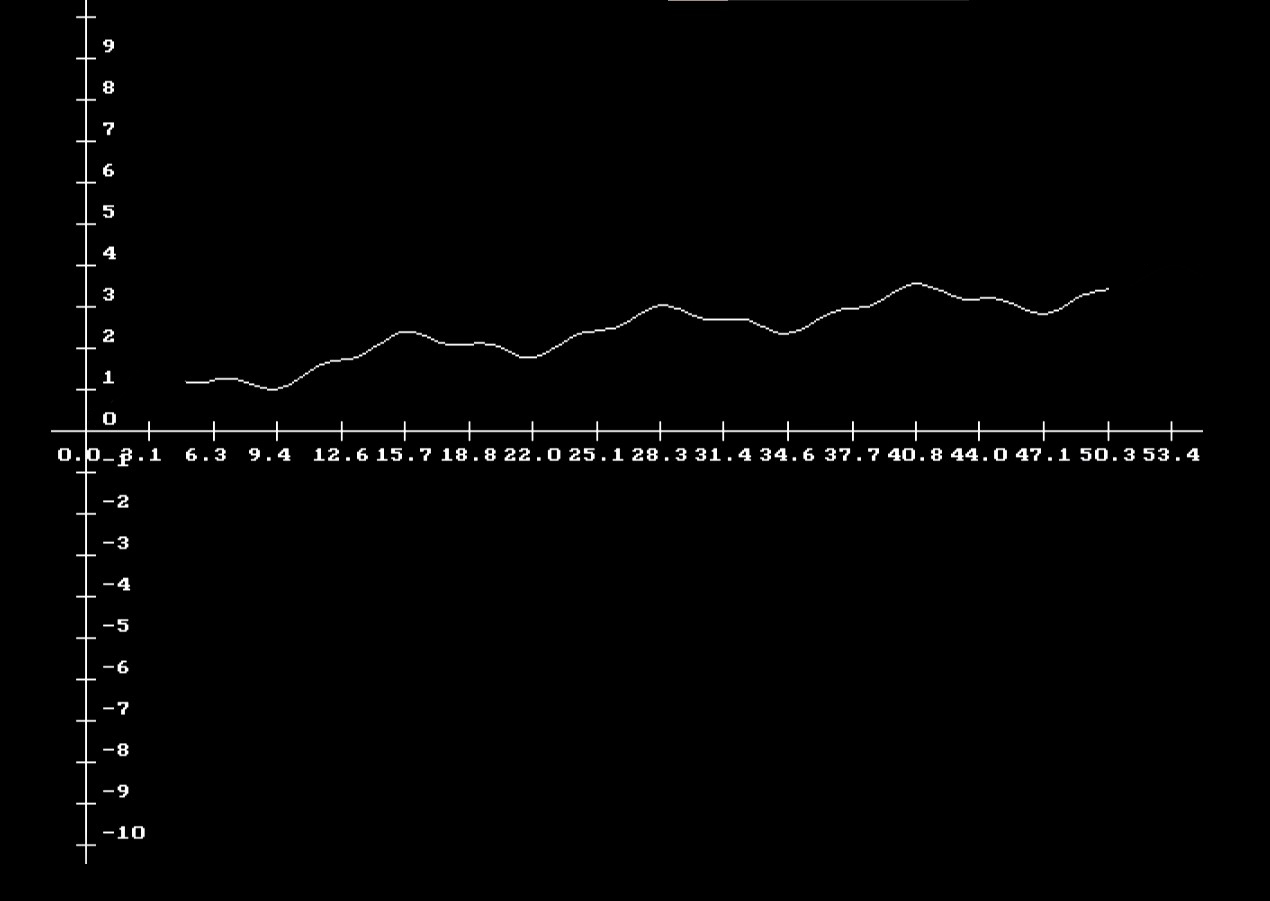


Рисунок 1. Пример запуска программы

**Вывод:** в результате данной лабораторной работы я изучил работу с видеосистемой в графическом режиме, вывел график заданной функции с масштабированием и разметкой осей.