**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)

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2023

**Цель работы**

Изучение работы с видеосистемой в графическом режиме, вывод графика заданной функции с масштабированием и разметкой осей.

**Формулировка задания**

1. Разработать программу для вывода на экран графика заданной функции.

2. Произвести разметку осей и проставить истинные значения точек.

3. Найти максимальное значение функции на заданном интервале и вывести в отдельное окно на экране.

**Теоретическая информация**

Характеристикой работы адаптера служит совокупность поддерживаемых им режимов, определяющих стандартное поведение адаптера и полностью описывающих все доступные для программиста средства управления адаптером и прочее. Режимы работы видеоадаптеров, будучи разнообразными, объединяются в две основные группы: текстовые и графические. Переключение между ними полностью изменяет логику работы видеоадаптера с видеобуфером.

В текстовом режиме видеоадаптер рассматривает экран как совокупность текстовых элементов, называемых текселами. Каждому текселу в текстовом режиме соответствуют два байта памяти видеобуфера: один байт хранит ASCII-код символа, а следующий байт, находящийся по нечетному адресу, содержит атрибуты отображения символа на экране.

Переключение адаптера в графический режим полностью меняет работу видеосистемы, позволяя контролировать цвет каждой точки экрана или пикселя. Количество строк пикселов и пикселов в каждой строке зависит от выбранного режима работы видеоадаптера, формируя матрицу пикселов на экране.

Использование графики в языке программирования C++ - это многоэтапный процесс. Необходимо определить тип видеоадаптера, установить подходящий режим его работы и выполнить инициализацию графической системы. После этого становятся доступными функции графической библиотеки graphics.h для создания основных графических элементов: линий, окружностей, эллипсов, прямоугольников, секторов, дуг и т.д. Также появляется возможность вывода текста с использованием различных шрифтов. Для использования этих функций необходимо инициализировать систему графики, загрузив соответствующий BGI-драйвер и выбрав режим работы адаптера.

Графические режимы, поддерживаемые библиотекой графики, определены символическими константами в заголовочном файле <graphics.h> в перечислимом типе graphics\_modes. Инициализация графической системы выполняется функцией initgraph(int \*graphdriver, int \*graphmode, char \*pathtodriver), которая загружает BGI-драйвер, определяемый указателем graphdriver, и устанавливает видеоадаптер в графический режим, задаваемый указателем graphmode. Аргумент pathtodriver указывает на строку, содержащую спецификацию файла BGI-драйвера.

Если инициализация сталкивается с противоречием между запрашиваемым режимом и типом видеоадаптера, или недостатком оперативной памяти и т.д., функция устанавливает код ошибки, доступный после вызова функции graphresult(). При завершении работы с функциями графической библиотеки рекомендуется вызвать функцию closegraph(), чтобы освободить память и восстановить текстовый режим адаптера. Эта функция также возвращает код ошибки и передает его по указателю, указанному в graphdriver. Функции setcolor(int color), setviewport(int left, int top, int right, int bottom, int clip), и outtextxy(int x, int y, char \*textstring) управляют цветом пикселов, окном вывода и выводом текста соответственно.

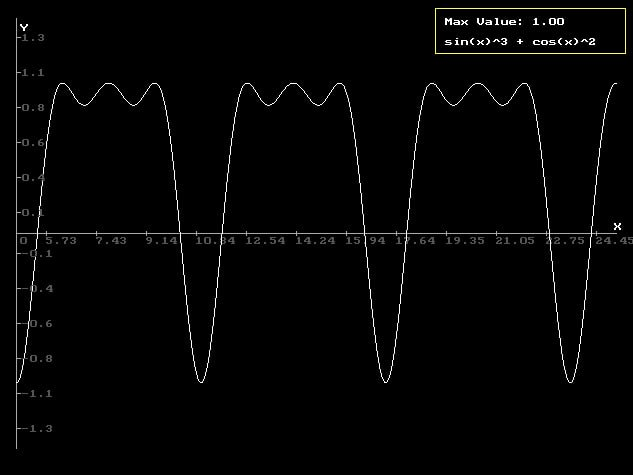
setcolor(int color) задает новый цвет для пикселов с кодом цвета 0.

setviewport(int left, int top, int right, int bottom, int clip) определяет новое графическое окно, устанавливая его координаты левого верхнего угла (left, top) и правого нижнего угла (right, bottom). Флаг clip указывает на использование усечения (clipping). Текущие координаты для функций графического вывода устанавливаются в левый верхний угол этого окна.

outtextxy(int x, int y, char \*textstring) выводит ASCII-строку текста, начиная с позиции (x, y), где textstring указывает на начало строки. Аргументы x и y четко определяют текущую позицию, используемую для вывода строки. Координаты X и Y измеряются относительно верхнего левого угла текущего графического окна.

**Результат работы программы**

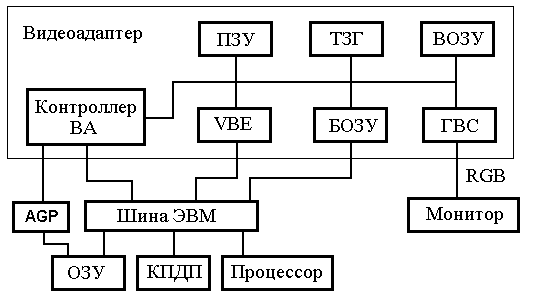
Программа строит график функции , находит максимальное значение функции в диапазоне и выводит всё это на экран.



**Блок-схема**



**Структурная схема аппаратных средств**



**Текст программы**

#include <conio.h>

#include <dos.h>

#include <graphics.h>

#include <math.h>

#include <stdio.h>

const float pi = 3.1415;

const float graph\_start = 3 \* pi / 2;

const float graph\_end = 8 \* pi;

const float scale = 150;

const float x1 = 20;

const float y1 = 20;

const float x2 = 620;

const float y2 = 450;

float func(float x) {

    return sin(x)\*sin(x)\*sin(x)+cos(x)\*cos(x);

}

int main() {

    int gdriver = DETECT, gmode;

    initgraph(&gdriver, &gmode, "C:\\TurboC\\BGI");

    int midy = (y1 + y2) / 2;

    float x\_incr = (graph\_end - graph\_start) / (x2 - x1);

    float y\_incr = 2 \* scale / (y2 - y1);

    int screen\_x = x1;

    int screen\_y = y1;

    setcolor(WHITE);

    outtextxy(x2 - 2, midy - 10, "X");

    outtextxy(x1 + 4, y1 + 6, "Y");

    setcolor(LIGHTGRAY);

    line(x1, midy, x2, midy);

    line(x1, y1, x1, y2);

    setcolor(DARKGRAY);

    outtextxy(x1 + 4, midy + 4, "0");

    float max\_y = func(graph\_start);

    int last\_x = screen\_x;

    int last\_y = midy - scale \* func(graph\_start);

    for (float x = graph\_start; x < graph\_end; x += x\_incr) {

        float ry = func(x);

        if (ry > max\_y)

            max\_y = ry;

        int y = midy - scale \* ry;

        setcolor(WHITE);

        line(last\_x, last\_y, screen\_x, y);

        last\_x = screen\_x;

        last\_y = y;

        if (screen\_x % 50 == 0) { //(x + x\_incr <= graph\_end)

            setcolor(LIGHTGRAY);

            line(screen\_x, midy - 2, screen\_x, midy + 2);

            setcolor(DARKGRAY);

            char label\_x[10];

            sprintf(label\_x, "%.2f", x);

            outtextxy(screen\_x, midy + 4, label\_x);

        }

        screen\_x++;

    }

    for (float y = 0; y < (midy - y1); y += y\_incr) {

        if (screen\_y % 50 == 0) { //

            setcolor(LIGHTGRAY);

            line(x1, midy - y, x1 + 4, midy - y);

            line(x1, midy + y, x1 + 4, midy + y);

            setcolor(DARKGRAY);

            char label\_up[10];

            sprintf(label\_up, "%.1f", y / scale);

            outtextxy(x1 + 6, midy - y - 3, label\_up);

            char label\_down[10];

            sprintf(label\_down, "%.1f", -y / scale);

            outtextxy(x1 + 6, midy + y - 3, label\_down);

        }

        screen\_y++;

    }

    int r\_left = getmaxx() - 200;

    int r\_top = 10;

    int r\_right = r\_left + 190;

    int r\_bottom = 55;

    setcolor(YELLOW);

    rectangle(r\_left, r\_top, r\_right, r\_bottom);

    floodfill((r\_left + r\_right) / 2, (r\_top + r\_bottom) / 2, BLACK);

    setcolor(WHITE);

    char label\_mv[20];

    sprintf(label\_mv, "Max Value: %.2f", max\_y);

    outtextxy(r\_left + 10, r\_top + 10, label\_mv);

    outtextxy(r\_left + 10, r\_top + 30, "sin(x)^3 + cos(x)^2");

    getch();

    closegraph();

    return 0;

}