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)

Студентка гр. 1361 _____ Горбунова Д. А.

Студент гр.1361 _____ Голубев Д. В.

Преподаватель _____ Гречухин М. Н.

Санкт-Петербург

2022

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

Интегральной характеристикой особенностей работы адаптера является совокупность поддерживаемых им режимов. Поведение адаптера в том или ином режиме является фактическим стандартом и полностью характеризует все особенности адаптера, доступные для программиста средства управления адаптером и т.п.

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.

Если видеоадаптер включен в текстовый режим, он рассматривает экран как совокупность так называемых текселов. Каждому текселу в текстовом режиме соответствуют два байта памяти видеобуфера. Байт по четному адресу хранит ASCII-код символа, а следующий за ним байт по нечетному адресу кодирует особенности отображения символа на экране. Этот байт называется байтом атрибута.

Переключение адаптера в один из графических режимов полностью изменяет логику работы аппаратуры видеосистемы. При работе в графическом режиме появляется возможность управлять цветом любой телевизионной точки экрана или пиксела. Число строк пикселов и число пикселов в каждой

строке зависит от режима работы видеоадаптера. Таким образом, экран в графическом режиме представляет собой матрицу пикселов.

Использование графики в языке C++ — 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 graphicx.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. Прежде чем использовать функции графической библиотеки C++, 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.

Инициализацию графической модели выполняет функция void far initgraph(int *graphdriver, int *graphmode, char * pathtodriver). При вызове она инициализирует графическую систему, загружая BGI-драйвер, определяемый указателем graphdriver, и устанавливая видеоадаптер в графический режим, задаваемый указателем graphmode. Аргумент pathtodriver указывает на ASCII-строку, хранящую спецификацию файла BGI-драйвера.

Если при выполнении инициализации возникает противоречие между запрашиваемым режимом и типом видеоадаптера, либо отсутствует достаточный объем свободной оперативной памяти и т.п., функция устанавливает код ошибки во внешней переменной, доступной после вызова функции graphresult(). Кроме того, код ошибки передается в точку вызова в ячейке памяти, на которую указывает graphdriver. Если функции графической библиотеки больше не нужны прикладной про-грамме, следует вызвать

функцию closegraph() "закрытия" графического режима и возвращения к текстовому режиму. Эта функция освобождает память, распределенную под драйверы графики, файлы шрифтов и промежуточные данные и восстанавливает режим работы адаптера в то состояние, в котором он находился до выполнения инициализации системы.

void setcolor (int color) устанавливает новый цвет пикселов, имеющих код цвета 0.

void setviewport (int left, int top, int right, int bottom, int clip) описывает новое графическое окно с координатами (столбец, строка) левого верхнего угла left, top, координатами правого нижнего угла right, bottom и значением флага усечения clip. В качестве начала текущих координат для функций графического вывода устанавливается левый верхний угол.

void outtextxy (int x, int y, char *textstring) ASCII-строку текста, на начало которой указывает textstring. Аргументы x и y явно специфицируют новую текущую позицию, используемую для вывода строки. Координаты X и Y измеряются относительно координат левого верхнего угла текущего графического окна.

Результат работы программы

Программа строит график функции $\sin^3\frac{x}{2}+\sqrt{x}$ в диапазоне $\left[\frac{3\pi}{2};\ 16\pi\right]$, находит максимальное значение функции и выводит всё это на экран.

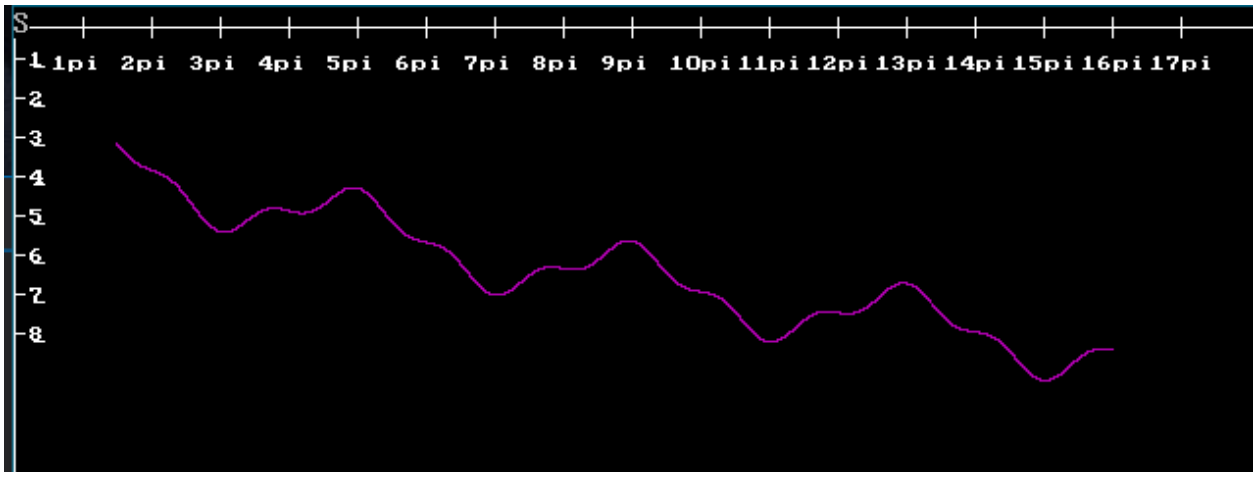
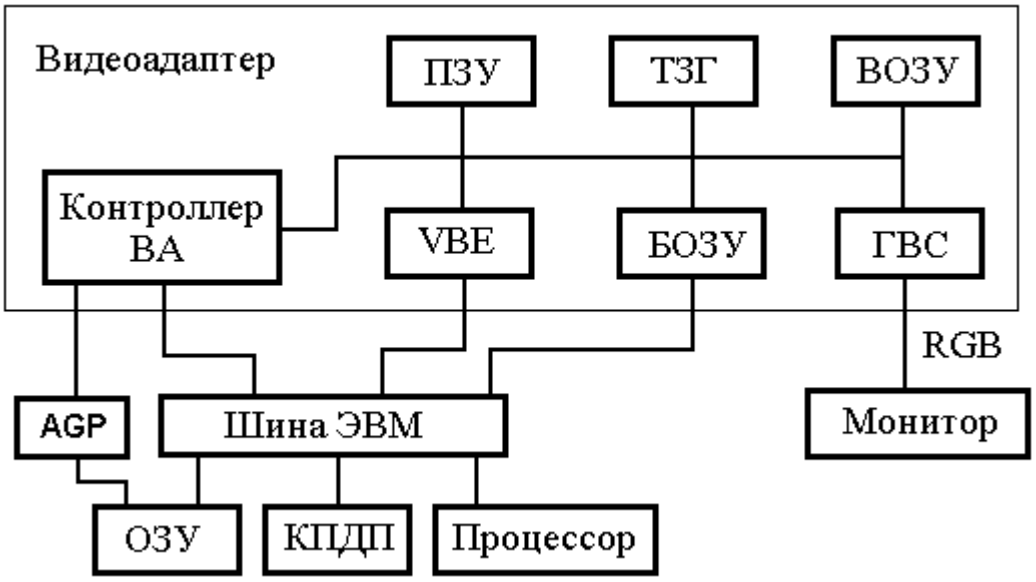


Рисунок 1 — Результат работы программы

Приложение 1. Блок-схема реализуемого кода

Приложение 2. Структурная схема аппаратных средств



Приложение 3. Исходный код программы

```
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

#define pi 3.14159265
#define dx 100
#define dy 100
#define x0 10
#define y0 10
#define Mixxi 1.5
#define Maxxi 16
int main() {
    int graph_driver, graph_mode, midx,midy;
    char str [3];
    char str2[16];
    int graph_error_code, k,j,i;
    double x1,y1,max=0,tabx=dx/pi,x,y;
    clrscr();
    graph_driver = DETECT;
    initgraph (&graph_driver, &graph_mode, "C:\\TURBOC3\\BGI");
    graph_error_code = graphresult();
    if (graph_error_code != grOk){
        printf("ERROR");
        getch();
        return 1;
    }
    midx=getmaxx();
    midy = getmaxy()/2-y0*x0;
    setviewport(0,0,midx,midy,1);
    line (x0,midy,x0,0);
    line (x0,midy,midx,midy);
    k=0;
    for (i=dx;k<=Maxxi;i+=dx){
        k++;
        sprintf (str,"%dpi",k);
        outtextxy(i-15,midy-20,str);
        line (i,midy+5,i,midy-5);
    }
    k=1;
    for (j=y0+28;k>=-1;j+=dy){
```

```
        sprintf (str,"%d",k);
        outtextxy(0,j-7,str);
        line (x0,j,x0+5,j);
        k--;
    }
    setviewport(0,0,midx,midy,0);
    x1=Mixxi*pi;
    do {
        y1 = sin(x1/2)*sin(x1/2)*sin(x1/2)-sqrt(x1);
        x= x1*tabx;
        y=y1*dy;
        putpixel (x, midy-y,5);
        if (y1>max) max= y1;
        x1+=0.0001;
    }
    while (x<=Maxxi*dx);
    sprintf (std2, "Max is %f\n", max);
    outtextxy(0, midy+midx, str2);
    getchar();
    closegraph();
    return 0;
}
```