**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. 8363 |  | Нерсисян А.С. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2019

**Цель работы.**

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

**Задание на лабораторную работу.**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Функция | Диапазон аргумента | |
| Начало | Конец |
| 11 |  |  |  |

1. Произвести разметку осей и проставить истинные значения точек.
2. 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 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 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.

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. Использование графической библиотеки С++ требует знакомства с моделью графической системы, применяемой компилятором для представления графической системы компьютера. Можно сказать, что сложность овладения деталями аппаратных средств видеоадаптеров сравнима со сложностью освоения графической модели. Однако достоинство графической модели заключается в ее относительной независимости от различных типов видеоадаптеров и открытости для дальнейших расширений. Появление новых типов видеоадаптеров не потребует большой переработки программ, так как все новые особенности аппаратуры будут учитываться в средствах библиотеки С++.

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. Сложность состоит в том, что фирма Borland International не раскрывает пока внутреннюю структуру драйвера.

Совокупность внешних переменных библиотеки графики и особенностей поведения мобильных функций образует модель графики С++.

**Инициализация и закрытие системы графики**

Инициализацию графической модели выполняет функция initgraph().

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. Третий аргумент функции initgraph() задает маршрут поиска файла, со-держащего .BGI-драйвер.

Помимо перевода видеоадаптера в заданный графический режим, функция initgraph() динамически распределяет оперативную память для загружаемого драйвера и хранения промежуточных результатов, возникающих при работе некоторых функций графики.

Если функции графической библиотеки больше не нужны прикладной программе, следует вызвать функцию closegraph() "закрытия" графического режима и возвращения к текстовому режиму.

closegraph().

**Обработка ошибок системы графики**

Защищенное от ошибок построение программы требует использования функции graphresult() после любого обращения к функциям detectgraph() и initgraph().

Возвращает значение внутреннего кода ошибки, установленного последним обращением к функциям графической библиотеки. Перед завершением сбрасывает код ошибки в 0.

**Определение и установка графического режима**

Установку режима выполняет функция setgraphmode(). Целая группа функций – getgraphmode(), getmaxmode(), getmodename() , getmoderange() - упрощает работу по определению текущего установленного режима. Две функции позволяют определить ширину и высоту экрана в пикселах для текущего видеорежима: getmaxx() и getmaxy(). Функция restorecrtmode() возвращает видеоадаптер в текстовый режим.

**Управление цветами и палитрами**

Возможности по выбору цветов принципиально различны для CGA-, EGA- и VGA-адаптеров, что обусловлено различной логикой построения аппаратных средств.

int getbkcolor(void)

Возвращает целое число, равное коду цвета фона.

int getmaxcolor(void)

Возвращает максимальное значение кода цвета пиксела минус 1. Это значение позволяет установить максимальное число цветов, которое может отображаться на экране. В зависимости от режима, в котором проведена инициализация системы графики, возвращаемое значение может быть равно 1, 3 или 15.

void setbkcolor (int color)

Устанавливает новый цвет пикселов, имеющих код цвета 0. Новый цвет фона задает значение аргумента color.

void setcolor (int color)

Устанавливает цвет, используемый функциями графического вывода в значение, заданное аргументом color. До того момента, пока цвет не установлен, используется максимальный (из палитры) номер цвета. В случае, если color задает недопустимый номер цвета для текущей палитры, текущий цвет остается неизменным.

**Задание окна экрана. Определение и установка графических координат**

Окно экрана в графическом режиме, или графическое окно (viewport), - это прямоугольная область экрана, заданная пиксельными координатами левого верхнего и правого нижнего углов. В графическом окне определены относительные координаты. С++ позволяет выполнять вывод текста и графических примитивов в графическое окно. При этом по желанию пользователя вывод, не вмещающийся в границы окна, может усекаться. Графическое окно может иметь отличающиеся от других участков экрана цвета фона и пикселов, маску заполнения и другие характеристики.

Для описания окна используется функция setviewport(). Текущие характеристики окна доступны программе через обращение к функции getviewsettings().

void far getviewsettings( struct viewporttype \*viewport)

Заполняет поля структурной переменной по шаблону viewporttype инфор-мацией о графическом окне. Описание структурной переменной выполняет вызывающая сторона. Функции передается указатель на описанную переменную. Шаблон viewporttype описан в <graphics.h>.

**Вывод текста в графическом режиме видеоадаптера**

Библиотека графики позволяет выводить на экран текст различными шрифтами. С++ имеет два типа шрифтов: битовый и сегментированный.

Каждому символу битового шрифта (bit-mapped font) ставится в соответствие матрица пикселов фиксированного размера. С++ использует в качестве битового шрифта таблицу знакогенератора для символов размером 8x8, установленную в компьютере перед инициализацией системы графики. Все изменения таблицы знакогенератора, сделанные, например, программами русификации, будут сохранены. Это позволяет, применяя функции, выводить текст русскими буквами и в графических режимах.

int textwidth ( char far \*textstring)

Возвращает ширину строки символов в пикселах, на которую указывает textstring. Использует информацию о текущем шрифте и установках масштаба знакоместа. Сама строка, на которую указывает textstring, на экран не выводится. Наиболее часто функция используется для установки нужных промежутков между символами текста, а также при вычислении таких масштабов для символов, которые позволяли бы уместить нужное число строк в фиксированной области экрана. В частности, обращение к функции

void outtextxy (int x, int y, char \*textstring)

Выводит ASCII-строку текста, на начало которой указывает textstring, ис-пользуя текущие цвет, установки направления, типа шрифта и выравнивания строки. Аргументы х и у явно специфицируют новую текущую позицию, ис-пользуемую для вывода строки. Координаты X и Y измеряются относительно координат левого верхнего угла текущего графического окна. В случае, когда текст выводится горизонтально и установлено выравнивание LEFT\_TEXT, функция outtext() продвигает координату X текущей позиции на значение, равное textwidth(textstring). В остальных случаях координата X текущей позиции остается неизменной. Если текст выводится в графическое окно с включенным усечением, он усекается на границах окна. Для сегментированных шрифтов усечение производится с точностью до пикселов, для битовых шрифтов оно происходит с точностью до символа. В случае, когда установлено выравнивание CENTER\_TEXT, но выводимая строка не помещается в текущем графическом окне, функция не выполняет вывод.

Функции способны выводить только нуль-терминированные строки, и для выполнения форматированного вывода в графических режимах выбранными сегментированными шрифтами поступают следующим образом. Сначала, ис-пользуя функцию стандартного вывода sprintf(), получают нужную форматную строку, а затем выводят ее с помощью функции outtextxy() выбранным шрифтом.

**Вывод графической информации: Параметры и атрибуты графического вывода**

Все функции библиотеки графики, генерирующие вывод информации на экран, работают в пределах текущего графического окна. Для графического вывода используется текущий цвет пиксела, установленный функцией setcolor().

При выводе отрезков прямых линий и графических примитивов система графики позволяет определить такой параметр, как стиль линии. С++ поддерживает ряд предопределенных стилей линий. Как и в случае маски заполнения, пользователь может описать собственный стиль линии. Для определения текущей установки стиля используется функция getlinesettings(). Выбор подходящего стиля выполняет функция setlinestyle().

void getlinesettings (struct linesettingstype \*lineinfo)

Возвращает информацию об установленном в текущий момент времени стиле "рисования" отрезков прямых линий и графических примитивов. Функция заполняет поля структурной переменной по шаблону struct linesettingstype. Структурную переменную описывает точка вызова и передает в функцию указатель lineinfo на эту переменную.

void setlinestyle (int linestyle, unsigned upattern, int thickness)

Устанавливает стиль "рисования" отрезков прямых линий и графических примитивов.

void setwritemode(int mode)

Устанавливает режим вывода отрезков прямых линий в значение, определяемое аргументом mode. Аргумент mode может принимать одно из двух значений, описанных в <graphics.h>: COPY\_PUT (0) - пикселы, лежащие на отрезке прямой линии, переопределяют пикселы на экране, и, таким образом, линия на экране имеет текущий цвет; XOR\_PUT (1) - пикселы, образующие линию, имеют код цвета, образуемый операцией исключающего ИЛИ (XOR) кода текущего цвета и кода цвета пикселов на экране, через которые линия проходит. В частности, можно стереть выведенную линию с экрана, выполнив вывод линии еще раз.

void getaspectratio (int \*xasp, int \*yasp)

Заполняет две переменные, описанные точкой вызова, значениями коэф-фициента сжатия для текущего видеорежима. Возвращаемые значения задают фактически физическую форму пиксела. Для размера пиксела по вертикали (значение, на которое указывает yasp), всегда возвращается 10 000. Если световое пятно на экране, соответствующее пикселу, является квадратным (как для адаптера VGA), то и значение "ширины" пиксела равно 10 000.

void setaspectratio (int xasp, int yasp) 45

Устанавливает новое значение коэффициента сжатия, которое будет ис-пользоваться системой графики при выводе геометрических примитивов - прямоугольников, дуг, окружностей, эллипсов. Аргумент xasp отображает в условных единицах ширину пиксела на экране, yasp - высоту пиксела.

void getfillpattern (char \* pattern)

Заполняет область памяти из 8 байт, описанную точкой вызова, текущим значением маски заполнения. Аргумент pattern указывает на начало описанной области памяти. Маска заполнения может иметь одно из предопределенных значений или описываться пользователем.

void setfillpattern (char \*upattern, int color)

Задает цвет пикселов и маску для заполнения областей экрана. По умолчанию используется белый цвет и маска заполнения, состоящая из матрицы единиц во всех битах. Таким образом, по умолчанию все пикселы заполняемой области имеют белый цвет. Аргумент upattern указывает на начало области из 8 байт, задающих новую маску заполнения. Первый байт задает пикселы самой верхней строки в пределах знакоместа. Старший бит первого байта соответствует самому левому пикселу знакоместа. Аргумент color задает цвет пикселов.

void getfillsettings( struct fillsettingstype \*fillinfo)

Заполняет поля структурной переменной по шаблону struct fillsettingstype информацией о текущей маске и цвете заполнения. Структурную переменную по шаблону struct fillsettingstype описывает точка вызова. Аргумент fillinfo указывает на описанную точкой вызова структурную переменную.

**Чтение-запись отдельных пикселов**

Базовой функцией любой графической библиотеки является функция вывода в заданные координаты пиксела специфицированного цвета. С++ имеет в своем составе две функции манипуляции отдельными пикселами экрана: getpixel() - для определения кода цвета пиксела и putpixel () - для вывода пиксела текущим цветом.

unsigned getpixel( int x, int у)

Определяет, лежит ли пиксел с координатами (х, у) в текущем графическом окне, и, если лежит, возвращает код цвета этого пиксела. В противном случае возвращается 0.

void putpixel(int x, int у, int pixelcolor)

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.

**Вывод отрезков прямых линий**

Целая группа функций библиотеки графики предназначена для вывода отрезков прямых линий. Далее приводится спецификация этих функций. Напомним, что на вывод отрезков прямых линий влияют режим вывода линии и стиль линии.

Выводимые отрезки прямых линий не пересекают границ текущего окна, если при описании окна включен режим "усечения" (clipping).

void line( int x1, int y1, int x2, int y2)

Выводит отрезок прямой линии между двумя явно специфицированными точками (x1, y1) и (х2, у2), используя текущие цвет, стиль, толщину и режим вывода линии. Координаты (x1, y1) и (х2, у2) задаются относительно левого верхнего угла текущего графического окна. Функция не изменяет текущую позицию.

void linerel(int dx, int dy)

Выводит отрезок прямой линии между текущей позицией (начало отрезка) и точкой, заданной горизонтальным смещением dx и вертикальным смещением dy от текущей позиции (конец отрезка). При выводе отрезка прямой используются текущие цвет, стиль, толщина и режим вывода линии. После вывода линии функция устанавливает новую текущую позицию, равную координатам конца отрезка.

**Вывод основных графических примитивов**

Библиотека графики содержит функции для вывода дуги окружности или целой окружности, эллиптической дуги или целого эллипса, кругового сектора, ломаной линии из нескольких отрезков прямой (полигона), прямоугольника, прямоугольной полосы заданного цвета и стиля заполнения, прямоугольника заданной толщины в аксонометрии.

void arc(int x, int y, int stangle, int endangle, int radius)

Выводит дугу окружности радиусом radius. Центр окружности задают ко-ординаты х, у. Аргументы stangle и endangle задают соответственно начальный и конечный углы выводимой дуги.

void bar(int left, int top, int right, int bottom)

Выводит полосу, заполненную текущим цветом с использованием текущей маски заполнения. Аргументы задают пиксельные координаты левого верхнего (left, top) и правого нижнего (right, bottom) углов заполняемой области экрана.

void bar3d(int left, int top, int right, int bottom, int depth, int topflag)

Выводит в изометрии "столбик" и заполняет его фронтальную поверхность текущим цветом с использованием текущей маски заполнения. Аргументы задают: пиксельные координаты левого верхнего (left, top) и правого нижнего (right, bottom) углов заполняемой области экрана; "глубину" (depth) в пикселах изображаемого столбца; необходимость изображения "верхней" поверхности столбца (topflag): если topflag = 0, верхняя поверхность не отображается.

void circle( int x, int y, int radius)

Выводит окружность заданного аргументом radius радиуса с центром, заданным координатами х и у.

void drawpoly(int numpoints, int polypoints[])

"Соединяет" отрезками прямых линий текущего цвета и стиля точки (полигон), координаты которых заданы парами значений. Эти пары расположены в массиве, на который указывает polypoints[]. Аргумент numpoints задает число соединяемых между собой точек.

void ellipse (int x, int y, int stangle, int endangle, int xradius, int yradius)

Выводит эллиптическую дугу или полный эллипс, используя текущий цвет. Аргументы задаютэ: пиксельные координаты центра эллипса (х, у); начальный угол дуги (stangle); конечный угол дуги (endangle); радиус эллипса по горизонтали (xradius); радиус эллипса по вертикали (yradius).

void fillellipse(int x, int y, int xradius, int yradius)

Выводит эллипс, заполненный текущим стилем. Аргументы функции задают: пиксельные координаты центра эллипса (х, у); радиус эллипса по горизонтали (xradius); радиус эллипса по вертикали (**yradius**).

void fillpoly(int numpoints, int \*polypoints)

Выводит контур полигона, заданного numpoints точками. Координаты точек заданы парами, расположенными в массиве, на который ссылается polypoints. Функция соединяет первую и последнюю точки и заполняет область внутри полигона текущим стилем.

void floodfill (int x, int y, int border)

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.

void pieslice( int x, int y, int stangle, int endangle, int radius)

Выводит контур кругового сектора и заполняет его внутреннюю область текущим стилем. Контур образован круговой дугой радиусом radius с координатами центра (х, у), проведенной, начиная от угла stangle до угла endangle, и радиусами, соединяющими центр с концевыми точками дуги.

void rectangle(int left, int top, int right, int bottom)

Выводит контур прямоугольника, заданного координатами левого верхнего (left, top) и правого нижнего (right, bottom) углов.

void sector(int x, int y, int stangle, int endangle, int xradius, int yradius)

Работает аналогично функции pieslice(), за исключением того, что выводится не круговая, а эллиптическая дуга. Аргумент xradius задает радиус эллипса по горизонтали, a yradius - радиус эллипса по вертикали.

**Алгоритмы**

Для получения значений функции на заданном отрезке используется данная функция:

void make\_fun(int N)

{

double x, dx=7.\*M\_PI/N;

int i;

for (i=0, x=M\_PI/2.; i<N; i++, x+=dx)

{

R[i]=(float)(pow(sin(x),2)-pow(cos(x),2));

if(Rmax<R[i]) Rmax=R[i];

}

}

что представляет собой цикл for по числу точек графика функции, где для расчета значения функции как аргумент используется значение х в данной точке.

Для нормирования значений функции используется цикл for по числу точек графика функции,

for (i=0;i<N;i++) RN[i]=Y0-(int)(R[i]/Rmax\*(float)(Y0-Y1));

где в каждой итерации берется ранее рассчитанное значение функции, и учитывая размеры окна нормируется и записывается в новый массив.

Для рисования графика функции используется цикл for по числу точек графика функции

for (i=0;i<N;i++) {

putpixel(i+X0,RN[i],12);

delay(20);

}

где в каждой итерации с задержкой на 20 миллисекунд, на экране выводится значение функции в данной точке, с атрибутом цвета 12 (Светло-красный).

**Текст программы**

#include "stdio.h"

#include "conio.h"

#include "math.h"

#include "dos.h"

#include "graphics.h"

#include "iostream.h"

#include "stdlib.h"

#include "windows.h"

#include "time.h"

int Xmax, Ymax, X0, X1, Y0, Y1, GrDr, GrM, RN[200];

float R[200], Rmax=0.;

char GrP[50]="..//BORLANDC//BGI"; // Расположение драйверов BGI

// Формирование массива значений отображаемой функции

void make\_fun(int N)

{

double x, dx=7.\*M\_PI/N;

int i;

for (i=0, x=M\_PI/2.; i<N; i++, x+=dx)

{

R[i]=(float)(pow(sin(x),2)-pow(cos(x),2)); // сама функция

if(Rmax<R[i]) Rmax=R[i];

}

}

// Программа вывода графика функции на экран

void main(void)

{

int i, N;

clrscr(); // Очистка экрана

gotoxy(20,5);

cprintf("Программа вывода графика функции на экран");

gotoxy(13,10);

cprintf("Для перехода в графический режим нажмите любую клавишу");

getch(); // Ожидаем нажатия клавиши...

detectgraph(&GrDr,&GrM); // Открыли графичесий режим

initgraph(&GrDr,&GrM,GrP);

if (graphresult()<0)

{

gotoxy(18,13);

cprintf("\nОШИБКА! НЕ МОГУ ОТКРЫТЬ ГРАФИЧЕСКИЙ РЕЖИМ...\n");

getch();

return;

}

else

{

Xmax=getmaxx();

Ymax=getmaxy(); // Установили размеры экрана

X0=40;

Y0=Ymax-200; // Левый нижний угол графика

X1=Xmax-200;

Y1=200; // Правый верхний угол графика

N = X1-X0; // Количество точек по оси X

char Ox[14][6] = {" ", "Pi", " ", "2Pi", " ", "3Pi", " ", "4Pi", " ", "5Pi", " ", "6Pi", " ", "7Pi"};

setlinestyle(0,1,3); // параметры линий

int textLen = textwidth("f(x)=sin^2(x)-cos^2(x) on Pi/2 to 7Pi");

outtextxy((Xmax/2)-(textLen/2), 30, "f(x)=sin^2(x)-cos^2(x) на отрезке от Pi/2 до 7Pi");

make\_fun(N); // Вычислили значения функции

for (i=0;i<N;i++) // Нормируем все значения функции

RN[i]=Y0-(int)(R[i]/Rmax\*(float)(Y0-Y1));

setlinestyle(0,1,1); // Рисуем координатные оси

line(X0, Y0+85, X0, Y1);

line(X0, Y0, X1+2, Y0);

outtextxy(X0+5,Y1,"f(x)"); // Помечаем оси

outtextxy(X1+3,Y0+1,"x");

outtextxy(X0-10, Y0-10, "0");

outtextxy(X0-10, Y1, "1");

outtextxy(X0-17, Y1+155, "-1");

for (i=1; i<14; i++) //Рисуем метки и подписи на оси Х

{

if(i%4==0 || i%6==0 || i%9==0 || i%11==0 || i%13==0)

{

line(X0+(int)(N/14)\*i+4, Y0-3, X0+(int)(N/14)\*i+4, Y0+3); //Метки

outtextxy(X0+(int)(N/14)\*i-15, Y0+5, Ox[i]); //Подписи

}

else

{

line(X0+(int)(N/14)\*i, Y0-3, X0+(int)(N/14)\*i, Y0+3); //Метки

outtextxy(X0+(int)(N/14)\*i-10, Y0+5, Ox[i]); //Подписи

}

};

for (i=0;i<N;i++)

{

// Рисуем график функции

putpixel(i+X0,RN[i],12);

delay(20);

}

window(10, 12, 70, 23);

outtextxy (200, 50, "Максимальное значение функции 1");

getch(); // Ожидаем нажатия клавиши...

closegraph(); // Закрываем графичесий режим

}

getch();

getch();

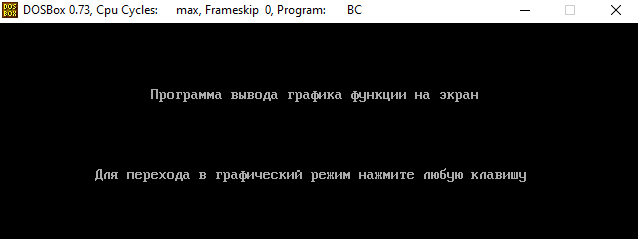
system("pause");

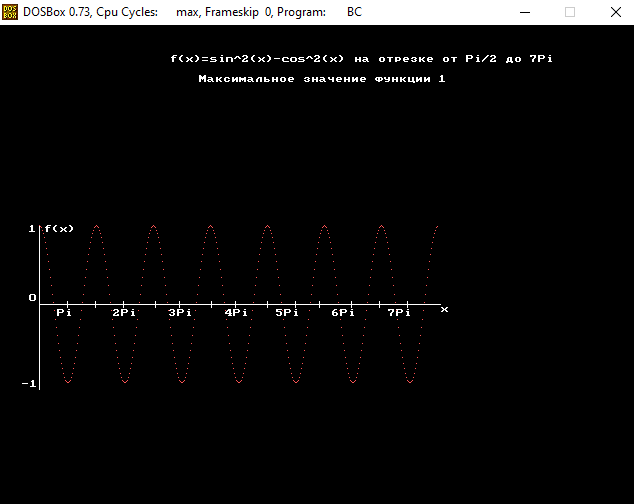
getch();

}

**Примеры запуска программы**

На рисунках демонстрирована работа программы. Программа выводит на экран график функции , а также максимальное значение функции на отрезке от до .

****

****

**Вывод**

В ходе выполнения данной лабораторной работы была изучена работа с видеосистемой в графическом режиме, вывод графика заданной функции с масштабированием и разметкой осей.

**Структурная схема аппаратных средств, используемых при выполнении программы с необходимой степенью детализации содержимого блоков.**

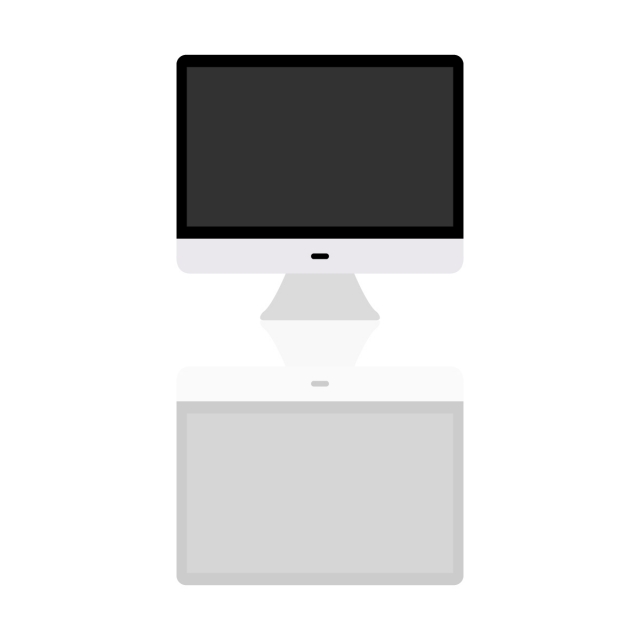
**ПЗУ**

**(RОM)**

**ОЗУ**

**(RAM)**

Контроллер



**ИП**

Монитор

Контроллер

**ЦП**

**(CPU)**

СИСТЕМНАЯ ШИНА