**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Организация ЭВМ и Систем»**

**Тема: «Исследование графического режима видеосистемы»**

**Вариант 4**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0302 |  | Чёрный Я.И. |
| Преподаватель |  | Жандаров В.В. |

Санкт-Петербург

2021

**Оглавление**

[1 Постановка задачи 3](#_Toc90501567)

[1.1 Параметры вывода надписей 3](#_Toc90501568)

[1.2 Дополнительная задача 3](#_Toc90501569)

[2 Краткие сведения о видеосистемах ПЭВМ, текстовом режиме их работы и функциях обслуживания текстового режима. 3](#_Toc90501570)

[3 Блок-схема программы 6](#_Toc90501571)

[4 Пример работы 7](#_Toc90501572)

[5 Текст программы 7](#_Toc90501573)

[6 Вывод 12](#_Toc90501574)

# Постановка задачи

Целью данной задачи является разработка программы, выводящую на экран график заданной функции на определённом интервале значений.

## Целевая функция на интервале

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Функция | Диапазон аргумента | |
| Начало | Конец |
| 4 | Cos2(x/2) +Sqrt(x) | 3π/2 | 18π |

## Дополнительная задача

Произвести разметку осей и проставить истинные значения точек. Найти максимальное значение функции на заданном интервале и вывести в отдельное окно на экране вместе с графиком.

# Краткие сведения о видеосистемах ПЭВМ, текстовом режиме их работы и функциях обслуживания текстового режима.

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++. Подробно эти элементы модели рассматриваются в следующих подразделах.

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. 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. Возможные значения для графических режимов даны в табл. 3.1. В табл. 3.2. приведены значения, определяющие графические драйверы при инициализации системы графики. Упомянутые в таблице символические константы определены в перечислимом типе graphics\_drivers из заголовочного файла <graphics.h>.

Третий аргумент функции initgraph() задает маршрут поиска файла, со­держащего .BGI-драйвер. Если файл не найден в заданной директории, функция просматривает текущий директорий. Если pathtodriver = NULL, драйвер должен располагаться в текущей директории. В случае, когда при вызове initgraph() па­раметры видеосистемы неизвестны, значение для graphdriver следует задать рав­ным указателю на DETECT.

Благодаря этому функция initgraph() вызывает другую библиотечную функцию – detectgraph() - для определения типа видеоадаптера, подходящего графического драйвера и графического режима максимального разрешения (максимального режима) для активного видеоадаптера системы. Значения для драйвера и максимального режима возвращаются в ячейках памяти, на которые указывают graphdriver и graphmode.

Возвращает текущий графический режим, установленный для графической модели функциями initgraph() или setgraphmode(). Возвращаемое значение соответствует номеру режима, установленному для инсталлированного драйвера графики. Возвращаемое значение соответствует числовому значению символических констант режима, перечисленных в табл. 3.1.

int getmaxmode(void)

Возвращает число, определяющее максимально возможный для инсталлиро­ванного. BGI-драйвера режим. Как и в предыдущем случае, возвращаемое значе­ние соответствует номеру режима, установленному для инсталлированного драйвера графики. Возвращаемое значение соответствует числовому значению символических констант режима, перечисленных в табл. 3.1.

int getmaxx(void)

int getmaxy(void)

Возвращают максимальные значения координат X и Y для текущего видеоре­жима. Например, для режима CGA0 getmaxx() возвращает значение 319, a getmaxy() -199. Функции особенно полезны для центрирования изображений и определения таких размеров знакомест при выводе текста в графическом режиме, чтобы текст помещался в заданную область экрана.

char \* getmodename(int mode\_number)

Выводит ASCII-строку текста, на начало которой указывает textstring, ис­пользуя текущие позицию, цвет и установки направления, типа шрифта и выравнивания строки. 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, текст усекается на границах окна. Для сегментированных шрифтов усечение производится с точностью до пикселов, для битовых шрифтов оно происходит с точностью до символа. Усечение строки может выполняться по одной границе или по обеим границам сразу в случае, когда задается выравнивание по центру.

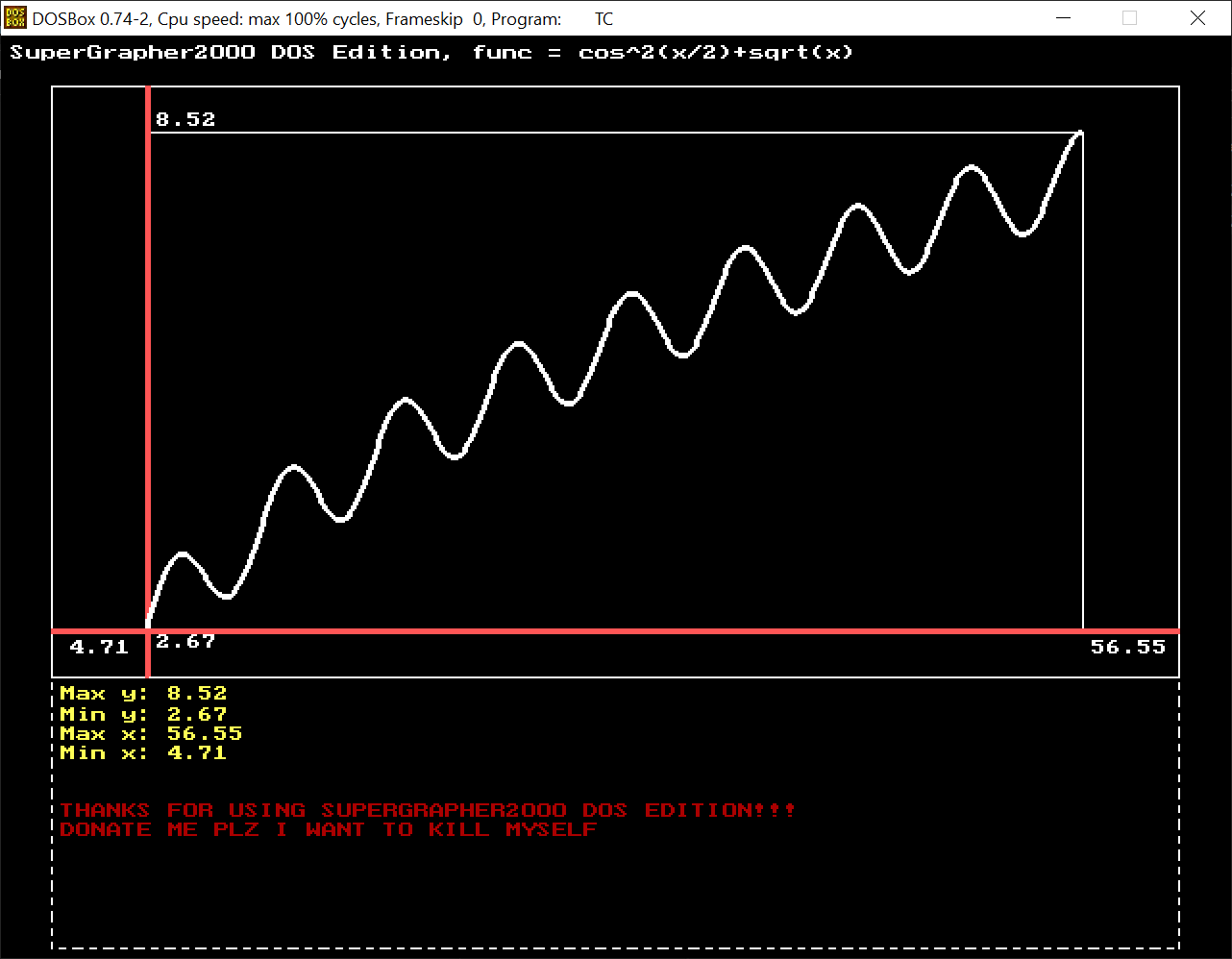
void outtextxy (int x, int y, char \*textstring)

Выводит ASCII-строку текста, на начало которой указывает textstring, ис­пользуя текущие цвет, установки направления, типа шрифта и выравнивания строки. Аргументы х и у явно специфицируют новую текущую позицию, ис­пользуемую 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.

# Блок-схема программы



# Пример работы



# Текст программы

MAIN.CPP

|  |
| --- |
| #include <iostream.h>  #include <graphics.h>  #include <math.h>  #include <stdlib.h>  #include <conio.h>  #define pi 3.1415  class Plotter  {  private:  class Point  {  public:  float x, y;  Point()  {  x = 0;  y = 0;  }  Point(float xp, float yp)  {  x = xp;  y = yp;  }  };  float round(float x)  {  int val = (int)(x \* 100 + 0.5f);  return ((float)val / 100);  }  int gDriver, gMode;  //main window pixel margin  int marginX, marginY, marginSmallX, marginSmallY;  int sizeX, sizeY;  float xMax, yMax, xMin, yMin;  float v, h, v1, h1, kv, kh;  Point\* realPoints;  Point\* plotPoints;  int nOfPoints;  void buildScale()  {  h1 = (xMax + xMin) / 2;  v1 = (yMax + yMin) / 2;  h = (sizeX / 2) / (xMax - h1);  v = (sizeY / 2) / (yMax - v1);  kh = (sizeX) / 2 - h1 \* h;  if (kh < 0)  kh = 0;  else if (kh > sizeX)  kh = sizeX;  kv = (sizeY) / 2 + v1 \* v;  if (kv < 0)  kv = 0;  else if (kv > sizeY)  kv = sizeY;  }  void buildRealPoints()  {  float step = 0.1f;  xMin = 3\*pi/2;  xMax = 18\*pi;  yMax = -HUGE\_VAL;  yMin = HUGE\_VAL;  nOfPoints = (int)floor(((xMax - xMin) / step)) + 10;  realPoints = new Point[nOfPoints];  int c = 0;  for (float x = xMin; x < xMax; x += step)  {  float y = cos(x / 2) \* cos(x / 2) + sqrt(x);  Point point(x, y);  realPoints[c] = point;  yMax = point.y > yMax ? point.y : yMax;  yMin = point.y < yMin ? point.y : yMin;  c++;  }  nOfPoints = c;  }  void buildPlotPoints()  {  buildScale();  plotPoints = new Point[nOfPoints];  for (int i = 0; i < nOfPoints; i++)  {  float x = (realPoints[i].x - h1) \* h + sizeX / 2;  float y = (-realPoints[i].y + v1) \* v + sizeY / 2;  Point p(x, y);  plotPoints[i] = p;  }  }  void drawPlot()  {  setcolor(WHITE);  for (int i = 1; i < nOfPoints; i++)  line(plotPoints[i - 1].x, plotPoints[i - 1].y, plotPoints[i].x, plotPoints[i].y);  }  void drawPlotText()  {  setviewport(0, 0, 0, 0, 0);  settextstyle(0, HORIZ\_DIR, 1);  outtextxy(5, 5, "SuperGrapher2000 DOS Edition, func = cos^2(x/2)+sqrt(x)");  //setviewport(0,0,0,0,0);  setlinestyle(CENTER\_LINE, 0, 1);  rectangle(  marginX - marginSmallX,  sizeY + marginY + marginSmallY,  sizeX + marginSmallX + marginX,  getmaxy() - 5);  setviewport(  marginX - marginSmallX,  sizeY + marginY + marginSmallY,  sizeX + marginSmallX + marginX,  getmaxy() - 5, 0);  setcolor(YELLOW);  char\* ymax = new char[10];  char\* ymin = new char[10];  char\* xmin = new char[10];  char\* xmax = new char[10];  ymin = gcvt(round(yMin), 10, ymin);  ymax = gcvt(round(yMax), 10, ymax);  xmin = gcvt(round(xMin), 10, xmin);  xmax = gcvt(round(xMax), 10, xmax);  //draw yMin, yMax  //5 - hor offset, -10 - vertical offset  outtextxy(5, 5, "Max y: ");  outtextxy(7 \* 8 + 5, 5, ymax);  outtextxy(5, 16, "Min y: ");  outtextxy(7 \* 8 + 5, 16, ymin);  outtextxy(5, 26, "Max x: ");  outtextxy(7 \* 8 + 5, 26, xmax);  outtextxy(5, 36, "Min x: ");  outtextxy(7 \* 8 + 5, 36, xmin);  settextstyle(0, HORIZ\_DIR, 1);  setcolor(RED);  outtextxy(5, 66, "THANKS FOR USING SUPERGRAPHER2000 DOS EDITION!!!");  outtextxy(5, 76, "DONATE ME PLZ I WANT TO KILL MYSELF");  }  void drawPlotOrds()  {  //int ordMargin = 10;  setlinestyle(SOLID\_LINE, 0, THICK\_WIDTH);  setcolor(12);  //verical  line((int)kh, sizeY + marginSmallY, (int)kh, -marginSmallY);  //horizontal  line(-marginSmallX, (int)kv, sizeX + marginSmallX, (int)kv);  setcolor(WHITE);  char\* ymax = new char[10];  char\* ymin = new char[10];  char\* xmin = new char[10];  char\* xmax = new char[10];  ymin = gcvt(round(yMin), 10, ymin);  ymax = gcvt(round(yMax), 10, ymax);  xmin = gcvt(round(xMin), 10, xmin);  xmax = gcvt(round(xMax), 10, xmax);  //draw yMin, yMax  //5 - hor offset, -10 - vertical offset  outtextxy((int)kh + 5, -10, ymax);  outtextxy((int)kh + 5, sizeY + 2, ymin);  //draw xMin, xMax  outtextxy(-8 \* 5, (int)kv + 5, xmin);  outtextxy(sizeX + 5, (int)kv + 5, xmax);  }  public:  Plotter()  {  marginX = 76;  marginY = 50;  marginSmallX = 50;  marginSmallY = 24;  }  int InitGraphics()  {  int gresult;  // Detect the graphics driver and mode  detectgraph(&gMode, &gDriver);  // initialize the graphics mode with initgraph  initgraph(&gMode, &gDriver, "C:\\BGI");  gresult = graphresult();  if (gresult != grOk)  {  cout << grapherrormsg(gresult);  return -1;  }  sizeX = getmaxx() - marginX \* 2 - 1;  sizeY = getmaxy() - marginY \* 2 - 1 - getmaxy() / 4;  setviewport(marginX, marginY, getmaxx() - marginX, getmaxy() - marginY, 0);  // set the background color  setbkcolor(BLACK);  // set the foreground color  setcolor(WHITE);  // draw a white color border with rectangle  rectangle(0, 0, sizeX, sizeY);  rectangle(-marginSmallX, -marginSmallY, sizeX + marginSmallX, sizeY + marginSmallY);  }  void Draw()  {  buildRealPoints();  buildPlotPoints();  drawPlotOrds();  drawPlot();  drawPlotText();  }  void Dispose()  {  delete[] realPoints, plotPoints;  closegraph();  clrscr();  }  };  int main()  {  Plotter plot;  if (!plot.InitGraphics())  return 0;  plot.Draw();  getch();  plot.Dispose();  return 1;  } |

# Вывод

В результате выполнения данной работы были получены навыки инициализации и управления графической видеосистемой ПЭВМ.