**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. 3311 |  | Шарпинский Д. А. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2024

**Краткие сведения о видеосистемах ПЭВМ, графическом режиме их работы и функциях обслуживания графического режима**

Видеосистема персонального компьютера (ПЭВМ) является важным компонентом, обеспечивающим отображение графической и текстовой информации на экране. Графический режим работы видеосистемы позволяет выводить на экран изображения, состоящие из пикселов, что делает возможным отображение сложных графических примитивов и изображений. В данном разделе рассмотрены основные аспекты видеосистем ПЭВМ, графического режима их работы и функций обслуживания графического режима.

Видеосистема ПЭВМ включает в себя видеоадаптер и монитор. Видеоадаптер отвечает за обработку графической информации и передачу её на монитор для отображения. Существует несколько типов видеоадаптеров, таких как CGA, EGA и VGA, каждый из которых поддерживает различные графические режимы и разрешения.

Графический режим работы видеосистемы позволяет отображать на экране изображения, состоящие из пикселов. В этом режиме каждый пиксел на экране может иметь свой собственный цвет, что делает возможным отображение сложных графических примитивов и изображений. Графический режим поддерживается различными видеоадаптерами и может быть настроен для работы с различными разрешениями и цветовыми палитрами.

Для работы с графическим режимом видеосистемы используются специальные функции, предоставляемые библиотекой графики. В языке программирования С++ для работы с графикой используется библиотека graphics.h, которая включает в себя набор функций для инициализации, закрытия и управления графическим режимом.

Перед началом работы с графическим режимом необходимо инициализировать систему графики. Для этого используется функция initgraph(), которая загружает соответствующий драйвер и устанавливает видеоадаптер в графический режим.

После инициализации системы графики можно установить цвета пикселов и палитры.

Графическое окно (viewport) - это прямоугольная область экрана, заданная пиксельными координатами левого верхнего и правого нижнего углов. Для задания окна используется функция setviewport().

Библиотека графики позволяет выводить текст в графическом режиме с использованием различных шрифтов. Для вывода текста используются функции outtext() и outtextxy().

Библиотека графики предоставляет функции для вывода основных графических примитивов, таких как отрезки прямых линий, окружности, эллипсы, прямоугольники и секторы.

**Задание на лабораторную работу**

Разработать программу для вывода на экран графика функции

Sin2(x) – Cos2(x) на промежутке от 3π/2 до 7π. 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.

**Алгоритмы и тексты отлаженных программ**

#include <GRAPHICS.H>

#include <MATH.H>

#include <CONIO.H>

#include <STDIO.H>

#define PI M\_PI

float f(float x){

    // (sin(x))\*\*2 - (cos(x))\*\*2 = -cos(2x)

    return -1 \* cos(2\*x);

}

void drawAxesLabels(float x\_left\_border, float x\_right\_border, float y\_bottom\_border, float y\_top\_border, float x\_step, float y\_step) {

    char label[50];

    float x, y;

    float screen\_x, screen\_y;

    float count = 2;

    float offsetX, scaleX;

    float offsetY, scaleY;

    setcolor(WHITE);

    for (x = x\_left\_border; x <= x\_right\_border; x += x\_step) { // draw labels on X-axis

        count += 1;

        offsetX = x - x\_left\_border;

        scaleX = 550 / (x\_right\_border - x\_left\_border);

        screen\_x = 50 + offsetX \* scaleX;

        line(screen\_x, 335, screen\_x, 325); // draw a small line

        sprintf(label, "%.1fpi", count / 2);

        outtextxy(screen\_x - 10, 340, label); // draw value of line

    }

    for (y = y\_bottom\_border + y\_step; y < y\_top\_border; y += y\_step) { // draw labels on Y-axis

        offsetY = y - y\_bottom\_border;

        scaleY = 350 / (y\_top\_border - y\_bottom\_border);

        screen\_y = 400 - offsetY \* scaleY;

        line(42, screen\_y, 57, screen\_y); // draw a small line

        sprintf(label, "%.1f", y);

        outtextxy(20, screen\_y - 10, label); // draw value of line

    }

}

int main(void) {

   int gdriver = DETECT, gmode, errorcode; // request auto detection

   float x\_left\_border = 3\*PI/2, x\_right\_border = 8\*PI; // vertical interval (task)

   float y\_bottom\_border = -2, y\_top\_border = 8; // horizontal interval

   float step = 0.01; // step

   float x, y; // variables in loop

   float screen\_x, screen\_y; // x, y coordinates in loop

   float y\_max\_value = -32000; // max function value

   int x\_max\_coordinate, y\_max\_coordinate; // x, y coordinates of max function value

   char max\_value[50];

   float scaleX, scaleY, offsetX, offsetY;

   initgraph(&gdriver, &gmode, "//tc//bgi"); // initialize graphics and local variables

   setlinestyle(0, 1, 2); // solid line, useless, thick width

   setcolor(WHITE);

   line (50, 330, 600, 330); // X line x1, y1, x2, y2

   line(595, 325, 600, 330);

   line(595, 335, 600, 330);

   outtextxy(590, 340, "X");

   line(50, 400, 50, 50); // Y line

   line(45, 55, 50, 50);

   line(55, 55, 50, 50);

   outtextxy(30, 50, "Y");

   drawAxesLabels(x\_left\_border, x\_right\_border, y\_bottom\_border, y\_top\_border, PI / 2, 1);

   for(x = x\_left\_border; x < x\_right\_border - PI; x += step) { // x from 3pi/2 to 17pi

      y = f(x);

      scaleX = 550 / (x\_right\_border - x\_left\_border);

      scaleY = 350 / (y\_top\_border - y\_bottom\_border);

      offsetX = x - x\_left\_border;

      offsetY = y - y\_bottom\_border;

      screen\_x = 50 + offsetX \* scaleX; // '50' is offset to 'oy' // 'x-x\_left\_border' is change '[xleft, xright]' to '[0, xright-xleft] // 550 / (x\_r\_border - x\_l\_border) is scale (size) = single step in the interval Δ𝑥 will be converted to a proportional number of pixels on the screen

      screen\_y = 400 - offsetY \* scaleY; // '400' is offset to 'ox' // 'y-y\_bottom\_border' is change '[ybottom, ytop]' to '[0, ytop-ybottom] // 350 / (y\_t\_border - y\_b\_border) is scale (size) = single step in the interval Δ𝑥 will be converted to a proportional number of pixels on the screen

      putpixel(screen\_x, screen\_y, WHITE); // put pixel on screen\_x, screen\_y coordinates

      if(y > y\_max\_value){

         x\_max\_coordinate = screen\_x;

         y\_max\_coordinate = screen\_y;

         y\_max\_value = y;

      }

   }

   settextstyle(0, 0, 2);

   sprintf(max\_value, "Max Value: %.2f", y\_max\_value);

   setcolor(RED);

   circle(x\_max\_coordinate, y\_max\_coordinate, 10); // draw a circle in coordinates with max function value

   outtextxy(x\_max\_coordinate + 10, y\_max\_coordinate-20, "max value");

   setcolor(WHITE);

   settextstyle(0, 0, 1);

   outtextxy(50, 420, "f(x) = (sin(x))\*\*2 - (cos(x))\*\*2");

   outtextxy(50, 450, max\_value);

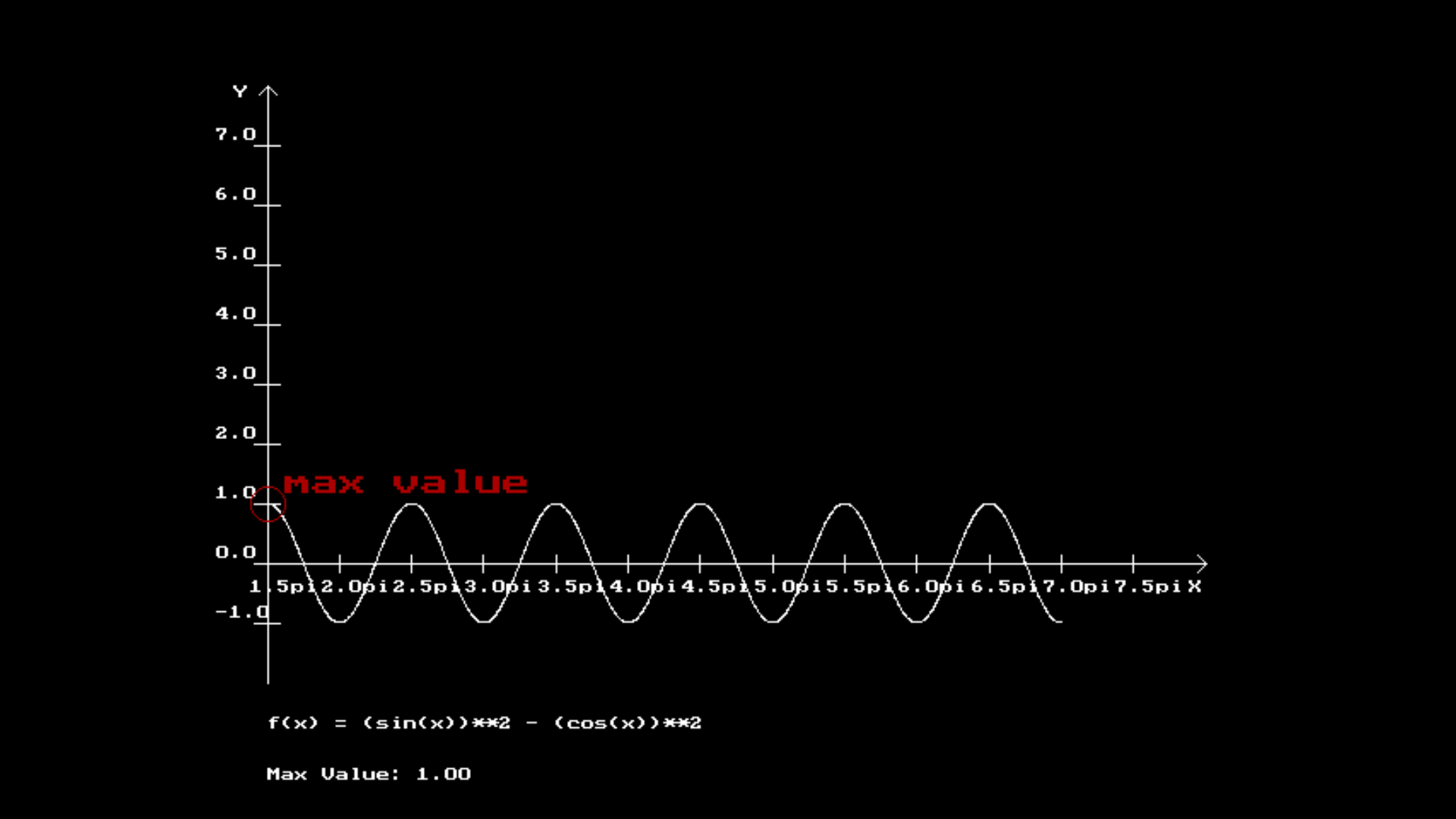
   getch(); // wait user action

   closegraph(); // clear and close

   return 0;

}

**Пример запуска программы**



**Выводы**

Видеосистема ПЭВМ и графический режим её работы являются важными компонентами для отображения графической информации на экране. Библиотека графики в языке программирования С++ предоставляет широкий набор функций для инициализации, управления и вывода графических примитивов, что делает возможным создание сложных графических приложений.