**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 |  | Сапронов К. Д. |
| --- | --- | --- |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2024

**Цель работы**

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

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

Видеосистема ПЭВМ является ключевым компонентом, обеспечивающим вывод графической и текстовой информации на экран. Графический режим работы видеосистемы позволяет отображать на экране изображения, состоящие из пикселей, что делает возможным вывод сложных графических элементов и изображений. В данном разделе рассматриваются основные особенности видеосистем ПК, графический режим их работы и функции для обслуживания этого режима.

Видеосистема ПЭВМ состоит из видеоадаптера и монитора. Видеоадаптер обрабатывает графическую информацию и передаёт 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 выводить изображения, состоящие из пикселей, где каждый пиксель может иметь свой цвет, что позволяет отображать сложные графические примитивы и изображения. Этот режим поддерживается различными видеоадаптерами и может быть настроен для работы с разными разрешениями и цветовыми палитрами.

Для работы с графическим режимом используются специальные функции, предоставляемые библиотекой графики. В языке программирования C++ для работы с графикой применяется библиотека graphics.h, содержащая функции для инициализации, завершения и управления графическим режимом.

Перед началом работы в графическом режиме необходимо инициализировать систему графики с помощью функции initgraph(), которая загружает нужный драйвер и переводит видеоадаптер в графический режим.

После инициализации системы графики можно настроить цвета пикселей и палитру. Графическое окно (viewport) — это прямоугольная область экрана, задаваемая координатами левого верхнего и правого нижнего углов. Для задания окна используется функция setviewport().

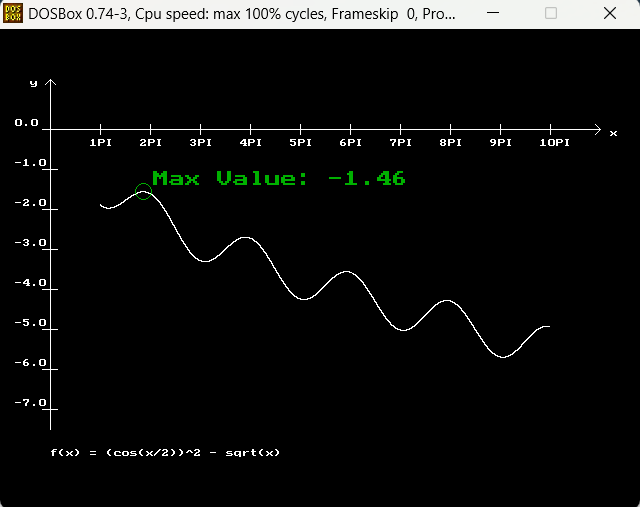
Библиотека графики также позволяет выводить текст в графическом режиме, используя различные шрифты. Для вывода текста применяются функции outtext() и outtextxy().

Кроме того, библиотека предоставляет функции для рисования базовых графических примитивов, таких как линии, окружности, эллипсы, прямоугольники и сектора.

**Задание (вариант 18)**

1. Разработать программу для вывода на экран графика функции Cos2(x/2)-Sqrt(x) на промежутке от π до 10π.
2. Произвести разметку осей и проставить истинные значения точек.
3. Найти максимальное значение функции на заданном интервале и вывести в отдельное окно на экране вместе с графиком.

**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.

**Текст программы**

#include <GRAPHICS.H>

#include <MATH.H>

#include <CONIO.H>

#include <STDIO.H>

#define PI M\_PI

// Function to draw labels on the X and Y axes

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 offsetX, scaleX;

float offsetY, scaleY;

int count = 1;

setcolor(WHITE);

// Draw labels on the X-axis starting from PI up to 10 \* PI

for (x = x\_left\_border + PI; x <= x\_right\_border; x += x\_step) {

offsetX = x - x\_left\_border;

scaleX = 500 / (x\_right\_border - x\_left\_border);

screen\_x = 50 + offsetX \* scaleX;

line(screen\_x, 105, screen\_x, 95); // Draw a small tick mark for the label

// Display X-axis labels in terms of PI

sprintf(label, "%dPI", count);

outtextxy(screen\_x - 10, 110, label);

count++;

}

// Draw labels on the Y-axis starting from the bottom up to the top

for (y = y\_bottom\_border; y <= y\_top\_border; y += y\_step) {

offsetY = y - y\_bottom\_border;

scaleY = 300 / (y\_top\_border - y\_bottom\_border);

screen\_y = 400 - offsetY \* scaleY;

line(42, screen\_y, 57, screen\_y); // Draw a small tick mark for the label

sprintf(label, "%.1f", y);

outtextxy(15, screen\_y - 10, label); // Display Y-axis label

}

}

int main(void) {

int gdriver = DETECT, gmode;

float x\_left\_border = 0, x\_right\_border = 10 \* PI;

float y\_bottom\_border = -7, y\_top\_border = 0;

float step = 0.00005; // Step size for smooth graph drawing

float x, y;

float screen\_x, screen\_y;

float y\_max\_value = -32000; // Initialize maximum value for the function

int x\_max\_coordinate, y\_max\_coordinate;

char max\_value[50];

float scaleX, scaleY, offsetX, offsetY;

// Initialize graphics mode

initgraph(&gdriver, &gmode, "C://TURBO3//BIN");

setlinestyle(0, 1, 2);

setcolor(WHITE);

// Draw X-axis at the top of the screen

line(50, 100, 600, 100);

line(595, 95, 600, 100);

line(595, 105, 600, 100);

outtextxy(610, 100, "x");

// Draw Y-axis

line(50, 420, 50, 50);

line(45, 55, 50, 50);

line(55, 55, 50, 50);

outtextxy(30, 50, "y");

// Draw the labels on the X and Y axes

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

// Scaling factors for converting to screen coordinates

scaleX = 500 / (x\_right\_border - x\_left\_border);

scaleY = 300 / (y\_top\_border - y\_bottom\_border);

// Plot the function graph

for (x = x\_left\_border + PI; x < x\_right\_border; x += step) {

y = pow(cos(x / 2), 2) - sqrt(x);

offsetX = x - x\_left\_border;

offsetY = y - y\_bottom\_border;

screen\_x = 50 + offsetX \* scaleX;

screen\_y = 400 - offsetY \* scaleY;

// Draw pixel if it fits within the screen boundaries

if (screen\_y >= 50 && screen\_y <= 400) {

putpixel(screen\_x, screen\_y, WHITE);

}

// Check if current y is the maximum value encountered

if (y > y\_max\_value) {

x\_max\_coordinate = screen\_x;

y\_max\_coordinate = screen\_y;

y\_max\_value = y;

}

}

// Display max value and mark it on the graph

settextstyle(0, 0, 2);

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

setcolor(GREEN);

circle(x\_max\_coordinate, y\_max\_coordinate, 8);

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

// Display the function formula

setcolor(WHITE);

settextstyle(0, 0, 1);

outtextxy(70, 420, "f(x) = (cos(x/2))^2 - sqrt(x)");

getch(); // Wait for user input to exit

closegraph(); // Close graphics mode

return 0;

}