Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Отчет по лабораторной работе №3**

**по дисциплине “Организация ЭВМ и систем”**

**на тему: “Исследование видеосистемы (графический режим)”**

Выполнил: Яловега Н.В. 9308

Принял: Жандаров В.В.

# **Содержание**

[Содержание 2](#__RefHeading___Toc407_355822346)

[1. Задание 3](#__RefHeading___Toc409_355822346)

[2. Краткие теоретические сведения 4](#__RefHeading___Toc411_355822346)

[3. Листинг программы 7](#__RefHeading___Toc413_355822346)

[4. Примеры запуска программ 10](#__RefHeading___Toc415_355822346)

[Вывод 11](#__RefHeading___Toc421_355822346)

# **1. Задание**

Общее задание:

Разработать программу для вывода на экран графика заданной функции.

Произвести разметку осей и проставить истинные значения точек.

Найти максимальное значение функции на заданном интервале и вывести в отдельное окно на экране.

Вариант 26(6)

Таблица 1. Входные данные для 26 (6) варианта.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Функция | Диапазон аргумента | |
| Начало | Конец |
| 6 | cos2(x/4)+sqrt(x) | 3π/2 | 16π |

# **2. Краткие теоретические сведения**

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.

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).

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.

Графические режимы, поддерживаемые библиотекой графики, задаются символическими константами, описанными в заголовочном файле <graphics.h> в перечислимом типе graphics\_modes.

Функция detectgraph() вызывается автоматически из функции инициализации видеосистемы initgraph(), если последняя вызывается со значением для graphdriver, равным указателю на DETECT.

Защищенное от ошибок построение программы требует использования функции graphresult() после любого обращения к функциям detectgraph() и initgraph(). Далее следует описание функций обработки ошибок, сообщающих внутренние коды ошибок графической библиотеки (graphresult()) или формирующей строку диагностического сообщения (grapherrormsg()).

После того, как проведена инициализация графической системы, может быть установлен другой, не превосходящий максимального, режим видеоадаптера и выбраны цвета для пикселов. Установку режима выполняет функция setgraphmode(). Целая группа функций – getgraphmode(), getmaxmode(), getmodename() , getmoderange() - упрощает работу по определению текущего ус­тановленного режима

После инициализации системы графики и уста­новки нужного видеорежима возможен выбор необходимых цветов пикселов. Возможности по выбору цветов принципиально различны для CGA-, EGA- и VGA-адаптеров, что обусловлено различной логикой построения аппаратных средств.

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.

Окно экрана в графическом режиме, или графическое окно (viewport), - это прямоугольная область экрана, заданная пиксельными координатами левого верхнего и правого нижнего углов.

Графические координаты X и Y измеряются в пикселах экрана относительно координат левого верхнего угла текуще­го окна. Функции графического вывода изменяют эти координаты в соответствии с объемом выведенной на экран инфор­мации. Текущие координаты в окне доступны через функции getx() и gety(). Установку нужных значений координат текущей позиции выполняют функции moveto() и moverel(). Кроме того, некоторые функции графического вывода позволяют задать текущую позицию (см., например, outtextxy()).

int getx (void)

int gety (void)

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)

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.

# **3. Листинг программы**

#include <graphics.h>

#include <dos.h>

#include <conio.h>

#include <stdio.h>

#include <math.h>

float func(float x)

{

return cos(x/4)\*cos(x/4)+sqrt(x);

}

int main()

{

const float PI = 3.1415;

int graph\_driver,

graph\_mode,

graph\_error\_code,

width,

height;

float scale,

oldx, oldy,

newx, newy,

max;

detectgraph(&graph\_driver,&graph\_mode);

initgraph(&graph\_driver, &graph\_mode, "C:\\TURBOCPP\\BGI");

graph\_error\_code = graphresult();

if (graph\_error\_code != grOk)

{

closegraph();

cputs("ERROR\n");

return 255;

}

setbkcolor(BLACK);

setcolor(CYAN);

width = getmaxx();

height = getmaxy();

line(width/2, 0, width/2, height);

line(0, height/2, width, height/2);

scale = 6;

setcolor(WHITE);

oldx = 3/2 \* PI;

oldy = func(oldx);

max = oldy;

while (oldx < 16\*PI)

{

newx = oldx + PI/16;

newy = func(newx);

line(oldx \* scale + width/2, height/2 - oldy \* scale, newx \* scale + width/2, height/2 - newy \* scale);

oldx = newx;

oldy = newy;

if (max < newy)

max = newy;

}

setcolor(RED);

moveto(width/2 + 5, height/2 + 5);

outtext("0");

moveto(width/2 + 5, 2);

outtext("y");

moveto(width - 15, height/2 + 5);

outtext("x");

setlinestyle(1,0,0);

line(3/2\*PI\*scale+width/2, height/2 - newy\*scale, 3/2\*PI\*scale + width/2, height/2 + 30 );

line(16\*PI\*scale+width/2, height/2 - newy\*scale, 16\*PI\*scale + width/2, height/2 + 30 );

moveto(3/2 \* PI \* scale + width/2 + 5, height/2 + 30);

outtext("3/2\*PI");

moveto(16 \* PI \* scale + width/2 - 40, height/2 + 30);

outtext("16\*PI");

setcolor(WHITE);

char s[80];

sprintf(s, "MAX VALUE: %f", max);

outtextxy (1, 1, s);

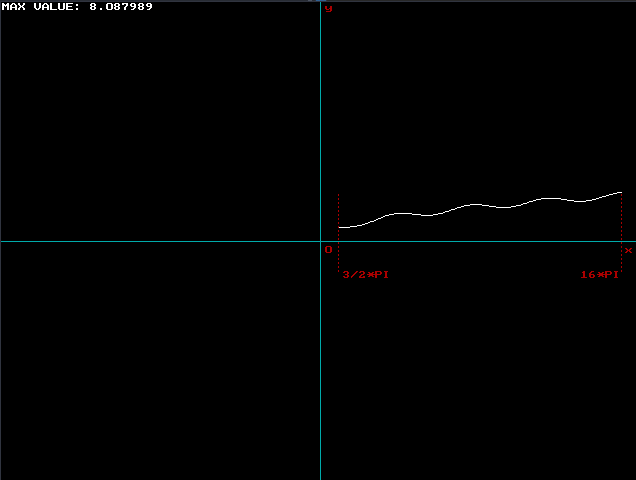
getch();

closegraph();

return 0;

}

# **4. Примеры запуска программ**

Рисунок 1: Пример работы программы

# **Вывод**

В ходе лабораторной работы научились пользоваться графическим режимом. Задание выполнено в полном объёме.