МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**ОТЧЁТ**

**по лабораторной работе № 4**

**«Закрашивание фигур*»***

по дисциплине:

**«Компьютерная графика»**

|  |  |
| --- | --- |
| Выполнил(а):  Дашкин С.М. Группа АТ-34 | Проверил:  Смагин С.М. |

Новосибирск

2024

**ЦЕЛЬ РАБОТЫ**

Изучить алгоритм заполнения произвольной фигуры, заданной своей границей.

**ЗАДАНИЕ**

Разработать программу, которая закрашивает произвольную фигуру (окружность с внутренними треугольниками и/или окружностями) по алгоритму заполнения с затравкой.

**ХОД РАБОТЫ:**

1. **setLocale()**:

- Устанавливает локаль для программы на русский язык, чтобы корректно отображались русские символы.

2. **createGrid(HDC hdc, int& cols, int& rows, int& cell\_side\_len)**:

- Создает сетку на холсте (HDC). Пользователь вводит количество столбцов и строк.

- Рисует вертикальные и горизонтальные линии сетки с заданным количеством столбцов и строк.

3. **fillCell(HDC hdc, int cols, int rows, int cell\_side\_len, int x, int y, char object)**:

- Заполняет ячейку сетки в заданных координатах (x, y) прямоугольником или кругом в зависимости от значения `object` ('r' для прямоугольника, 'c' для круга).

4. **getCellColor(HDC hdc, int rows, int cell\_side\_len, int x, int y)**:

- Возвращает цвет пикселя в заданных координатах (x, y) на сетке.

5. **drawLine(HDC hdc, int cols, int rows, int cell\_side\_len, int x1, int y1, int x2, int y2)**:

- Рисует линию между двумя точками (x1, y1) и (x2, y2) на сетке, используя алгоритм Брезенхема.

6. **drawTriangle(HDC hdc, int cols, int rows, int cell\_side\_len)**:

- Рисует треугольник, запрашивая у пользователя координаты трех точек и используя функцию `drawLine` для рисования сторон треугольника.

7. **circ\_8(HDC hdc, int cols, int rows, int cell\_side\_len, int ix0, int iy0, int x, int y, int col)**:

- Рисует 8 симметричных точек относительно центра окружности для алгоритма Брезенхема.

8. **drawCircle(HDC hdc, int cols, int rows, int cell\_side\_len)**:

- Рисует окружность, запрашивая у пользователя координаты центра и радиус, и используя алгоритм Брезенхема для рисования окружности.

9. **fillFigure(HDC hdc, int cols, int rows, int cell\_side\_len, int x, int y, COLORREF color)**:

- Заполняет фигуру, начиная с заданной точки (x, y), используя алгоритм заливки с затравкой.

10**. main()**:

- Основная функция программы, которая инициализирует локаль, создает окно консоли и получает контекст устройства (HDC).

- Создает сетку и предлагает пользователю выбрать действие: выбрать цвет, нарисовать круг, нарисовать треугольник или закрасить фигуру.

- В зависимости от выбора пользователя вызывает соответствующие функции для выполнения действий.

**ЛИСТИНГ:**

#include <iostream>

#include <windows.h>

#include <windowsx.h>

#include <vector>

#include <locale.h>

#define X\_OFFSET 500

#define Y\_OFFSET 40

void setLocale**()** **{**

setlocale**(**LC\_ALL**,** "Russian"**);**

**}**

void createGrid**(**HDC hdc**,** int**&** cols**,** int**&** rows**,** int**&** cell\_side\_len**)** **{**

std**::**cout **<<** "Введите размер сетки(cols rows): "**;**

std**::**cin **>>** cols **>>** rows**;**

cell\_side\_len **\*=** 100 **/** **(**float**)**rows**;**

**for** **(**int i **=** 0**;** i **<=** cols**;** **++**i**)** **{**

MoveToEx**(**hdc**,** X\_OFFSET **+** i **\*** cell\_side\_len**,** Y\_OFFSET**,** **NULL);**

LineTo**(**hdc**,** X\_OFFSET **+** i **\*** cell\_side\_len**,** Y\_OFFSET **+** rows **\*** cell\_side\_len**);**

**}**

**for** **(**int i **=** 0**;** i **<=** rows**;** **++**i**)** **{**

MoveToEx**(**hdc**,** X\_OFFSET**,** Y\_OFFSET **+** i **\*** cell\_side\_len**,** **NULL);**

LineTo**(**hdc**,** X\_OFFSET **+** cols **\*** cell\_side\_len**,** Y\_OFFSET **+** i **\*** cell\_side\_len**);**

**}**

**}**

void fillCell**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**,** int x**,** int y**,** char object**)** **{**

**if** **(**x **<** 1 **||** x **>** cols **||** y **<** 1 **||** y **>** rows**)** **return;**

int xPos **=** X\_OFFSET **+** **(**x **-** 1**)** **\*** cell\_side\_len**;**

int yPos **=** Y\_OFFSET **+** **(**rows **-** y**)** **\*** cell\_side\_len**;**

**if** **(**object **==** 'r'**)** **{**

Rectangle**(**hdc**,** xPos**,** yPos**,** xPos **+** cell\_side\_len**,** yPos **+** cell\_side\_len**);**

**}**

**else** **if** **(**object **==** 'c'**)** **{**

Ellipse**(**hdc**,** xPos**,** yPos**,** xPos **+** cell\_side\_len**,** yPos **+** cell\_side\_len**);**

**}**

**}**

COLORREF getCellColor**(**HDC hdc**,** int rows**,** int cell\_side\_len**,** int x**,** int y**)** **{**

**return** GetPixel**(**hdc**,** X\_OFFSET **+** **(**x **-** 1**)** **\*** cell\_side\_len **+** 1**,** Y\_OFFSET **+** **(**rows **-** y**)** **\*** cell\_side\_len **+** 1**);**

**}**

void drawLine**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**,** int x1**,** int y1**,** int x2**,** int y2**)** **{**

int xSign **=** x1 **>** x2 **?** **-**1 **:** 1**;**

int ySign **=** y1 **>** y2 **?** **-**1 **:** 1**;**

int dx **=** abs**(**x2 **-** x1**),** dy **=** abs**(**y2 **-** y1**);**

int err **=** dx **-** dy**;**

int x **=** x1**,** y **=** y1**;**

**while** **(true)** **{**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** x**,** y**,** 'r'**);**

**if** **(**x **==** x2 **&&** y **==** y2**)** **break;**

int e2 **=** err **\*** 2**;**

**if** **(**e2 **>** **-**dy**)** **{**

err **-=** dy**;**

x **+=** xSign**;**

**}**

**if** **(**e2 **<** dx**)** **{**

err **+=** dx**;**

y **+=** ySign**;**

**}**

**}**

**}**

void drawTriangle**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**)** **{**

int P1**[**2**],** P2**[**2**],** P3**[**2**];**

std**::**cout **<<** "Введите 3 точки (x1, y1, x2, y2, x3, y3): "**;**

std**::**cin **>>** P1**[**0**]** **>>** P1**[**1**]** **>>** P2**[**0**]** **>>** P2**[**1**]** **>>** P3**[**0**]** **>>** P3**[**1**];**

drawLine**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** P1**[**0**],** P1**[**1**],** P2**[**0**],** P2**[**1**]);**

drawLine**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** P3**[**0**],** P3**[**1**],** P2**[**0**],** P2**[**1**]);**

drawLine**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** P1**[**0**],** P1**[**1**],** P3**[**0**],** P3**[**1**]);**

**}**

void circ\_8**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**,** int ix0**,** int iy0**,** int x**,** int y**,** int col**)** **{**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **+** x**,** iy0 **+** y**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **+** x**,** iy0 **-** y**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **-** x**,** iy0 **+** y**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **-** x**,** iy0 **-** y**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **+** y**,** iy0 **+** x**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **+** y**,** iy0 **-** x**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **-** y**,** iy0 **+** x**,** col**);**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0 **-** y**,** iy0 **-** x**,** col**);**

**}**

void drawCircle**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**)** **{**

int ix0**,** iy0**,** ir**;**

std**::**cout **<<** "Введите центр окружности и радиус(x, y, r): "**;**

std**::**cin **>>** ix0 **>>** iy0 **>>** ir**;**

int x **=** 0**,** y **=** ir**,** d **=** 3 **-** **(**ir **<<** 1**);**

**while** **(**x **<** y**)** **{**

circ\_8**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0**,** iy0**,** x**,** y**,** 'r'**);**

**if** **(**d **<** 0**)** **{**

d **+=** **(**x **<<** 2**)** **+** 6**;**

**}**

**else** **{**

d **+=** **((**x **-** y**)** **<<** 2**)** **+** 10**;**

**--**y**;**

**}**

**++**x**;**

**}**

**if** **(**x **==** y**)** **{**

circ\_8**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** ix0**,** iy0**,** x**,** y**,** 'r'**);**

**}**

**}**

void fillFigure**(**HDC hdc**,** int cols**,** int rows**,** int cell\_side\_len**,** int x**,** int y**,** COLORREF color**)** **{**

LOGBRUSH logBrush**;**

GetObject**((**HBRUSH**)::**GetStockObject**(**DC\_BRUSH**),** **sizeof(**LOGBRUSH**),** **&**logBrush**);**

**if** **(**logBrush**.**lbColor **==** color**)** **return;**

std**::**vector**<**std**::**pair**<**int**,** int**>>** stack**;**

stack**.**push\_back**({** x**,** y **});**

**while** **(!**stack**.**empty**())** **{**

std**::**pair**<**int**,** int**>** point **=** stack**.**back**();**

stack**.**pop\_back**();**

int cx **=** point**.**first**;**

int cy **=** point**.**second**;**

**if** **(**getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** cx**,** cy**)** **!=** color**)** **continue;**

fillCell**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** cx**,** cy**,** 'r'**);**

**if** **(**cy **>** 1 **&&** getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** cx**,** cy **-** 1**)** **==** color**)** **{**

stack**.**push\_back**({** cx**,** cy **-** 1 **});**

**}**

**if** **(**cy **<** rows **&&** getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** cx**,** cy **+** 1**)** **==** color**)** **{**

stack**.**push\_back**({** cx**,** cy **+** 1 **});**

**}**

**if** **(**cx **>** 1 **&&** getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** cx **-** 1**,** cy**)** **==** color**)** **{**

stack**.**push\_back**({** cx **-** 1**,** cy **});**

**}**

**if** **(**cx **<** cols **&&** getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** cx **+** 1**,** cy**)** **==** color**)** **{**

stack**.**push\_back**({** cx **+** 1**,** cy **});**

**}**

**}**

**}**

int main**()** **{**

setLocale**();**

HWND hwnd **=** GetConsoleWindow**();**

HDC hdc **=** GetDC**(**hwnd**);**

HPEN pen **=** CreatePen**(**PS\_SOLID**,** 1**,** RGB**(**255**,** 255**,** 255**));**

HBRUSH brush **=** **(**HBRUSH**)::**GetStockObject**(**DC\_BRUSH**);**

SelectObject**(**hdc**,** pen**);**

SelectObject**(**hdc**,** brush**);**

int cols**,** rows**,** cell\_side\_len **=** 4**;**

createGrid**(**hdc**,** cols**,** rows**,** cell\_side\_len**);**

**for** **(**int i**,** x**,** y**,** r**,** g**,** b**;;)** **{**

std**::**cout **<<** "1. Выбрать цвет \n2. Нарисовать круг \n3. Нарисовать треугольник \n4. Закрасить фигуру\n"**;**

std**::**cin **>>** i**;**

**switch** **(**i**)** **{**

**case** 1**:** **{**

std**::**cout **<<** "Введите значение от 0 до 255 (r g b): "**;**

std**::**cin **>>** r **>>** g **>>** b**;**

SetDCBrushColor**(**hdc**,** RGB**(**r**,** g**,** b**));**

**break;**

**}**

**case** 2**:** **{**

drawCircle**(**hdc**,** cols**,** rows**,** cell\_side\_len**);**

**break;**

**}**

**case** 3**:** **{**

drawTriangle**(**hdc**,** cols**,** rows**,** cell\_side\_len**);**

**break;**

**}**

**case** 4**:** **{**

std**::**cout **<<** "Введите начальную точку для заливки(x y): "**;**

std**::**cin **>>** x **>>** y**;**

fillFigure**(**hdc**,** cols**,** rows**,** cell\_side\_len**,** x**,** y**,** getCellColor**(**hdc**,** rows**,** cell\_side\_len**,** x**,** y**));**

**break;**

**}**

**default:**

**goto** end**;**

**}**

**}**

end**:**

DeleteObject**(**pen**);**

DeleteObject**(**brush**);**

ReleaseDC**(**hwnd**,** hdc**);**

int i**;**

std**::**cin **>>** i**;**

**}**

**ТЕСТИРОВАНИЕ:**

