Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные системы и системное программирование

Отчёт
к лабораторной работе
на тему

Основы программирования в Win 32 API. Оконное приложение Win 32 с минимальной достаточной функциональностью. Обработка основных оконных сообщений.

Студент: гр.153502
Александрёнок И.А.

Проверил: Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Возобновление, закрепление и развитие навыков программирования оконных приложений Windows: структура приложения, цикл обработки сообщений, организация взаимодействия посредством сообщений, создание и использование окон и оконных элементов управления, использование базовых средств графики Windows.

Реализовать игру "Сапер" с графическим интерфейсом, позволяющим пользователю открывать ячейки поля и помечать мины.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Application Programming Interface) – это набор функций и процедур, предоставляемых операционной системой Windows для разработки приложений на языке программирования C/C++. Оконное приложение Win32 – это приложение, которое состоит из одного или нескольких окон, в которых происходит взаимодействие с пользователем. Для создания окна необходимо зарегистрировать класс окна с помощью функции RegisterClassEx и создать окно с помощью функции CreateWindowEx. Окно может иметь различные свойства, такие как заголовок, размеры, стиль и обработчики сообщений. Важным аспектом программирования в Win32 API является обработка оконных сообщений. Оконные сообщения – это события, которые происходят в окне, например, нажатие кнопки мыши или клавиши, изменение размера окна и другие действия пользователя. Для обработки оконных сообщений необходимо определить функцию оконной процедуры (WndProc), которая будет вызываться системой при возникновении сообщения. В функции WndProc нужно обрабатывать различные типы сообщений с помощью условных операторов и выполнять соответствующие действия.

3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Оконное приложение с простым пользовательским интерфейсом (рисунок 1). В начальном состоянии всё поле является закрытым. В процессе игры пользователь вскрывает клетки, где символ “#” представляет собой пустую клетку. В случае вскрытия клетки с бомбой выводится соответствующее сообщение происходит завершение игры. Если игрок успешно вскрывает все клетки, кроме бомб - игра завершается и выводится сообщение об успехе.

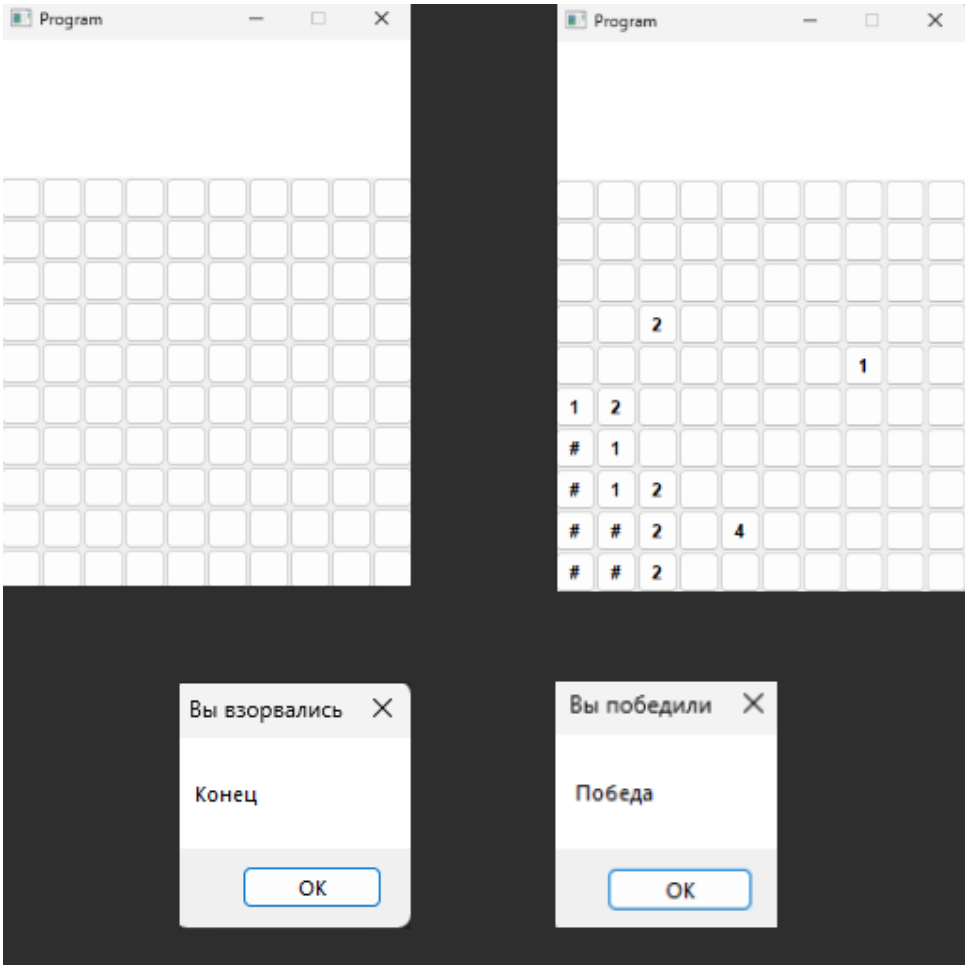

Рисунок 1 – Графический интерфейс программы

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Начало работы с классическими приложениями для Windows, которые используют API Win32 [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/desktop-programming

ПРИЛОЖЕНИЕ А
Исходный код программы

Файл MineSweeper.h

```
#pragma once
#include <utility>
using namespace std;
class MineSweeper {
/*
field - числовой массив, значение ячейки каждого - количество мин в соседних клетках
-1 - обозначение бомбы
fieldStatus - булевый массив, открыта ли ячейка в игровом окне
*/
private:
int mineCount = 15;
int fieldSize = 10;
void CreateField();
void GenerateFieldData();
void MemClear();
void IncreaseNeighbours(int i,int j);
public:
int** field;
bool** fieldStatus;
bool** markStatus;
int unrevealedCount;
MineSweeper();
void ResetField();
int GetFieldSize();
int GetMineCount();
pair<int, int> ExtractCoordinates(int x) {
int j = x % fieldSize;
int i = (x - j) / fieldSize;
return make_pair(i, j);
}
~MineSweeper();
};
```

Файл MineSweeper.cpp

```
#include "Minesweeper.h"
#include<cstdlib>
#include <iostream>

using namespace std;

MineSweeper::MineSweeper() {
srand(time(0));
CreateField();
GenerateFieldData();
}

MineSweeper::~MineSweeper() {
MemClear();
}


void MineSweeper::ResetField() {
MemClear();
CreateField();
GenerateFieldData();
}

void MineSweeper::MemClear() {
for (int i = 0; i < fieldSize; i++) {
delete[] field[i];
delete[] fieldStatus[i];
delete[] markStatus[i];
}
delete[] field;
delete[] fieldStatus;
delete[] markStatus;
}

void MineSweeper::CreateField() {

unrevealedCount = fieldSize * fieldSize;


field = new int* [fieldSize];
for (int i = 0; i < fieldSize; i++) {
field[i] = new int[fieldSize];
for (int j = 0; j < fieldSize; j++) {
field[i][j] = 0;
```

```
}
}

fieldStatus = new bool* [fieldSize];
for (int i = 0; i < fieldSize; i++) {
fieldStatus[i] = new bool[fieldSize];
for (int j = 0; j < fieldSize; j++) {
fieldStatus[i][j] = 0;
}
}
markStatus = new bool* [fieldSize];
for (int i = 0; i < fieldSize; i++) {
markStatus[i] = new bool[fieldSize];
for (int j = 0; j < fieldSize; j++) {
markStatus[i][j] = 0;
}
}
}

void MineSweeper::IncreaseNeighbours(int i, int j) {
for (int k = i - 1; k <= i + 1; k++) {
if (k < 0 || k == fieldSize) continue;
for (int p = j - 1; p <= j + 1; p++) {
if (p < 0 || p == fieldSize) continue;
if (k == i && j == p) continue;
if (field[k][p] != -1) {
field[k][p]++;
}
}
}
}


void MineSweeper::GenerateFieldData() {
int totalMines = 0;

while (totalMines < mineCount) {
int i = rand() % fieldSize, j = rand() % fieldSize;
if (field[i][j] == -1) {
continue;
}
else {
field[i][j] = -1;
totalMines++;
IncreaseNeighbours(i, j);
```

```
}
}

}

int MineSweeper::GetFieldSize() {
return fieldSize;
}
int MineSweeper::GetMineCount() {
return mineCount;
}
```

Файл main.cpp

```
   #pragma comment(linker,"\"/manifestdependency:type='win32' \
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \
processorArchitecture='*' publicKeyToken='6595b64144ccf1df'
language='*'\"")

#include <windows.h>
#include<format>
#include "Minesweeper.h"
#include <utility>
#include <commctrl.h>

MineSweeper game = MineSweeper();
int mineCount = game.GetMineCount();
int width, height;

//т.к. кнопки по умолчанию не обрабатывают нажатие ПКМ, переопределение процедуры
LRESULT CALLBACK SubclassWindowProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam, UINT_PTR uIdSubclass, DWORD_PTR dwRefData) {
switch (uMsg) {
case WM_NCDESTROY:
RemoveWindowSubclass(hWnd, SubclassWindowProc, uIdSubclass);
break;

case WM_RBUTTONDOWN:

int bId = GetDlgCtrlID(hWnd);
std::pair<int, int>coords = game.ExtractCoordinates(bId);
if (game.markStatus[coords.first][coords.second]) {
SendMessage(hWnd, WM_SETTEXT, 0, (LPARAM)(L" "));
game.markStatus[coords.first][coords.second] = false;
```

```
    }
    else {
    SendMessage(hWnd, WM_SETTEXT, 0, (LPARAM)(L"*"));
    game.markStatus[coords.first][coords.second] = true;
    }

    break;
    }

    return DefSubclassProc(hWnd, uMsg, wParam, lParam);
    }

    bool UncoverTiles(HWND hwnd, int iX, int iY);
    LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam);

    //точка входа
    int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance, PWSTR pCmdLine, int nCmdShow)
    {

    const wchar_t CLASS_NAME[] = L"Sample Window Class";
    width = game.GetFieldSize() * 30 + 16;
    height = game.GetFieldSize() * 30 + 140;



    //Регистрация окна

    WNDCLASS wc = { };

    wc.lpfnWndProc = WindowProc;        //процедура окна
    wc.hInstance = hInstance;           //дескриптор приложения
    wc.lpszClassName = CLASS_NAME;      //класс окна

    RegisterClass(&wc);

    //Создание окна

    HWND hwnd = CreateWindowEx(
    0,                          //Доп. стили (прим. прозрачные окна)
    CLASS_NAME,                     //Имя класса
    L"Program",                   //Заголовок окна
    WS_CAPTION    |    WS_SYSMENU    |    WS_MINIMIZEBOX    |
WS_CLIPCHILDREN,           //Стиль окна
```

```
// Координаты, ширина, высота окна
CW_USEDEFAULT, CW_USEDEFAULT, width, height,

NULL,
NULL,
hInstance,
NULL
);

if (hwnd == NULL)
{
return 0;
}

ShowWindow(hwnd, nCmdShow);

// все работает на сообщениях
MSG msg = { };
while (GetMessage(&msg, NULL, 0, 0) > 0)
{
TranslateMessage(&msg);
DispatchMessage(&msg);
}

return 0;
}

//процедура для обработки тех самых сообщений
LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
switch (uMsg)
{
case WM_CREATE:
{
//Создание внутренностей окна
int size = game.GetFieldSize();
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {


HWND hButton = CreateWindow(
L"BUTTON",
reinterpret_cast<LPCWSTR>(L" "),
```

```
    WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_NOTIFY,
    // x  y   w   h
    j * 30, 100 + i * 30, 30, 30, hwnd, reinterpret_cast<HMENU>(i * size + j),
0, 0
    );
    SetWindowSubclass(hButton, SubclassWindowProc, 1, 0);
    }
    }

    }

    return 0;



    case WM_COMMAND:
    {
    if (HIWORD(wParam) == WM_RBUTTONUP) {
    SendMessage(GetDlgItem(hwnd, LOWORD(wParam)), WM_SETTEXT, 0,
LPARAM(L"*"));
    }
    else {
    pair<int, int> data = game.ExtractCoordinates(LOWORD(wParam));
//координаты игровом на поле | first = i | second = j
    UncoverTiles(hwnd, data.first, data.second);
    }



    }
    return 0;

    case WM_DESTROY:
    PostQuitMessage(0);
    return 0;

    case WM_PAINT:
    {
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hwnd, &ps);
    SetBkColor(hdc, RGB(0, 255, 0));
    EndPaint(hwnd, &ps);
    }
    return 0;
```

```
}
return DefWindowProc(hwnd, uMsg, wParam, lParam);
}

void CALLBACK UncoverTile(HWND hwnd, int id, int iX, int iY, bool isZero = TRUE) {
//открытие клетки
game.fieldStatus[iX][iY] = true;
if (game.field[iX][iY] == -1) {
MessageBox(hwnd, L"Конец", L"Вы взорвались", MB_OK);
PostQuitMessage(0);
}
game.unrevealedCount--;
HWND bHWnd = GetDlgItem(hwnd, id);



if (isZero) {
SendMessage(bHWnd, WM_SETTEXT, 0, (LPARAM)(L"#"));
}
else {
std::string tempstr = std::format("{}", game.field[iX][iY]);
std::wstring temp = std::wstring(tempstr.begin(), tempstr.end());
SendMessage(bHWnd, WM_SETTEXT, 0, (LPARAM)(temp.c_str()));
}

//сделать кнопку вскрытой клетки поля неактивной
SendMessage(bHWnd, WM_ENABLE, 0, 0);

//конец игры?
if (game.unrevealedCount == mineCount) {
MessageBox(hwnd, L"Победа", L"Вы победили", MB_OK);
PostQuitMessage(0);
}
}


bool CALLBACK UncoverTiles(HWND hwnd, int iX, int iY) {
//если уже открыта - уйти
if (game.fieldStatus[iX][iY])
return true;

int fieldSize = game.GetFieldSize();

//если клетка "пограничная" - вскрыть только ее
```

```
if (game.field[iX][iY] != 0) {
UncoverTile(hwnd, iX * fieldSize + iY, iX, iY, FALSE);
return false;
}


UncoverTile(hwnd, iX * fieldSize + iY, iX, iY);
//открытие соседних клеток
for (int k = iX - 1; k <= iX + 1; k++) {
if (k < 0 || k == fieldSize) continue;
for (int p = iY - 1; p <= iY + 1; p++) {
if (p < 0 || p == fieldSize) continue;
if (k == iX && iY == p) continue;
UncoverTiles(hwnd, k, p);
}
}
return true;
}
```