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

Логвинович М.В.

Проверил: Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 ЦЕЛЬ РАБОТЫ 3](#_Toc147333809)

[2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc147333810)

[3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 5](#_Toc147333811)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 7](#_Toc147333812)

[ПРИЛОЖЕНИЕ А 8](#_Toc147333813)

# **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) представляет собой мощный набор функций и процедур, предоставляемых операционной системой Windows для разработки приложений на языке программирования C/C++. Этот интерфейс обеспечивает программистам доступ к различным функциональным возможностям операционной системы Windows, включая создание и управление оконными приложениями.

Оконные приложения в контексте разработки на Win32 API представляют собой приложения, в которых пользователь взаимодействует с программой через окна и элементы управления, такие как кнопки, текстовые поля и другие. Оконные приложения обеспечивают более удобный и графический интерфейс по сравнению с консольными приложениями, и они широко используются в современных приложениях.

Для создания окна в приложении Win32 необходимо зарегистрировать класс окна с помощью функции RegisterClassEx, которая определяет характеристики окна, такие как его имя, обработчик оконной процедуры, стиль и другие параметры. Зарегистрированный класс окна служит шаблоном для создания экземпляров окон. Затем необходимо создать окно с использованием функции CreateWindowEx. Эта функция создает экземпляр окна на основе зарегистрированного класса и возвращает дескриптор окна. При создании окна вы также можете определить его свойства, такие как заголовок, размеры, начальное положение и стиль.

Для обработки пользовательского взаимодействия и управления окном необходимо знать, как обрабатывать оконные сообщения. Оконные сообщения представляют собой события, которые происходят в окне, такие как нажатие кнопки мыши, клавиши клавиатуры, 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 используются конструкции условных операторов для обработки различных типов сообщений и выполнения соответствующих действий. Это позволяет приложению реагировать на действия пользователя и взаимодействовать с ним через интерфейс окна.

# 

# **3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

Оконное приложение представляет собой интуитивно понятный пользовательский интерфейс (см. Рисунок 1). При запуске приложения все клетки поля находятся в закрытом состоянии. В ходе игры пользователь взаимодействует с приложением, открывая клетки, в которых символ "/" обозначает пустую клетку, а символ “#” – предполагаемую клетку с миной. В случае успешного завершения игры, где пользователь вскрыл все клетки, кроме клеток с бомбами, приложение поздравляет игрока соответствующим сообщением и завершает игровой сеанс. Если же пользователь случайно открывает клетку с миной, приложение немедленно выдает соответствующее сообщение и завершает игру (см. Рисунок 2). Приложение предоставляет возможность переключения между режимами сложности (см. Рисунок 3). Выбор сложности влияет на количество бомб и размер игрового поля.

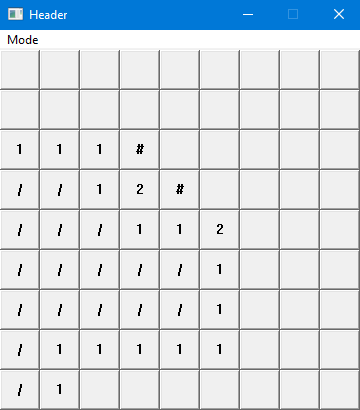


Рисунок 1 – Графический интерфейс программы

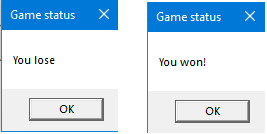


Рисунок 2 – Сообщения о результатах игры

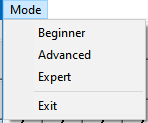


Рисунок 3 – Меню переключения режима сложности

# **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.cpp**

#include <unordered\_set>

#include <ctime>

#include <cstdlib>

using namespace std;

enum Mode {

BEGINNER, ADVANCED, EXPERT

};

struct hashFunction

{

size\_t operator()(const std::pair<short, short>& x) const {

return x.first ^ x.second;

}

};

class Minesweeper {

public:

Minesweeper(Mode mode = BEGINNER) {

switch (mode)

{

case BEGINNER:

fieldSize = 9;

break;

case ADVANCED:

fieldSize = 14;

break;

case EXPERT:

fieldSize = 20;

break;

}

opened\_cells = fieldSize \* fieldSize;

setField();

setMines();

}

short getFieldSize() {

return fieldSize;

}

wchar\_t getStatus(short i, short j) {

return field[i][j];

}

wchar\_t setAsMine(short i, short j) {

switch (field[i][j]) {

case '.':

field[i][j] = L'\*';

opened\_cells--;

break;

case L'\*':

opened\_cells++;

field[i][j] = '.';

break;

}

return field[i][j];

}

bool checkWinner() {

return opened\_cells == 0;

}

bool openCell(short i, short j) {

if (mines.contains(std::pair(i, j))) {

if (first) {

mines.erase(std::pair(i, j));

first = false;

return true;

}

else return false;

}

first = false;

return true;

}

Mode getMode() {

switch (fieldSize) {

case 9:

return BEGINNER;

break;

case 14:

return ADVANCED;

break;

case 20:

return EXPERT;

break;

}

}

wchar\_t checkCell(short i, short j) {

if (field[i][j] != '.' && field[i][j] != L'\*') return field[i][j];

int counter = 0;

for (short x = max(i - 1, 0); x <= min(i + 1, fieldSize - 1); ++x) {

for (short y = max(j - 1, 0); y <= min(j + 1, fieldSize - 1); ++y) {

if (mines.contains(pair(x, y))) counter++;

}

}

if (counter == 0) {

field[i][j] = L'/';

}

else {

field[i][j] = counter + '0';

}

opened\_cells--;

return field[i][j];

}

private:

short fieldSize;

wchar\_t\*\* field;

bool first = true;

int opened\_cells;

std::unordered\_set<std::pair<short, short>, hashFunction> mines;

void setField() {

field = new wchar\_t\* [fieldSize];

for (int i = 0; i < fieldSize; ++i) {

field[i] = new wchar\_t[fieldSize];

for (int j = 0; j < fieldSize; ++j) {

field[i][j] = '.'; // Set the default value as dot '.'

}

}

}

void setMines() {

srand(time(NULL));

if(!mines.empty()) mines.clear();

while (mines.size() < fieldSize) {

short x = static\_cast<short>(rand() % fieldSize);

short y = static\_cast<short>(rand() % fieldSize);

std::pair<short, short> mine(x, y);

if (!mines.contains(mine)) {

mines.insert(mine);

}

}

}

};

**Файл Source.cpp**

#include <Windows.h>

#include "Minesweeper.cpp"

#include <commctrl.h>

void AddMenu(HWND);

void updateMode(Mode, HWND, int, int);

void AddMinesweeperField(HWND hWnd);

void CALLBACK UpdateField(short i, short j, int size, HWND hWnd);

void CALLBACK OpenCell(std::pair<short, short> ind, HWND hWnd);

std::pair<short, short> getButtonIndices(short id);

#define MENU\_ACTION\_1 1

#define MENU\_ACTION\_2 2

#define MENU\_ACTION\_3 3

#define MENU\_ACTION\_4 4

#define IDC\_FIRST\_BUTTON 5

Minesweeper mine = Minesweeper();

LRESULT CALLBACK ButtonProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam, UINT\_PTR uIdSubclass, DWORD\_PTR dwRefData) {

switch (uMsg) {

case WM\_RBUTTONDOWN:

{

int buttonID = GetDlgCtrlID(hWnd);

std::pair<short, short> indices = getButtonIndices(buttonID);

wchar\_t buttonStatus = mine.setAsMine(indices.first, indices.second);

if(buttonStatus == '.') SendMessage(hWnd, WM\_SETTEXT, 0, (LPARAM)(L""));

else if(buttonStatus == '\*') {

SendMessage(hWnd, WM\_SETTEXT, 0, (LPARAM)(L"\*"));

if (mine.checkWinner()) {

MessageBox(hWnd, L"You won!", L"Game status", MB\_OK);

PostQuitMessage(0);

}

}

break;

}

default:

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

}

}

LRESULT CALLBACK WindowProcedure(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

switch (uMsg) {

case WM\_COMMAND:

switch (wParam) {

case MENU\_ACTION\_1:

if (mine.getMode() != BEGINNER) updateMode(BEGINNER, hWnd, 27, 28);

break;

case MENU\_ACTION\_2:

{

if (mine.getMode() != ADVANCED) updateMode(ADVANCED, hWnd, 50, 75);

break;

}

case MENU\_ACTION\_3:

{

if (mine.getMode() != ADVANCED) updateMode(EXPERT, hWnd, 120, 120);

break;

}

case MENU\_ACTION\_4:

{

int result = MessageBox(hWnd, L"Are you sure?", L"Exit", MB\_YESNO);

if (result == IDYES) {

PostQuitMessage(0);

}

break;

}

default: {

int id = LOWORD(wParam);

HWND button = GetDlgItem(hWnd, id);

OpenCell(getButtonIndices(id), hWnd);

break;

}

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_CREATE:

AddMenu(hWnd);

AddMinesweeperField(hWnd);

break;

}

return DefWindowProc(hWnd, uMsg, wParam, lParam);

}

WNDCLASS NewWindowClass(HINSTANCE hInstance) {

WNDCLASS wc{ sizeof(WNDCLASS) };

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wc.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wc.hIcon = LoadIcon(nullptr, IDI\_APPLICATION);

wc.hInstance = hInstance;

wc.lpszClassName = L"App";

wc.lpfnWndProc = WindowProcedure;

wc.lpszMenuName = nullptr;

wc.style = CS\_VREDRAW | CS\_HREDRAW;

return wc;

}

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

MSG msg{};

HWND hwnd{};

WNDCLASS wc = NewWindowClass(hInstance);

if (!RegisterClass(&wc)) {

return EXIT\_FAILURE;

}

hwnd = CreateWindow(wc.lpszClassName, L"Header", WS\_CAPTION | WS\_SYSMENU | WS\_MINIMIZEBOX, CW\_USEDEFAULT, CW\_USEDEFAULT, mine.getFieldSize() \* 45 - 27, mine.getFieldSize() \* 50 - 28, nullptr, nullptr, wc.hInstance, nullptr);

if (hwnd == INVALID\_HANDLE\_VALUE) {

return EXIT\_FAILURE;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&msg, nullptr, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return EXIT\_SUCCESS;

}

void AddMenu(HWND hWnd) {

HMENU rootMenu = CreateMenu();

HMENU subMenu = CreateMenu();

AppendMenu(subMenu, MF\_STRING, MENU\_ACTION\_1, L"Beginner");

AppendMenu(subMenu, MF\_STRING, MENU\_ACTION\_2, L"Advanced");

AppendMenu(subMenu, MF\_STRING, MENU\_ACTION\_3, L"Expert");

AppendMenu(subMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(subMenu, MF\_STRING, MENU\_ACTION\_4, L"Exit");

AppendMenu(rootMenu, MF\_POPUP, (UINT\_PTR)subMenu, L"Mode");

SetMenu(hWnd, rootMenu);

}

void AddMinesweeperField(HWND hWnd) {

int size = mine.getFieldSize();

for (int i = 0; i < size; ++i) {

for (int j = 0; j < size; ++j) {

int buttonId = IDC\_FIRST\_BUTTON + i \* size + j;

HWND button = CreateWindow(L"BUTTON", L"", WS\_VISIBLE | WS\_CHILD, 40 \* j, 40 \* i, 40, 40, hWnd, reinterpret\_cast<HMENU>(buttonId), NULL, NULL);

SetWindowSubclass(button, ButtonProc, 0, 0);

}

}

}

void CALLBACK OpenCell(std::pair<short, short> ind, HWND hWnd) {

// end game

if (!mine.openCell(ind.first, ind.second)) {

MessageBox(hWnd, L"You lose", L"Game status", MB\_OK);

PostQuitMessage(0);

}

else

{

// open cell

UpdateField(ind.first, ind.second, mine.getFieldSize(), hWnd);

}

}

std::pair<short, short> getButtonIndices(short id) {

short row\_index = (id - 5) / mine.getFieldSize();

short column\_index = (id - 5) % mine.getFieldSize();

return std::pair<short, short>(row\_index, column\_index);

}

void CALLBACK UpdateField(short i, short j, int size, HWND hWnd) {

wchar\_t status = mine.getStatus(i, j);

if (status == '/') return;

HWND button = GetDlgItem(hWnd, IDC\_FIRST\_BUTTON + i \* size + j);

wchar\_t wcharDigit[2];

wcharDigit[0] = mine.checkCell(i, j);

wcharDigit[1] = L'\0';

if (mine.checkWinner()) {

MessageBox(hWnd, L"You won!", L"Game status", MB\_OK);

PostQuitMessage(0);

}

SendMessage(button, WM\_SETTEXT, 0, (LPARAM)(wcharDigit));

if (wcharDigit[0] == '/') {

for (short x = max(i - 1, 0); x <= min(i + 1, size - 1); ++x) {

for (short y = max(j - 1, 0); y <= min(j + 1, size - 1); ++y) {

if (x != i || y != j) {

UpdateField(x, y, size, hWnd);

}

}

}

}

}

void updateMode(Mode mode, HWND hWnd, int dx, int dy) {

HWND button;

for (int i = 0; i < mine.getFieldSize(); ++i) {

for (int j = 0; j < mine.getFieldSize(); ++j) {

int buttonId = IDC\_FIRST\_BUTTON + i \* mine.getFieldSize() + j;

button = GetDlgItem(hWnd, buttonId);

DestroyWindow(button);

}

}

mine = Minesweeper(mode);

RECT clientRect = { 0, 0, 600, 600 };

int newWidth = mine.getFieldSize() \* 45 - dx;

int newHeight = mine.getFieldSize() \* 50 - dy;

AdjustWindowRect(&clientRect, WS\_OVERLAPPEDWINDOW, FALSE); // Adjust for window borders

SetWindowPos(hWnd, nullptr, 0, 0, newWidth, newHeight, SWP\_NOMOVE | SWP\_NOZORDER);

UpdateWindow(hWnd);

AddMinesweeperField(hWnd);

}