Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №1

на тему

Основы программирования в Win32 API. Оконное приложение Win32 с минимальной функциональной достаточностью. Обработка основных оконных сообщений

Выполнил: студент группы 153503

Жечко Иван Сергеевич

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[3 Результаты выполнения лабораторной работы 5](#_Toc146728179)

[Выводы 7](#_Toc146728180)

[Список использованных источников 8](#_Toc146728181)

[Приложение А (обязательное) Листинг кода 9](#_Toc146728182)

## 1 ЦЕЛЬ РАБОТЫ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API с минимальной функциональной достаточностью, использующего основные оконные сообщений. Для реализации цели была выбрана игра «Сапер».

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Интерфейс прикладного программирования Win32 API представляет собой набор функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы Windows. Этот API предоставляет средства для управления и взаимодействия с операционной системой, а также для создания приложений, которые работают в среде Windows.

Win32 API включает в себя более 3000 функций, которые позволяют реализовать различные сервисы операционной системы. Эти функции могут быть вызваны из приложений для выполнения разнообразных задач, таких как создание процессов и обработка оконных сообщений.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

­– Win32 API (Application Programming Interface): Win32 API — это набор функций и подпрограмм, предоставляемых операционной системой Windows для взаимодействия с приложениями. Он предоставляет доступ к различным функциональным возможностям Windows, таким как создание окон, обработка сообщений, работа с файлами и др.

– Обработка событий и сообщений: в Windows-приложениях взаимодействие с пользователем осуществляется через обработку событий и сообщений.

– Создание пользовательского интерфейса: для создания графического интерфейса приложения были использованы стандартные элементы управления Windows. Эти элементы были созданы и настроены с использованием Win32 API функций.

– Работа с окнами и элементами управления: для создания оконного приложения была использована функция CreateWindowEx, которая создает окно с указанными характеристиками, такими как размер, положение и стиль.

Для создания

– Интерфейс пользователя: графический интерфейс приложения разработан с учетом стандартных принципов пользовательского интерфейса, таких как расположение элементов и размещение текста.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы была разработана игра «Сапёр». Игра представляет собой поле из ячеек, которые можно открывать (рисунок 1).

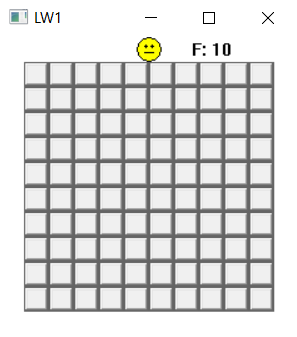


Рисунок 1 – Начальный экран

Открытые ячейки содержат информацию о количестве мин в ячейках вокруг. Ячейки могут быть помечены флагами, чтобы пометить места где могут быть мины (рисунок 2).

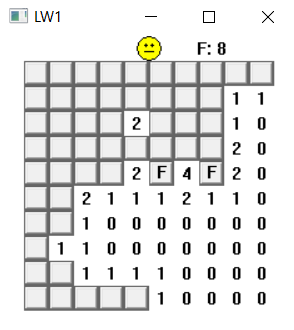


Рисунок 2 ─ Частично открытое поле с флагами

Пользователь побеждает, если открывает все ячейки без мин.

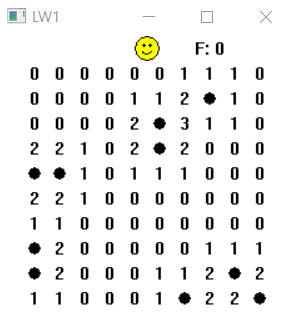


Рисунок 3 – Победа

Пользователь проигрывает, если открывает ячейку с миной.

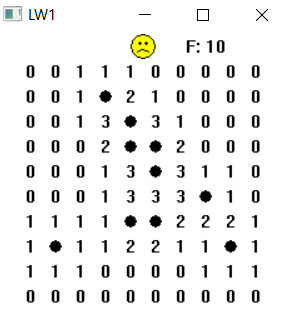


Рисунок 4 – Поражение

## ВЫВОДЫ

В результате выполнения работы было создано оконное приложение, содержащее игру “Сапер”. Были добавлены ключевые механики игры: открытие ячеек, размещение флажков, победа, поражение. В результате, было создано в Win32 API оконное приложение с базовой функциональностью и обработкой базовых сообщений.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. ─ СПб: Питер, 2008. ─ 592 с.: ил.

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 19.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

Листинг 1 – Файл minesweeper.cpp

#include <windows.h>

#include <tchar.h>

#include <time.h>

const int minecount = 10;

const int cCols = 10;

const int cRows = 10;

const int offset = 20;

int cellsLeft = cCols \* cRows - minecount;

int flagcount = minecount;

int gamestate = 0;

HWND mainWindow;

int\*\* getMinefield() {

srand(time(NULL));

int\*\* mines = new int\* [cCols];

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

mines[i] = new int[cRows];

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

mines[i][j] = 0;

}

}

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

int val = rand() % (cRows \* cCols);

while (mines[val % cCols][val / cRows] == 1) {

val = rand() % (cRows \* cCols);

}

mines[val % cCols][val / cRows] = 1;

}

int test = mines[6][2];

return mines;

}

HWND\*\* getButtonfield() {

HWND\*\* buttons = new HWND \* [cCols];

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

buttons[i] = new HWND[cRows];

}

return buttons;

}

int\*\* minefield = getMinefield();

HWND\*\* buttonfield = getButtonfield();

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK RBHookProc(int nCode, WPARAM wParam, LPARAM lParam);

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

{

// Window class.

const wchar\_t CLASS\_NAME[] = L"Window";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

const int window\_x = 255;

const int window\_y = 280;

// Window creation

HWND hwnd = CreateWindowEx(

0,

CLASS\_NAME,

L"LW1",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, window\_x, window\_y,

NULL,

NULL,

hInstance,

NULL

);

mainWindow = hwnd;

if (hwnd == NULL)

{

return 0;

}

ShowWindow(hwnd, nCmdShow);

// Message loop.

MSG msg = {};

while (GetMessage(&msg, NULL, 0, 0) > 0)

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

switch (uMsg)

{

case WM\_DESTROY: {

PostQuitMessage(0);

return 0;

}

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

HBRUSH brush = CreateSolidBrush(

RGB(255, 255, 255)

);

int test = gamestate;

FillRect(hdc, &ps.rcPaint, brush); //фон

brush = CreateSolidBrush(

RGB(255, 255, 0)

);

SelectObject(hdc, brush);

Ellipse(hdc, 110, 0, 130, 20);

brush = CreateSolidBrush(

RGB(0, 0, 0)

);

SelectObject(hdc, brush);

Ellipse(hdc, 116, 6, 119, 9);

Ellipse(hdc, 121, 6, 124, 9);

if (gamestate == 0) {

MoveToEx(hdc, 116, 12, NULL);

LineTo(hdc, 124, 12);

}

if (gamestate == 1) {

MoveToEx(hdc, 116, 12, NULL);

AngleArc(hdc, 119, 8, 6, 240, 90);

}

if (gamestate == 2) {

MoveToEx(hdc, 124, 14, NULL);

AngleArc(hdc, 119, 18, 6, 60, 90);

}

wchar\_t buffer[256];

wsprintfW(buffer, L"F: %d", flagcount);

RECT textRect = { 150, 0, 190, 20 };

DrawText(hdc, buffer, -1, &textRect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

for (int i = 0; i < cCols; i++)

{

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

RECT buttonRect = { offset + i \* 20, offset + j \* 20, offset + i \* 20 + 20, offset + j \* 20 + 20 };

wchar\_t buffer[256];

if (minefield[i][j] == 1) {

brush = CreateSolidBrush(

RGB(0, 0, 0)

);

SelectObject(hdc, brush);

Ellipse(hdc, offset + i \* 20 + 5, offset + j \* 20 + 5, offset + i \* 20 + 20 - 5, offset + j \* 20 + 20 - 5);

}

else {

int cnt = 0;

if (i > 0 && j > 0 && minefield[i - 1][j - 1] == 1) { cnt++; }

if (i > 0 && minefield[i - 1][j] == 1) { cnt++; }

if (i > 0 && j < cRows - 1 && minefield[i - 1][j + 1] == 1) { cnt++; }

if (j > 0 && minefield[i][j - 1] == 1) { cnt++; }

if (j < cRows - 1 && minefield[i][j + 1] == 1) { cnt++; }

if (i < cCols - 1 && j > 0 && minefield[i + 1][j - 1] == 1) { cnt++; }

if (i < cCols - 1 && minefield[i + 1][j] == 1) { cnt++; }

if (i < cCols - 1 && j < cRows - 1 && minefield[i + 1][j + 1] == 1) { cnt++; }

wsprintfW(buffer, L"%d", cnt);

DrawText(hdc, buffer, -1, &buttonRect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

}

}

}

EndPaint(hwnd, &ps);

return 0;

}

case WM\_CREATE:

{

HHOOK hHook = SetWindowsHookEx(WH\_MOUSE\_LL, RBHookProc, NULL, 0);

auto buttons = new HWND[cCols][cRows];

for (int i = 0; i < cCols; i++)

{

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

buttonfield[i][j] = CreateWindow(

L"BUTTON", // Button class

L"", // Button text

WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON, // Styles

i \* 20 + offset, // x position

j \* 20 + offset, // y position

20, // Button width

20, // Button height

hwnd, // Parent window

NULL,

(HINSTANCE)GetWindowLongPtr(hwnd, GWLP\_HINSTANCE),

NULL);

}

}

return 0;

}

case WM\_COMMAND: {

if (!IsWindow((HWND)lParam)) { //событытие для уже открытой плитки

return 0;

}

if (wParam == WM\_RBUTTONDOWN) { //нажатие правой кнопкой мыши

TCHAR text[256];

GetWindowText((HWND)lParam, text, 256);

if (\_tcslen(text) > 0) {

SetWindowText((HWND)lParam, L"");

flagcount++;

RECT textRect = { 150, 0, 190, 20 };

RedrawWindow(hwnd, &textRect, NULL, RDW\_INVALIDATE | RDW\_INTERNALPAINT); //обновляется только часть экрана

}

else if (flagcount > 0) {

SetWindowText((HWND)lParam, L"F");

flagcount--;

RECT textRect = { 150, 0, 190, 20 };

RedrawWindow(hwnd, &textRect, NULL, RDW\_INVALIDATE | RDW\_INTERNALPAINT);

}

return 0;

}

TCHAR text[256];

GetWindowText((HWND)lParam, text, 256);

if (\_tcslen(text) > 0) { //кнопка с флагом

return 0;

}

RECT buttonRect;

RECT windowRect;

GetWindowRect((HWND)lParam, &buttonRect);

GetWindowRect((HWND)hwnd, &windowRect);

int buttonX = (buttonRect.left - windowRect.left - 28) / 20;

int buttonY = (buttonRect.top - windowRect.top - 51) / 20;

DestroyWindow((HWND)lParam);

int test = minefield[1][4];

if (minefield[buttonX][buttonY] == 1) { //проигрыш

gamestate = 2;

while (FindWindowEx(hwnd, NULL, NULL, NULL) != NULL)

{

auto hWndChild = FindWindowEx(hwnd, NULL, NULL, NULL);

DestroyWindow(hWndChild);

}

RedrawWindow(hwnd, NULL, NULL, RDW\_INVALIDATE | RDW\_INTERNALPAINT);

}

else {

cellsLeft--;

if (cellsLeft == 0) { //победа

gamestate = 1;

while (FindWindowEx(hwnd, NULL, NULL, NULL) != NULL)

{

auto hWndChild = FindWindowEx(hwnd, NULL, NULL, NULL);

DestroyWindow(hWndChild);

}

RedrawWindow(hwnd, NULL, NULL, RDW\_INVALIDATE | RDW\_INTERNALPAINT);

}

int cnt = 0; // соседние мины

int i = buttonX;

int j = buttonY;

int test = minefield[6][2];

if (i > 0 && j > 0 && minefield[i - 1][j - 1] == 1) { cnt++; }

if (i > 0 && minefield[i - 1][j] == 1) { cnt++; }

if (i > 0 && j < cRows - 1 && minefield[i - 1][j + 1] == 1) { cnt++; }

if (j > 0 && minefield[i][j - 1] == 1) { cnt++; }

if (j < cRows - 1 && minefield[i][j + 1] == 1) { cnt++; }

if (i < cCols - 1 && j > 0 && minefield[i + 1][j - 1] == 1) { cnt++; }

if (i < cCols - 1 && minefield[i + 1][j] == 1) { cnt++; }

if (i < cCols - 1 && j < cRows - 1 && minefield[i + 1][j + 1] == 1) { cnt++; }

if (cnt == 0) { //открывать соседние

if (i > 0 && j > 0) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i - 1][j - 1]); }

if (i > 0) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i - 1][j]); }

if (i > 0 && j < cRows - 1) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i - 1][j + 1]); }

if (j > 0) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i][j - 1]); }

if (j < cRows - 1) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i][j + 1]); }

if (i < cCols - 1 && j > 0) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i + 1][j - 1]); }

if (i < cCols - 1) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i + 1][j]); }

if (i < cCols - 1 && j < cRows - 1) { SendMessage(hwnd, uMsg, wParam, (LPARAM)buttonfield[i + 1][j + 1]); }

}

}

return 0;

}

}

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

}

LRESULT CALLBACK RBHookProc(int nCode, WPARAM wParam, LPARAM lParam) //winhook для передачи события нажатия ПКМ

{

if (wParam == WM\_RBUTTONDOWN)

{

RECT windowRect;

GetWindowRect(mainWindow, &windowRect);

POINT pt;

GetCursorPos(&pt);

int mouseX = (pt.x - windowRect.left - 28) / 20;

int mouseY = (pt.y - windowRect.top - 51) / 20;

SendMessage(mainWindow, WM\_COMMAND, wParam, (LPARAM)buttonfield[mouseX][mouseY]);

}

return CallNextHookEx(NULL, nCode, wParam, lParam);

}