Занятие 12

26.04.2021

* Задание на **одну** неделю.

**WinAPI: разработка приложений с графическим интерфейсом**

Обработка текстового файла

Обработать заданный текстовый файл:

1. Вычислить частоту встречаемости символов в строке;
2. Вычислить частоту встречаемости слов в строке.

***Требования к интерфейсу***

Окно приложения (стиль окна WS\_OVERLAPPEDWINDOW) разделено на три части:

* список, каждый элемент которого содержит строку файла; WM\_CREATE, CreateWindow(listbox)
* диаграмма А для выбранной строки файла;
* диаграмма В для выбранной строки файла.

Диаграммы должны содержать поясняющие подписи. Столбиковая горизонтальная диаграмма

Отображение масштабируется при изменении размеров окна.

Source.cpp

#ifndef UNICODE

#define UNICODE

#endif

#include <windows.h>

#include"resource.h"

#include<string>

#include<fstream>

#include<map>

#include<sstream>

#include<vector>

#include<algorithm>

#include <iterator>

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

//void SelectColor(HWND hwndDlg, COLORREF& color);

//BOOL CALLBACK Edit\_R(HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM lParam);

static HINSTANCE hInst;

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PWSTR pCmdLine, int nCmdShow) {

// Register the window class.

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

WNDCLASS wc = {};

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

wc.lpszMenuName = MAKEINTRESOURCE(IDR\_MENU1);

HWND hMainWindow = NULL;

RegisterClass(&wc);

// Create the window.

HWND hwnd =

CreateWindowEx(0, // Optional window styles.

CLASS\_NAME, // Window class

L"Learn to Program Windows", // Window text

WS\_OVERLAPPEDWINDOW, // Window style

// Size and position

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, // Parent window

NULL, // Menu

hInstance, // Instance handle

NULL // Additional application data

);

HWND hWndButton = CreateWindow(L"button", L"Choose", WS\_CHILD | WS\_VISIBLE, 50, 0, 60, 20, hwnd, (HMENU)1001, hInst, NULL);

CreateWindow(L"Edit", NULL, WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 0, 0, 40, 20, hwnd, (HMENU)1002, NULL, NULL);

//SetWindowText(hWndButton, "Click Me");

/\*if (hwnd == NULL)

{

return 0;

}\*/

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

// Run the message loop.

MSG msg = {};

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam,

LPARAM lParam) {

static HDC hdc;

static RECT rect;

static HBRUSH brush;

static HPEN pen, pen\_old;

static PAINTSTRUCT ps;

static size\_t number;

static TCHAR str\_number[2];

static HWND hListBox;

static std::map<char, int> symbols\_frequance;

static std::map<std::string, int> words\_frequance;

static std::vector<std::string> strings;

switch (uMsg) {

case WM\_CREATE: {

hListBox = CreateWindow(L"LISTBOX", NULL, WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 0, 20, 300, 300, hwnd, NULL, NULL, NULL);

pen = CreatePen(PS\_SOLID, 5, RGB(0, 0, 0));

brush = CreateSolidBrush(RGB(255, 255, 255));

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

case WM\_COMMAND: {

HMENU hMenu = GetMenu(hwnd);

switch (LOWORD(wParam)) {

case IDC\_BUTTON\_CHOOSE: {

GetDlgItemText(hwnd, IDC\_EDIT\_NUMBER, str\_number, sizeof(str\_number));

number=std::stoi(str\_number);

std::string str = strings[number-1];

std::stringstream in\_str(str);

std::string word;

while (getline(in\_str, word, ' ')) {

words\_frequance[word]++;

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

char symbol = word[i];

symbols\_frequance[symbol]++;

}

}

break;

}

case ID\_MENU\_FILEOPEN: {

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof(ofn));

TCHAR file\_name[300] = { '\0' };

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFile = file\_name;

ofn.nMaxFile = sizeof(file\_name);

ofn.lpstrFilter = L"All Files\0\*.\*\0\0";

ofn.nFilterIndex = 1;

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.Flags = OFN\_FILEMUSTEXIST | OFN\_PATHMUSTEXIST;

if (GetOpenFileName(&ofn)) {

std::ifstream in(file\_name);

std::string str;

while (!in.eof()) {

getline(in, str);

std::wstring temp = std::wstring(str.begin(), str.end());

SendMessage(hListBox, LB\_ADDSTRING, NULL, (LPARAM)(temp.c\_str()));

strings.push\_back(str);

}

}

InvalidateRect(hwnd, 0, 1);

break;

}

default:

break;

}

}

case WM\_PAINT:{

hdc = BeginPaint(hwnd, &ps);

GetClientRect(hwnd, &rect);

SelectObject(hdc, brush);

FillRect(hdc, &rect, brush);

MoveToEx(hdc, 0, 0, 0);

SetMapMode(hdc, MM\_ANISOTROPIC);

//words\_frequance

SetWindowExtEx(hdc, 3, words\_frequance.size(), NULL);

SetViewportExtEx(hdc, rect.right / 3, rect.bottom, NULL);

SetViewportOrgEx(hdc, rect.right / 3, 0, NULL);

pen = CreatePen(PS\_SOLID, 0, RGB(255, 0, 0));

pen\_old = (HPEN)SelectObject(hdc, pen);

MoveToEx(hdc, 0, 0, 0);

int v[5];

int i = 0;

for (auto& item : words\_frequance) {

v[i] = item.second;

i++;

}

for (size\_t i = 0; i < words\_frequance.size(); i++)

{

Rectangle(hdc, 0, i, v[i], i + 1);

}

//symbols\_frequance

//std::map<char,int>::iterator it=(std::max\_element(symbols\_frequance.begin(), symbols\_frequance.end(),

// [](const std::pair<char, int>& p1, const std::pair<char, int>& p2) {

// return p1.second < p2.second; }));

//int max\_value = it->second;

SetWindowExtEx(hdc, 10, symbols\_frequance.size(), NULL);

SetViewportExtEx(hdc, 2\*rect.right/3 , rect.bottom, NULL);

SetViewportOrgEx(hdc, 2\*rect.right/3, 0, NULL);

i = 0;

for (auto& item : symbols\_frequance) {

v[i] = item.second;

i++;

}

for (size\_t i = 0; i < symbols\_frequance.size(); i++)

{

Rectangle(hdc, 0, i, v[i], i + 1);

}

SelectObject(hdc, pen\_old);

DeleteObject(pen);

EndPaint(hwnd, &ps);

break;

}

case WM\_SIZE:

InvalidateRect(hwnd, NULL, TRUE);

break;

UpdateWindow(hwnd);

return 0;

}

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

}

Resource.h

#define IDR\_MENU1 101

#define ID\_MENU\_FILEOPEN 40001

#define IDC\_BUTTON\_CHOOSE 1001

#define IDC\_EDIT\_NUMBER 1002