#pragma once

//необходимо подключить, что бы была возможность работы с именами сообщений, а не с их числовым представлением

#include <windows.h>

#include <windows.h>

#define ID\_COUNT 1

#define ID\_EDIT1 2

#define ID\_EDIT2 3

#define ID\_EXIT 4

#define ID\_STATIC 5

#define ID\_STATIC1 6

#define ID\_STATIC2 7

typedef void EventFunction(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);

class SuperExtendedClass {

private:

 typedef void (SuperExtendedClass::\*EventFunctionSuper)(WPARAM wParam, LPARAM lParam);

//узел, описывающий событие формы

 struct WindowNode {

 UINT message;

 SuperExtendedClass::EventFunctionSuper func;

 };

 WindowNode \*windowNodes;

 int windowsNodesSize;

 HINSTANCE hInstance;

 //узел, описывающий элемент управления

 struct ListNode {

 int key;

 HWND handle;

 //EventFunction \*func;

 SuperExtendedClass::EventFunctionSuper func;

 };

 int nodesSize;

 ListNode\* nodes;

 int width;

 int height;

 HWND hWnd;

 void paintEvent(WPARAM wParam, LPARAM lParam) {

#define NUM 1000

#define TWOPI (2 \* 3.14159)

 PAINTSTRUCT ps;

 POINT pt[NUM];

 HDC hdc = BeginPaint(hWnd, &ps) ;

 HPEN pen = CreatePen(1,1,78);

 SelectObject(hdc, pen);

 /\*

 4 графика

 скорость работы

 1. добавление

 2. удаление

 3. поиск

 4. полная очистка

 f(x) - множество точек (конечное множетво)

 1. Выбраем шаг изменения X, X - количество доб. данных

 2. напрмер 1000

 int getTimeList(int numberOfData) {

 List \*temp = new List();

 НачалоЗамера

 for (int i=0;i<numberOfData;i++) {}

 temp->add(random(0, 1000000);

 }

 ОкончаниеЗАмера

 delete temp;

 вернуть разницу времени

 }

 PointList \*temp = new PointList();

 XMax = 100000;

 XCur = 0;

 while (XCur < XMax) {

 int x = xCur;

 int y = getTimeLisr(xCur);

 temp.Add(x, y);

 XCur += 1000;

 }

 PointList \*tempTwoList = new PointList();

 PointList \*tempTree = new PointList();

 PointList \*tempHash = new ...

 y(i+1) >= y(i)

 //(0,1) (1,1) (2,4) (3,8) (4,16) - множество точек

 temp.RemoveMoreY(YMax); //удалить все точки, у которых y >= YMax

 Нормировака по Y

 1. Находили MaxY

 2. Нормировали Y

 3. Инвертировали Y

 4. приводили к нужному масштабу

 Нормировака по X

 XMax1, xMax2, xMax3, xMax4 - у каждого графика структуры данных

 XSuperMax = Max(XMax1, xMax2, xMax3, xMax4)

 X [0; бесконечности)

 X [0; XSuperMax)

 X = X / XSuperMax

 Обл. определения - [0;бесконечности)

 1. вызнаете односвяз. список

 2. двух связ. список

 3. бинароное дерево

 4. хэш

 Start - тек. время

 for (int i=0;i<1000;i++) {

 XXXX

 }

 End - время окончания

 End - start = время выполнения == время выполнения XXXX \* 1000

 \*/

 MoveToEx(hdc, 0, cyClient / 2, NULL) ;

 LineTo(hdc, cxClient, cyClient / 2) ;

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

 pt[i].x = i \* cxClient / NUM ;

 pt[i].y = (int) (cyClient / 2 \* (1 - sin (TWOPI \* i / NUM))) ;

 }

 Polyline(hdc, pt, NUM) ;

 DeleteObject(pen);

 EndPaint(hWnd, &ps);

#undef TWOPI

#undef NUM

 }

 void calculateEvent(WPARAM wParam, LPARAM lParam) {

 //Получаем текст из первого поля ввода

 //получаем количество символов в первом поле ввода

 int lengthA = getTextLength(ID\_EDIT1);

 //выделяем память под них, не забывая о завершающем 0 (+ 1)

 char \*str1 = new char[lengthA + 1];

 //копируем текст из поля ввода в массив символов

 getText(ID\_EDIT1, str1, lengthA + 1);

 //по полной аналогии получаем текст из второго поля ввода

 int lengthB = getTextLength(ID\_EDIT2);

 char \*str2 = new char[lengthB + 1];

 getText(ID\_EDIT2, str2, lengthB + 1);

 //находим максимальную длинну числа

 int maxSize = (lengthA > lengthB) ? lengthA : lengthB;

 //сумма может быть на 1 символ больше максимального числа (напр 99 + 6)

 maxSize += 1;

 //мы будем выводить сообщение типа "A + B = Z ", где Z - ответ.

 //Кроме самого числа есть еще 9 символов - "А + В = ", учитываем из при нахождении размера

 maxSize += 9;

 //выделяем память под ответ

 char \*str3 = new char[maxSize + 1];

 //Преобразуем строки в целочисленные числа

 int a = atoi(str1);

 int b = atoi(str2);

 //находим сумму

 int c = a + b;

 //Формируем строку ответа, записывая в неё подсчитанные значения

 sprintf\_s(str3, maxSize, "A + B = %i ", c);

 //выводим текст в 3ю надпись

 setText(ID\_STATIC2, str3);

 delete str3;

 delete str2;

 delete str1;

 }

 int cxClient, cyClient;

 void sizeChangedEvent(WPARAM wParam, LPARAM lParam) {

 cxClient = LOWORD(lParam) ;

 cyClient = HIWORD(lParam) ;

 }

 void addEventToMainWindow(UINT message, EventFunctionSuper func) {

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

 if (windowNodes[i].message != message) {

 continue;

 }

 windowNodes[i].func = func;

 return;

 }

 WindowNode \*newNodes = new WindowNode[windowsNodesSize + 1];

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

 newNodes[i] = windowNodes[i];

 }

 newNodes[windowsNodesSize].message = message;

 newNodes[windowsNodesSize].func = func;

 windowsNodesSize++;

 if (windowNodes != NULL) {

 delete windowNodes;

 }

 windowNodes = newNodes;

 }

 void init(HINSTANCE \_hInstance) {

 nodesSize = 0;

 nodes = NULL;

 hInstance = \_hInstance;

 }

 void setText(int id, char\* newText) {

 SetDlgItemText(hWnd, id, newText);

 }

 void getText(int id, char\* buffer, int maxCount) {

 HWND h = getNode(id);

 GetWindowText(h, buffer, maxCount);

 }

 int getTextLength(int id) {

 HWND h = getNode(id);

 return GetWindowTextLength(h);

 }

 void setLimit(int id, int limit) {

 HWND h = getNode(id);

 //мы пока не знаем, что это - comboBox или TextBox, поэтому

 //посылаем сообщение как ComboBox

 int res = SendMessage(h, CB\_LIMITTEXT, limit, 0);

 //Если не получилось

 if (res == 0) {

 //то посылаем сообщение как TextBox

 res = SendMessage(h, EM\_LIMITTEXT, limit, 0);

 }

 }

 void createLabel(LPCSTR str, int left, int top, int width, int height, int id) {

 HWND handle = CreateWindow("static", str, WS\_CHILD|WS\_VISIBLE|SS\_LEFT|WS\_EX\_TRANSPARENT|WS\_BORDER, left, top, width, height, hWnd, (HMENU)id, hInstance, NULL);

 addNode(id, handle);

 }

 void createTextBox(int left, int top, int width, int height, int id) {

 HWND handle = CreateWindow("edit", NULL, WS\_CHILD|WS\_VISIBLE|ES\_LEFT|WS\_BORDER, left, top, width, height, hWnd, (HMENU)id, hInstance, NULL);

 addNode(id, handle);

 }

 void createButton(LPCSTR str, int left, int top, int width, int height, int id) {

 HWND handle = CreateWindow("button", str, WS\_CHILD|WS\_VISIBLE|BS\_DEFPUSHBUTTON, left, top, width, height, hWnd, (HMENU)id, hInstance, NULL);

 addNode(id, handle);

 }

 void addNode(int id, HWND handle) {

 ListNode \*newNodes = new ListNode[nodesSize + 1];

 memmove(newNodes, nodes, sizeof(ListNode) \* nodesSize);

 if (nodes != NULL) {

 delete nodes;

 }

 nodes = newNodes;

 nodes[nodesSize].key = id;

 nodes[nodesSize].handle = handle;

 nodes[nodesSize].func = NULL;

 nodesSize++;

 }

 HWND getNode(int id) {

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

 if (nodes[i].key != id) {

 continue;

 }

 return nodes[i].handle;

 }

 assert(false);

 return 0;

 }

 bool addEventToElement(int id, EventFunctionSuper func) {

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

 if (nodes[i].key != id) {

 continue;

 }

 nodes[i].func = func;

 return true;

 }

 return false;

 }

public:

 void createInterface() {

 //Текст-заголовок первого поля ввода

 createLabel("A: ", 10, 10, 20, 20, ID\_STATIC);

 //Поле ввода 1

 createTextBox(30, 10, 80, 20, ID\_EDIT1);

 //Текст 2

 createLabel("B: ", 10, 40, 20, 20, ID\_STATIC1);

 //Поле ввода 2

 createTextBox(30, 40, 80, 20, ID\_EDIT2);

 //Текст-ответ

 createLabel("A + B = ", 10, 70, 100, 20, ID\_STATIC2);

 //Кнопка "Считать"

 createButton("Считать", 10, 100, 100, 30, ID\_COUNT);

 //Кнопка "Выход"

 createButton("Выход", 10, 130, 100, 30, ID\_EXIT);

 //устанавливаем, что бы пользователь не мог ввести больше 3х символов в поля ввода

 setLimit(ID\_EDIT1, 3);

 setLimit(ID\_EDIT2, 3);

 //привязываем, что бы при нажатии на элемент управления с индексом ID\_COUNT срабатывал метод calculateEvent

 addEventToElement(ID\_COUNT, &SuperExtendedClass::calculateEvent);

 }

 HWND createMainWindow(LPCSTR szProgName, \_TCHAR \*title, int left, int top, int \_width, int \_height) {

 width = \_width;

 height = \_height;

 hWnd = CreateWindow(szProgName, title, WS\_DLGFRAME |WS\_BORDER | WS\_SYSMENU, left, top, width, height, (HWND)NULL, (HMENU)NULL, hInstance, NULL);

 return hWnd;

 }

 SuperExtendedClass(HINSTANCE hInst) {

 init(hInst);

 windowNodes = NULL;

 windowsNodesSize = 0;

 addEventToMainWindow(WM\_PAINT, &SuperExtendedClass::paintEvent);

 addEventToMainWindow(WM\_SIZE, &SuperExtendedClass::sizeChangedEvent);

 }

 //обработка сообшений для всего окна

 bool processWindowEvent(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam) {

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

 if (windowNodes[i].message != messg) {

 continue;

 }

 EventFunctionSuper s2 = windowNodes[i].func;

 SuperExtendedClass \*temp = this;

 (temp->\*s2)(wParam, lParam);

 return true;

 }

 return false;

 }

 //обработка сообщений для элементов окна

 bool processEvent(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam) {

 int localId;

 switch(messg) {

 case WM\_COMMAND:

 //Обработка сообщений от элементов управления

 localId = LOWORD(wParam);

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

 if (nodes[i].key != localId) {

 continue;

 }

 if (nodes[i].func == NULL) {

 return false;

 }

 EventFunctionSuper s2 = nodes[i].func;

 SuperExtendedClass \*temp = this;

 (temp->\*s2)(wParam, lParam);

 return true;

 }

 return false;

 default:

 return false;

 }

 }

 virtual ~SuperExtendedClass() {

 if (windowNodes != NULL) {

 delete windowNodes;

 }

 windowNodes = NULL;

 windowsNodesSize = 0;

 if (nodes != NULL) {

 delete nodes;

 }

 nodesSize = 0;

 nodes = NULL;

 }

};

// l1.cpp : Defines the entry point for the console application.

//

#include <string.h>

#include "stdafx.h"

/\*

2. Объем используемой памяти

1. Скорость работы

3. Удобство сопровождения

4. скорость разработки

Файл - последовательность байт

Пусть файл занимает 1000000000 (K) байт

от 0 до (K / N + 1)

fgetc() - 1 обращение к диску 100

//1 обращение за 1 символом

//1 обращение за N символами

ДОбавлен. в строку - 5 обращений к памяти 50

150 \* K

150 \* K / 1000 + 150

150 единиц при чтени файла целиком

\*/

void simpleAdd(char \*&str1, char \*addStr) {

 int len = strlen(str1);

 int len2 = strlen(addStr);

 char \*newStr = new char[len + len2 + 1];

 //while (\*str1++ = \*str2++);

 memmove(newStr, str1, sizeof(char) \* len);

 memmove(&newStr[len], addStr, sizeof(char) \* (len2 + 1));

 //strcat(newStr, addStr);

 delete str1;

 str1 = newStr;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

 FILE \*f = fopen("D:\\1.txt", "rt");

 if (f == NULL) {

 return 0;

 }

 char \*curStr = new char[1];

 curStr[0] = 0;

 const int N = 10;

 char buffer[N];

 while (true) {

 if (fgets(buffer, N - 1, f) == NULL) {

 break;

 }

 simpleAdd(curStr, buffer);

 }

 //fgets(

 fclose(f);

 delete curStr;

 return 0;

}