Министерство образования и науки РФ

Севастопольский государственный университет

Институт информационных технологий и управления в технических сетях

Лабораторная работа №7

**ОРГАНИЗАЦИЯ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ. БИБЛИОТЕКА MFC.**

Выполнил:

ст.гр.ИСб-22д

Воронин И.Ю.

Проверил:

Тимофеев И.С.

Севастополь

2015

1.Цель работы

Приобретение практических навыков в написании объектно-ориентированных программ с использованием MFC. Освоение особенностей отладки объектно-ориентированных программ с графическим интерфейсом.

2.Вариант задания

Вариант задания соответствует лабораторной работе №7. Для своего варианта необходимо реализовать графический интерфейс пользователя, используя возможности библиотеки классов MFC

***Вариант 10***

Написать программу учета книг в библиотеке. Сведения о книгах содержат: фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать выбор с помощью меню и выполнение следующих функций:

– добавление данных о книгах, вновь поступающих в библиотеку;

– удаление данных о списываемых книгах;

– выдача сведений о всех книгах, упорядоченных по фамилиям авторов;

– выдача сведений о всех книгах, упорядоченных по годам издания.

Хранение данных организовать с применением контейнерного класса vector.

3.Текст программы

Класс Book:

#include "stdafx.h"

#pragma once

typedef int us;

class Book

{

private:

CString author;

CString title;

us year;

us amount;

public:

Book();

Book(CString \_title, CString \_author, us \_year, us \_amount);

void Set\_all(CString \_title, CString \_author, us \_year, us \_amount);

CString getAuthor();

CString getTitle();

us getYear();

us getAmount();

~Book();

};

#include "stdafx.h"

#pragma once

Book::Book(){

author = "";

title = "";

year = 0;

amount = 0;

}

void Book::Set\_all(CString \_title, CString \_author, us \_year, us \_amount){

author = \_author;

title = \_title;

year = \_year;

amount = \_amount;

}

Book::Book(CString \_title, CString \_author, us \_year, us \_amount){

author = \_author;

title = \_title;

year = \_year;

amount = \_amount;

}

CString Book::getAuthor(){

return author;

}

CString Book::getTitle(){

return title;

}

us Book::getYear(){

return year;

}

us Book::getAmount(){

return amount;

}

Book::~Book(){

}

Класс Librarian:

#pragma once

#include "stdafx.h"

class Librarian{

public:

Librarian();

void Swap(Book &a, Book &b){

Book temp;

temp = a;

a = b;

b = temp;

}

void DeleteBook(std::vector <Book>&);

void AuthorSort(std::vector <Book>&);

void YearSort(std::vector <Book>&);

~Librarian();

};

#include "stdafx.h"

#include "Librarian.h"

Librarian::Librarian(){

}

void Librarian::AuthorSort(std::vector <Book> &book){

for (int i = 0; i < book.size() - 1; i++){

for (int j = i + 1; j < book.size(); j++){

if (book[i].getAuthor() > book[j].getAuthor()){

Swap(book[i], book[j]);

}

}

}

}

void Librarian::YearSort(std::vector <Book> &book){

for (int i = 0; i < book.size() - 1; i++){

for (int j = i + 1; j < book.size(); j++){

if (book[i].getYear() > book[j].getYear()){

Swap(book[i], book[j]);

}

}

}

}

void Librarian::DeleteBook(std::vector <Book> &book){

CString del\_flag;

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

if (book[i].getTitle() == del\_flag) {

book.erase(book.begin() + i);

}

}

}

Librarian::~Librarian(){

}

Класс ListCtrl

#pragma once

#ifndef \_\_AFXWIN\_H\_\_

#error "включить stdafx.h до включения этого файла в PCH"

#endif

#include "resource.h" // основные символы

class CBookApp : public CWinApp

{

public:

CBookApp();

// Переопределение

public:

virtual BOOL InitInstance();

// Реализация

DECLARE\_MESSAGE\_MAP()

};

extern CBookApp theApp;

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CBookApp

BEGIN\_MESSAGE\_MAP(CBookApp, CWinApp)

ON\_COMMAND(ID\_HELP, &CWinApp::OnHelp)

END\_MESSAGE\_MAP()

// создание CBookApp

CBookApp::CBookApp()

{

// поддержка диспетчера перезагрузки

m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_RESTART;

// TODO: добавьте код создания,

// Размещает весь важный код инициализации в InitInstance

}

// Единственный объект CBookApp

CBookApp theApp;

// инициализация CBookApp

BOOL CBookApp::InitInstance(){

INITCOMMONCONTROLSEX InitCtrls;

InitCtrls.dwSize = sizeof(InitCtrls);

// Выберите этот параметр для включения всех общих классов управления, которые необходимо использовать

// в вашем приложении.

InitCtrls.dwICC = ICC\_WIN95\_CLASSES;

InitCommonControlsEx(&InitCtrls);

CWinApp::InitInstance();

AfxEnableControlContainer();

// Создать диспетчер оболочки, в случае, если диалоговое окно содержит

// представление дерева оболочки или какие-либо его элементы управления.

CShellManager \*pShellManager = new CShellManager;

// Активация визуального диспетчера "Классический Windows" для включения элементов управления MFC

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));

SetRegistryKey(\_T("Локальные приложения, созданные с помощью мастера приложений"));

CMainDlg dlg;

m\_pMainWnd = &dlg;

INT\_PTR nResponse = dlg.DoModal();

if (nResponse == IDOK){

}

else if (nResponse == IDCANCEL){}

else if (nResponse == -1)

{

TRACE(traceAppMsg, 0, "Предупреждение. Не удалось создать диалоговое окно, поэтому работа приложения неожиданно завершена.\n");

TRACE(traceAppMsg, 0, "Предупреждение. При использовании элементов управления MFC для диалогового окна невозможно #define \_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS.\n");

}

// Удалить диспетчер оболочки, созданный выше.

if (pShellManager != NULL)

{

delete pShellManager;

}

return FALSE;

}

Класс CMainDlg

class CMainDlg : public CDialogEx

{

// Создание

public:

CMainDlg(CWnd\* pParent = NULL); // стандартный конструктор

// Данные диалогового окна

enum { IDD = IDD\_LISTCTRL\_DIALOG };

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

HICON m\_hIcon;

// Созданные функции схемы сообщений

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

};

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// Данные диалогового окна

enum { IDD = IDD\_ABOUTBOX };

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD){

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX){

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

CMainDlg::CMainDlg(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(CMainDlg::IDD, pParent){

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CMainDlg::DoDataExchange(CDataExchange\* pDX){

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CMainDlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

END\_MESSAGE\_MAP()

BOOL CMainDlg::OnInitDialog(){

CDialogEx::OnInitDialog();

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL){

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty()){

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Задает значок для этого диалогового окна. Среда делает это автоматически,

// если главное окно приложения не является диалоговым

SetIcon(m\_hIcon, TRUE); // Крупный значок

SetIcon(m\_hIcon, FALSE); // Мелкий значок

// TODO: добавьте дополнительную инициализацию

return TRUE; // возврат значения TRUE, если фокус не передан элементу управления

}

void CMainDlg::OnSysCommand(UINT nID, LPARAM lParam){

if ((nID & 0xFFF0) == IDM\_ABOUTBOX){

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else{

CDialogEx::OnSysCommand(nID, lParam);

}

}

void CMainDlg::OnPaint(){

if (IsIconic()){

CPaintDC dc(this); // контекст устройства для рисования

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Выравнивание значка по центру клиентского прямоугольника

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Нарисуйте значок

dc.DrawIcon(x, y, m\_hIcon);

}

else{

CDialogEx::OnPaint();}

}

HCURSOR CMainDlg::OnQueryDragIcon(){

return static\_cast<HCURSOR>(m\_hIcon);

}

4.Выполнения программы

Данная программа способна добавлять, изменять и удалять записи о книгах, которые содержат в себе 2 текстовых и два целочисленных поля.

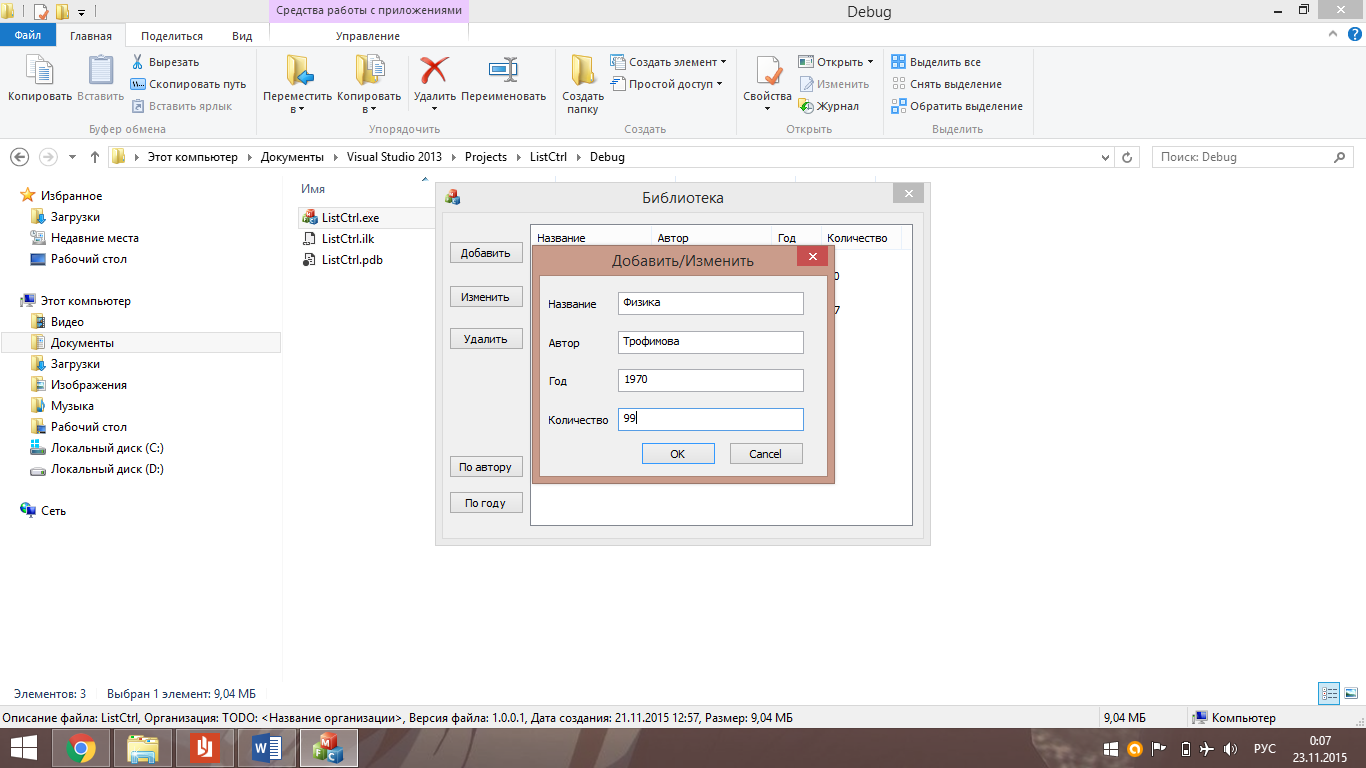


Рисунок 4.1 – Добавление элемента.

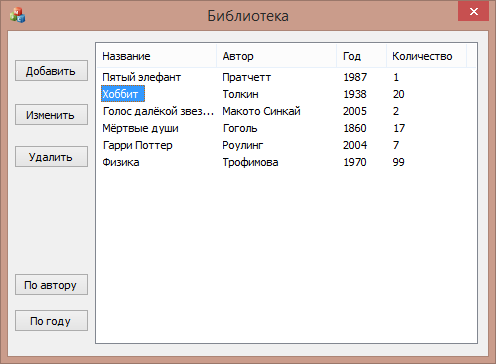


Рисунок 4.2 – Просмотр и выделение элемента.

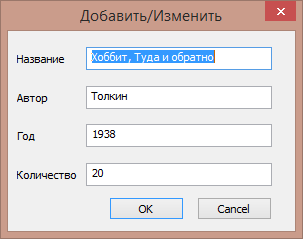
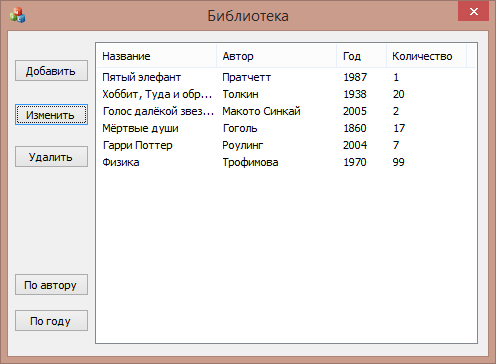


Рисунок 4.3 – Изменение элемента Хоббит.

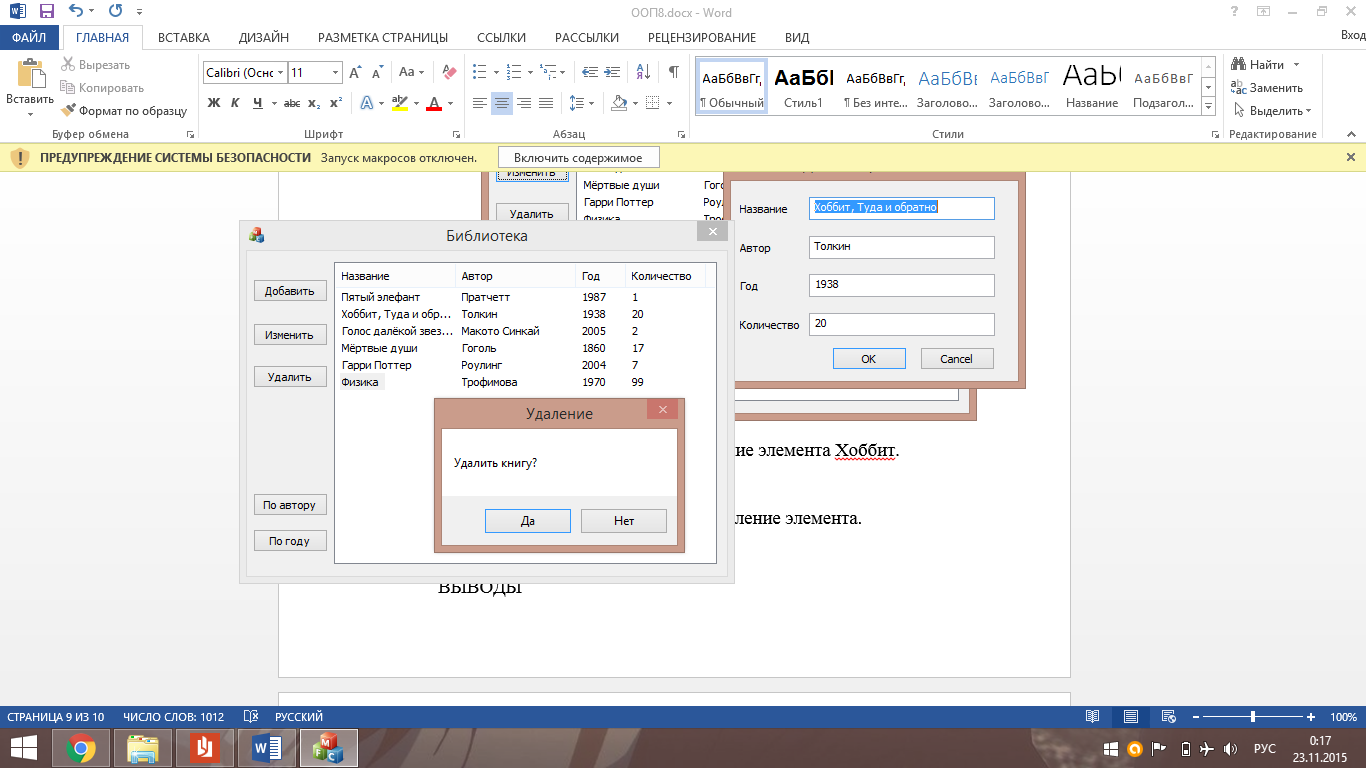


Рисунок 4.4 – Удаление элемента.

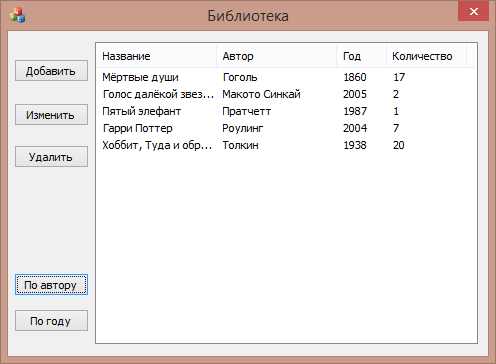


Рисунок 4.5 – Сортировка по автору.

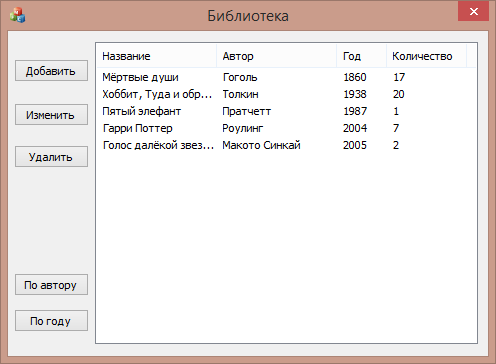


Рисунок 4.5 – Сортировка по годам.

ВЫВОДЫ

В данной лабораторной работе были изучены понятия классов и структуры библиотеки MFC. Была написана программа, которая выполняет функции библиотеки по работе к записями о книгах.