***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №1**

**По дисциплине ППвИС за IV семестр**

**Тема: «Основы создания Windows-приложений в системе**

**Microsoft Visual Studio C++. Процедурный стиль»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Муравьев Г.Л.

Брест 2019

Цель работы: изучение особенностей создания оконных приложений в системе Microsoft Visual Studio C++.

Содержание отчета:

1. Описание каркасов Windows приложений (Empty, Simple или аналогичных)
   1. Характеристика интерфейса, предоставляемых возможностей
   2. Файловый состав (структура проекта, дерево папок, состав, назначение файлов и их соподчиненность по включению)
   3. Функциональный состав (схема иерархии функции приложения, назначение и прототипы функций)
2. Описание ТКП: алгоритм работы (приложение 1), диаграмма прецедентов (приложение 2), диаграмма состояний (приложение 3), компонентов (рис. 7.1).
3. Выводы:
   1. О последовательности инициализации ТКП
   2. О причинах и условиях посылки сообщений ТКП, рассмотренных в работе
   3. Перечень ситуаций «Перерисовки»
   4. Выводы по способу реализации завершения работы приложения

Ход работы

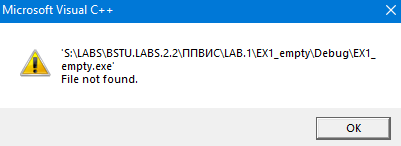
1. Изучим теоретический материал об особенностях оконных приложений.
2. Создадим каркас (с именем EX1\_empty) Windows-приложения (тип Empty) средствами мастера Win32 Application (рис. 2.1).

Рисунок 2.1



Насколько видим, пустой каркас не содержит в себе файлов. А поэтому и запуск программы не может быть осуществлен (рис. 2.2).

Рисунок 2.2



1. Создадим файл. Назовем его произвольным образом (рис. 3.1).

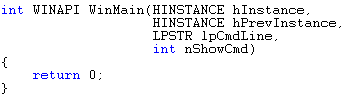
Рисунок 3.1



Включим в файл <windows.h>.



Создадим пустую функцию WinMain.



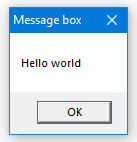
Выполним приложение.



Включим в приложение вывод окна сообщения (рис 3.2).



Рисунок 3.2 – message box



Посмотрим также на работу библиотеки классов MFC. Для этого также вызовем окно сообщения, но используя уже функцию AfxMessageBox из <afxwin.h>

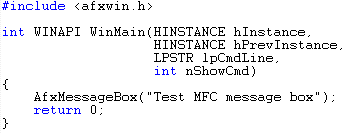
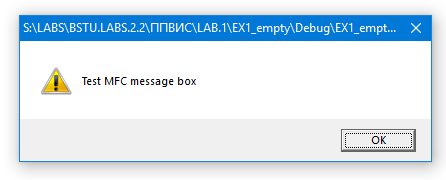
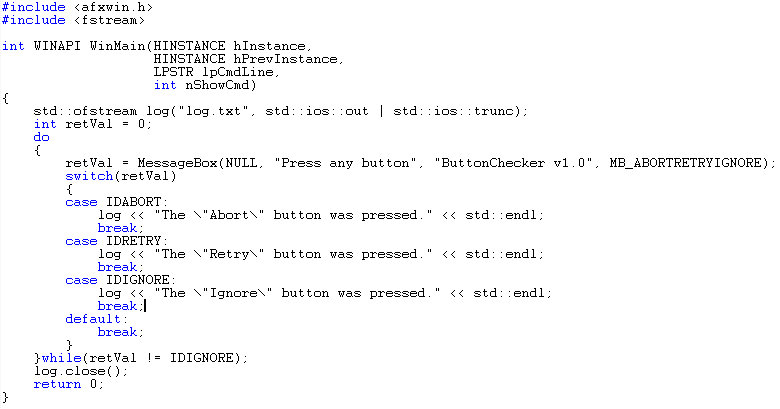


Рисунок 3.3 – MFC message box



Проанализируем возвращаемый код для MessageBox.



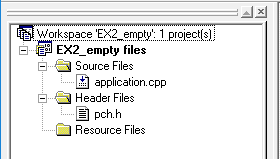
Результат работы из файла “log.txt”:



1. Создадим проект с именем EX2\_empty из двух файлов – один, главный, с текстом программы, а второй – заголовочный с командами препроцессора (рис. 4.1)

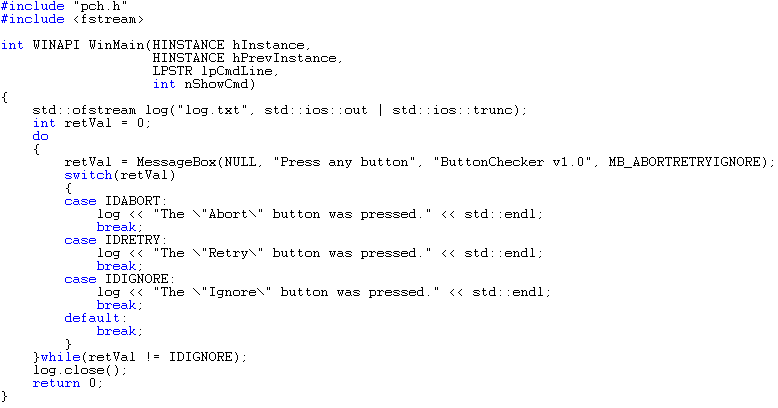
.

Рисунок 4.1

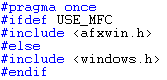


Повторим действия из п. 3, добавив при этом в файл pch.h команды препроцессора для подключения нужных заголовков и подключив его в файл application.cpp:

* application.cpp



* pch.h

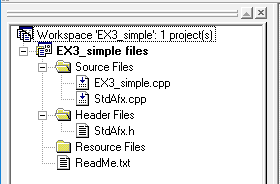


Результат выполнения при этом остается неизменным:



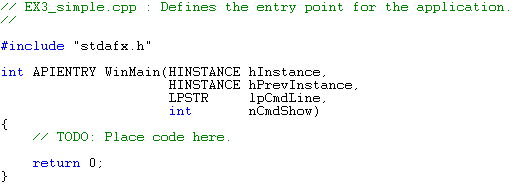
1. Создадим каркас (с именем EX3\_simple) Windows-приложения (тип Simple) средствами мастера Win32 Application (рис. 4.1).

Рисунок 4.1 – Win32 Application

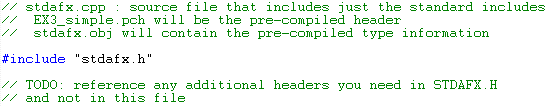


Как можно наблюдать, мастером было созданы файлы:

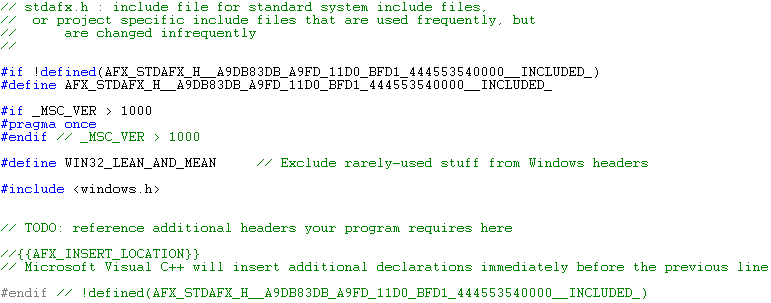
* EX3\_simple.cpp



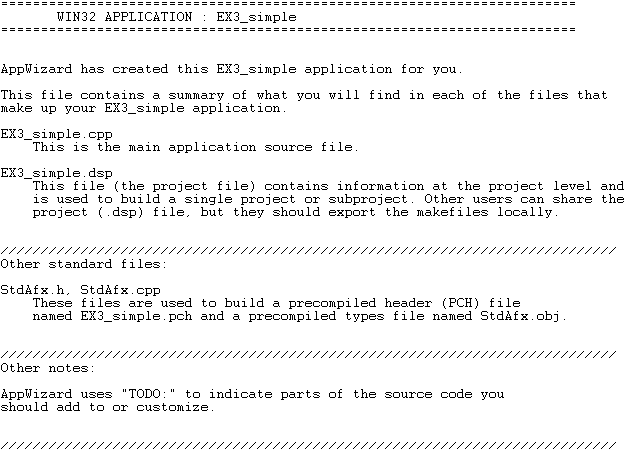
* StdAfx.cpp



* StdAfx.h



* ReadMe.txt

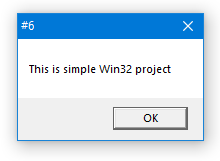


1. Модифицируем полученный в прошлом пункте каркас путем вставки в WinMain функции MessageBox.



Получаем ожидаемый результат (рис 6.1).

Рисунок 6.1



**Работа с типовым каркасом приложения**

7. Изучим теоретический материал о сообщениях и типовом каркасе приложения.

Создадим приложение (с именем EX4\_simple) и вставим в основной файл проекта текст ТКП.

#include "stdafx.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);

char szWindowStyle [ ] = "myWindowStyle";

//===============================

// ГЛАВНАЯ ФУНКЦИЯ

int WINAPI WinMain (HINSTANCE hInst, //дескриптор приложения

HINSTANCE hPreInst, //всегда NULL

LPSTR lpszCmdLine, //командная строка

int nCmdShow) //окно при первом выводе

{

HWND hWnd; //дескриптор окна

MSG lpMsg; //структура хранения сообщений

WNDCLASS wcApp; //структура описания стиля окна

wcApp.lpszClassName = szWindowStyle; //имя стиля окна

wcApp.hInstance = hInst; //дескриптор приложения

wcApp.lpfnWndProc = WndProc; //указатель на обработчик сообщений

wcApp.hCursor = LoadCursor(NULL, IDC\_ARROW); //курсор - “стрелка”

wcApp.hIcon = 0; //без использования пиктограммы

wcApp.lpszMenuName = 0; //дескриптор меню (без меню)

wcApp.hbrBackground = (HBRUSH) GetStockObject (WHITE\_BRUSH); //цвет фона

wcApp.style = CS\_HREDRAW | CS\_VREDRAW; //перерисовывать окно

wcApp.cbClsExtra = 0; //число доп. байт для WNDCLASS

wcApp.cbWndExtra = 0; //общее число доп. байт

if ( ! RegisterClass (&wcApp) ) //регистрация окна

return 0;

hWnd = CreateWindow (szWindowStyle,

"ТИПОВОЙ КАРКАС Windows-приложения … ",

WS\_OVERLAPPEDWINDOW, //окно перекрывающееся, меню, кнопки

CW\_USEDEFAULT, //координата х - левый верхний угол окна

CW\_USEDEFAULT, //координата у - левый верхний угол окна

CW\_USEDEFAULT, //ширина окна в единицах устройства

CW\_USEDEFAULT, //высота окна в единицах устройства

(HWND) NULL, //указатель на родительское окно

(HMENU) NULL, //зависит от стиля окна (указатель меню)

hInst, //дескриптор приложения

NULL ); //адрес дополнительной информации об окне

ShowWindow (hWnd, nCmdShow); //вывод окна

UpdateWindow (hWnd); //перерисовка окна

while (GetMessage ( &lpMsg, NULL, 0, 0) )

{

TranslateMessage (&lpMsg); //преобразование виртуальных клавиш

DispatchMessage (&lpMsg); //передача сообщения обработчику

}

return ( lpMsg.wParam );

}

//================================

// ФУНКЦИЯ-ОБРАБОТЧИК ГЛАВНОГО ОКНА. (имя выбирает пользователь)

LRESULT CALLBACK WndProc (HWND hWnd, //дескриптор окна

UINT messg, //код сообщения

WPARAM wParam, LPARAM lParam)

{

HDC hdc; //дескриптор контекста устройства

PAINTSTRUCT ps; //структура для клиентской области окна

switch (messg)

{

case WM\_PAINT: //перерисовать окно

hdc = BeginPaint (hWnd, &ps);

**//-----Начало фрагмента пользователя**

**//-----Конец фрагмента пользователя**

ValidateRect (hWnd,NULL);

EndPaint (hWnd, &ps);

break;

case WM\_DESTROY: //послать сообщение о завершении приложения

PostQuitMessage (0);

break;

default:

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

}

return 0;

}

ТКП, как видим из листинга, состоит из 2-х функций: WinMain (основная функция) и WndProc (обработчик сообщений). В WinMain описывается и регистрируется класс окна, создается и выводится главное окно на основе этого класса и выполняется передача сообщений обработчику. WndProc принимает и обрабатывает полученное сообщение.



Рисунок 7.1 – Структура ТКП

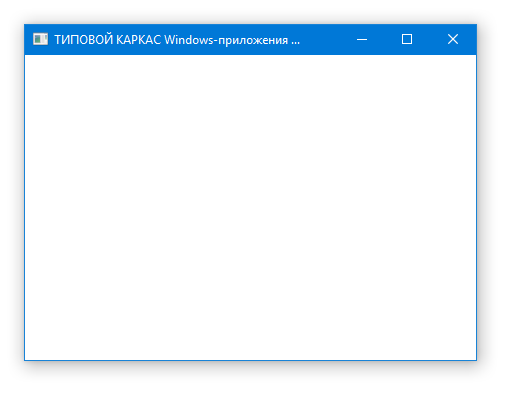


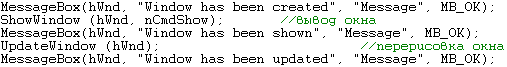
Рисунок 7.2 – Выполнение ТКП

7.2. Модифицировать приложение, последовательно размещая функцию MessageBox с сообщением об выполненной функции:

а) после создания окна



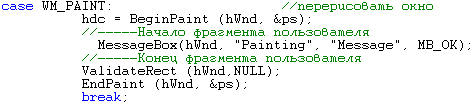
б) после вывода и после обновления окна



в) сразу в секции case WM\_DESTROY

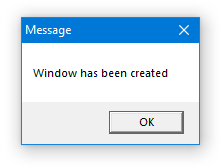


г) в секции case WM\_PAINT в место для вставки фрагмента пользователся.

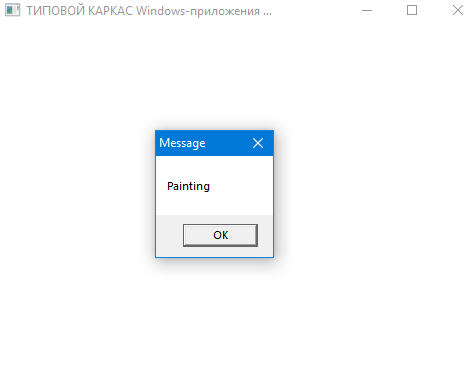


Посмотрим на результат выполнения (по порядку):

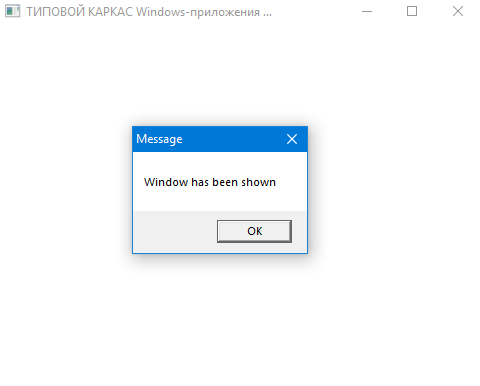
* После CreateWindow



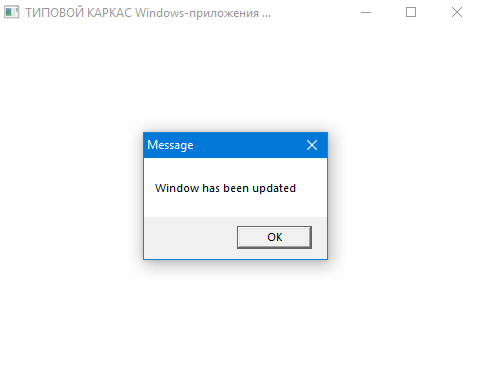
* В case WM\_PAINT



* После ShowWindow

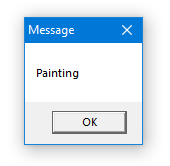


* После UpdateWindow

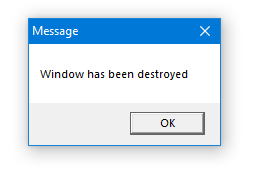


* Когда мы пытаемся изменить размер окна или максимизировать его.

Притом это окно сообщения можно передвигать только в случае если оно было вызвано максимизацией главного окна (возможно



* В case WM\_DESTROY.

Сообщение отправляется в WinProc после того, как окно было удалено с экрана (Windows Dev Center). Поэтому на момент появления этого сообщения главное окно исчезло.

Закомментируем блок обработки сообщения WM\_DESTROY.



Запустим прилажение и закроем окно. Просмотривая процессы можно видеть, что процес до сих пор активен:



8. Повторим создание приложения для закрепления.

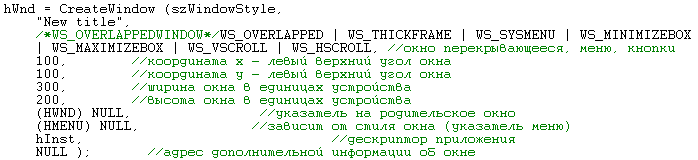
9. Изучим теоретический материал об описании и создании окна приложения. Создадим приложение с именем EX5\_simple, модифицировав приложение на базе ТКП. Заменим стиль WS\_OVERLAPPEDWINDOW на поэлементные стили через оператор ИЛИ.



Внесем изменения в структуру WNDCLASS.



Внесем изменения в функцию CreateWindow.



Внесем изменения в функцию ShowWindow.

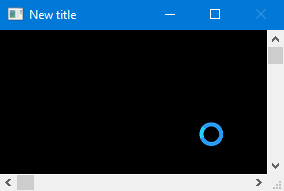


Выполнив все эти преобразования и запустив приложение, получаем:

* В изначальном состоянии свернутое окно

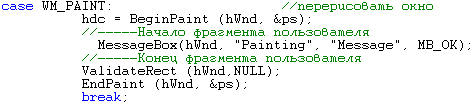
C:\Users\Ilya\AppData\Local\Microsoft\Windows\INetCache\Content.Word\window_minimized.png

* В развернутом виде



1. Добавим в обработчик секцию обработки сообщения WM\_PAINT как вывод соответствующего MessageBox. Проанализировать работу секции.

Воспользуемся кодом из п. 7.2.



По результату проведения анализа работы секции WM\_PAINT установлено, что сообщение о перерисовке отсылается при: развертывании окна, максимизации окна, выхода из режима максимизации окна и попытке изменения размеров окна.

1. Обеспечим попеременное переключение окна – «спрятано», «показано», «свернуто», «развернуто».

Вывод:

а) Последовательность инициализации ТКП. Анализируя код ТКП, имеем: Сначала инициализируется класс окна. Она содержит в себе описание стиля окна, имя класса, дескриптор курсора, дескриптор иконки, дескриптор меню, кисть, а также дескриптор приложения, к которому принадлежит данное окно и указатель на обработчик сообщений данного окна. Данная структура (WNDCLASS) регистрируется. Поле этого происходит инициализация окна через функцию CreateWindow, его последующее отображение и обновление: ShowWindow и UpdateWindow. Далее идет инициализация структуры MSG через функцию GetMessage, которая считывает очередное сообщение. Инициализированная переменна данной структуры обрабатывается функцией TranslateMessage и пересылается в соответствующий обработчик сообщений.

б) Причины и условия посылки сообщений ТКП. Сообщения в Windows-приложения поступают через операционную систему в очередь сообщений. После этого функцией GetMessage выбирается самое первое сообщение в очереди. Сообщения также могут быть отправлены вручную через функцию SendMessage. Сообщения в Windows существуют для общения между различными сущностями (окна, люди).

в) Перерисовка в Windows происходит при:

* Перемещении окна
* Максимизации окна
* Выводе окна из режима минимизации
* Изменении размеров окна
* Вызове функции ShowWindow
* Вызове функции RedrawWindow

г) Завершение работы может быть осуществлено с помощью функции PostQuitMessage, которая отсылает сообщение WM\_QUIT в очередь сообщений. Данная функция может быть помещена в обработчик сообщения WM\_DESTROY главного окна, которое отсылается функцией DestroyWindow. Можно организовать уточнение намерение закрытия окна, и только после этого вызывать функцию DestoryWindow. При попытке закрыть окно, обычно, высылается сообщение WM\_CLOSE.

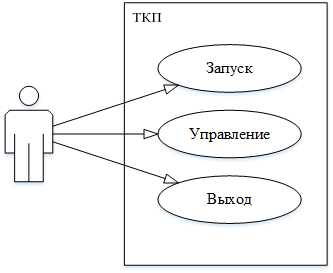
Приложение 1

Алгоритм выполнения ТКП



Приложение 1

Диаграмма прецедентов



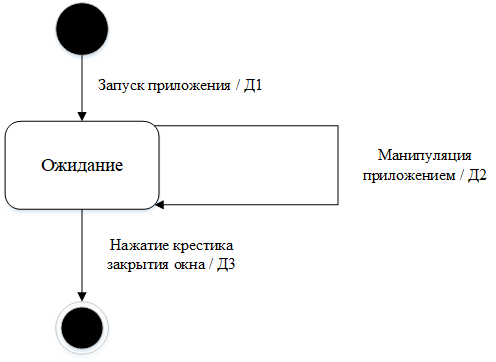
Запуск: создается система окон – настраивается система обработки сообщений – визуализируется окна – обновляется окна – окно активно, приложение ждет.

Управление: перемещение, свертывание, максимизация и др.

Выход: удаление окон, высвобождение ресурсов.

Приложение 2

Диаграмма состояний



Д1: создается система окон – настраивается система обработки сообщений – визуализируется окна – обновляется окна.

Д2: перерисовка окна (если требуется)

Д3: обработка сообщения WM\_CLOSE – закрытие окна – высвобождение ресурсов