***Министерство образования Республики Беларусь***

***Учреждение образования***

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

***Кафедра ИИТ***

**Лабораторная работа №1**

**По дисциплине МРЗвИС за IV семестр**

**Тема: «Многопоточность»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Анфилец С.В.

Брест 2019

Цель: написать программу, в которой «одновременно» выполняются два потока. Процесс выполнения каждого потока должен отражаться при помощи индикатора.

Ход работы.

1. Создадим проект MFC (рис. 1.1, 1.2, 1.3).

Рисунок 1.1 – Создание MFC приложения

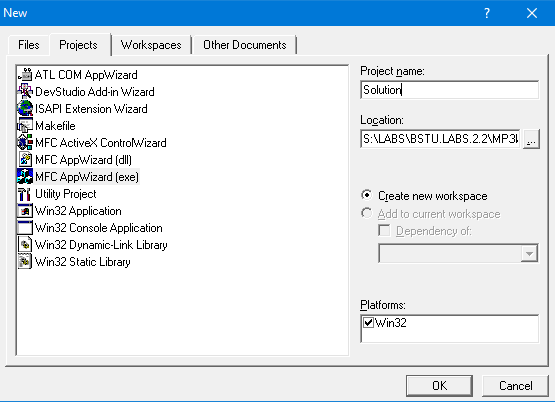


Рисунок 1.2 – Древо MFC проекта

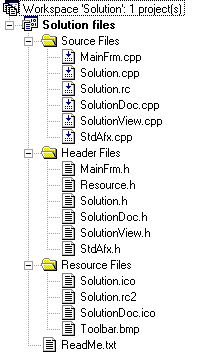


Рисунок 1.3 – Ресурсы MFC проекта

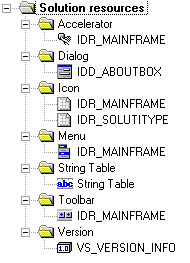


Рисунок 1.4 – Классы MFC проекта



1. Создадим ресурс диалогового окна. Выполним эту операцию через Insert->Resource. (рис. 2.1, 2.2).

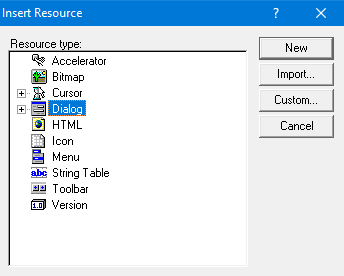
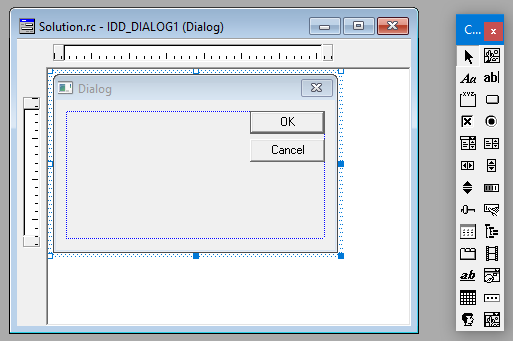


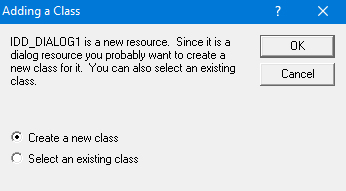
Рисунок 2.1 – создание диалогового окна

Рисунок 2.2 – редактор диалогового окна



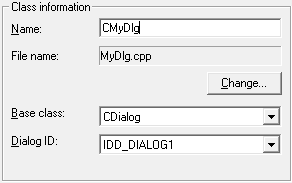
1. Создадим класс диалогового окна. Для этого щелкнем на диалоговом окне правой кнопкой мыши и в появившемся контекстном меню выберем пункт ClassWizard. Приложение сообщает нам, что не создано класса и предлагает нам его создать (рис. 3.1).

Рисунок 3.1 – Подтверждение создания класса



Создадим его, выбрав базовым классом CDialog. Имя же ему дадим CMyDlg (рис. 3.2).

Рисунок 3.2 – Заполнение информации о классе



**Программирование класса диалогового окна.**

1. Зададим максимальные значения для потоков.

enum {nMax1 = 1000000, nMax2=500000};

Создадим переменную для работы с таймером.

private:

int m\_nTimer;

Создадим глобальные переменные, с которыми будут работать функции.

int m\_nCount1 = 0;

int m\_nCount2 = 0;

Создадим сами функции:

UINT FirstThread(LPVOID pParam)

{

volatile int nTmp;

for(m\_nCount1 = 0; m\_nCount1 < CMyDlg::nMax1;

::InterlockedIncrement((LPLONG)&m\_nCount1))

{

for(nTmp = 0; nTmp <= 1000; nTmp++) {}

}

m\_nCount1 = 0;

return 0;

}

UINT SecondThread(LPVOID pParam)

{

volatile int nTmp;

for(m\_nCount2 = 0; m\_nCount1 < CMyDlg::nMax2;

::InterlockedIncrement((LPLONG)&m\_nCount2))

{

for(nTmp = 0; nTmp <= 1000; nTmp++) {}

}

m\_nCount2 = 0;

return 0;

}

Тут мы использовали служебное слово volatile для того, чтобы компилятор не хранил наш счетчик в регистре. Функция ::InterlockedIncrement запрещает доступ к переменной до во время её увеличения. В конце выполнения мы возвращаем наши счетчики в исходное положение.

Добавим обработчики для кнопок диалогового окна. Для этого воспользуемся выберем в контекстном меню “Events…” (рис. 4.1)

void CMyDlg::OnOK()

{

m\_nTimer = SetTimer(1, 100, NULL);

ASSERT (m\_nTimer!=0);

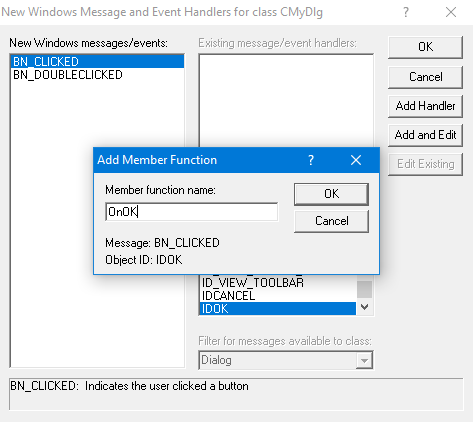
GetDlgItem(IDOK)->EnableWindow(FALSE);

AfxBeginThread(FirstThread, GetSafeHwnd(), THREAD\_PRIORITY\_NORMAL);

AfxBeginThread(SecondThread, GetSafeHwnd(), THREAD\_PRIORITY\_NORMAL);

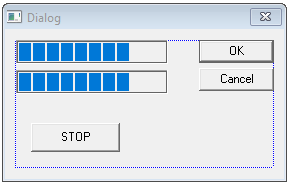
}

Рисунок 4.1 – Создание обработчика события OnOK



Добавим пару дополнительных элементов в наше диалоговое окно, а именно: 2 линии прогресса и кнопку STOP (рис. 4.2).

Рисунок 4.2 – Добавлены дополнительные элементы



Добавим обработчик к кнопке STOP. Он будет или выключать наше приложение, если оно еще не начало своё выполнение, или завершать выполнение процессов.

void CMyDlg::OnStop()

{

if(m\_nCount1 == 0 && m\_nCount2 == 0)

{

CDialog::OnCancel();

}

else

{

m\_nCount1 = nMax1;

m\_nCount2 = nMax2;

GetDlgItem(IDOK)->EnableWindow(TRUE);

}

}

Для отображения текущего состояния каждого потока будем использовать добавленные стоки прогресса (см. рис. 4.2). Обрабатывать будем с помощью созданного ранее таймера, а вернее его обработчика.

void CMyDlg::OnTimer(UINT nIDEvent)

{

CProgressCtrl\* pBar1 = (CProgressCtrl\*) GetDlgItem(IDC\_PROGRESS1);

CProgressCtrl\* pBar2 = (CProgressCtrl\*) GetDlgItem(IDC\_PROGRESS2);

pBar1->SetPos(m\_nCount1\*100/nMax1);

pBar2->SetPos(m\_nCount2\*100/nMax2);

if(m\_nCount1 == 0 && m\_nCount2 == 0)

{

GetDlgItem(IDOK)->EnableWindow(TRUE);

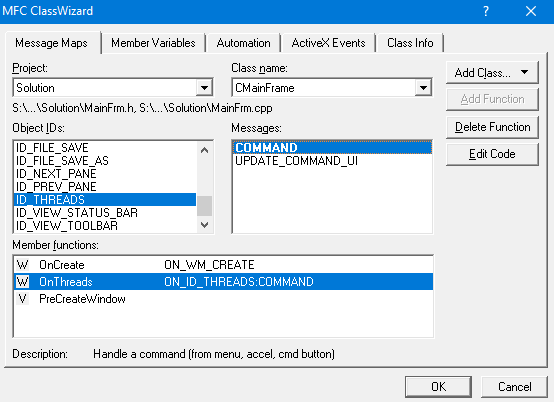
}

CDialog::OnTimer(nIDEvent);

}

1. Добавим пункт меню для наблюдения наших потоков (рис 5.1)

Рисунок 5.1 – Добавлена функция для обработки пункта меню



Отредактируем функцию-обработчик, чтобы она создавала наше диалоговое окно.

void CMainFrame::OnThreads()

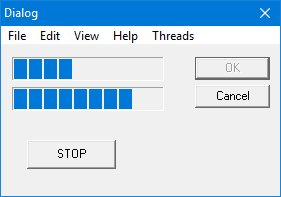
{

CMyDlg d;

d.DoModal();

}

**Тестирование работы приложения**. Результаты выполнения работы на рис. 5.2.



Все клавиши работают корректно.

Вывод: В данной лабораторной работе были изучены основы многопоточности в MFC приложении.