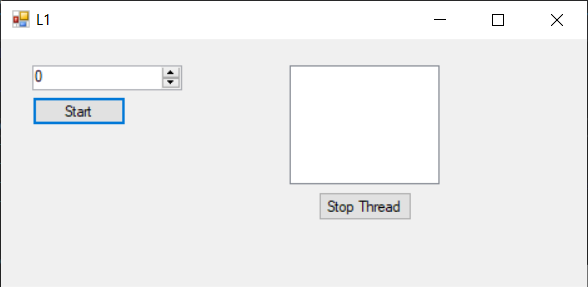
Отчет по ЛР 1  
Пупынина М.А   
Группа АС-17-04

Задача: реализовать возможность открытия, закрытия потоков через диалоговое окно(C#), связанное с консольным приложением(С++) с помощью событий.

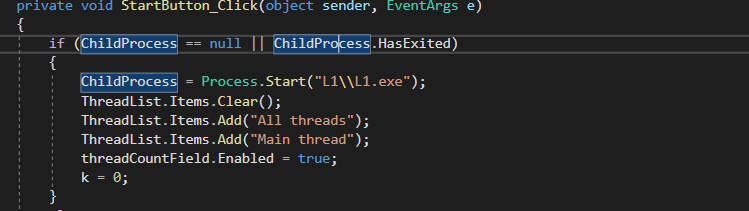
Реализация:

1. Создание диалогового приложения

Создадим диалоговое окно для управления консольным приложением:

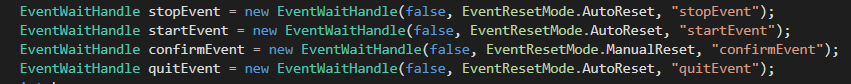


Для того, чтобы консольное приложение запускалось при работе диалогового, а также при его закрытии диалоговое возобновляло его работу, пропишем его открытие на нажатие кнопки Start:



Принцип работы:

С помощью функции CreateEvent () создаются необходимые нам события (открытие потока, закрытие потока, закрытие консольного приложения, подтверждение):



При нажатии кнопки Start запускается консольное приложение. В случае, если оно уже запущено, мы открываем столько потоков, сколько было указано в диалоговом приложении.

Для каждого потока по очереди триггерим событие startEvent, ждем, пока нам вернется ConfirmEvent, отображаем создание потока в диалоговом окне.

Логика закрытия потока та же:

stopEvent.Set();

Ждем confirmEvent;

Отображаем закрытие потока.

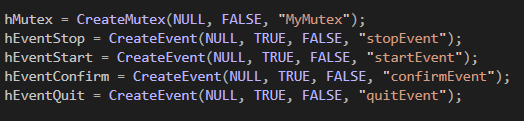
В случае закрытия главного потока (если открытых пользователем потоков не осталось) сработает событие quitEvent, которое оповестит консольное приложение о просьбе закрыться. Остальной алгоритм остается прежним

2)Создание консольного приложения

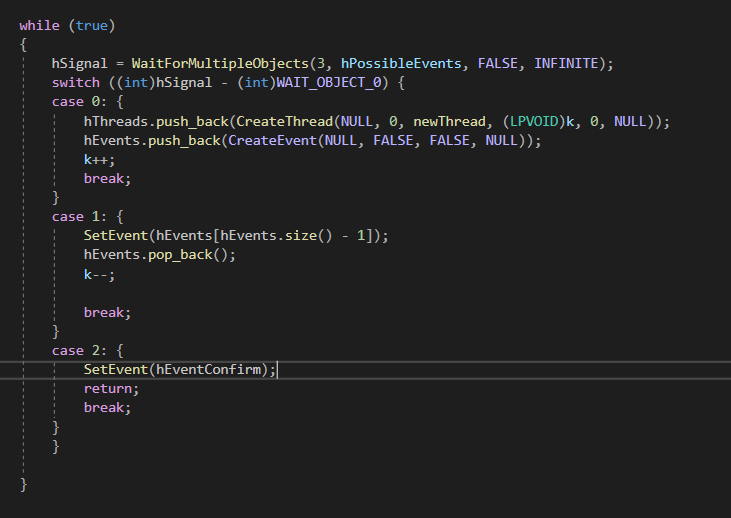
Как уже указано ранее, C++ и C# общаются при помощи специальных объектов ядра – событий.

Для того, чтобы обеспечить правильную связь, необходимо одинаково наименовать события.

Создание объекта события происходит также по функции CreateEvent():

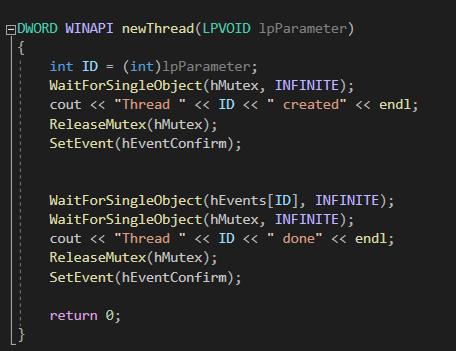


Задача консольного приложения – ожидание изменения состояния событий и принятие соответствующих действий. Ожидание изменения состояния, идентификация события и реакция реализованы в данном цикле:



Ожидание события реализуется с помощью функции семейства WaitFor – WaitForMultipleObjects(). Данная функция ожидает срабатывания хотя бы одного из событий. С помощью возвращаемого значения этой функции -WAIT\_OBJECT\_0 – вычисляется индекс сработавшего события.

1. В случае перехвата события на создание потока используем функцию CreateThread(). Функция newThread реализована слежующим образом:



Здесь уже используется WaitForSingleObject(), а также для синхронизации потоков используется Mutex. Механизм его работы схож с механизмом работы семафора: мьютекс может занять лишь один поток, и только после его освобождения дается доступ следующему. Основное отличие от семафора заключается в том, что освободить мьютекс может только занявший его поток.