Выполнила: Белоусова Е., ИП-911

Цель: приобрести навыки разработки многопоточных приложений на основе различных реализаций потоков.
Задание 1: протестировать программы, разобранные в Лекции 8.
Задание 2: написать многопоточную программу, каждый поток которой увеличивает значение глобальной переменной на определенную величину. Проверить совпадает ли значение этой переменной с ожидаемым значением после выполнения всех потоков.

Описание работы программы

Протестируем программы, разобранные в Лекции 8.

```
PS D:\семестр 5\os\лаб8> ./1.exe
parent
child
parent
thread_is_over!
```

Visual C++

```
PS D:\семестр 5\os\лаб8> ./2.exe
parent
child
parent
thread_is_over!
```

C++ 11

```
PS D:\семестр 5\os\лаб8> ./3.exe
parent
child
parent
thread_is_over!
```

Win32 API

В этих программах мы создаем новый поток, передаем ему функцию Thread для выполнения и в качестве аргумента передаем ссылку на функцию g. Далее параллельно выполняется цикл while(q) в main и такой же цикл в функции Thread.

Поток использует код и сегменты данных процесса с другими потоками в процессе, но имеет собственные уникальные значения регистра, пространство стека и адрес текущей инструкции.

В C++11 для того чтобы создать новый поток нужно создать объект класса thread и инициализировать передав в конструктор имя функции которая должна выполнятся в потоке. Для синхронизации используется метод join().

Метод join возвращает выполнение программе, когда поток заканчивает выполнение, после чего объект класса thread можно безопасно уничтожить.

В Visual C++ с помощью _beginthread создаем новый поток. Передаем в качестве аргументов начальный адрес процедуры для выполнения нового потока, размер стека нового потока или 0, список аргументов, передаваемый в новый поток или NULL.

В Win32 API с помощью CreateThread создаем новый поток. В качестве аргументов передаем указатель на структуру SECURITY_ATTRIBUTES (которая определяет, может ли возвращаемый дескриптор наследоваться дочерними процессами. Если lpThreadAttributes имеет значение NULL, дескриптор не может быть унаследован), размер стека в байтах, указатель на функцию, которая будет выполняться в потоке, указатель на переменную, которая должна быть передана потоку, флаг, контролирующий создание потока и указатель на переменную, которая получает идентификатор потока (если этот параметр равен нулю, идентификатор потока не возвращается).

Value	Meaning
0	The thread runs immediately after creation.
CREATE_SUSPENDED 0x00000004	The thread is created in a suspended state, and does not run until the ResumeThread function is called.
STACK_SIZE_PARAM_IS_A_RESERVATION 0x00010000	The *dwStackSize* parameter specifies the initial reserve size of the stack. If this flag is not specified, *dwStackSize* specifies the commit size.

Написать многопоточную программу, каждый поток которой будет увеличивать значение глобальной переменной на определенную величину.

Проверим, совпадает ли значение этой переменной с ожидаемым значением после выполнения всех потоков. Ожидаемое значение – 44000.

```
PS D:\семестр 5\os\лаб8> ./4.exe
43626
PS D:\семестр 5\os\лаб8> ./4.exe
43000
PS D:\семестр 5\os\лаб8> ./4.exe
44000
PS D:\семестр 5\os\лаб8> ./4.exe
41720
PS D:\семестр 5\os\лаб8> ./4.exe
44000
PS D:\семестр 5\os\лаб8> ./4.exe
44000
PS D:\семестр 5\os\лаб8> ./4.exe
41855
```

Мы не всегда получаем ожидаемое значение, т.к. потоки используют общие ресурсы и вычисления конфликтуют.

Листинг

```
#include <windows.h>
#include <thread>
#include <stdio.h>

int SUM = 0;

void sum(int N)
{
   for(int i = 0; i < N; i++)
   {
      SUM++;
   }
   //printf("%d\n", SUM);
}

int main()
{
   std::thread th1(sum, 1000);
   std::thread th2(sum, 3000);
   std::thread th3(sum, 40000);

   th1.join();
   th2.join();
   th3.join();
   printf("%d\n", SUM);
   return 0;
}
```