Задание 4. Общие и частные переменные в OpenMP: программа «Скрытая ошибка»

Изучите конструкции для управления работой с данными shared и private. Напишите программу, в которой создается k нитей, и каждая нить выводит на экран свой номер через переменную rank следующим образом:

```
rank = omp_get_thread_num();
printf("I am %d thread.\n", rank);
```

Экспериментами определите, общей или частной должна быть переменная rank.

Входные данные: целое число k – количество нитей в параллельной области.

Выходные данные: k строк вида «`I am <Номер нити>.`».

Пример входных и выходных данных

Входные данные	Выходные данные
3	I am 0 thread. I am 1 thread. I am 2 thread.

Задание 5. Общие и частные переменные в OpenMP: параметр reduction

1. Напишите программу, в которой две нити параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью оператора if языка C. Для сложения результатов вычисления нитей воспользуйтесь OpenMP-параметром reduction.

Входные данные: целое число N – количество чисел.

Выходные данные: каждая нить выводит свою частичную сумму в формате «`[Номер_нити]: Sum = <частичная_сумма>`», один раз выводится общая сумма в формате «`Sum = <сумма>`».

Пример входных и выходных данных

Входные данные	Выходные данные
4	[0]: Sum = 3 [1]: Sum = 7 Sum = 10

2*. Модифицируйте программу таким образом, чтобы она работала для k нитей.

Входные данные: целое число k – количество нитей, целое число N – количество чисел.

Выходные данные: каждая нить выводит свою частичную сумму в формате «`[Номер_нити]: Sum = <частичная_сумма>`», один раз выводится общая сумма в формате «`Sum = <сумма>`».

Указания к заданию 4. Общие и частные переменные в OpenMP: программа «Скрытая ошибка»

1. Создайте проект omp_hide_error в Microsoft Visual Studio 2010 с поддержкой OpenMP (см. указания к заданию 1).
2. В функции main создайте параллельную область с k нитями. Вставьте следующий код:

```
rank = omp_get_thread_num();
printf("I am %d thread.\n", rank);
```

3. Определите переменную rank как общую. Для этого достаточно объявить переменную rank до начала параллельной области. Скомпилируйте и запустите ваше приложение. Верный ли результат выдает программа?
4. Добавьте в параллельную область код, имитирующий длительные вычисления, следующим образом:

```
rank = omp_get_thread_num();
Sleep(100); // Имитация длительных вычислений
printf("I am %d thread.\n", rank);
```

Справка: функция Sleep приостановит выполнение программы на указанный интервал времени, заданный во входном параметре в миллисекундах. Для использования данной функции в вашей программе необходимо подключить заголовочный файл Windows.h.

5. Скомпилируйте и запустите ваше приложение. Верный ли результат выдает программа?
6. Переопределите переменную rank как частную. Для этого добавьте к директиве parallel параметр private(), в круглые скобки поместите переменную rank:

```
#pragma omp parallel private(rank)
```

7. Скомпилируйте и запустите ваше приложение. Объясните все полученные выше результаты.

Указания к заданию 5. Общие и частные переменные в OpenMP: параметр reduction

1. Создайте проект omp_reduction в Microsoft Visual Studio 2010 с поддержкой OpenMP (см. указания к заданию 1).
2. В функции main создайте параллельную область с 2-я нитями.
3. Для распределения вычислений по нитям в параллельной области напишите следующую конструкцию:

```
if (omp_get_thread_num() == 0) {
```

```
    // вычисления для нити с номером 0
}
else {
    // вычисления для нити с номером 1
}
```

Напишите для нити с номером 0 цикл, вычисляющий сумму чисел от 1 до N/2, а для нити с номером 1 – от N/2 до N. Частичные суммы запишите в переменную sum.

4. *Общей или частной должна быть переменная sum?* Переменная sum должна быть с одной стороны частной, чтобы избежать ошибки потери слагаемого при одновременной записи в нее двумя нитями, а с другой стороны – должна быть общей, чтобы иметь возможность сложить частичные суммы, подсчитанные нитями. Для таких случаев удобно использовать OpenMP-параметр reduction.
5. Вставьте в директиве parallel параметр reduction:

   ```
   #pragma omp parallel reduction(+:sum)
   ```

   Синтаксис параметра reduction:

   reduction(*операция***:***список***)**, где

   операция – это одна из операций **+**, *****, **-**, **&**, |, ^, **&&**, || (для языка C);

   список – это список общих переменных, для каждой из которых создаются локальные копии в каждой нити. Над локальными копиями переменных после выполнения всех операторов параллельной области выполняется *операция*. Локальные копии инициализируются соответственно типу *операции* (для аддитивных операций – 0 или его аналоги, для мультипликативных операций – 1 или ее аналоги).

6. Скомпилируйте и запустите ваше приложение. Убедитесь, что выдается верный результат.