Задание 6. Распараллеливание циклов в OpenMP: программа «Сумма чисел»

Изучите OpenMP-директиву параллельного выполнения цикла for. Напишите программу, в которой k нитей параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью OpenMP-директивы for.

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>`».

Пример входных и выходных данных

Входные данные	Выходные данные
2 4	[0]: Sum = 3 [1]: Sum = 7 Sum = 10
2 2	[0]: Sum = 1 [1]: Sum = 2 Sum = 3
3 2	[0]: Sum = 1 [1]: Sum = 2 [2]: Sum = 0 Sum = 3

Задание 7. Распараллеливание циклов в OpenMP: параметр schedule

Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания 6 таким образом, чтобы дополнительно выводилось на экран сообщение о том, какая нить, какую итерацию цикла выполняет:

`[<Номер нити>]: calculation of the iteration number <Номер итерации>.`

Задайте k = 4, N = 10. Заполните следующую таблицу распределения итераций цикла по нитям в зависимости от параметра schedule:

Номер итерации	Значение параметра schedule						
	static	**static, 1**	**static, 2**	**dynamic**	**dynamic, 2**	**guided**	**guided, 2**
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

Задание 8. Распараллеливание циклов в OpenMP: программа «Матрица»

Напишите OpenMP-программу, которая вычисляет произведение двух квадратных матриц $A \times B = C$ размера $n \times n$. Используйте следующую формулу:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ . & . & . & \dots & . \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ . & . & . & \dots & . \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{pmatrix}$$

$$c_{im} = \sum_{j=1}^{n} a_{ij} \cdot b_{jm}; i = 1,2, \dots, n;\ m = 1,2, \dots, n$$

$$C = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} \cdot b_{j1} & \sum_{j=1}^{n} a_{1j} \cdot b_{j2} & \sum_{j=1}^{n} a_{1j} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{1j} \cdot b_{jn} \\ \sum_{j=1}^{n} a_{2j} \cdot b_{j1} & \sum_{j=1}^{n} a_{2j} \cdot b_{j2} & \sum_{j=1}^{n} a_{2j} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{2j} \cdot b_{jn} \\ \dots & \dots & \dots & \dots & \dots \\ \sum_{j=1}^{n} a_{nj} \cdot b_{j1} & \sum_{j=1}^{n} a_{nj} \cdot b_{j2} & \sum_{j=1}^{n} a_{nj} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{nj} \cdot b_{jn} \end{pmatrix}$$

Входные данные: целое число n, $1 \le n \le 10$, n^2 вещественных элементов матрицы A и n^2 вещественных элементов матрицы B.
Выходные данные: n^2 вещественных элементов матрицы C.

Пример входных и выходных данных

Входные данные	Выходные данные
2 1 4 5 3	14 44

Указания к заданию 6. Распараллеливание циклов в OpenMP: программа «Сумма чисел»

1. Откройте проект omp_reduction в Microsoft Visual Studio 2010 (см. указания к заданию 5).
2. Определите параметр k. В параллельной области функции main задайте k нитей.
3. В параллельную область вставьте директиву for, которая самостоятельно будет производить распределение итераций по нитям:

```
#pragma omp for
```

После директивы for должен идти только оператор for языка C. Счетчик цикла должен принимать все (!) значения от 1 до N:

```
#pragma omp for
   for(i=1; i<=N; i++) {
       // вычисление суммы чисел от 1 до N }
```

4. В результате выполнения такого цикла каждая из k нитей будет выполнять k-ю часть всех имеющихся итераций цикла. При этом на вид параллельных циклов накладывается ограничение: программа не должна зависеть от того, какая именно нить, какую итерацию параллельного цикла выполнит, т.е. **итерации цикла должны быть не зависимы**!

   В данной задаче это условие выполняется, т.к. очередность сложения слагаемых не важна.

5. Для сложения частичных сумм используйте параметр reduction.
6. Скомпилируйте и запустите ваше приложение. Убедитесь, что выдается верный результат.

Указания к заданию 7. Распараллеливание циклов в OpenMP: параметр schedule

1. Откройте проект omp_reduction в Microsoft Visual Studio 2010 (см. указания к заданию 6).

1. В параллельную область вставьте вывод сообщения «`[<Номер нити>]: calculation of the iteration number <Номер итерации>.`».
2. Добавьте для директивы for параметр schedule, который задает, каким образом итерации цикла распределяются между нитями. Присваивая ему поочередно значения из таблицы, компилируйте и запускайте ваше приложение. Результаты выполнения запишите в таблицу, данную в задании.

   Синтаксис параметра schedule:

   `schedule(``type[, chunk]`**`),`**

   где type задает тип распределения итераций; основные значения следующие:

   `static` – блочно-циклическое распределение итераций цикла; размер блока – chunk. Первый блок из chunk итераций выполняет нулевая нить, второй блок — следующая и т.д. до последней нити, затем распределение снова начинается с нулевой нити. Если значение chunk не указано, то все множество итераций делится на непрерывные куски примерно одинакового размера (конкретный способ зависит от реализации), и полученные порции итераций распределяются между нитями.

   `dynamic` – динамическое распределение итераций с фиксированным

размером блока: сначала каждая нить получает chunk итераций (по умолчанию chunk=1), та нить, которая заканчивает выполнение своей порции итераций, получает первую свободную порцию из chunk итераций. Освободившиеся нити получают новые порции итераций до тех пор, пока все порции не будут исчерпаны. Последняя порция может содержать меньше итераций, чем все остальные.

`guided` – динамическое распределение итераций, при котором размер порции уменьшается с некоторого начального значения до величины chunk (по умолчанию chunk=1) пропорционально количеству еще не распределенных итераций, деленному на количество нитей, выполняющих цикл. Размер первоначально выделяемого блока зависит от реализации. В ряде случаев такое распределение позволяет аккуратнее разделить работу и сбалансировать загрузку нитей. Количество итераций в последней порции может оказаться меньше значения chunk.

Указания к заданию 8. Распараллеливание циклов в OpenMP: программа «Матрица»

1. Создайте проект omp_matrix в Microsoft Visual Studio 2010 с поддержкой OpenMP (см. указания к заданию 1).
2. Напишите последовательную программу, вычисляющую произведение матриц по приведенной в задании формуле.
3. Определите код, который можно распараллелить, и задайте параллельную область с помощью директивы parallel.
4. Определите цикл for, подходящий для распараллеливания. Используйте для него OpenMP-директиву for.
5. Изучите все переменные в параллельной области. Определите, общие переменные, частные и reduction.
6. Скомпилируйте и запустите ваше приложение. Убедитесь, что выдается верный результат.