Отчёт по выполнению  
Лабораторной работы №3  
«Основы OpenMP»

Выполнил:  
студент гр. МОиАИС 184-1  
 Ибраев Ерлан.

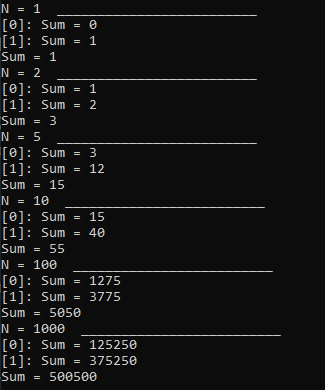
**Цель работы:** Знакомство с базовыми элементами технологии распараллеливания OpenMP.

**Задание 1.** Общие и частные переменные в OpenMP: параметр reduction

Напишите программу, в которой две нити параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью оператора if языка С. Для сложения результатов вы-числения нитей воспользуйтесь OpenMP-параметром reduction. Выполните расчёты для N Î {1,2,5,10,100,1000}

Входные данные: целое число N – количество чисел.

Выходные данные:



int Sum(int start, int end)   
{  
 int sum = 0;  
 for (int i = start; i <= end; i++)  
 sum += i;  
 return sum;  
}

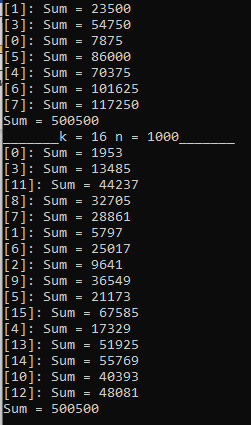
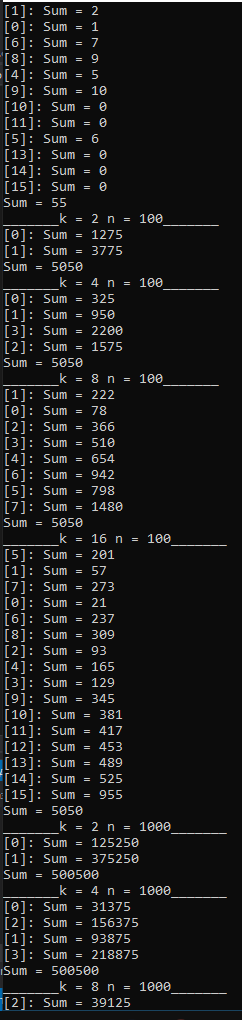
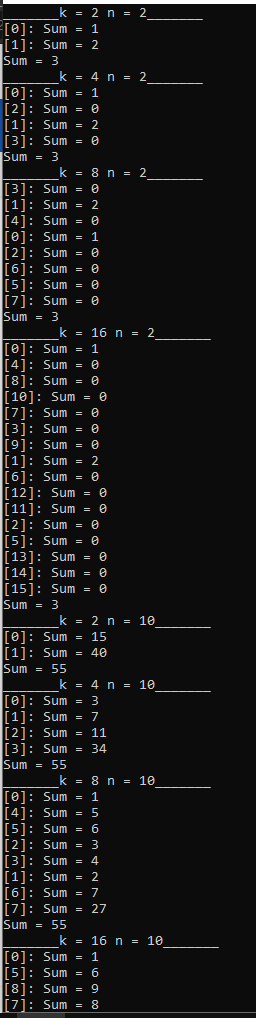
void Task1(int n)  
{  
 int sum = 0;  
 printf("N = %d \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n", n);  
#pragma omp parallel num\_threads(2) reduction (+ : sum)  
 {  
 int currentThread = omp\_get\_thread\_num();  
 if (currentThread == 0)  
 {  
 sum = Sum(0, n / 2);  
 printf("[0]: Sum = %d\n", sum);  
 }  
 if (currentThread == 1)  
 {  
 sum = Sum(n / 2 + 1, n);  
 printf("[1]: Sum = %d\n", sum);  
 }  
 }  
 printf("Sum = %d\n", sum);  
}

**Вывод:** выполнение задачи делится на два потока, следовательно алгоритм выполняется быстрей.

**Задание 2:** Общие и частные переменные в OpenMP: параметр reduction.

Модифицируйте программу таким образом, чтобы она работала для k нитей. Покажите работу программы для N Î {2,10,100,1000}, k Î {2,4,8,16}.

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



void Task2(int n, int k)  
{  
 int sum = 0;  
 printf("\_\_\_\_\_\_\_k = %d n = %d\_\_\_\_\_\_\_\n", k, n);  
#pragma omp parallel num\_threads(k) reduction(+ : sum)  
 {  
 int current = omp\_get\_thread\_num();  
 if (k >= n)  
 if (current < n)   
 sum += current + 1;  
 else   
 {  
 if (current + 1 != k)  
 sum += Sum(n / k \* current + 1, n / k \* (current + 1));  
 else sum += Sum(n / k \* current + 1, n);  
 }  
 printf("[%d]: Sum = %d\n", current, sum);  
 }  
 printf("Sum = %d\n", sum);

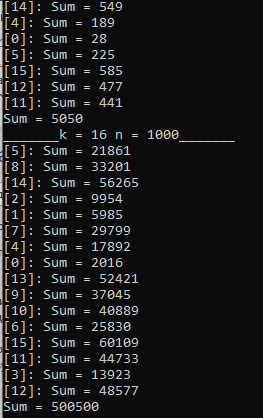
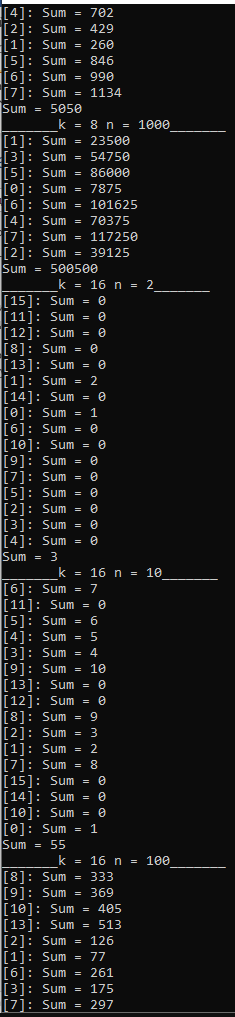
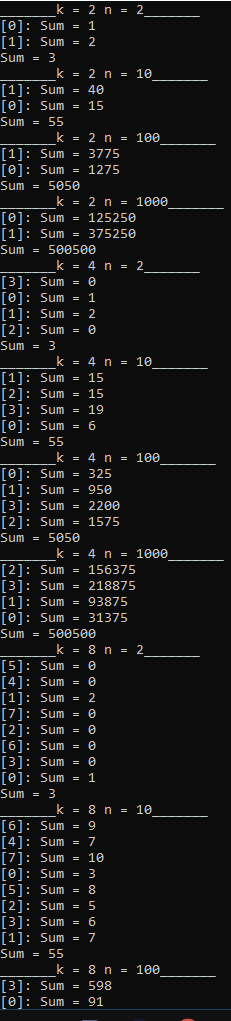
}

**Вывод:** программа сделана таким образом, что, если задачу невозможно поделить на все заданные потоки.

**Задание 3:** 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. Покажите работу программы для N Î {2,10,100,1000}, k Î {2,4,8,16}.

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

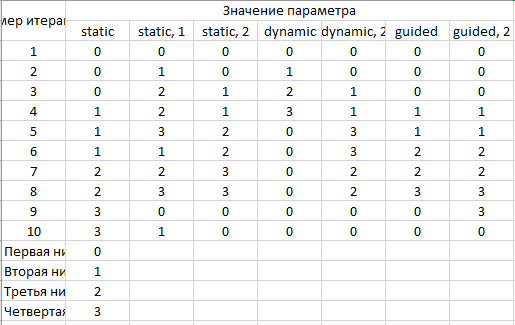


void Task3(int k, int n)   
{  
 int sum = 0;  
 printf("\_\_\_\_\_\_\_k = %d n = %d\_\_\_\_\_\_\_\n", k, n);  
#pragma omp parallel num\_threads(k) reduction(+ : sum)  
 {  
 int current = omp\_get\_thread\_num();  
#pragma omp for  
 for (int j = 1; j <= n; j++) sum += j;  
 printf("[%d]: Sum = %d\n", current, sum);  
 }  
 printf("Sum = %d\n", sum);  
}

**Вывод:** в директиве параллельного выполнения цикла for такая же проблема, что и во втором задании. Но директива for умеет рационально разбивать циклы между указанными количеством потоков.

**Задание 4:** Распараллеливание циклов в OpenMP: параметр schedule

Изучите параметр schedule директивы for. Модифицируйте программу «Сумма чисел» из задания 3 таким образом, чтобы дополнительно выводилось на экран сообщение о том, какая нить, какую итерацию цикла выполняет:



void Task4()  
{  
 int sum = 0, i;  
#pragma omp parallel num\_threads(4) reduction(+ : sum)  
 {  
#pragma omp for schedule(guided)  
 for (i = 1; i <= 10; i++) {  
 int currentThread = omp\_get\_thread\_num();  
 sum += i;  
 printf("[%d]: Sum = %d\n", currentThread, sum);  
 printf("[%d]: calculation of the iteration number %d\n", currentThread, i);  
 }  
 }  
 printf("Sum = %d\n", sum);  
}

**Вывод:** в процессе выполнения задания 4 были изучены параметры директивы schedule. Static означает, что chunk будут выделены до выполнения потоков и передаваемый в директиву – из размер (т.е. блочноциклическое распределение итераций цикла; размер блока – chunk). Dynamic работает так же как и static за тем лишь исключением, что chunk выделяются каждому потоку во время выполнения программы. Guided похож на Dynamic, но размер chunk постоянно уменьшается, пока не станет равным 1.

**Задание 5:** Распараллеливание циклов в OpenMP: параметр schedule

Реализовать вычисление числа π с требуемой точностью ε по формуле

Для вычисления интеграла использовать метод левых прямоугольников.

Исследовать зависимость времени работы алгоритма от числа потоков {k=1,2,3,…,20} и желаемой точности (ε=0.001, 0.0001, 0.00001, …, 0.000000001, … - остановка, когда время последнего расчёта превысило 2 мин). Полученную зависимость показать на графике.

Для достижения необходимой точности на каждом шаге увеличивать количество разбиений в 2 раза, пока уточнение результата не станет изменяться меньше, чем на ε.



**Вывод:** на графике видно, что с повышением точности – увеличивается время выполнение задачи на всех количествах потоков. На одном потоке при большой точности заметны серьезные просадки по времени. Количество потоков оказывает разное влияние: когда-то в большую часть, когда-то – в меньшею сторону

void Task5()  
{  
 double eps = 0.001;  
 double prev = 0;  
 double curr = 1;  
 double dt;  
 int n = 1;  
 int k = 1;  
 string out;  
 clock\_t start, end;  
 do  
 {  
 do  
 {  
 start = clock();  
 do  
 {  
#pragma omp parallel for num\_threads(k) schedule(dynamic)  
 for (int i = 0; i < k; i++)  
 {  
 int ThCurr = omp\_get\_thread\_num();  
 prev = curr;  
 curr = LeftRectangle(0, 1, n);  
 n \*= 2;  
 //printf("[%d] accuracy:%f. Needed eps=%f, Current K=%d, Current N=%d\n", ThCurr, abs(curr - prev), eps, k, n);  
 }  
 } while (abs(curr - prev) > eps);  
 end = clock();  
 n = 1;  
 dt = (double)(end - start) / CLOCKS\_PER\_SEC;  
 k++;  
 printf("Значение интеграла с точностью %f = %f; Затрачено %f сек.\n", eps, curr, dt);  
 out += "" + to\_string(eps) + ";" + to\_string(k) + ";" + to\_string(dt) + "" + "\n";  
 } while (k <= 20 && dt < 120);  
 k = 1;  
 eps /= 10;  
 } while (dt < 120);  
 ofstream F;  
 F.open("C:/Users/Grost/Desktop/ATPP/ConsoleApplication5/Task5.csv", ios::out);  
 F << out;  
 F.close();  
}

double f(double x)  
{  
 return 4 / (1 + pow(x, 2));  
}  
double LeftRectangle(int a, int b, double n)  
{  
 double h = (double)(b - a) / n;  
 double sum = 0;  
 double x;  
 for (int i = 0; i < n; i++)  
 {  
 x = a + i \* h;  
 sum += f(x);  
 }  
 return sum \* h;  
}