

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования

**"Южно-Уральский государственный университет
(национальный исследовательский университет)"**

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении практического задания № 4

по дисциплине

«Теория, методы и средства
параллельной обработки информации»

Выполнил:

студент группы КЭ-303

Старостенок Д.В.

Проверил:

доцент кафедры СП

Маковецкая Т.Ю.

Задание:

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:

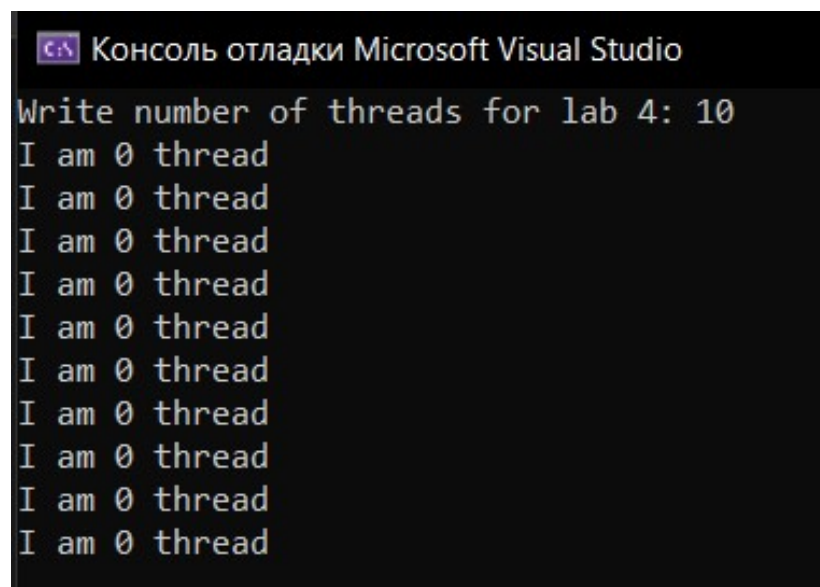
```
rank = omp_get_thread_num();  
printf("I am %d thread.\n", rank);
```

Экспериментами определите, общей или частной должна быть переменная rank.

Листинг программы:

```
int number_threads_3;  
printf("Write number of threads for lab 4: ");  
scanf_s("%d", &number_threads_3);  
int rank = omp_get_thread_num();  
  
#pragma omp parallel num_threads(number_threads_3)  
{  
    printf("I am %d thread\n", rank);  
}
```

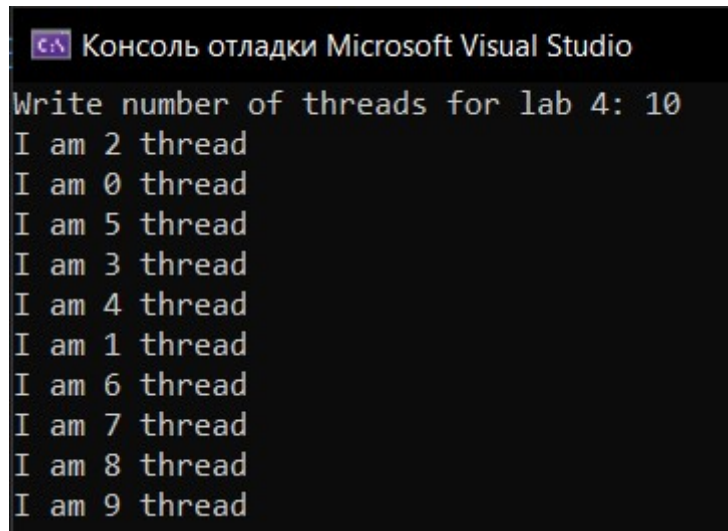
Результат выполнения программы с типом shared для rank (Рис. 1).



```
C:\> Консоль отладки Microsoft Visual Studio  
Write number of threads for lab 4: 10  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread  
I am 0 thread
```

Рис. 1 – Результат выполнения программы shared

Результат выполнения программы с типом private для rank (Рис. 2).



```
Консоль отладки Microsoft Visual Studio
Write number of threads for lab 4: 10
I am 2 thread
I am 0 thread
I am 5 thread
I am 3 thread
I am 4 thread
I am 1 thread
I am 6 thread
I am 7 thread
I am 8 thread
I am 9 thread
```

Рис. 2 – Результат выполнения программы private

Ответы на вопросы:

1) Для чего нужны частные переменные? Не противоречит ли их существование реализуемой OpenMP модели программирования в общей памяти? Приведите содержательный пример частной переменной.

Частные переменные необходимы для того, чтобы каждая нить имела свою локальную копию переменной внутри параллельного региона, а не общую. Это позволяет избежать некорректных результатов при одновременном обращении нескольких нитей к одной переменной.

Использование частных переменных не противоречит модели программирования в общей памяти, так как каждая нить имеет доступ только к своей локальной копии переменной внутри параллельной области.

Пример:

```
int main() {
    int sum = 0;
    #pragma omp parallel for private(sum)
    for (int i = 0; i < 10; i++) {
        sum += i;
        int thread_id = omp_get_thread_num();
        std::cout << "Thread " << thread_id << ": local sum = " <<
sum << std::endl;
    }
    return 0;
}
```

2) Какие новые области видимости появляются в параллельной программе? Как они задаются?

В параллельной программе появляются две области видимости: `shared` и `private`. Область видимости `shared` обозначает переменные, доступные для чтения и записи всеми нитями в параллельном регионе, а область видимости `private` обозначает переменные, которые должны иметь свою локальную копию для каждой нити в параллельном регионе. Они задаются с помощью директив OpenMP, таких как `shared` и `private`.

3) Продемонстрируйте конфликт обращений к переменной `rank` в написанной программе? Всегда ли он возникает? Как его предотвратить?

Конфликт обращений к переменной `rank` возникает, когда несколько нитей пытаются одновременно обратиться к общей переменной `rank` без использования директивы `private`. В таком случае значение переменной может быть изменено несколькими нитями одновременно, что приведет к некорректным результатам. Чтобы предотвратить конфликт обращений к переменной `rank`, необходимо использовать директиву `private` для того, чтобы каждая нить имела свою локальную копию переменной внутри параллельной области. Также можно использовать синхронизацию для контроля доступа к общей переменной.

Пример:

```
int rank = 0;
#pragma omp parallel num_threads(4)
{
    rank = omp_get_thread_num();
    printf("I am %d thread.\n", rank);
    rank++; // конфликт обращений к переменной rank
}
printf("Final value of rank: %d\n", rank);
return 0;
```

Переменная `rank` общая и каждая нить увеличивает ее значение на 1, что может привести к конфликту обращений и непредсказуемому значению переменной после выполнения параллельной области.