**Отчет лабораторной работы №6**

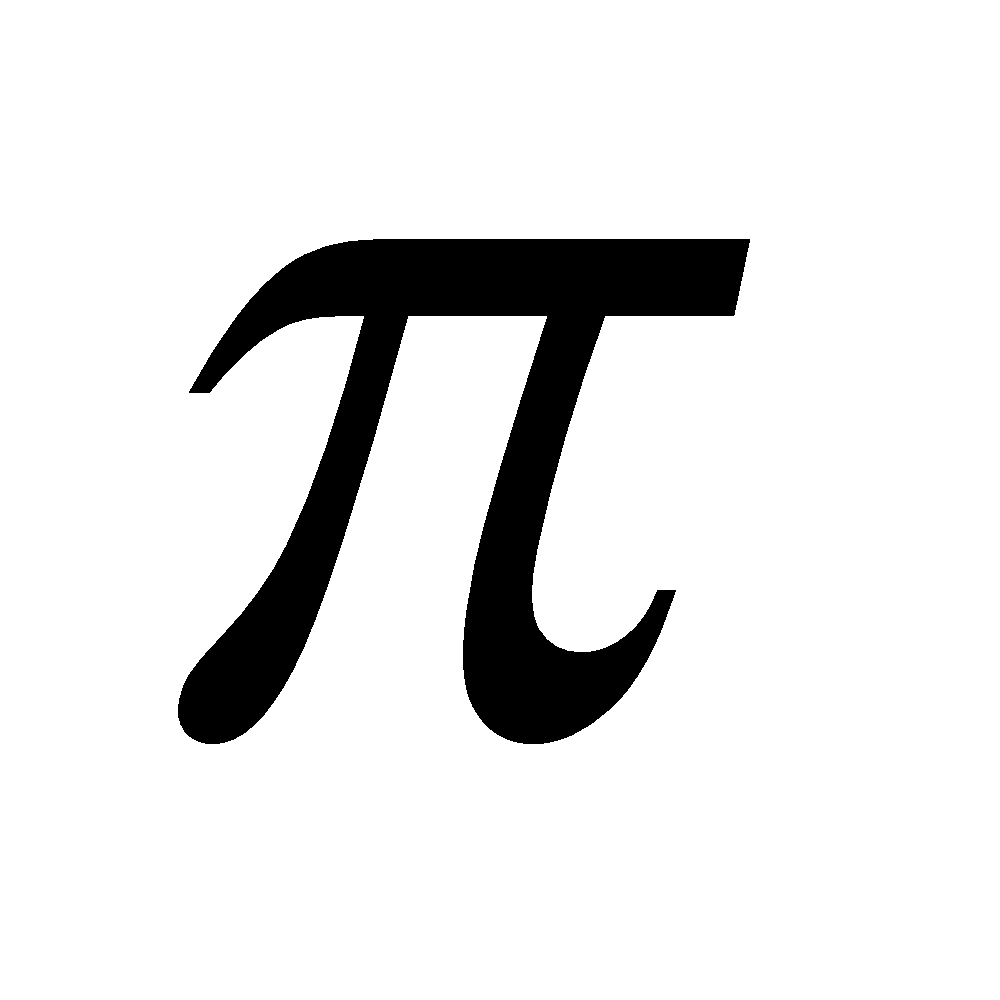
**Тема «** **Технология OpenMP.** **Потокобезопасные функции»**

**Султанов Шакир КА**

Студенты должны знать:

1. Директивы OpenMP.
2. Потокобезопасные функции.

**Задание**

Вычислить число методом Монте Карло.

Написать функцию random(), вычисляющую псевдослучайную последовательность целых чисел с помощью линейного конгруэнтного генератора. Функция должна быть потокобезопасной.

Провести вычислительный эксперимент. Заполнить таблицу

**КОД**

**#include <stdio.h>**

**#include "omp.h"**

**static long MULTIPLIER = 1366;**

**static long ADDEND = 150889;**

**static long PMOD = 714025;**

**long random\_last = 0;**

**#pragma omp threadprivate(random\_last)**

**double random()**

**{**

**long random\_next;**

**random\_next = (MULTIPLIER \* random\_last + ADDEND) % PMOD;**

**random\_last = random\_next;**

**return ((double)random\_next / (double)PMOD);**

**}**

**void main()**

**{**

**long num\_trials;**

**printf("Введите кол-во шагов:");**

**scanf("%ld", &num\_trials);**

**long i;**

**long Ncirc = 0;**

**double pi, x, y;**

**double r = 1.0; // radius of circle. Side of squrare is 2\*r**

**//seed(0,-r, r); // The circle and square are centered at the origin**

**unsigned long long pseed[10], iseed;**

**int n;**

**printf("Введите кол-во потоков:");**

**scanf("%d", &n);**

**#pragma omp parallel num\_threads(n)**

**{**

**#pragma omp single**

**{ int nthreads = omp\_get\_num\_threads();**

**iseed = PMOD / MULTIPLIER; // just pick a seed**

**pseed[0] = iseed;**

**int mult\_n = MULTIPLIER;**

**for (i = 1; i < nthreads; ++i)**

**{**

**iseed = (unsigned long long)((MULTIPLIER \* iseed) % PMOD);**

**pseed[i] = iseed;**

**mult\_n = (mult\_n \* MULTIPLIER) % PMOD;**

**}**

**}**

**random\_last = (unsigned long long) pseed[omp\_get\_thread\_num()];**

**#pragma omp for private (x, y) reduction (+:Ncirc)**

**for (i = 0;i < num\_trials; i++)**

**{**

**x = random();**

**y = random();**

**if ((x\*x + y \* y) <= r \* r) Ncirc++;**

**}**

**}**

**pi = 4.0 \* ((double)Ncirc / (double)num\_trials);**

**printf("\n %ld trials, pi is %f \n", num\_trials, pi);**

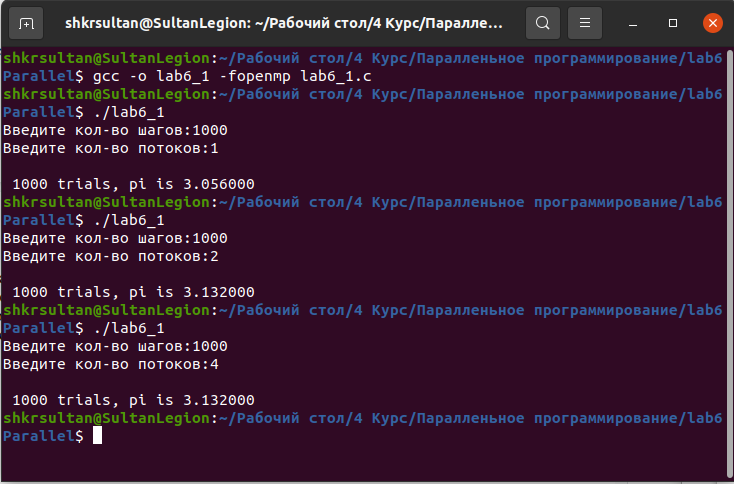
**char c = getchar();**

**}**

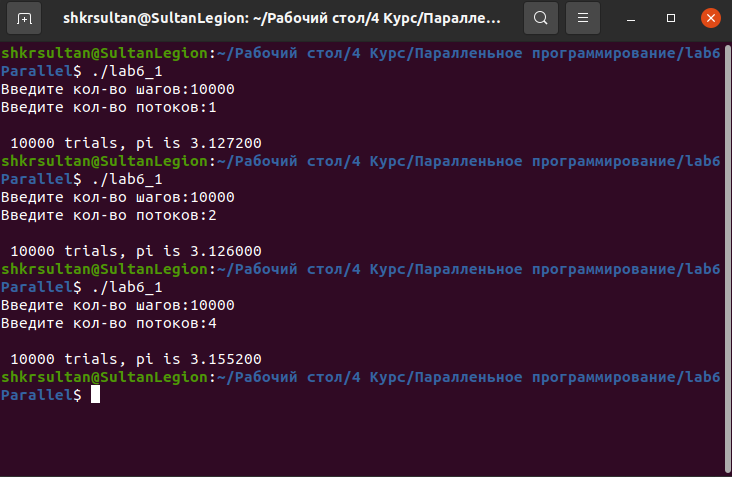
**Результат**

|  |  |  |  |
| --- | --- | --- | --- |
| **Steps** | **One thread** | **2 threads** | **4 threads** |
| 1000 | 3.056 | 3.132 | 3.132 |
| 10000 | 3.1272 | 3.126 | 3.1552 |
| 100000 | 3.13416 | 3.13928 | 3.14512 |
| 1000000 | 3.140944 | 3.141132 | 3.141552 |

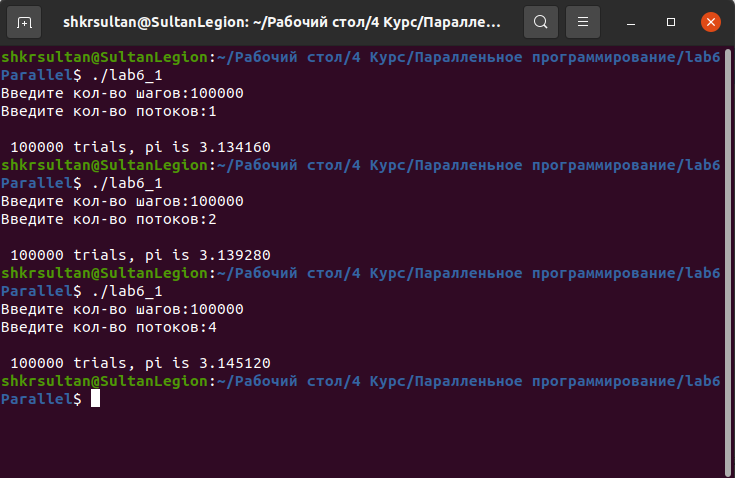
**1000**



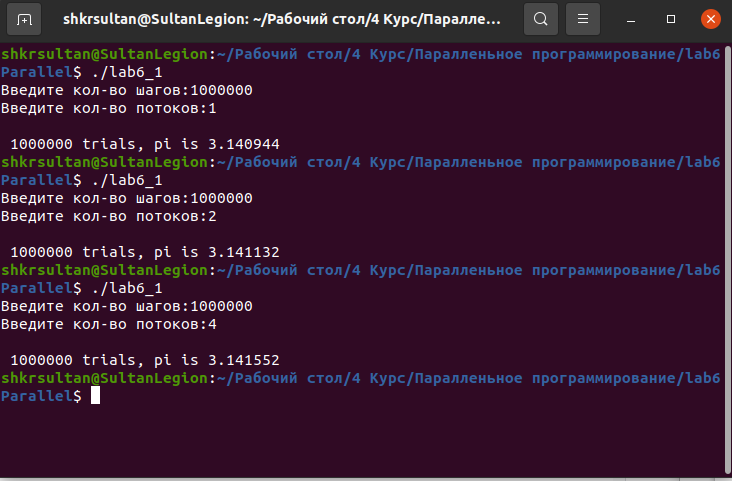
**10000**

****

**100000**

****

**1000000**

****