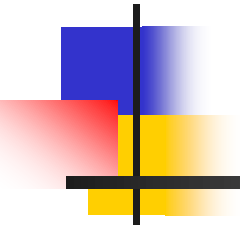
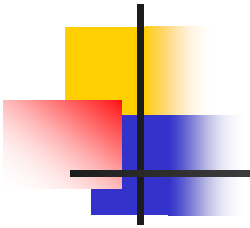


Введение в параллельное программирование

Олег Николаевич Граничин

Осень 2017

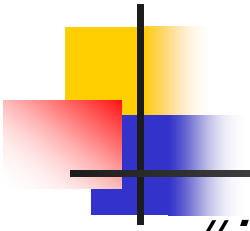
Санкт-Петербургский государственный университет



Практика. C++, OpenMP

- *Задача:* Вычислить среднее арифметическое значение набора большого количества чисел

Заголовок



- *#include "stdafx.h"*
- *#include <omp.h>*
- *#include <cstdio>*
- *#include <iostream>*
- *#include <vector>*
- *#include <algorithm>*
- *#include <ctime>*
- *#include <windows.h>*
- *#include <chrono>*
- *#include <thread>*

Обычная процедура

```
void conventional_procedure(std::vector<double> vec) {
        double mean = 0.0;
        for (int i = 0; i < vec.size(); i++) {
                mean = mean + vec.at(i);
                std::this_thread::sleep_for(std::chrono::
milliseconds(100));
        }
        mean /= vec.size();
        std::cout << "Mean value: " << mean << "\n";
}
```

Процедура с распараллеливанием

```
void parallel_procedure(std::vector<double> vec) {
        double mean = 0.0;
        #pragma omp parallel num_threads(4)
        {
                #pragma omp for reduction(+:mean)
                for (int i = 0; i < vec.size(); i++) {
                        mean = mean + vec.at(i);
                        std::this_thread::sleep_for(...);
                }
        }
        mean /= vec.size();
        std::cout << "Mean value: " << mean << "\n";
}
```

Процедура с «конвейером»

```
#pragma omp parallel num_threads(4)
{
        int ip = 1;
        for (int i = 2; i <= aSize; i *= 2) {
                #pragma omp for
                for (int k = 0; k < aSize; k += i) {
                        a[k] = a[k] + a[k + ip];
                std::this_thread::sleep_for(...);
                }
                ip = i;
        }

}
```

Главная процедура 1

```
int main()
{
    DWORD dwStart, dwDiff;

    // Set "Open MP Support" to True in Project -> <Project Name> Properties -> C/C++ -> Language
    #ifdef _OPENMP
        std::cout << "OpenMP is supported\n\n";
    #endif // _OPENMP

    // Set the size of a vector
    int vSize = 128;
```

Главная процедура 2

```
// Create a vector and initialize with random values
std::vector<double> vec(vSize);
std::srand(static_cast<double>(std::time(0)));
std::generate(vec.begin(), vec.end(), std::rand);

// Test the conventional procedure
std::cout << "Conventional procedure:\n";
dwStart = GetTickCount();
conventional_procedure(vec);
dwDiff = GetTickCount() - dwStart;
std::cout << "Time: " << dwDiff << " milliseconds or "
<< dwDiff / 1000 << " seconds\n\n";
```

Главная процедура 3

```
// Test the parallel procedure
std::cout << "Parallel procedure:\n";
dwStart = GetTickCount();
parallel_procedure(vec);
dwDiff = GetTickCount() - dwStart;
std::cout << "Time: " << dwDiff << " milliseconds or "
<< dwDiff / 1000 << " seconds\n\n";

// Test the parallel procedure 2
std::cout << "Parallel procedure:\n";
dwStart = GetTickCount();
parallel_bin_procedure(vec);
dwDiff = GetTickCount() - dwStart;
```

Главная процедура 4

```
        std::cout << "Time: " << dwDiff << " milliseconds or "
    << dwDiff / 1000 << " seconds\n\n";

        getchar();

        return 0;
    }
```

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Благодарю за внимание!

Вопросы?

Санкт-Петербург