

ИД3-3

Баханкова Дарья Сергеевна

БПИ215

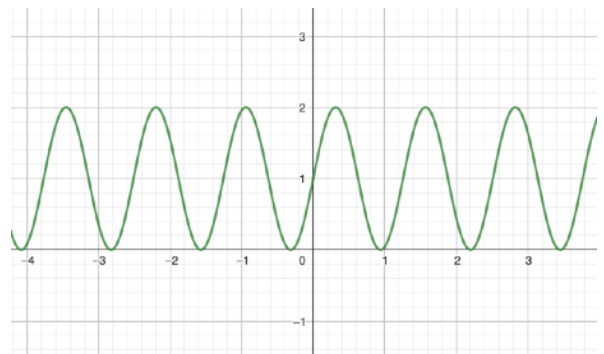
Вариант 34

https://github.com/Dashbah/Bakhankova_Asm_HW4

34. Задача для агронома. Председатель дачного кооператива Сидоров В.И. получил указание, что в связи с составлением единого земельного кадастра, необходимо представить справку о площади занимаемых земель. Известно, что территория с запада и востока параллельна меридианам, на севере ограничена параллелью, а с юга выходят к реке, описываемой функцией $f(x)$. **Требуется создать многопоточное приложение, вычисляющее площадь угодий методом адаптивной квадратуры. При решении использовать парадигму рекурсивного параллелизма.** Замечание: кривизну Земли из-за малой занимаемой площади не учитывать.

Предисловие

Будем считать, что функция нам дана ($f(x)=\sin(5x)+1$) и площадь ограничена снизу осью ОХ. Входные параметры - левая и правая границы. Некорректными считаются случаи, когда левая граница больше правой.



Задача решена с помощью вычисления интеграла методом адаптивной квадратуры.

Исходя из требований к заданию, оно выполнено на **8 баллов**

Метод адаптивной квадратуры:

Интервал разбивается на две части и для каждой из них рекурсивно вычисляется интеграл. Их вызовы независимы, т.е. каждый из них работает над своей частью общих данных. Значит, их можно выполнить параллельно.

```
double qIntegral(double left_, double right_, double f_left, double f_right, double integr_l_now) {
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
    double mid = (left_ + right_) / 2;
    double f_mid = func(x: mid);

    //Аппроксимация по левому отрезку
    double l_integral = (f_left + f_mid) * (mid - left_) / 2;
    //Аппроксимация по правому отрезку
    double r_integral = (f_mid + f_right) * (right_ - mid) / 2;

    if (abs(lcpp_x: (l_integral + r_integral) - integr_l_now) > EPS) {
        //Рекурсия для интегрирования обоих значений
        l_integral = qIntegral(left_, right_: mid, f_left, f_right: f_mid, integr_l_now: l_integral);
        r_integral = qIntegral(left_: mid, right_, f_left: f_mid, f_right, integr_l_now: r_integral);
    }

    return (l_integral + r_integral);
}
```

The screenshot shows a C++ IDE with the following components:

- Project Explorer:** Shows the project structure for "Bakhankova_Asm_HW4", including files like CMakeLists.txt, main.cpp, Func.cpp, inputs.cpp, and README.md.
- Code Editor:** Displays the source code for "Func.cpp". The code implements a parallel integration algorithm using `std::thread` and `std::async`. It calculates the integral of a function `f(x) = x^2` over the interval `[left, right]` using a recursive approach with parallel threads. The code includes comments in Russian and C++ code.
- Run Console:** Shows the output of the program. It prompts for console input, file input, and random generation. The output shows the result `5.088642`, which matches the expected result `5.088643`. The execution time is `14552`.
- Bottom Bar:** Shows the status bar with the current file `Func.cpp` and the active window `Bakhankova_Asm_HW4`.

Один поток (14552мс)

```
33 return (l_integral + r_integral);
34 }
35
36 void qIntegralVoid(double left_, double right_, double f_left, double f_right, double intgrl_now, double *res) {
37     *res = qIntegral(left_, right_, f_left, f_right, intgrl_now);
38 }
39
40 double getResult() {
41     auto n::unsigned int = std::thread::hardware_concurrency();
42
43     auto len::double = (right - left) / n;
44
45     std::vector<std::thread> threads;
46     std::vector<double> part_res;
47     part_res.resize(n);
48
49     double result = 0;
50     for (auto i = 0; i < n; ++i) {
51         threads.emplace_back( &qIntegralVoid, left + len * i, left + len * (i + 1), func(x::left + len * i),
52                               func(x::left + len * (i + 1)), (func(x::left + len * i) + func(x::left + len * (i + 1))) * le
53
54     }
55     for (auto &thread : threads) thread.join();
56     for (auto &part : part_res) result += part;
57 }
58
59 int main() {
60     int type;
61     for (int i = 0; i < 3; ++i) {
62         std::cout << "For " << (i == 0 ? "console input" : i == 1 ? "file input" : "random generation") << " enter " << i << " ";
63         if (i % 3 == 2) std::cout << "\n";
64         if (i % 3 == 2) system("clear");
65         type = i;
66     }
67     double left, right;
68     if (type == 0) {
69         std::cout << "Left, right: ";
70         left = 1;
71         right = 10;
72     } else if (type == 1) {
73         std::ifstream f("input.txt");
74         if (!f.is_open()) {
75             std::cout << "Error opening file\n";
76             return 1;
77         }
78         f >> left;
79         f >> right;
80     } else {
81         left = 1;
82         right = 10;
83     }
84     double result = getResult();
85     std::cout << "Result: " << result << "\n";
86     std::cout << "Expected result: 5.008643\n";
87     std::cout << "Execution time: " << std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::high_resolution_clock::now() - start).count() << "ms\n";
88     return 0;
89 }
```

Run: Bakhankova_Asm_HW4

For console input enter 1
For file input enter 2
For random generation enter 3
1

Left, right:
1 10

Result: 5.008642
Expected result: 5.008643
Execution time: 12481

Process finished with exit code 0

8 потоков(12481мс)

Источники:

http://templet.ssau.ru/wiki/_media/presentations/%D0%B2%D0%B2%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F.pdf

<https://studfile.net/preview/16404441/page:6/>