Выполнила Юдина Екатерина БПИ198

Вариант 29

Текст задания:

29. Вычислить интеграл:

$$\int_a^b f(x)dx,$$

используя метод прямоугольников. Входные данные: вещественные числа a и b, функция f(x) задается с использованием описания в программе в виде отдельной функции. При суммировании использовать принцип дихотомии. Протестировать на различных функциях.

Источники:

Программа выполнена с помощью знаний, полученных на лекциях и на семинарах по дисциплине ABC. Также использовались следующие источники информации:

Сайт преподавателя: http://www.softcraft.ru/

Сайт по многопоточности: https://zen.yandex.ru/media/nuancesprog/c-chast-1-mnogopotochnost-konkurentnost-i-parallelizm-osnovy-5ec7ec813dcf52100c15887a

Сайт по многопоточности: https://nuancesprog-ru.turbopages.org/nuancesprog.ru/s/p/5452/

https://metanit.com/cpp/tutorial/1.1.php

Изучение принципа дихотомии: http://www.machinelearning.ru/wiki/index.php?title=Методы_дихотомии

Изучение способов интегрирования: https://reshit.ru/Reshenie-integralov

Комментарии к работе:

Программа должна считать интеграл функции в указанном диапазоне. Метод прямоугольников [разбиение отрезка интегрирования на части (прямоугольники) с длиной по Ox равной STEP и высотой равной среднему значению на границах данного шага(принцип дихотимии); итоговый результат это сумма площадей всех таких частей] в вычислении интеграла позволяет разбить программу на потоки, выполняющие схожие вычисления в цикле, а затем соединить временные значения вычислений в единый результат. Следовательно используемый подход -- Итеративный параллелизм

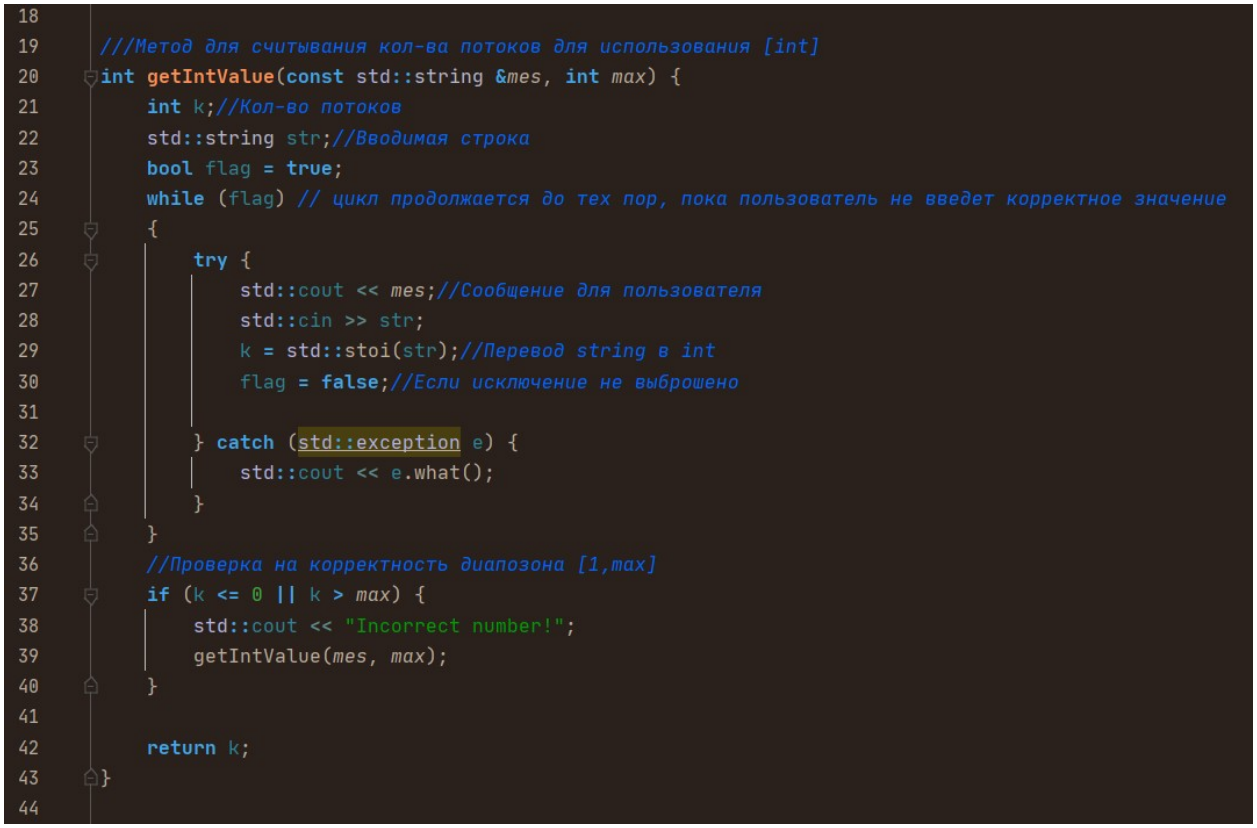
Текст программы:

Ссылка на GitHub: https://github.com/KateJud/ThreadsABC

Глобальные переменные и сама интегрируемая функция

```
9
10  const double STEP = 0.01;//Шаг
11  double a;//Нижняя граница
12  double b;//Верхняя граница
13
14  ///Функция для интегрирования
15  double f(double x) {
16      return x * x;
17  }
18
```

Считывание количества потоков

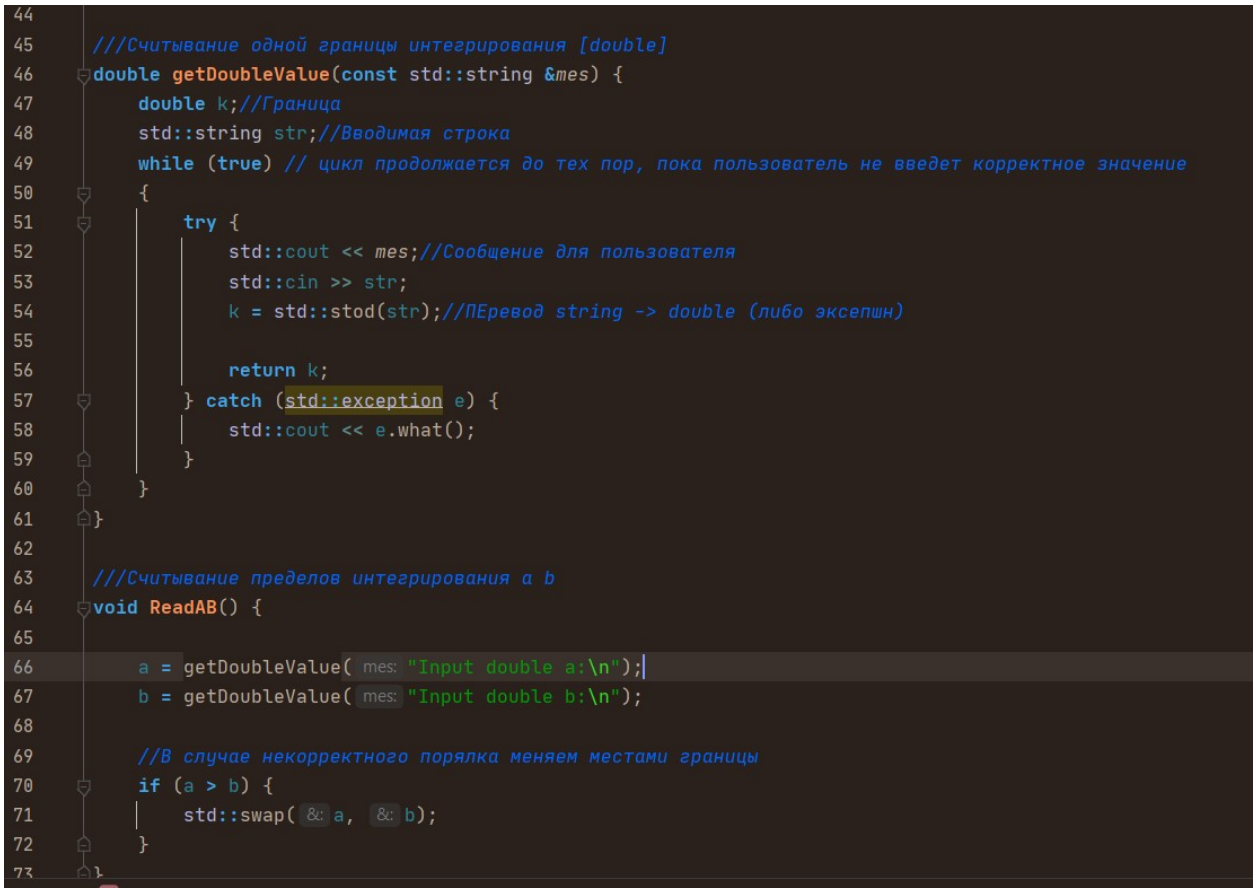


```cpp
///Метод для считывания кол-ва потоков для использования [int]
int getIntValue(const std::string &mes, int max) {
    int k;//Кол-во потоков
    std::string str;//Вводимая строка
    bool flag = true;
    while (flag) // цикл продолжается до тех пор, пока пользователь не введет корректное значение
    {
        try {
            std::cout << mes;//Сообщение для пользователя
            std::cin >> str;
            k = std::stoi(str);//Перевод string в int
            flag = false;//Если исключение не выброшено

        } catch (std::exception e) {
            std::cout << e.what();
        }
    }
    //Проверка на корректность диапозона [1,max]
    if (k <= 0 || k > max) {
        std::cout << "Incorrect number!";
        getIntValue(mes, max);
    }

    return k;
}
```

Считывание границ интегрирования



```cpp
///Считывание одной границы интегрирования [double]
double getDoubleValue(const std::string &mes) {
    double k;//Граница
    std::string str;//Вводимая строка
    while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение
    {
        try {
            std::cout << mes;//Сообщение для пользователя
            std::cin >> str;
            k = std::stod(str);//ПЕревод string -> double (либо эксепшн)

            return k;
        } catch (std::exception e) {
            std::cout << e.what();
        }
    }
}

///Считывание пределов интегрирования a b
void ReadAB() {

    a = getDoubleValue( mes: "Input double a:\n");
    b = getDoubleValue( mes: "Input double b:\n");

    //В случае некорректного порялка меняем местами границы
    if (a > b) {
        std::swap( &: a, &: b);
    }
}
```

Функция для дочерних потоков

```
74
75  //стартовая функция для дочерних потоков
76  //номер потока, кол-во потоков, сумма для одного потока
77  void integral(int iTread, int iTN, double &sum) {
78      for (double i = a + iTread * STEP; i < b; i += iTN * STEP) {//i+кол-во_потоков* шаг [начинаем №потока*STEP]
79          sum += STEP * (f(i) + f( x: i + STEP)) / 2;//шаг*h//Принцип Дихотомии
80      }
81  }
82
```

I-ая часть main (Получение необходимых данных от пользователя)

```
82
83  //Итеративный параллелизм
84  int main() {
85
86      //ВВод границ
87      ReadAB();
88      //Определение количество потоков
89      int max = static_cast<int>((a + b) / STEP ) + 1;
90      std::string mes = {"Enter the desired number of threads to use in the range [1,"};
91      mes += std::to_string(max) + "]..";
92      int threadNumber = getIntValue(mes, max);
93      std::vector<std::thread> thr;//Вектор потоков
94      thr.reserve(threadNumber);//Кол-во потоков
95      //std::thread *thr=new std::thread [threadNumber];
96      auto *sum = new double[threadNumber];
97
98      std::cout << "Upper bound: " << b << "\nLow bound: " << a << std::endl;
99      std::cout << "Upper bound: " << b << "\nLow bound: " << a << std::endl;
100     std::cout << "Step: " << STEP << std::endl;
101     std::cout << "Number of threads: " << threadNumber << std::endl;
102
```

II-ая часть main (Разбиение на потоки и вычисление интеграла)

```
103
104     auto begin = std::chrono::steady_clock::now();
105     //Создание потоков
106     for (int i = 0; i < threadNumber; i++) {
107         sum[i] = 0;
108         thr.emplace_back(std::thread{integral, i, threadNumber, std::ref( &: sum[i])});//Инициализируем поток[i]
109     }
110
111     double rez = 0; //для результата
112     // Завершение потоков
113     for (int i = 0; i < threadNumber; i++) {
114         thr[i].join();
115         rez += sum[i];//Суммируем суммы посчитанные одним потоком
116     }
117
118     auto end = std::chrono::steady_clock::now();
119     std::cout << std::endl;
120     auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
121     std::cout << "milliseconds: " << elapsed_ms.count() << std::endl;
122     printf( _Format: "Result is: %.3f", rez);
123
124     return 0;
125
126 }
```

Результат работы программы:

при f(x)=x^2:

```
Input double a:
2
Input double b:
4
Enter the desired number of threads to use in the range [1,601]..1
Upper bound: 4
Low bound: 2
Upper bound: 4
Low bound: 2
Step: 0.01
Number of threads: 1

milliseconds: 3
Result is: 18.827
Process finished with exit code 0
```

```
Input double a:
2
Input double b:
4
Enter the desired number of threads to use in the range [1,601]..13
Upper bound: 4
Low bound: 2
Upper bound: 4
Low bound: 2
Step: 0.01
Number of threads: 13

milliseconds: 9
Result is: 18.827
Process finished with exit code 0
```

```
Input double a:
2
Input double b:
4
Enter the desired number of threads to use in the range [1,601]..600
Upper bound: 4
Low bound: 2
Upper bound: 4
Low bound: 2
Step: 0.01
Number of threads: 600

milliseconds: 310
Result is: 18.667
Process finished with exit code 0
```

при f(x)=sin(x):

```
Input double a:
1
Input double b:
2
Enter the desired number of threads to use in the range [1,301]..4
Upper bound: 2
Low bound: 1
Upper bound: 2
Low bound: 1
Step: 0.01
Number of threads: 4

milliseconds: 12
Result is: 0.956
Process finished with exit code 0
```

(Проверка на корректность)

```
Input double a:
www
invalid stod argumentInput double a:
2
Input double b:
3
Enter the desired number of threads to use in the range [1,501]..-3
Incorrect number!Enter the desired number of threads to
 use in the range [1,501]..w
invalid stoi argumentEnter the desired number of threads to use in the range [1,501]..4
Upper b
ound: 3
Low bound: 2
Upper bound: 3
Low bound: 2
Step: 0.01
Number of threads: 4

milliseconds: 6
Result is: 0.574
Process finished with exit code 0
```

P.S. Все результаты были проверены с помощью онлайн калькулятора: https://www.integral-calculator.ru/

Следовательно программа выполняет заданную задачу.