**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук Департамент программной инженерии Дисциплина: «Архитектура вычислительных систем»

**Вычисление определенного интеграла.**

**Вариант 29.**

Пояснительная записка

**Выполнил:**   
Бураков Даниил  
*студент гр. БПИ197.*

**Москва**2020

Содержание

[1. Текст задания 2](#_Toc55056090)

[2. Применяемые расчетные методы 3](#_Toc55056091)

[2.1. Теория решения задания 3](#_Toc55056092)

[2.2. Дополнительный функционал программы 3](#_Toc55056093)

[3. Тестирование программы 4](#_Toc55056094)

[3.1. Корректные значения 4](#_Toc55056095)

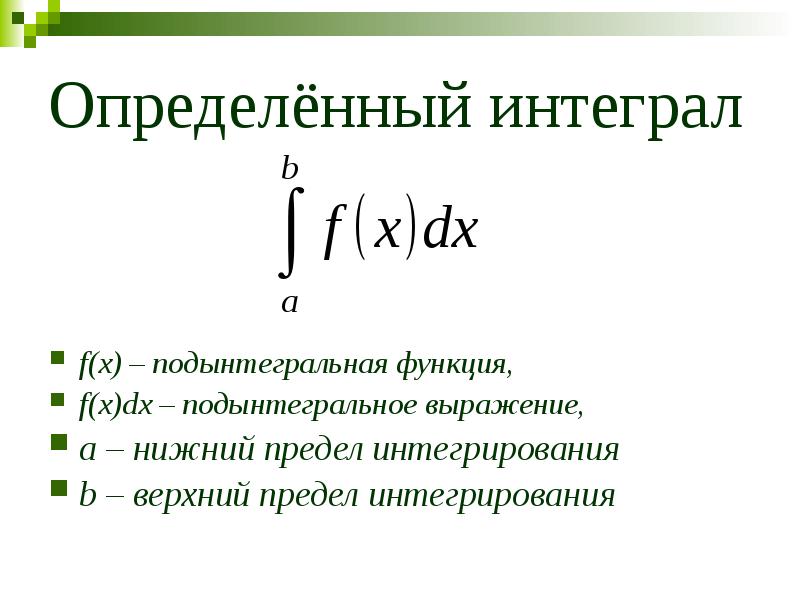
[3.2. Некорректные значения 4](#_Toc55056096)

[ПРИЛОЖЕНИЕ 1.](#_Toc55056097) [Список литературы 5](#_Toc55056098)

[ПРИЛОЖЕНИЕ 2.](#_Toc55056099) [Код программы 6](#_Toc55056100)

1. Текст задания

Вычислить интеграл, 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.



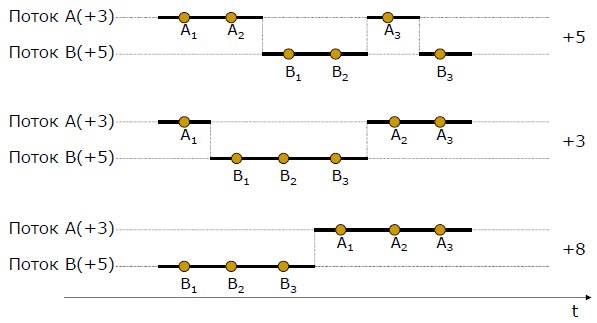
1. Применяемые расчетные методы
   1. Теория решения задания

Для решения данной задачи был использование принцип многопоточного программирования “Производители и потребители”. Мы считаем интеграл с помощью рекурсивной функции Integral. Интервал интегрирования на каждом такте применения этой функции делится на 2 части. К каждой части применяется та же функция, но только уже в другом потоке. Так продолжается, пока длинна разбиения не будет меньше константы эпсилон, которая задается в программе.

В итоге получаем, что на каждой итерации алгоритма создаются два потока-производителя. При обратном ходе рекурсии они передают значения в поток-потребитель. Буфер в данном случае- дополнительная переменная res в параметрах функции.

* 1. Теоретические сведения

Одной из типовых задач, требующих синхронизации, является задача producerconsumer (производитель-потребитель). Пусть два потока обмениваются информацией через буфер ограниченного размера. Производитель добавляет информацию в буфер, а потребитель извлекает ее оттуда.



Два потока, обменивающихся информацией через циклический буфер

Функциональность потоков производителя и потребителя можно записать следующим образом.

Producer:

while(true) {

PrepareData(); // Подготовить данные

Put(Data); // Поместить данные в циклический буфер

}

Consumer:

while(true) {

Get(&Data); // Считать данные из циклического буфера

UseData(); // Использовать данные

}

Сразу необходимо отметить, что если буфер пуст, то потребитель должен ждать, пока в нем появятся данные, а если буфер полон, то производитель должен ждать появления свободного элемента. В данном случае реализация производителя и потребителя будет иметь следующий вид.

Producer:

while(true) {

PrepareData(); // Подготовить данные

while( ! Put(Data) ) // Разместить данные

57

;

}

Consumer:

while(true) {

while( ! Get(&Data) ) // Считать данные

;

UseData(); // Использовать данные

}

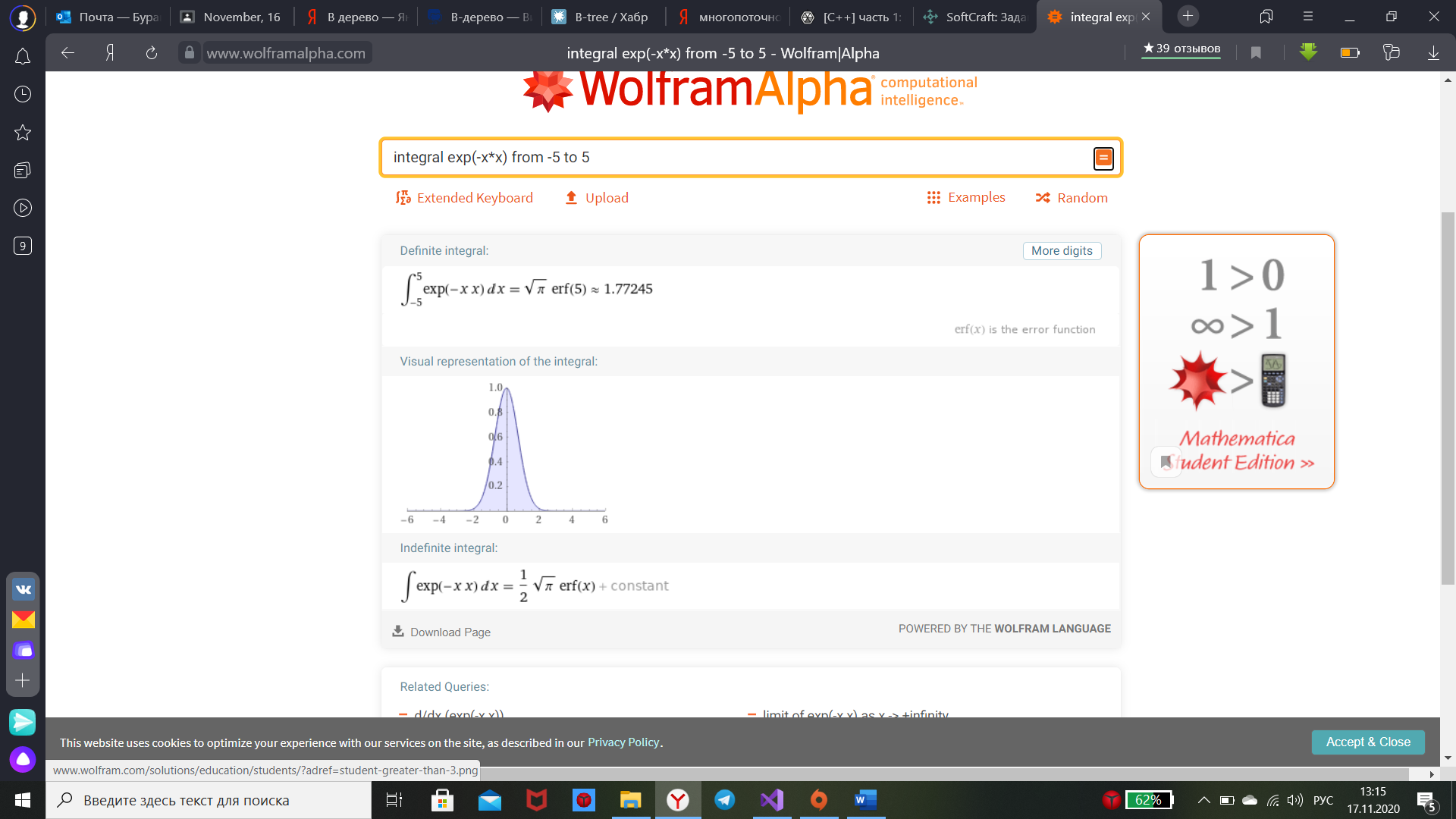
Задача "Производители-Потребители" заключается в обеспечении согласованного доступа нескольких потоков к разделяемому циклическому буферу. Корректное решение должно удовлетворять следующим условиям:

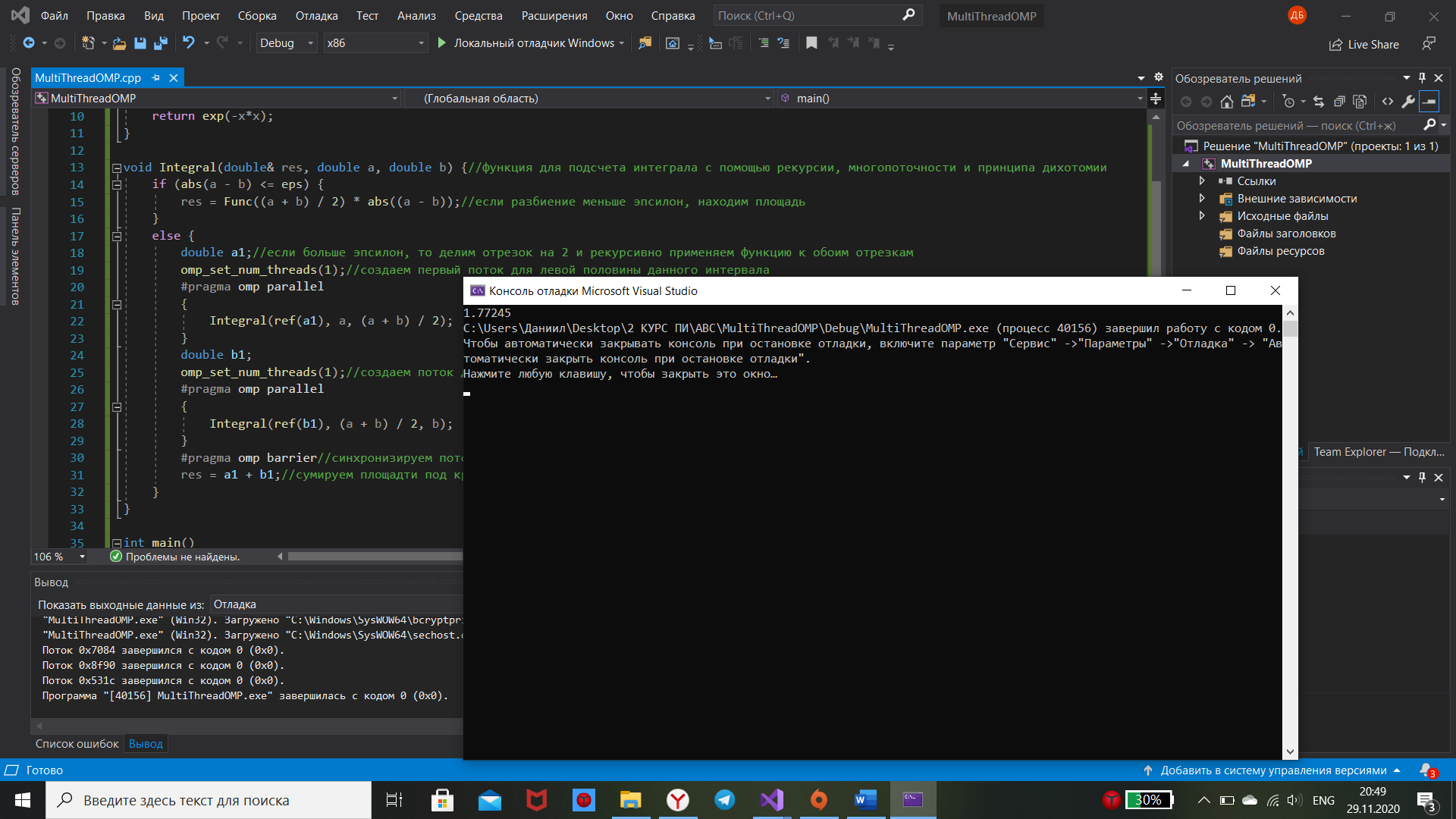
* потоки выполняются параллельно;
* одновременно в критической секции, связанной с каждым критическим ресурсом, должно находиться не более одного потока;
* потоки должны завершить работу в течение конечного времени;
* потоки должны корректно использовать операции с циклическим буфером.

1. Тестирование программы
   1. Корректные значения

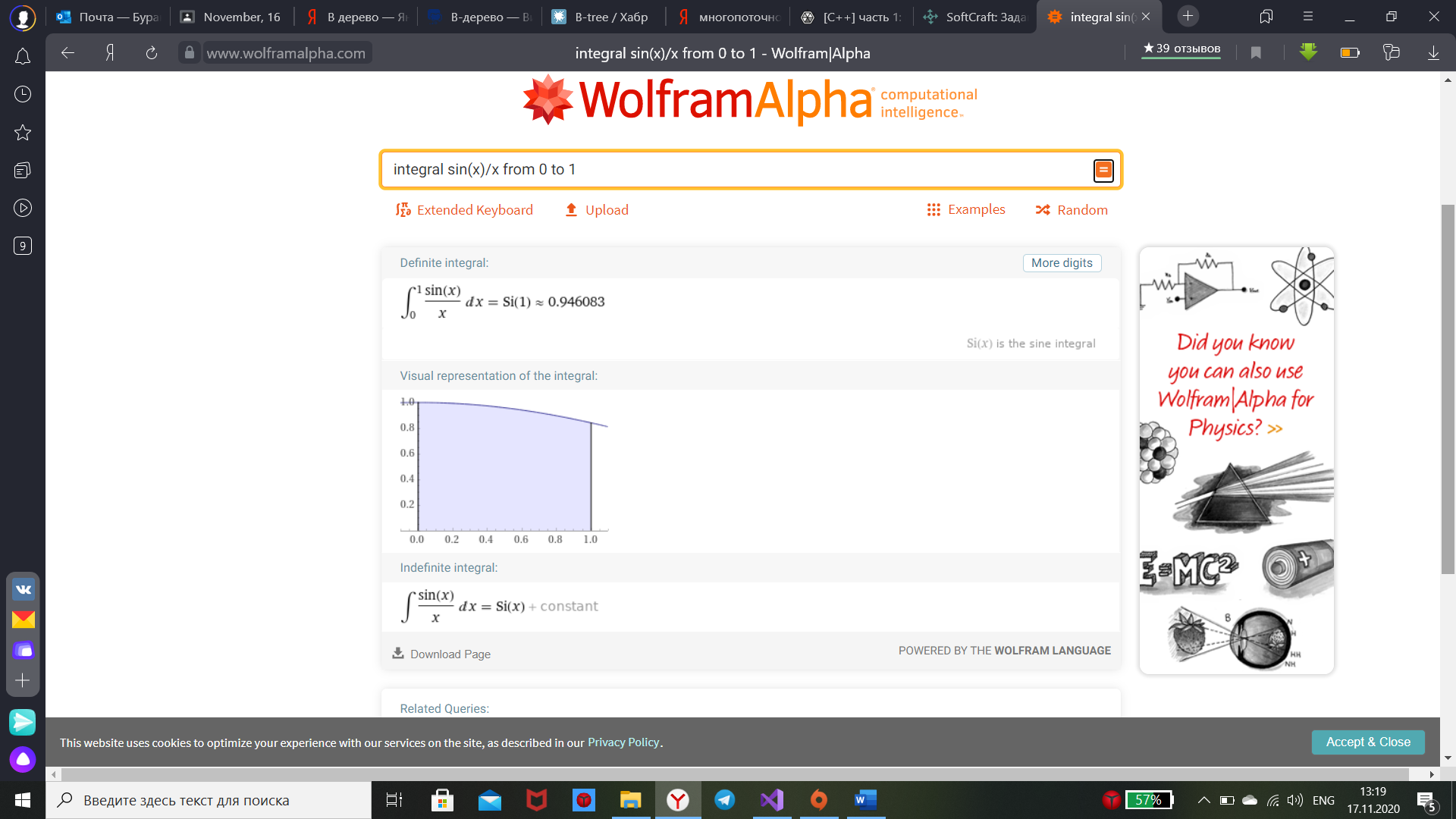
Тесты будут заключаться в том, что мы посчитаем интеграл через **Вольфрам математика** и проверим с помощью программы.

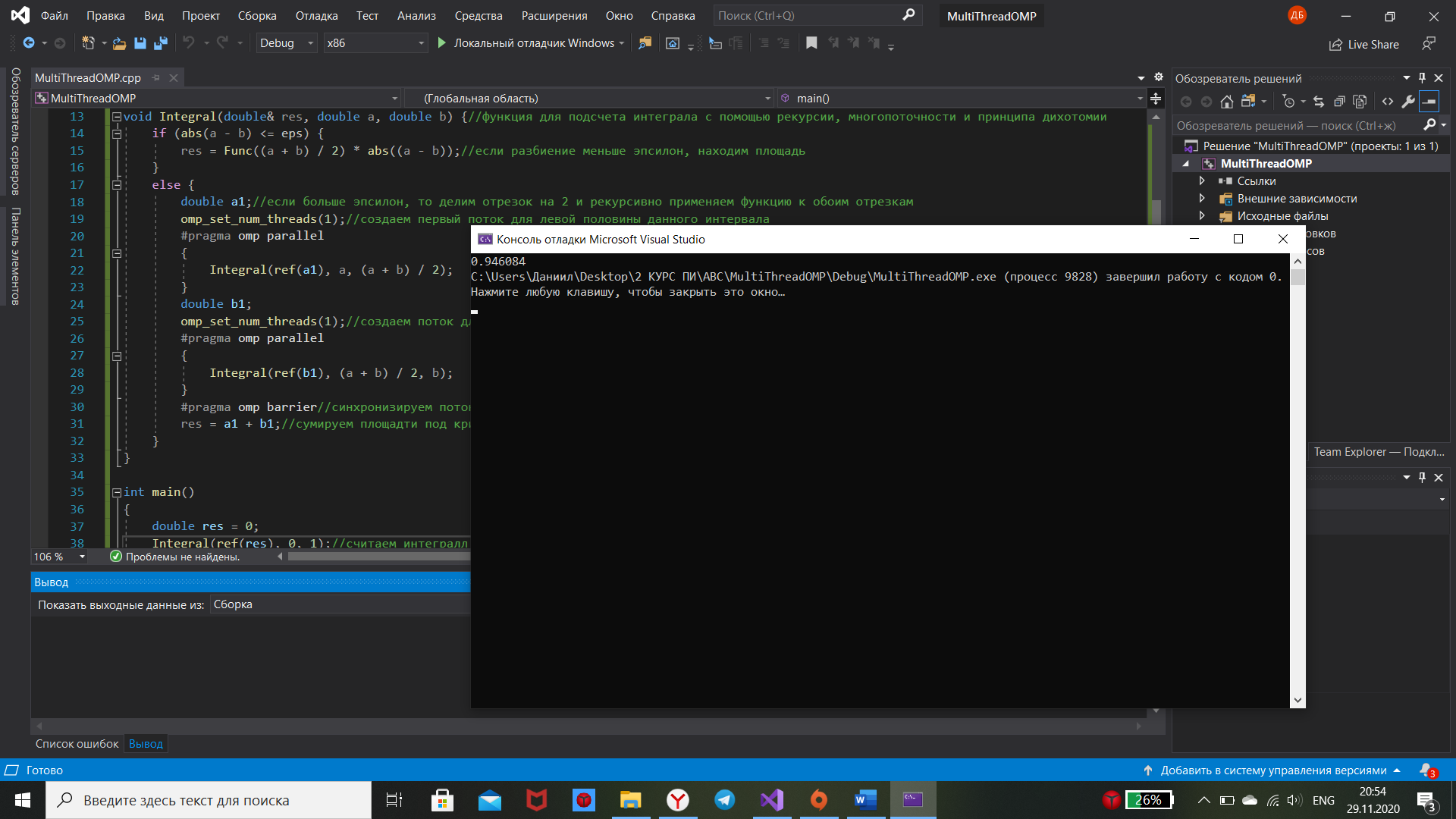
Тест 1.



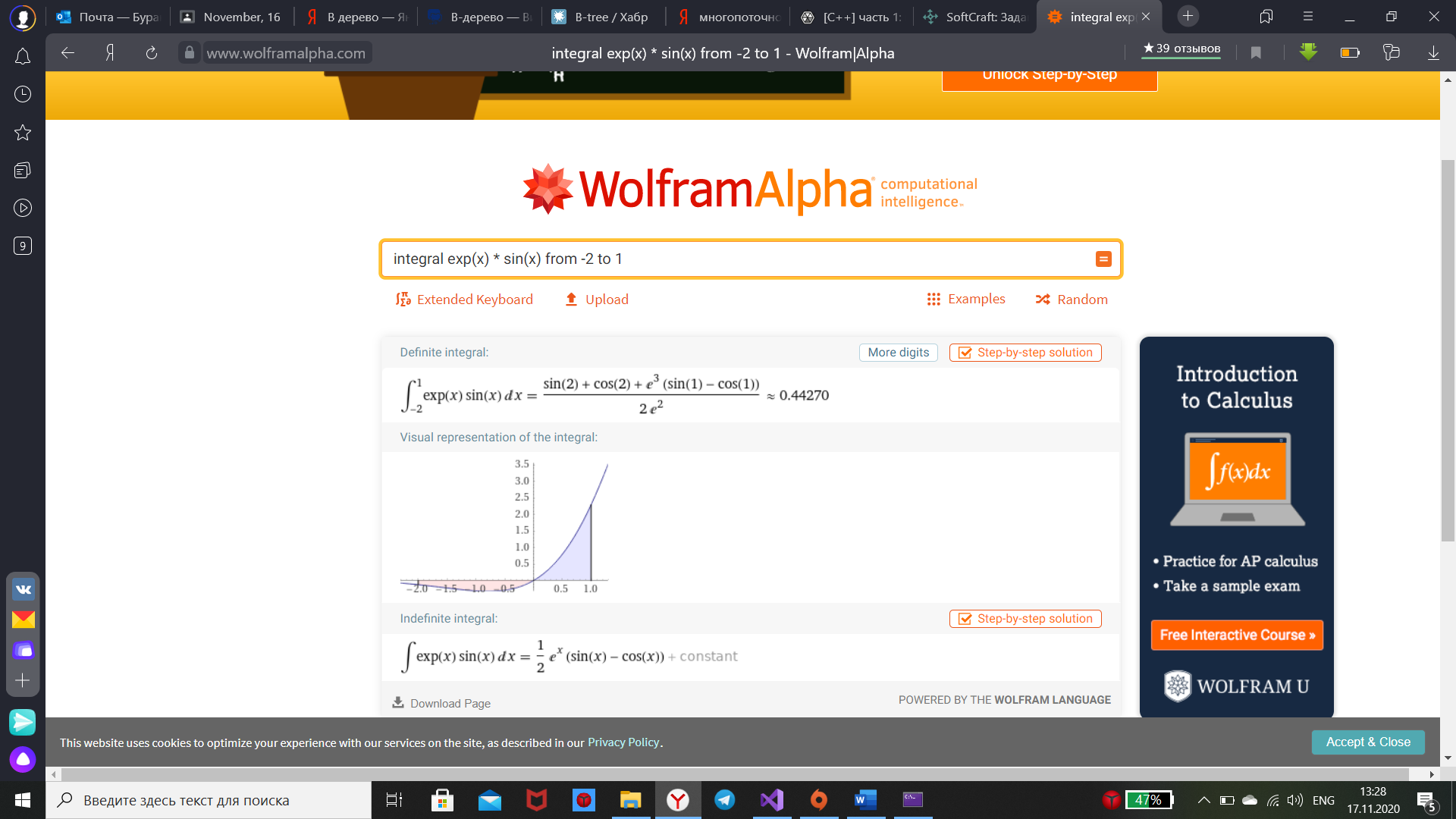


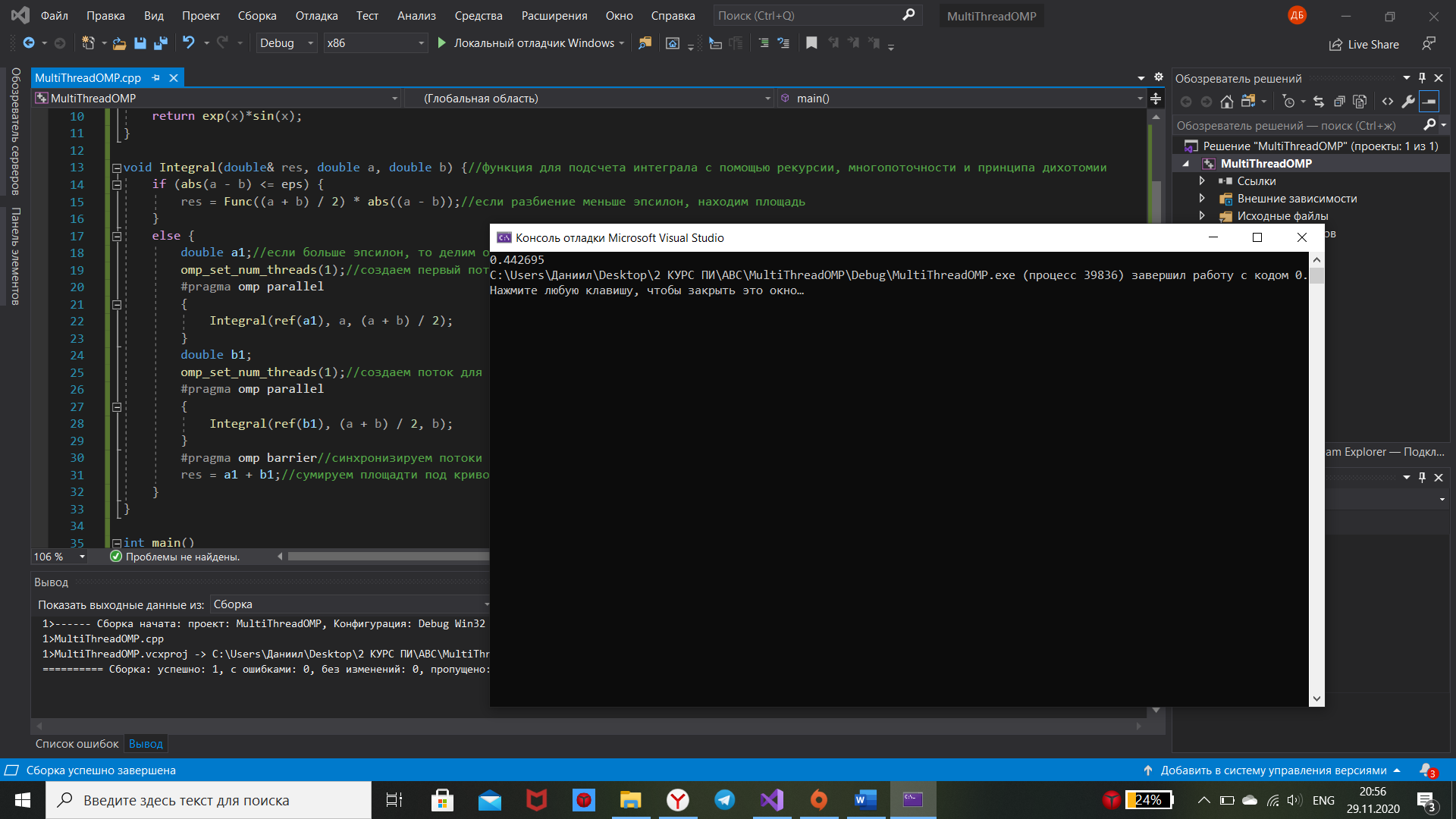
Тест 1 Верен.





Тест 2 Верен.

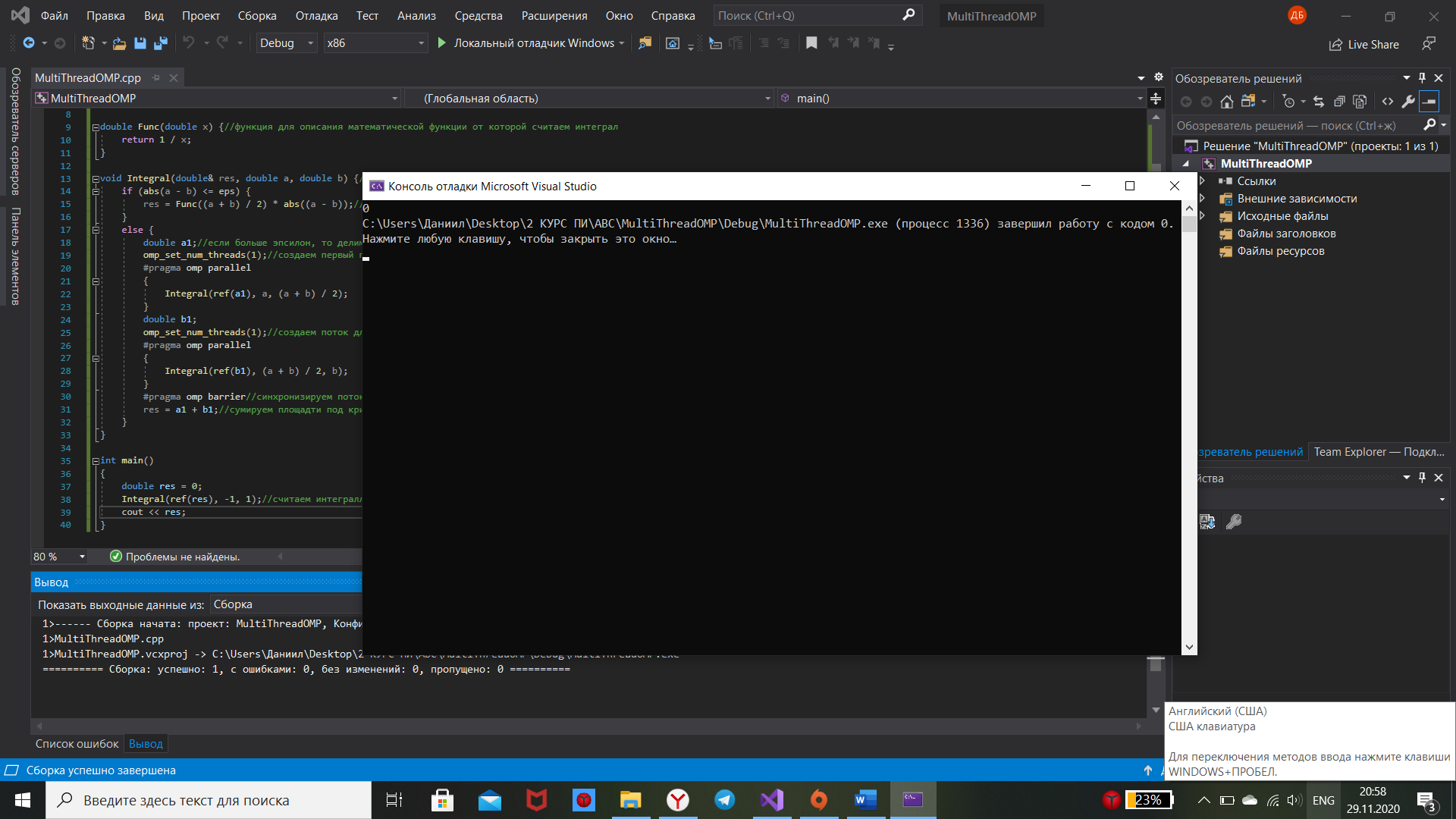




Тест 3 Верен.

* 1. Некорректные значения

При вводе функций, которые на данном интервале не дифференцируемые программа выводит некорректные значения.



Так при вводе функции 1/x на интервале от -1 до 1 программа выдаст 0. Хотя этот интеграл неисчислим.

Также при вводе большого интервала глубина рекурсии будет слишком большой и из-за такого количества потоков программа вылетает.

ПРИЛОЖЕНИЕ 1

Список литературы

1. Операционные системы – аспекты параллелизма. https://intuit.ru/studies/courses/4447/983/lecture/14923?page=4

ПРИЛОЖЕНИЕ 2

Код программы

#include <iostream>

#include <math.h>

#include <omp.h>

using namespace std;

const double eps = 0.01;//константа эпсилон - диаметр разбиения данного интеграла

double Func(double x) {//функция для описания математической функции от которой считаем интеграл

return 1 / x;

}

void Integral(double& res, double a, double b) {//функция для подсчета интеграла с помощью рекурсии, многопоточности и принципа дихотомии

if (abs(a - b) <= eps) {

res = Func((a + b) / 2) \* abs((a - b));//если разбиение меньше эпсилон, находим площадь

}

else {

double a1;//если больше эпсилон, то делим отрезок на 2 и рекурсивно применяем функцию к обоим отрезкам

omp\_set\_num\_threads(1);//создаем первый поток для левой половины данного интервала

#pragma omp parallel

{

Integral(ref(a1), a, (a + b) / 2);

}

double b1;

omp\_set\_num\_threads(1);//создаем поток для левой половины интервала

#pragma omp parallel

{

Integral(ref(b1), (a + b) / 2, b);

}

#pragma omp barrier//синхронизируем потоки

res = a1 + b1;//сумируем площадти под кривой на 2 отрезках

}

}

int main()

{

double res = 0;

Integral(ref(res), -1, 1);//считаем интегралл от функции и выводим значение

cout << res;

}