Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Решение инженерных задач на основе циклических программ»

Выполнил:

Студент 1 курса 9 группы

Павлович Ян Андреевич

Преподаватель: асс. Андронова М.В.

2023, Минск

5. В соответствии со своим вариантом написать программы для вычисления площади криволинейной трапеции по исходным данным из таблицы, приведенной ниже, методом ***трапеций*** и методом ***парабол***. Для всех вариантов принять **n** = 200. Сравнить результаты, которые должны отличаться на небольшую величину.

Снимок экрана 2023-10-31 в 16.43.17

#include <iostream>

//подключаем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе, что мы используем пространство имен std

**int** main() {

setlocale(LC\_CTYPE, "Russian"); //добавляем возможность вывода на русском языке

**double** h, x, a = 1.00, b = 3.00, n = 200.00, s = 0, sum1 = 0.00, sum2 = 0.00, s2; //объявляем переменные

x = a;

h = (b - a) / n; //вычисляем высоту трапеции (длина шага программы)

cout << "Шаг: " << h << endl;

**for** (**int** i = 0; i <= n; i++) {

    //добавляем оператор for для вычисления площади всей фигуры путем сложения трапеций с высотой h

s = s + h \* (((sin(x) + 1) + (sin(x + h) + 1)) / 2);

x += h;

}

cout << "Интеграл методом трапеции равен: " << s << endl; //выводим результат

x = a; //присваиваем x исходное значение

**for** (**int** j = 2; j < n; j += 2) {

    //добавляем оператор for для вычисления суммы четных значений x

x += j \* h;

sum1 += sin(x) + 1;

}

sum1 \*= 2; //умножаем сумму на два исходя из формулы

x = a; //присваиваем x исходное значение

**for** (**int** d = 1; d < n; d += 2) {

    //добавляем оператор for для вычисления суммы нечетных значений x

x += d \* h;

sum2 += sin(x) + 1;

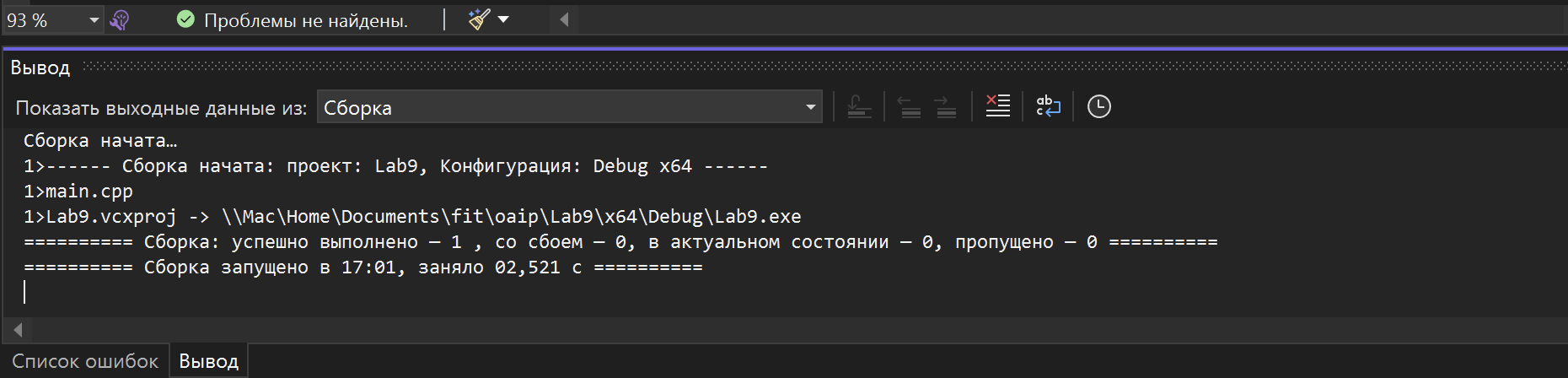
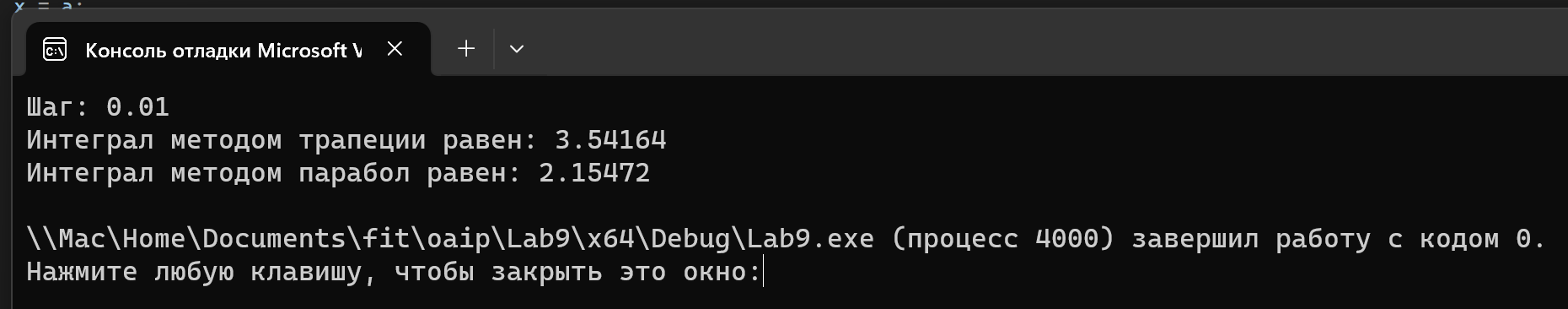
}

sum2 \*= 4; //умножаем сумму на четыре исходя из формулы

s2 = h / 3 \* ((sin(0) + 1) + (sin(n) + 1) + sum1 + sum2); //вычисляем площадь методом парабол

cout << "Интеграл методом парабол равен: " << s2 << endl; //выводим результат

}



В соответствии со своим вариантом найти отрезок (значения **a** и **b**), который содержит один корень, ***отделив корни*** уравнения ***графическим*** методом для исходных данных из таблицы, приведенной ниже. Если корней несколько, то выбрать один из отрезков. Написать программу вычисления корня уравнения методом ***дихотомии***. Точность вычислений принять равной **e** = 0,0001 для всех вариантов. Найти корень уравнения с помощью приложения Excel. Сравнить результаты.

Снимок экрана 2023-10-31 в 17.05.00

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**double** f(**double** x) {

    //добавляем в функцию нашу функцию, из которой находим корни уравнения корень

**return** 2 \* x + pow(x, 3) - 7;

}

**double** function(**double** a, **double** b, **double** Eps) {

//добавляем функцию для нахождения корня методом дихотомии

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность работы с русским языком

**double** c, d;

**while** (b - a >= 2 \* Eps) { //пока длина интервала больше удвоенной погрешности

        c = (a + b) / 2; //находим середину интервала

**if** (f(c) == 0.0) { //если значение функции в середине интервала равно 0, то возвращаем эту точку

**return** c;

        } **else** **if** (f(a) \* f(c) < 0) { //если знаки функции в начале и середине интервала разные, то корень находится в левой половине интервала

            b = c;

        } **else {** //иначе корень находится в правой половине интервала

            a = c;

}

    }

    cout << "Значение a: " << a << endl << "Значение b: " << b << endl;

**return** (a + b) / 2; //возвращаем середину интервала

}

**int** main() {

**double** a = -3.0, b = 3.0, Eps = 0.0001; //объявляем начало/конец интервала и погрешность

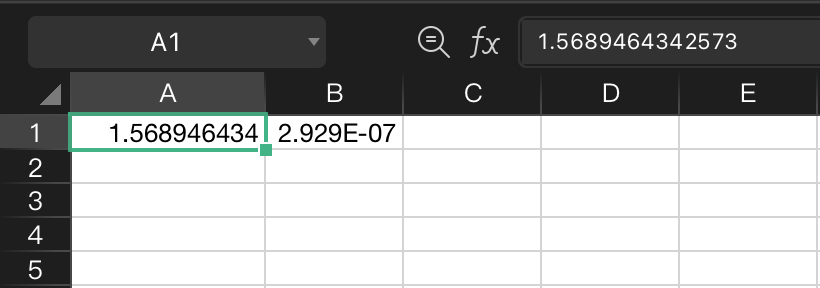
**double** root = function(a, b, Eps); //находим корень вызывая функцию со значениями

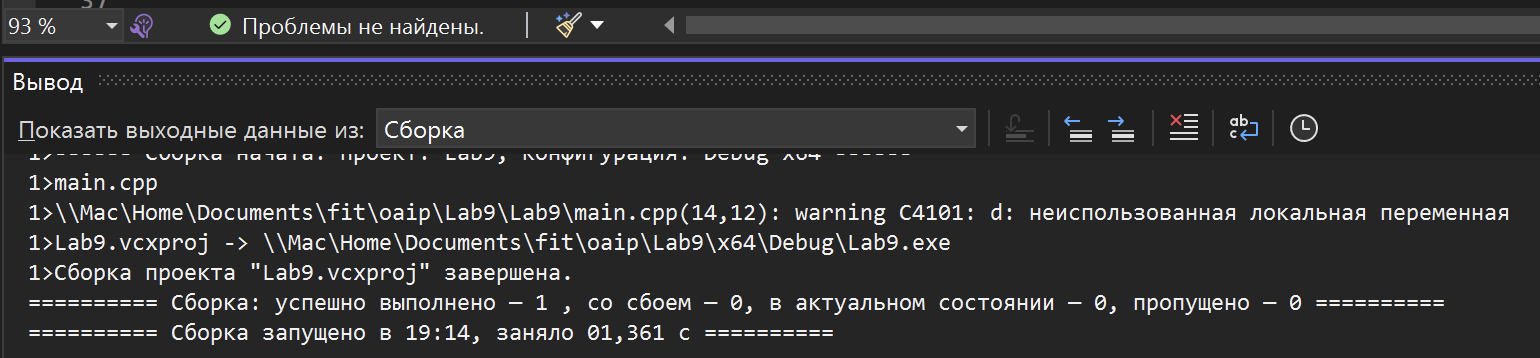
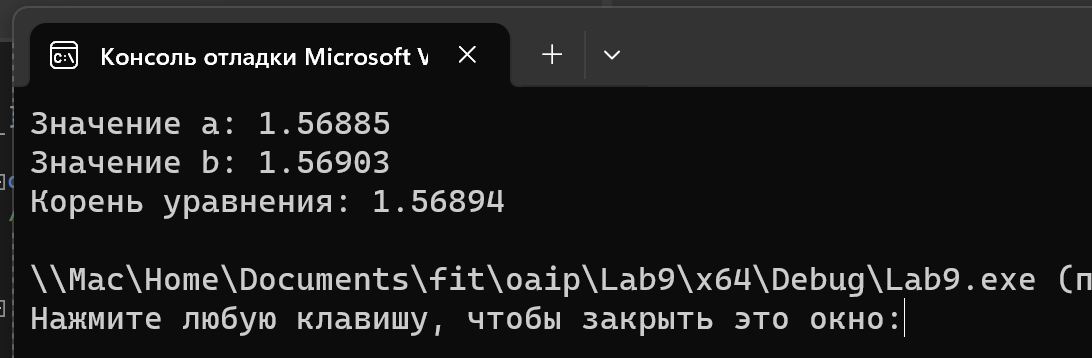
    cout << "Корень уравнения: " << root << endl; //выводим корень

**return** 0;

}

Результаты Exel и программы совпали.





**Дополнительные задания**

5. В соответствии со своим вариантом написать программы для вычисления площади криволинейной трапеции по исходным данным из таблицы, приведенной ниже, методом ***трапеций*** и методом ***парабол***. Для всех вариантов принять **n** = 200. Сравнить результаты, которые должны отличаться на небольшую величину.

Снимок экрана 2023-11-01 в 21.07.08

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность вывода на русском языке

**double** h, x, a = 4.00, b = 7.00, n = 200.00, s = 0, sum1 = 0.00, sum2 = 0.00, s2; //объявляем переменные

    x = a;

    h = (b - a) / n; //вычисляем высоту трапеции (длина шага программы)

    cout << "Шаг: " << h << endl;

**for** (**int** i = 0; i <= n; i++) {

        //добавляем оператор for для вычисления площади всей фигуры путем сложения трапеций с высотой h

        s = s + h \* ((pow(cos(x), 3) + pow(cos(x + h), 3)) / 2);

        x += h;

    }

    cout << "Интеграл методом трапеции равен: " << s << endl;

    //выводим результат

    x = a; //присваиваем x исходное значение

**for** (**int** j = 2; j < n; j += 2) {

        //добавляем оператор for для вычисления суммы четных значений x

        x += j \* h;

        sum1 += pow(cos(x), 3);

    }

    sum1 \*= 2; //умножаем сумму на два исходя из формулы

    x = a; //присваиваем x исходное значение

**for** (**int** d = 1; d < n; d += 2) {

        //добавляем оператор for для вычисления суммы нечетных значений x

        x += d \* h;

        sum2 += pow(cos(x), 3);

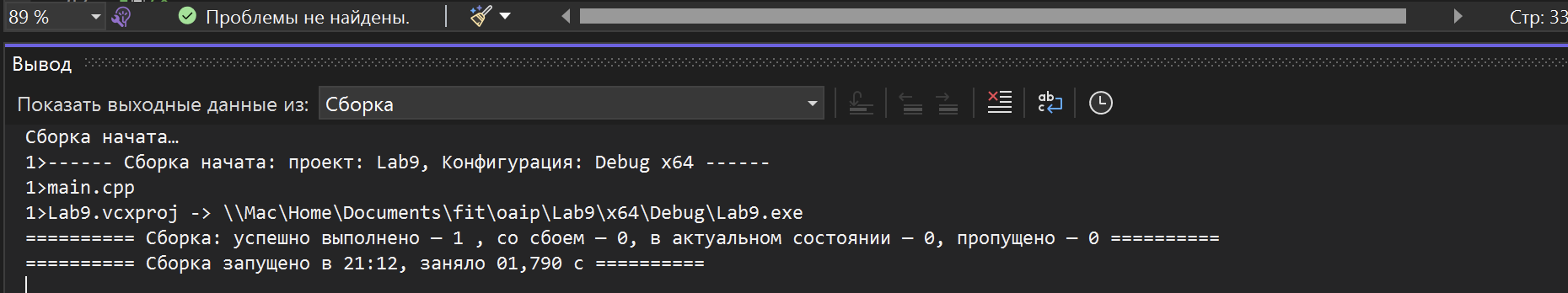
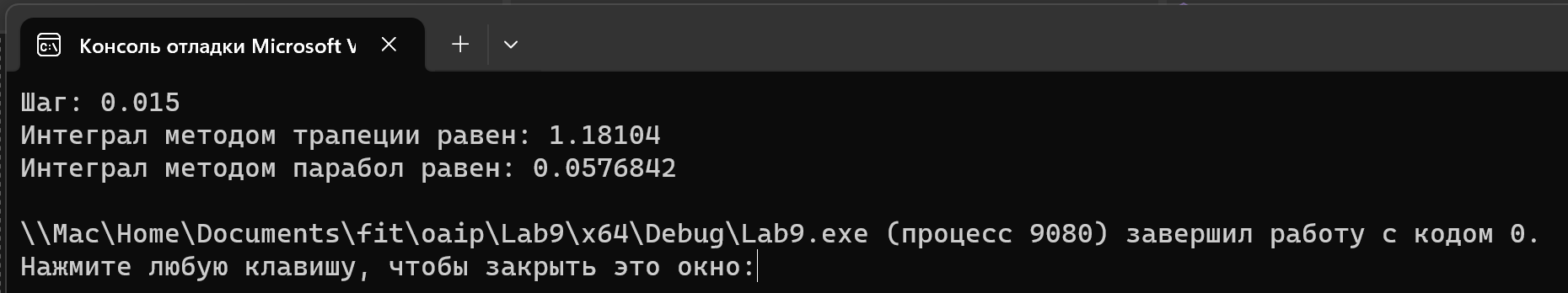
    }

    sum2 \*= 4;//умножаем сумму на четыре исходя из формулы

    s2 = h / 3 \* ((pow(cos(0), 3)) + (pow(cos(n), 3)) + sum1 + sum2);//вычисляем площадь методом парабол

    cout << "Интеграл методом парабол равен: " << s2 << endl; //выводим результат

}



6. В соответствии со своим вариантом найти отрезок (значения **a** и **b**), который содержит один корень, ***отделив корни*** уравнения ***графическим*** методом для исходных данных из таблицы, приведенной ниже. Если корней несколько, то выбрать один из отрезков. Написать программу вычисления корня уравнения методом ***дихотомии***. Точность вычислений принять равной **e** = 0,0001 для всех вариантов. Найти корень уравнения с помощью приложения Excel. Сравнить результаты.

Снимок экрана 2023-11-01 в 21.27.07

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**double** f(**double** x) {

    //добавляем в функцию нашу функцию, из которой находим корни уравнения корень

**return** cos(x) + x - 7;

}

**double** function(**double** a, **double** b, **double** Eps) {

//добавляем функцию для нахождения корня методом дихотомии

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность работы с русским языком

**double** c, d;

**while** (b - a >= 2 \* Eps) { //пока длина интервала больше удвоенной погрешности

        c = (a + b) / 2; //находим середину интервала

**if** (f(c) == 0.0) { //если значение функции в середине интервала равно 0, то возвращаем эту точку

**return** c;

        } **else** **if** (f(a) \* f(c) < 0) { //если знаки функции в начале и середине интервала разные, то корень находится в левой половине интервала

            b = c;

        } **else {** //иначе корень находится в правой половине интервала

            a = c;

}

    }

    cout << "Значение a: " << a << endl << "Значение b: " << b << endl;

**return** (a + b) / 2; //возвращаем середину интервала

}

**int** main() {

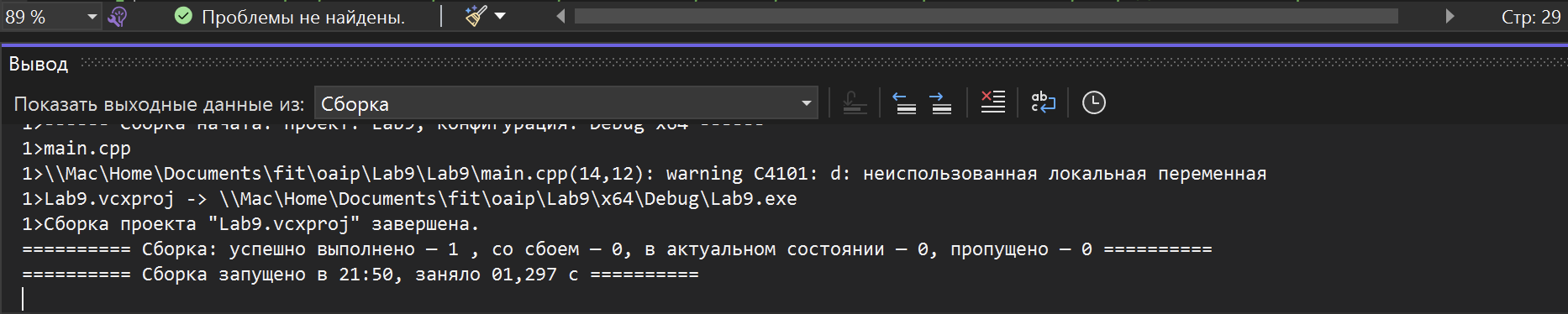
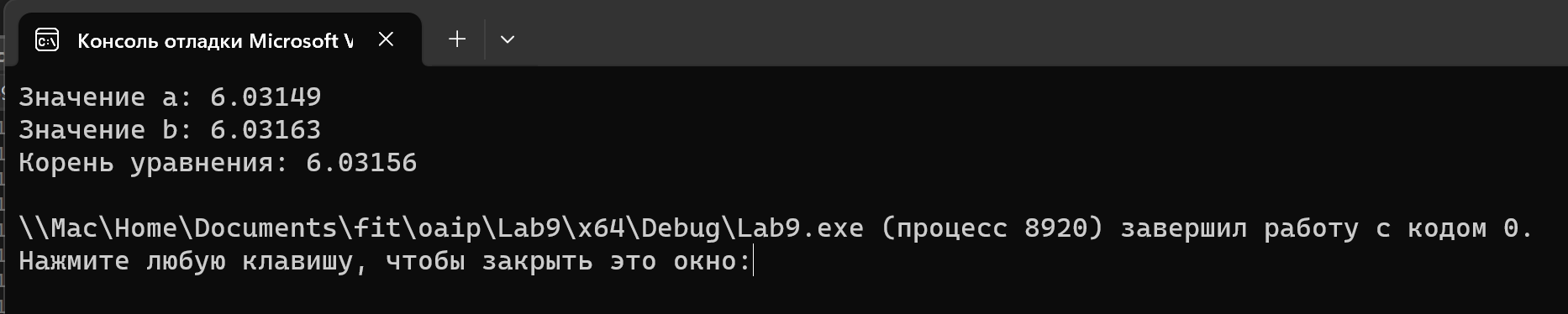
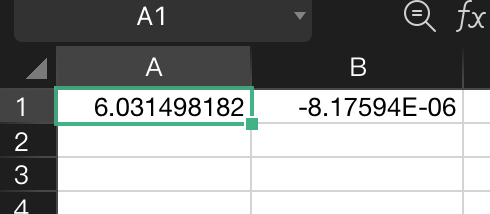
**double** a = -9.0, b = 9.0, Eps = 0.0001; //объявляем начало/конец интервала и погрешность

**double** root = function(a, b, Eps); //находим корень вызывая функцию со значениями

    cout << "Корень уравнения: " << root << endl; //выводим корень

**return** 0;

}



1. В соответствии со своим вариантом написать программы для вычисления площади криволинейной трапеции по исходным данным из таблицы, приведенной ниже, методом ***трапеций*** и методом ***парабол***. Для всех вариантов принять **n** = 200. Сравнить результаты, которые должны отличаться на небольшую величину.

Снимок экрана 2023-11-01 в 21.15.48

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность вывода на русском языке

**double** h, x, a = 1.00, b = 5.00, n = 200.00, s = 0, sum1 = 0.00, sum2 = 0.00, s2; //объявляем переменные

    x = a;

    h = (b - a) / n; //вычисляем высоту трапеции (длина шага программы)

    cout << "Шаг: " << h << endl;

**for** (**int** i = 0; i <= n; i++) {

        //добавляем оператор for для вычисления площади всей фигуры путем сложения трапеций с высотой h

        s = s + h \* (((1 + pow(x, 3)) + 1 + pow(x + h, 3)) / 2);

        x += h;

    }

    cout << "Интеграл методом трапеции равен: " << s << endl;

    //выводим результат

    x = a; //присваиваем x исходное значение

**for** (**int** j = 2; j < n; j += 2) {

        //добавляем оператор for для вычисления суммы четных значений x

        x += j \* h;

        sum1 += 1 + pow(x, 3);

    }

    sum1 \*= 2; //умножаем сумму на два исходя из формулы

    x = a; //присваиваем x исходное значение

**for** (**int** d = 1; d < n; d += 2) {

        //добавляем оператор for для вычисления суммы нечетных значений x

        x += d \* h;

        sum2 += 1 + pow(x, 3);

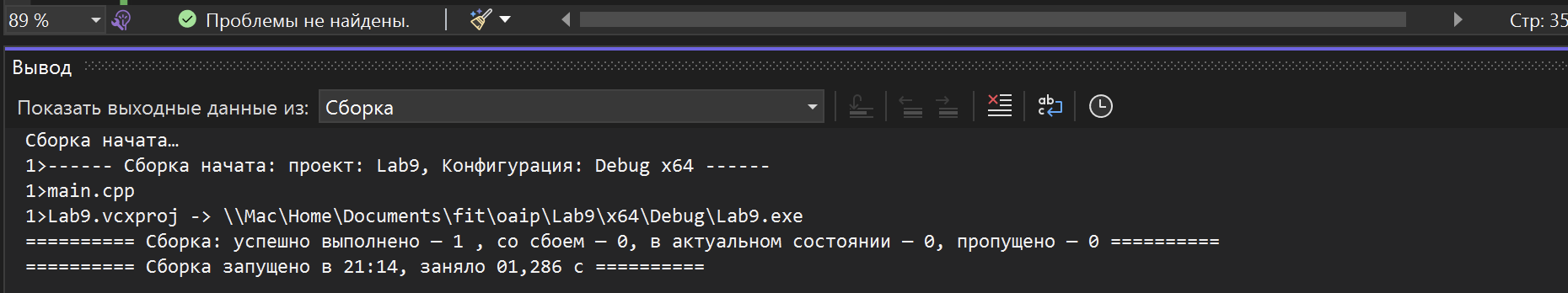
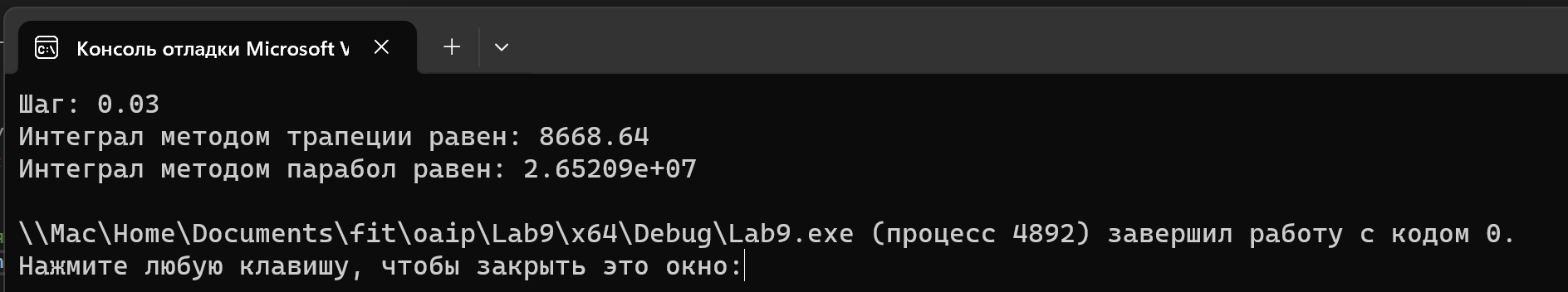
    }

    sum2 \*= 4;//умножаем сумму на четыре исходя из формулы

    s2 = h / 3 \* ((1 + pow(0, 3)) + (1 + pow(n, 3)) + sum1 + sum2);//вычисляем площадь методом парабол

    cout << "Интеграл методом парабол равен: " << s2 << endl; //выводим результат

}



6. В соответствии со своим вариантом найти отрезок (значения **a** и **b**), который содержит один корень, ***отделив корни*** уравнения ***графическим*** методом для исходных данных из таблицы, приведенной ниже. Если корней несколько, то выбрать один из отрезков. Написать программу вычисления корня уравнения методом ***дихотомии***. Точность вычислений принять равной **e** = 0,0001 для всех вариантов. Найти корень уравнения с помощью приложения Excel. Сравнить результаты.

Снимок экрана 2023-11-01 в 21.27.27

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**double** f(**double** x) {

    //добавляем в функцию нашу функцию, из которой находим корни уравнения корень

**return** 5 \* x - 1 + pow(x, 3);

}

**double** function(**double** a, **double** b, **double** Eps) {

//добавляем функцию для нахождения корня методом дихотомии

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность работы с русским языком

**double** c, d;

**while** (b - a >= 2 \* Eps) { //пока длина интервала больше удвоенной погрешности

        c = (a + b) / 2; //находим середину интервала

**if** (f(c) == 0.0) { //если значение функции в середине интервала равно 0, то возвращаем эту точку

**return** c;

        } **else** **if** (f(a) \* f(c) < 0) { //если знаки функции в начале и середине интервала разные, то корень находится в левой половине интервала

            b = c;

        } **else {** //иначе корень находится в правой половине интервала

            a = c;

}

    }

    cout << "Значение a: " << a << endl << "Значение b: " << b << endl;

**return** (a + b) / 2; //возвращаем середину интервала

}

**int** main() {

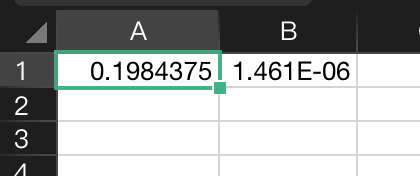
**double** a = -3.0, b = 3.0, Eps = 0.0001; //объявляем начало/конец интервала и погрешность

**double** root = function(a, b, Eps); //находим корень вызывая функцию со значениями

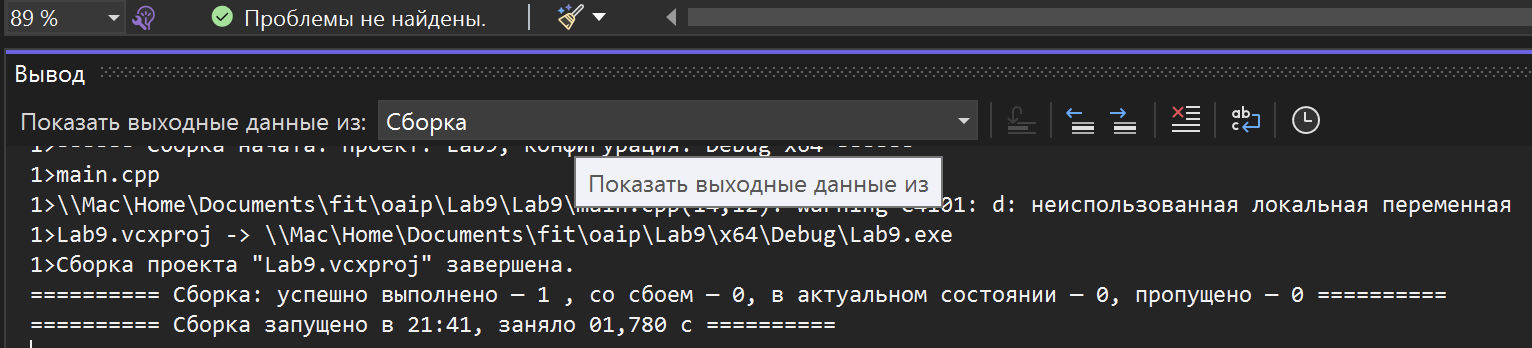
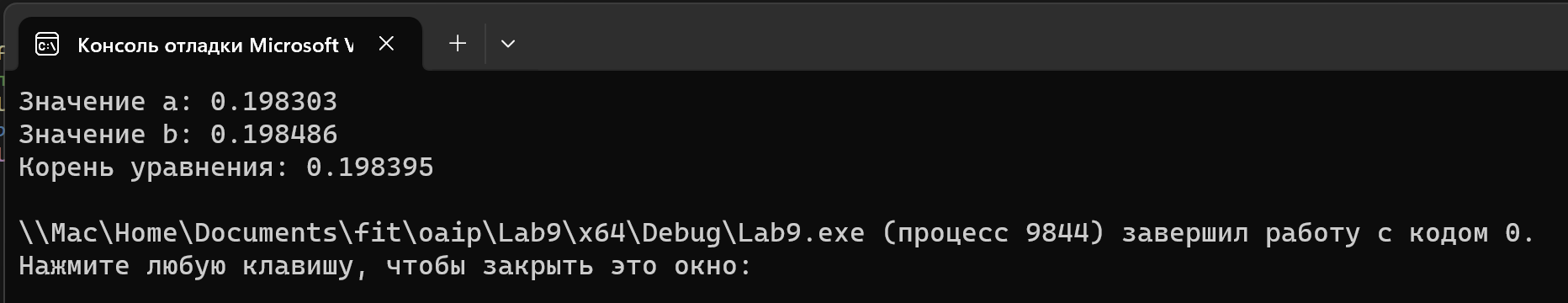
    cout << "Корень уравнения: " << root << endl; //выводим корень

**return** 0;

}



Результаты сходятся с погрешностью 0.0001



5. В соответствии со своим вариантом написать программы для вычисления площади криволинейной трапеции по исходным данным из таблицы, приведенной ниже, методом ***трапеций*** и методом ***парабол***. Для всех вариантов принять **n** = 200. Сравнить результаты, которые должны отличаться на небольшую величину.

Снимок экрана 2023-11-01 в 21.23.21

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**int** main() {

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность вывода на русском языке

**double** h, x, a = 8.00, b = 14.00, n = 200.00, s = 0, sum1 = 0.00, sum2 = 0.00, s2; //объявляем переменные

    x = a;

    h = (b - a) / n; //вычисляем высоту трапеции (длина шага программы)

    cout << "Шаг: " << h << endl;

**for** (**int** i = 0; i <= n; i++) {

        //добавляем оператор for для вычисления площади всей фигуры путем сложения трапеций с высотой h

        s = s + h \* (((2 + pow(x, 3)) + 2 + pow(x + h, 3)) / 2);

        x += h;

    }

    cout << "Интеграл методом трапеции равен: " << s << endl;

    //выводим результат

    x = a; //присваиваем x исходное значение

**for** (**int** j = 2; j < n; j += 2) {

        //добавляем оператор for для вычисления суммы четных значений x

        x += j \* h;

        sum1 += 2 + pow(x, 3);

    }

    sum1 \*= 2; //умножаем сумму на два исходя из формулы

    x = a; //присваиваем x исходное значение

**for** (**int** d = 1; d < n; d += 2) {

        //добавляем оператор for для вычисления суммы нечетных значений x

        x += d \* h;

        sum2 += 2 + pow(x, 3);

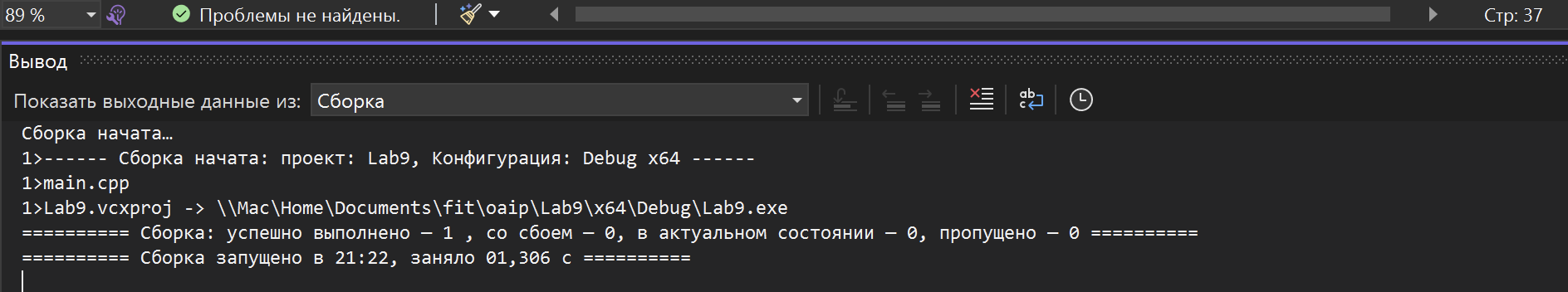
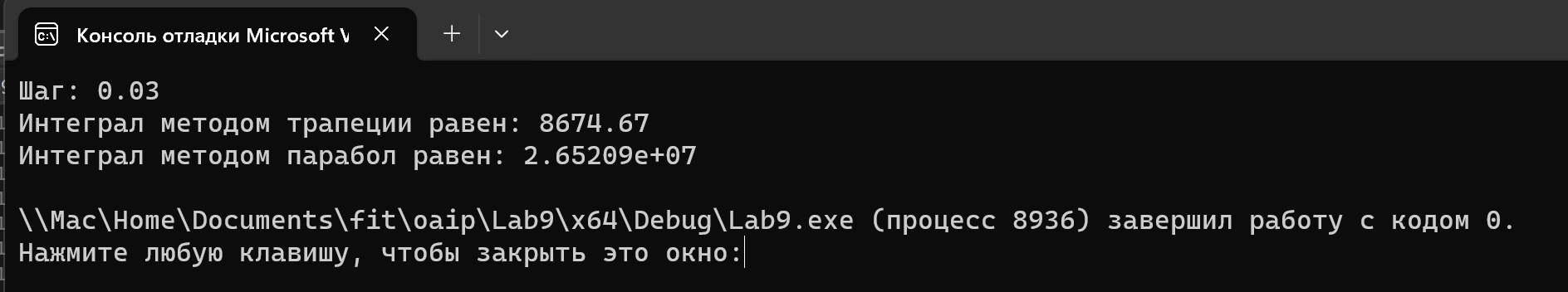
    }

    sum2 \*= 4;//умножаем сумму на четыре исходя из формулы

    s2 = h / 3 \* ((2 + pow(0, 3)) + (2 + pow(n, 3)) + sum1 + sum2);//вычисляем площадь методом парабол

    cout << "Интеграл методом парабол равен: " << s2 << endl; //выводим результат

}



6. В соответствии со своим вариантом найти отрезок (значения **a** и **b**), который содержит один корень, ***отделив корни*** уравнения ***графическим*** методом для исходных данных из таблицы, приведенной ниже. Если корней несколько, то выбрать один из отрезков. Написать программу вычисления корня уравнения методом ***дихотомии***. Точность вычислений принять равной **e** = 0,0001 для всех вариантов. Найти корень уравнения с помощью приложения Excel. Сравнить результаты.

Снимок экрана 2023-11-01 в 21.27.42

#include <iostream>

//добавляем библиотеку для работы с вводом/выводом

**using** **namespace** std;

//объявляем программе: что мы работаем с пространством имен std

**double** f(**double** x) {

    //добавляем в функцию нашу функцию, из которой находим корни уравнения корень

**return** exp(x) + x - 4;

}

**double** function(**double** a, **double** b, **double** Eps) {

//добавляем функцию для нахождения корня методом дихотомии

    setlocale(LC\_CTYPE, "Russian"); //добавляем возможность работы с русским языком

**double** c, d;

**while** (b - a >= 2 \* Eps) { //пока длина интервала больше удвоенной погрешности

        c = (a + b) / 2; //находим середину интервала

**if** (f(c) == 0.0) { //если значение функции в середине интервала равно 0, то возвращаем эту точку

**return** c;

        } **else** **if** (f(a) \* f(c) < 0) { //если знаки функции в начале и середине интервала разные, то корень находится в левой половине интервала

            b = c;

        } **else {** //иначе корень находится в правой половине интервала

            a = c;

}

    }

    cout << "Значение a: " << a << endl << "Значение b: " << b << endl;

**return** (a + b) / 2; //возвращаем середину интервала

}

**int** main() {

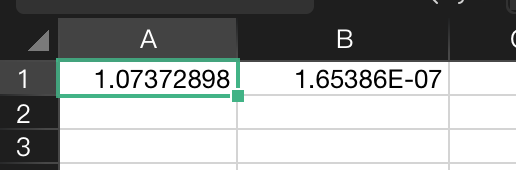
**double** a = -3.0, b = 3.0, Eps = 0.0001; //объявляем начало/конец интервала и погрешность

**double** root = function(a, b, Eps); //находим корень вызывая функцию со значениями

    cout << "Корень уравнения: " << root << endl; //выводим корень

**return** 0;

}



Результаты сходятся с погрешностью 0.0001

