**Курманова Амира БПИ223**

**Вариант 32**

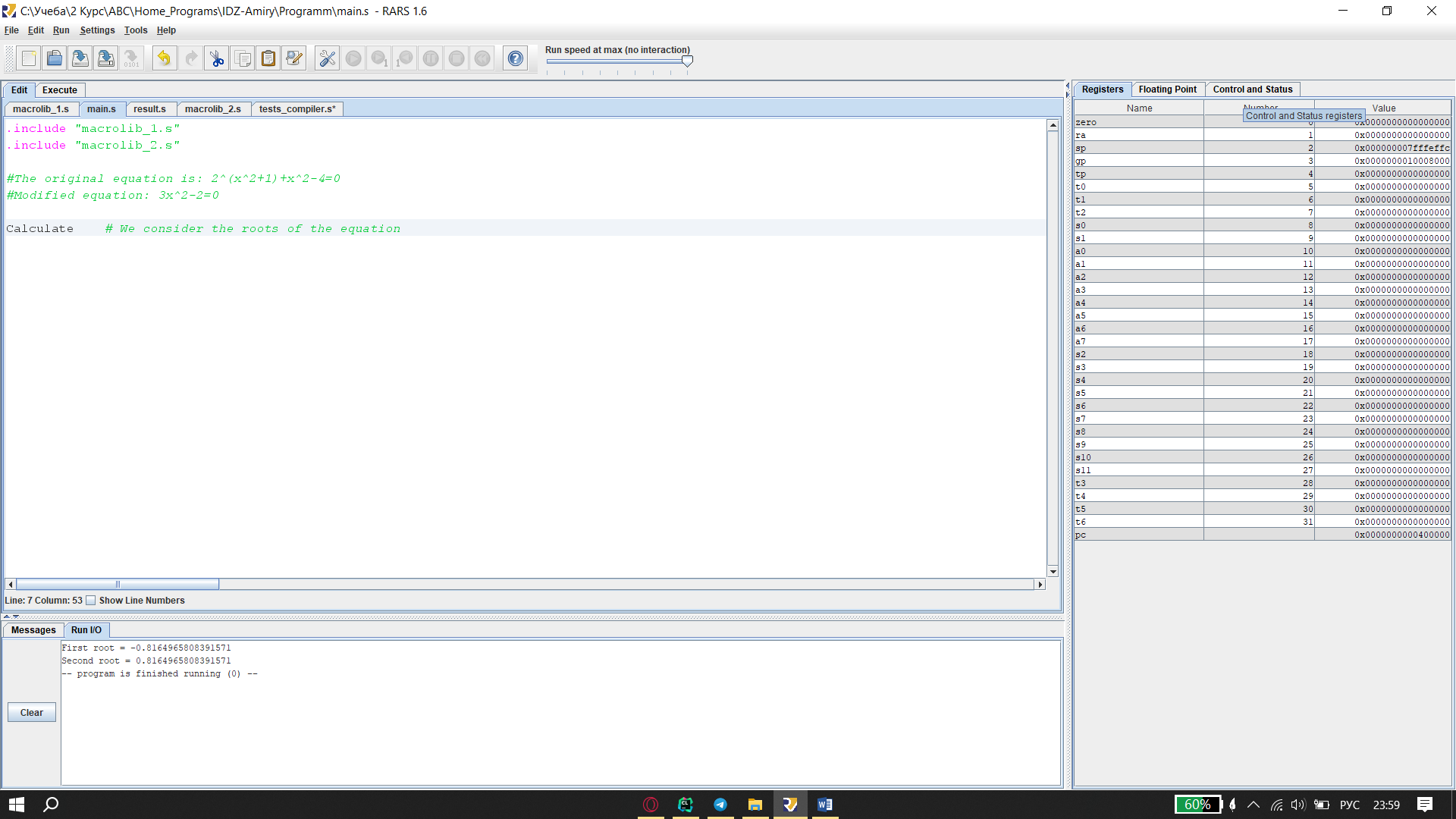
**Условие:**

Разработать программу, определяющую корень уравнения (модифицированное уравнение) методом половинного деления с точностью от 0,001 до 0,00000001 в диапазоне [-1;1] (модифицированный диапазон). Если диапазон некорректен, то подобрать корректный диапазон.

**Отчет:**

Тестирующая программа не требуется.

Cкриншот основной программы:



Скриншот дополнительной программы на языке С++:

#include <iostream>  
#include <cmath>  
#include <iomanip>  
  
double func(double x) {  
 return 3 \* x \* x - 2;  
}  
  
double bisection(double a, double b, double tolerance) {  
 double mid;  
 while ((b - a) / 2.0 > tolerance) {  
 mid = (a + b) / 2.0;  
 if (func(mid) == 0.0) {  
 break; // Корень найден  
 } else if (func(a) \* func(mid) < 0) {  
 b = mid;  
 } else {  
 a = mid;  
 }  
 }  
 return (a + b) / 2.0;  
}  
  
int main() {  
 // Задаем начальный диапазон и точность  
 double a = -1, b = 1, tolerance = 0.001;  
  
 // Проверяем, содержит ли диапазон отрицательный корень  
 if (func(a) \* func(0) > 0) {  
 std::cout << "Нет отрицательного корня в диапазоне [" << a << "; 0]." << std::endl;  
 } else {  
 // Ищем отрицательный корень  
 while (tolerance >= 0.00000001) {  
 double root = bisection(a, 0, tolerance);  
 std::cout << "Neg root 3x^2 - 2 = 0 with accuracy " << tolerance << ": ";  
 std::cout << std::fixed << std::setprecision(10) << root << std::endl;  
 tolerance /= 10.0; // Уменьшаем точность  
 }  
 }  
  
 // Восстанавливаем начальную точность для поиска положительного корня  
 tolerance = 0.001;  
  
 // Проверяем, содержит ли диапазон положительный корень  
 if (func(0) \* func(b) > 0) {  
 std::cout << "Нет положительного корня в диапазоне [0; " << b << "]." << std::endl;  
 } else {  
 // Ищем положительный корень  
 while (tolerance >= 0.00000001) {  
 double root = bisection(0, b, tolerance);  
 std::cout << "Положительный корень уравнения 3x^2 - 2 = 0 с точностью " << tolerance << ": ";  
 std::cout << std::fixed << std::setprecision(10) << root << std::endl;  
 tolerance /= 10.0; // Уменьшаем точность  
 }  
 }  
  
 return 0;  
}