Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОССУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №7-8

По дисциплине «Архитектуры вычислительных систем»

По теме «Программирование арифметического сопроцессора»

Выполнил:

студент гр. 653505

Хлопцев А. А.

Проверил:

Шиманский В. В.

Минск 2018

Оглавление

[1 Цель работы 3](#_Toc510038475)

[2 Постановка задачи 4](#_Toc510038476)

[3 Краткие теоретические сведения 5](#_Toc510038477)

[3.1 Квадратно уравнение 5](#_Toc510038478)

[3.2 Разложение функций в степенной ряд Тейлора (Маклорена) 6](#_Toc510038479)

[4 Алгоритмы выполнения 8](#_Toc510038480)

[4.1 Решение квадратного уравнения 8](#_Toc510038481)

[4.2 Вычисление ряда Маклорена для sin(x) 8](#_Toc510038482)

[5 Демонстрация работы программы 9](#_Toc510038483)

[6 Вывод 13](#_Toc510038484)

[7 Приложение. Код программы 14](#_Toc510038485)

# Цель работы

Изучить как решаются простейшие математические задачи с помощью сопроцессора в языке ассемблер.

# Постановка задачи

1. Написать программу, находящую решение квадратного уравнения ax2 + bx + c = 0 с помощью сопроцессора.
2. Значение аргумента x изменяется от a до b с шагом h. Для каждого x найти значения функции Y(x), суммы S(x) и число итераций n, при котором достигается требуемая точность ε = |Y(x)-S(x)|. Результат вывести в виде таблицы. Значения a, b, h и ε вводятся с клавиатуры.



# Краткие теоретические сведения

## Квадратно уравнение

*Квадратное уравнение* — это уравнение вида *ax*2 + *bx* + *c* = 0, где коэффициенты *a*, *b* и *c* — произвольные числа, причем a ≠ 0.

Прежде, чем изучать конкретные методы решения, заметим, что все квадратные уравнения можно условно разделить на три класса:

1. Не имеют корней;
2. Имеют ровно один корень;
3. Имеют два различных корня.

В этом состоит важное отличие квадратных уравнений от линейных, где корень всегда существует и единственен. Как определить, сколько корней имеет уравнение? Для этого существует замечательная вещь — **дискриминант**.

**Дискриминант**

Пусть дано квадратное уравнение *ax*2 + *bx* + *c* = 0. Тогда дискриминант — это просто число *D* = *b*2 − 4*ac*.

**Корни квадратного уравнения**

Теперь перейдем, собственно, к решению. Если дискриминант *D* > 0, корни можно найти по формулам:

Формула корней квадратного уравнения

Основная формула корней квадратного уравнения

Когда *D* = 0, можно использовать любую из этих формул — получится одно и то же число, которое и будет ответом. Наконец, если *D* < 0, корней нет — ничего считать не надо.

**Неполные квадратные уравнения**

Бывает, что квадратное уравнение несколько отличается от того, что дано в определении. Например:

1. *x*2 + 9*x* = 0;
2. *x*2 − 16 = 0.

Несложно заметить, что в этих уравнениях отсутствует одно из слагаемых. Такие квадратные уравнения решаются даже легче, чем стандартные: в них даже не потребуется считать дискриминант. Итак, введем новое понятие:

Уравнение *ax*2 + *bx* + *c* = 0 называется неполным квадратным уравнением, если *b* = 0 или *c* = 0, т.е. коэффициент при переменной *x* или свободный элемент равен нулю.

Разумеется, возможен совсем тяжелый случай, когда оба этих коэффициента равны нулю: *b* = *c* = 0. В этом случае уравнение принимает вид a*x*2 = 0. Очевидно, такое уравнение имеет единственный корень: *x* = 0.

Рассмотрим остальные случаи. Пусть *b* = 0, тогда получим неполное квадратное уравнение вида *ax*2 + *c* = 0. Немного преобразуем его:

Решение неполного квадратного уравнения

Решение неполного квадратного уравнения

Поскольку арифметический квадратный корень существует только из неотрицательного числа, последнее равенство имеет смысл исключительно при (−*c*/*a*) ≥ 0. Вывод:

1. Если в неполном квадратном уравнении вида *ax*2 + *c* = 0 выполнено неравенство (−*c*/*a*) ≥ 0, корней будет два. Формула дана выше;
2. Если же (−*c*/*a*) < 0, корней нет.

Теперь разберемся с уравнениями вида *ax*2 + *bx* = 0, в которых свободный элемент равен нулю. Тут все просто: корней всегда будет два. Достаточно разложить многочлен на множители:

Разложение уравнения на множители

Вынесение общего множителя за скобку

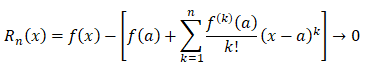
Произведение равно нулю, когда хотя бы один из множителей равен нулю.

## Разложение функций в степенной ряд Тейлора (Маклорена)

Если функция  допускает в некоторой окрестности  точки  разложение в степенной ряд по степеням , то этот ряд (ряд Тейлора) имеет вид



При  ряд Тейлора называют также рядом Маклорена. Последнее равенство справедливо, если при  остаточный член ряда Тейлора



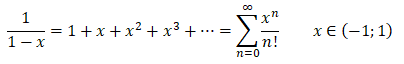
при 

Для оценки остаточного члена можно пользоваться формулой (форма Лагранжа)



Пользуясь основными разложениями, а также формулой для геометрической прогрессии, можно во многих случаях просто получать разложение данной функции в степенной ряд, причем отпадает необходимость исследования остаточного члена. Иногда при разложении полезно использовать почленное дифференцирование или интегрирование. При разложении в степенные ряды рациональных функций рекомендуется разлагать эти функции на простейшие дроби.

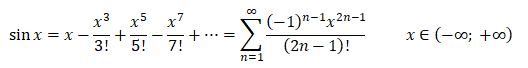
**I.**  Бесконечная убывающая геометрическая прогрессия со знаменателем .



**II.**  Разложение экспоненты в ряд Маклорена



**III.** Разложение синуса в ряд Маклорена



И т.д

# Алгоритмы выполнения

## Решение квадратного уравнения

**Алгоритм:**

1. Считываем входные данные: a, b, c
2. Кладём данные в регистры a – r0, b – r1, c – r2
3. Находим b2, результат сохраняем в r3
4. Кладём в r4 – 4
5. Находим 4\*a, результат в r4
6. Находим 4\*a\*c, результат в r4
7. Вычисляем дискриминант: D = b2 - 4\*a\*c (r3 – r4), результат в r3
8. Если D > 0:
9. Находим корень из D – r3
10. Кладём –b в r1
11. Кладём в r4 – 2
12. Находим 2\*a – r0
13. Высчитываем –b + sqrt(D) (r1 + r3) – результат сохраняем в r5
14. Считаем первый корень: x1 = (-b + sqrt(D))/2\*a – результат в r5
15. Высчитываем –b - sqrt(D) (r1 - r3) – результат сохраняем в r6
16. Считаем второй корень: x2 = (-b - sqrt(D))/2\*a – результат в r6
17. Перемещаем корни в r0, r1
18. Если D = 0
19. Кладём –b в r1
20. Находим 2\*a – r0
21. Кладём в r4 – 2
22. Высчитываем x = -b / 2 \* a, результат в r1
23. Кладём в r0, r1 корни x
24. Если D < 0 – выводим сообщение об отсутствии корней

## Вычисление ряда Маклорена для sin(x)

**Алгоритм:**

1. Считываем данные: a, b, h, e
2. Запускаем цикл от a до b с шагом h
3. Для каждого значения x из цикла, запускаем функцию создания ряда Маклорена
4. В r0 – x, это наш ряд Маклорена (S)
5. В r1 – x, для подсчёта числителя
6. В r2 – x, это наш числитель
7. В r3 – 1, это основание факториала
8. В r4 – 1, это значение факториала (знаменатель)
9. В r5 – 1,значение для увеличение основания факториала
10. Запускаем бесконечный цикл, до тех пор, пока не будет найден ответ или его отсутствие (|sin(x) – S| < e)
11. Увеличиваем значение знаменателя на x (r2 \* r1)
12. Опять увеличиваем значение знаменателя на x (r2 \* r1)
13. Увеличиваем основание факториала на 2 (r3 + 2 \* r5)
14. Пересчитываем факториал для нового основания r4 \*= r4 \* (r3 + 1) \* (r3 + 2)
15. Вычисляем новый член прогрессии S(i) = r2 / r4, результат в r6
16. По чётности/нечётности I определяем, добавить член к S или отнять (неч +, чёт -)
17. Выводим результат (если мы не достигли результата за 1000 шагов, говорим что он очень большой)

# Демонстрация работы программы

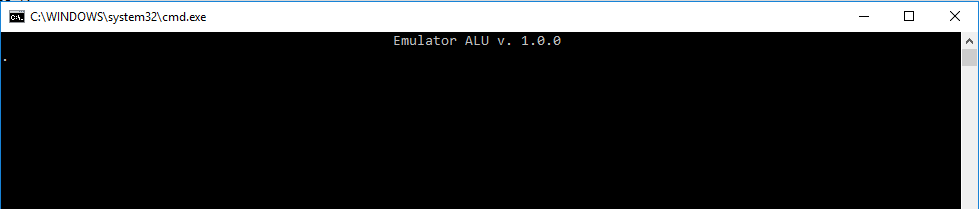


Рисунок 1 – Запуск программы

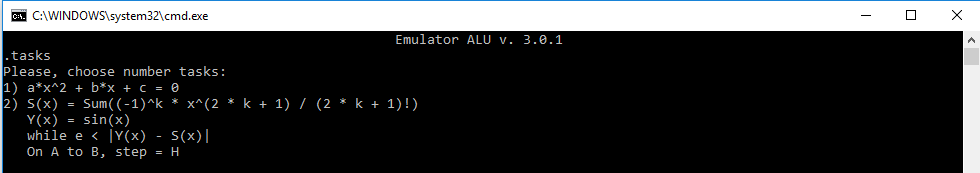


Рисунок 2 – Меню задач

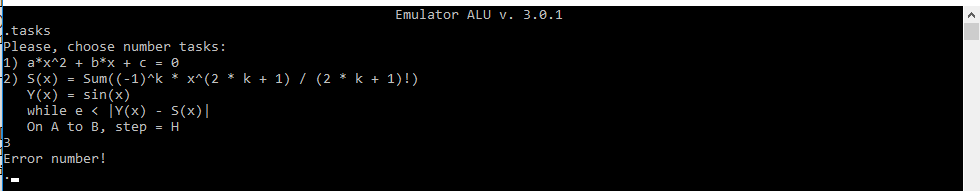


Рисунок 2 – Проверка наличия выбранной задачи

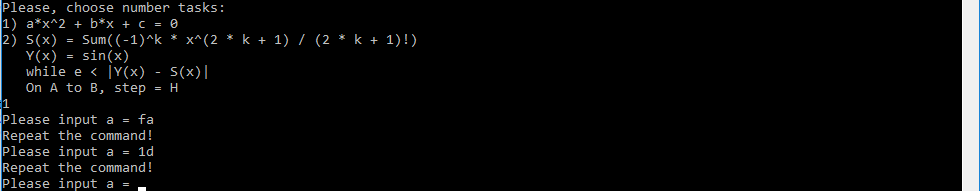
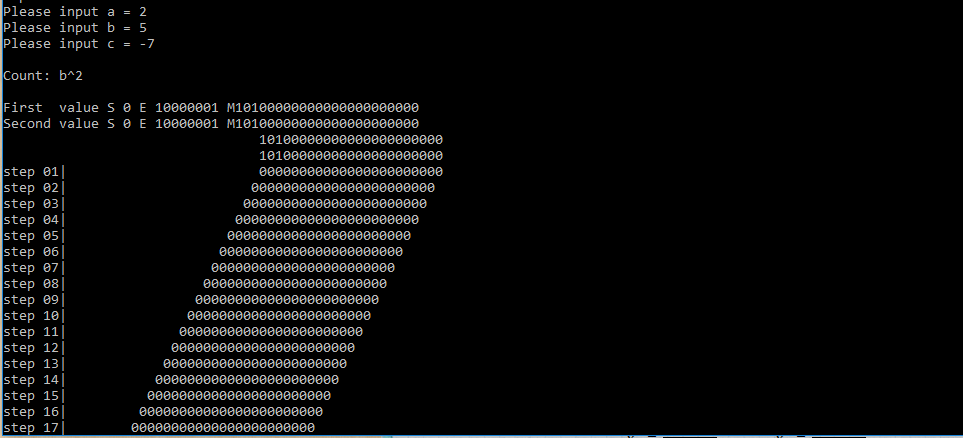
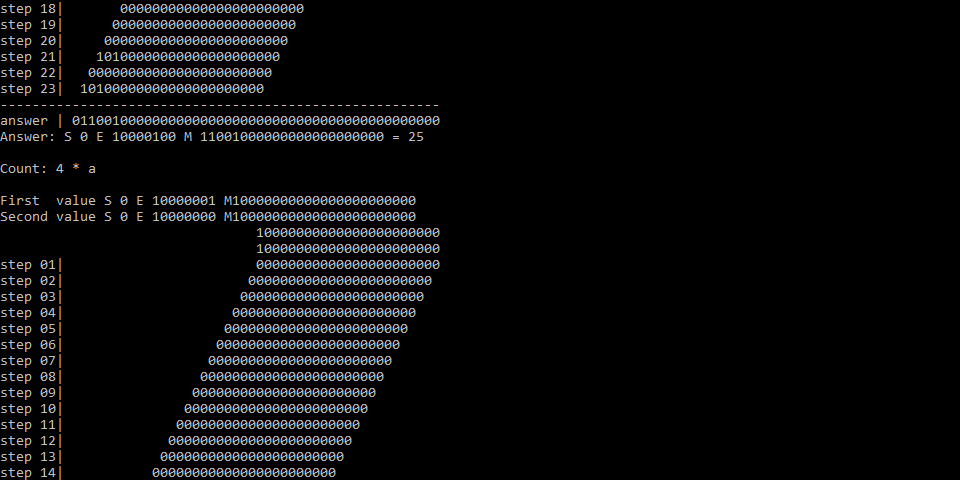


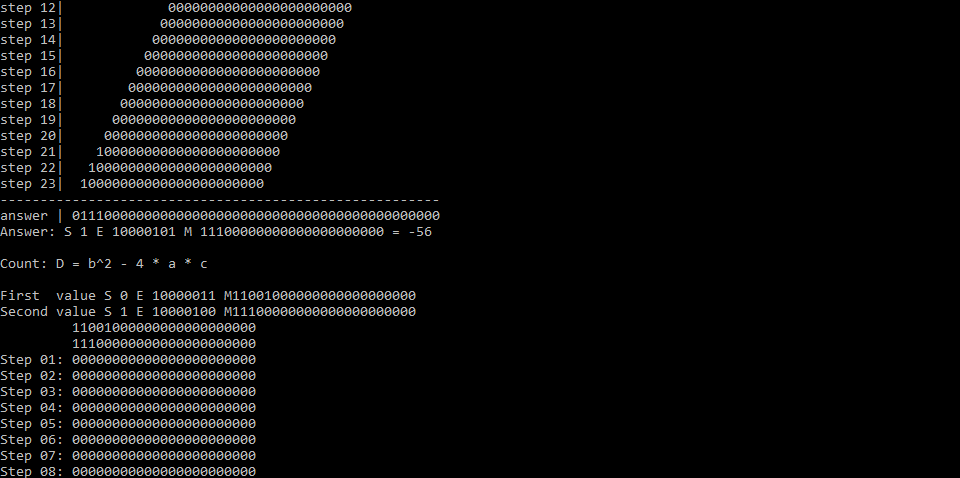
Рисунок 3 – Проверка вводимых данных

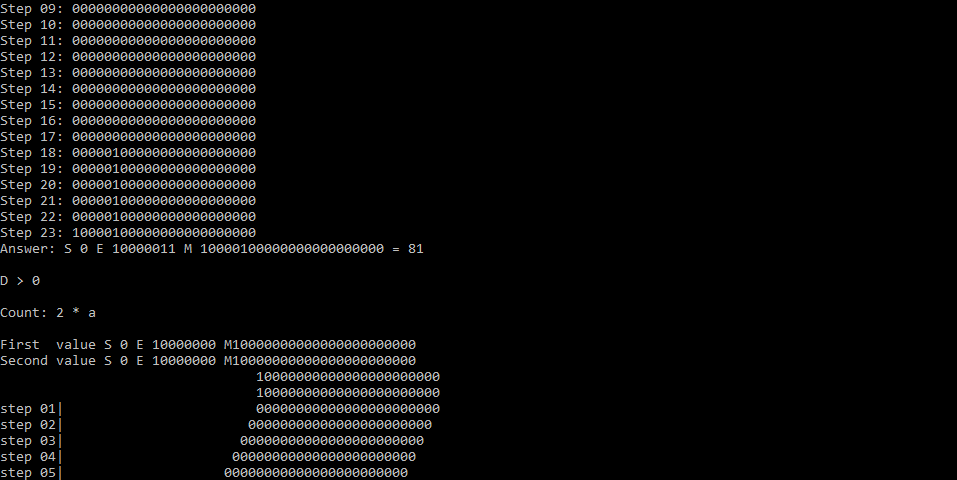
(10-38 <= a, b, c <= 1038)

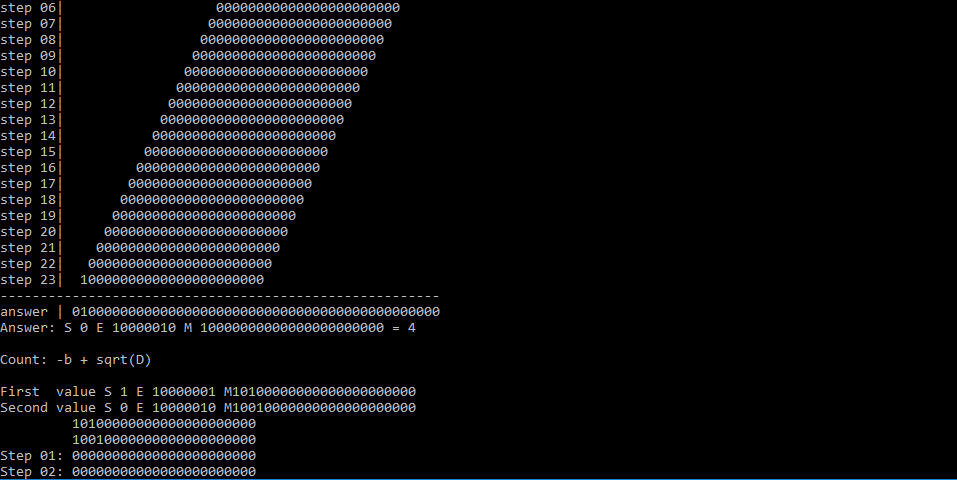


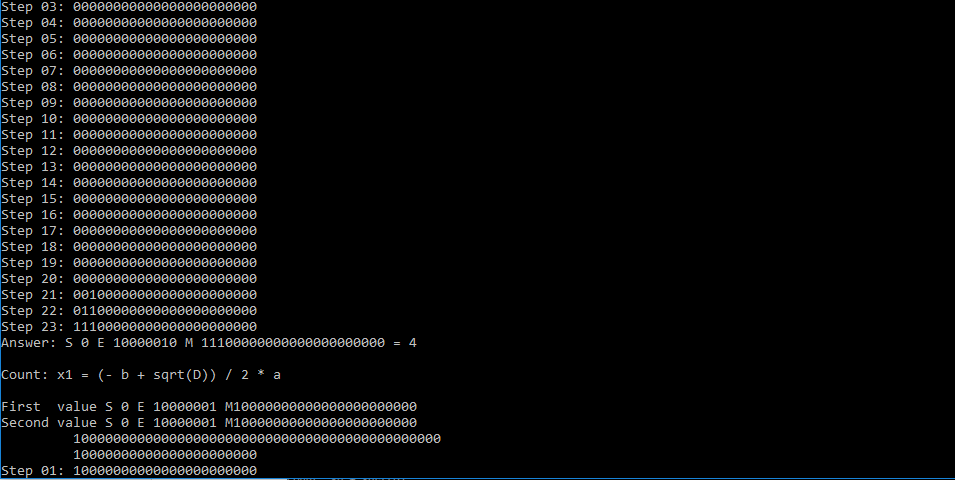


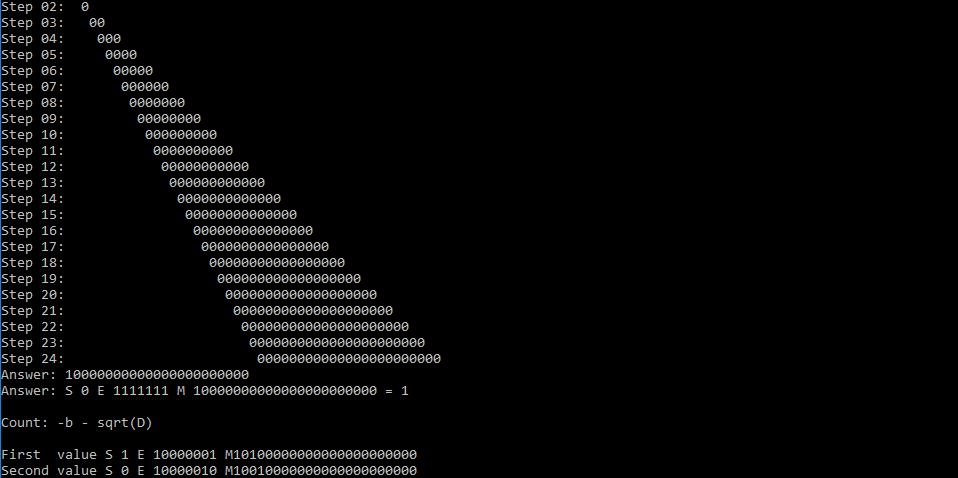


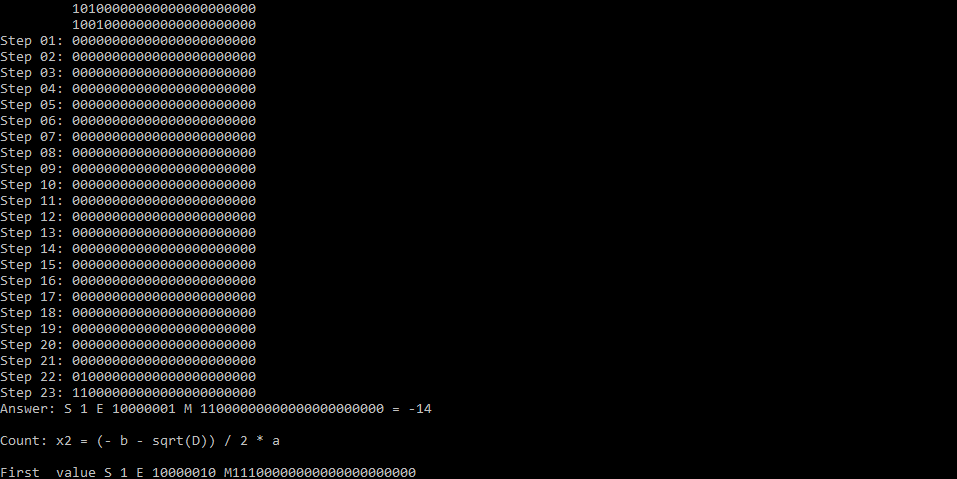












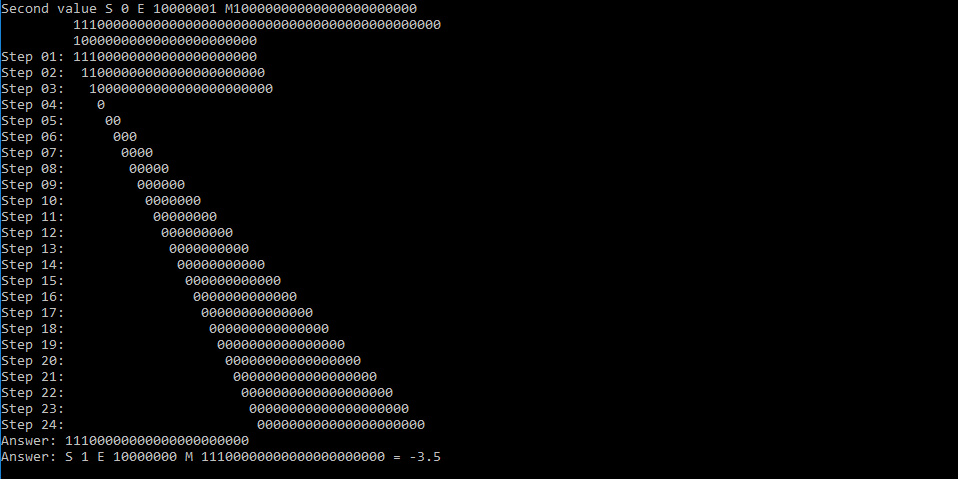




Рисунок 4 - 15 – Пример решения задачи по поиску дискриминанта

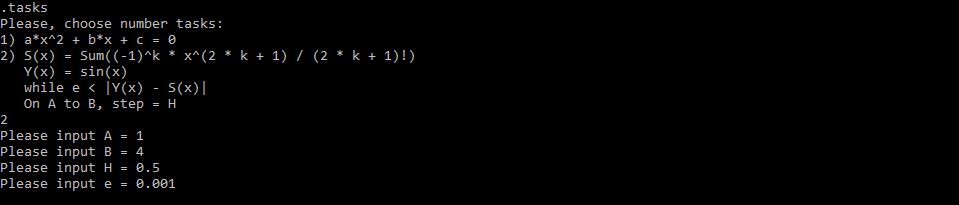


Рисунок 16 – Входные данные для ряда Маклорена

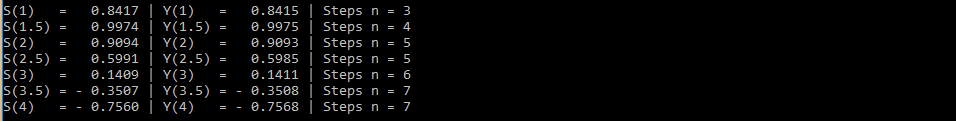


Рисунок 17 – Результат выполнения задачи, ряд Маклорена

# Вывод

В результате выполнения лабораторных работ №7-8 были созданы и реализованы алгоритмы нахождения решения квадратного уравнения и разложения синуса в ряд Маклорена с помощью сопроцессора и арифметических операций с плавающей точкой.

# Приложение. Код программы

private static void MyTasks()

{

Console.WriteLine("Please, choose number tasks:");

Console.WriteLine("1) a\*x^2 + b\*x + c = 0");

Console.WriteLine("2) S(x) = Sum((-1)^k \* x^(2 \* k + 1) / (2 \* k + 1)!)\n Y(x) = sin(x)\n while e < |Y(x) - S(x)|\n On A to B, step = H");

string s = Console.ReadLine();

int x;

try

{

x = Int32.Parse(s);

if (x != 1 && x != 2)

throw new Exception();

}

catch

{

Console.WriteLine("Error number!");

return;

}

if (x == 1)

Tasks.FirstTasks();

else

if (x == 2)

Tasks.SecondTasks();

}

public static class Tasks

{

static public void FirstTasks()

{

double a, b, c;

while (true)

{

Console.Write("Please input a = ");

string s = Console.ReadLine();

if (double.TryParse(s, out a))

break;

else

Console.WriteLine("Repeat the command!");

}

while (true)

{

Console.Write("Please input b = ");

string s = Console.ReadLine();

if (double.TryParse(s, out b))

break;

else

Console.WriteLine("Repeat the command!");

}

while (true)

{

Console.Write("Please input c = ");

string s = Console.ReadLine();

if (double.TryParse(s, out c))

break;

else

Console.WriteLine("Repeat the command!");

}

SCPU.Task1(a, b, c);

}

static public void SecondTasks()

{

double a, b, h, e;

while (true)

{

Console.Write("Please input A = ");

string s = Console.ReadLine();

if (double.TryParse(s, out a))

break;

else

Console.WriteLine("Repeat the command!");

}

while (true)

{

Console.Write("Please input B = ");

string s = Console.ReadLine();

if (double.TryParse(s, out b))

break;

else

Console.WriteLine("Repeat the command!");

}

while (true)

{

Console.Write("Please input H = ");

string s = Console.ReadLine();

if (double.TryParse(s, out h))

break;

else

Console.WriteLine("Repeat the command!");

}

if (a > b)

{

Console.WriteLine("Error a > b");

return;

}

if (a + h < a)

{

Console.WriteLine("Error a + n < a");

return;

}

while (true)

{

Console.Write("Please input e = ");

string s = Console.ReadLine();

if (double.TryParse(s, out e))

break;

else

Console.WriteLine("Repeat the command!");

}

string ans = "";

for (double i = a; i <= b; i += h)

{

ans += SCPU.Task2(i, e);

ans += "\n";

}

Console.WriteLine(ans);

}

}

public static void Task1(double a, double b, double c)

{

(reg["r0"] as SoReg).In(a);

(reg["r1"] as SoReg).In(b);

(reg["r2"] as SoReg).In(c);

Console.WriteLine("\nCount: b^2\n");

Visual.VisualCommandFMul("r1", "r1");

(reg["r3"] as SoReg).In((reg["r1"] as SoReg).val \* (reg["r1"] as SoReg).val);

Console.WriteLine("\nCount: 4 \* a\n");

(reg["r4"] as SoReg).In(4);

Visual.VisualCommandFMul("r4", "r0");

(reg["r4"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r0"] as SoReg).val);

Console.WriteLine("\nCount: 4 \* a \* c\n");

Visual.VisualCommandFMul("r4", "r2");

(reg["r4"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r2"] as SoReg).val);

Console.WriteLine("\nCount: D = b^2 - 4 \* a \* c\n");

byte x = 0;

if ((reg["r3"] as SoReg).val - (reg["r4"] as SoReg).val < 0)

x = 1;

Visual.VisualCommandFSub(x, "r3", "r4");

(reg["r3"] as SoReg).In((reg["r3"] as SoReg).val - (reg["r4"] as SoReg).val);

if ((reg["r3"] as SoReg).val > 0)

{

Console.WriteLine("\nD > 0");

(reg["r3"] as SoReg).In(Math.Sqrt((reg["r3"] as SoReg).val));

(reg["r1"] as SoReg).In(-(reg["r1"] as SoReg).val);

Console.WriteLine("\nCount: 2 \* a\n");

(reg["r4"] as SoReg).In(2);

Visual.VisualCommandFMul("r4", "r0");

(reg["r0"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r0"] as SoReg).val);

Console.WriteLine("\nCount: -b + sqrt(D)\n");

x = 0;

if ((reg["r1"] as SoReg).val + (reg["r3"] as SoReg).val < 0)

x = 1;

Visual.VisualCommandFAdd(x, "r1", "r3");

(reg["r5"] as SoReg).In((reg["r1"] as SoReg).val + (reg["r3"] as SoReg).val);

Console.WriteLine("\nCount: x1 = (- b + sqrt(D)) / 2 \* a \n");

Visual.VisualCommandFDiv("r5", "r0");

(reg["r5"] as SoReg).In((reg["r5"] as SoReg).val / (reg["r0"] as SoReg).val);

Console.WriteLine("\nCount: -b - sqrt(D)\n");

x = 0;

if ((reg["r1"] as SoReg).val - (reg["r3"] as SoReg).val < 0)

x = 1;

Visual.VisualCommandFSub(x, "r1", "r3");

(reg["r6"] as SoReg).In((reg["r1"] as SoReg).val - (reg["r3"] as SoReg).val);

Console.WriteLine("\nCount: x2 = (- b - sqrt(D)) / 2 \* a \n");

Visual.VisualCommandFDiv("r6", "r0");

(reg["r6"] as SoReg).In((reg["r6"] as SoReg).val / (reg["r0"] as SoReg).val);

(reg["r0"] as SoReg).In((reg["r5"] as SoReg).val);

(reg["r1"] as SoReg).In((reg["r6"] as SoReg).val);

Console.WriteLine("\nAnswer x1 = {0}(r0), x2 = {1}(r1)\n", (reg["r0"] as SoReg).val, (reg["r1"] as SoReg).val);

}

else

if (Math.Abs((reg["r3"] as SoReg).val) <= 0.0000001)

{

Console.WriteLine("\nD = 0");

(reg["r1"] as SoReg).In(-(reg["r1"] as SoReg).val);

Console.WriteLine("\nCount: 2 \* a\n");

(reg["r4"] as SoReg).In(2);

Visual.VisualCommandFMul("r4", "r0");

(reg["r0"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r0"] as SoReg).val);

Console.WriteLine("\nCount: x1,2 = -b / 2 \* a \n");

Visual.VisualCommandFDiv("r1", "r0");

(reg["r1"] as SoReg).In((reg["r1"] as SoReg).val / (reg["r0"] as SoReg).val);

(reg["r0"] as SoReg).In((reg["r1"] as SoReg).val);

Console.WriteLine("\nAnswer x1 = {0}(r0), x2 = {1}(r1)\n", (reg["r0"] as SoReg).val, (reg["r1"] as SoReg).val);

}

else

{

Console.WriteLine("\nRoots are not present!");

return;

}

}

public static string Task2(double x, double e)

{

Clear();

MExit = true;

double y = Math.Sin(x);

int n = 0;

bool sign = true;

(reg["r0"] as SoReg).In(x);

(reg["r1"] as SoReg).In(x);

(reg["r2"] as SoReg).In(x);

(reg["r3"] as SoReg).In(1);

(reg["r4"] as SoReg).In(1);

(reg["r5"] as SoReg).In(1);

while (true)

{

Console.WriteLine("\nS{0} = {1}x^{2} / {2}! = {3}", ++n, sign ? "+" : "-", (int)(reg["r3"] as SoReg).val, (reg["r0"] as SoReg).val);

if (Math.Abs(y - (reg["r0"] as SoReg).val) <= e)

break;

sign = sign ? false : true;

Console.WriteLine("\nCount: x^{0} \n", (int)(reg["r3"] as SoReg).val + 2);

Visual.VisualCommandFMul("r2", "r1");

(reg["r2"] as SoReg).In((reg["r2"] as SoReg).val \* (reg["r1"] as SoReg).val);

Visual.VisualCommandFMul("r2", "r1");

(reg["r2"] as SoReg).In((reg["r2"] as SoReg).val \* (reg["r1"] as SoReg).val);

Console.WriteLine("\nCount: ({0} + 2)! \n", (int)(reg["r3"] as SoReg).val);

Visual.VisualCommandFAdd(0, "r3", "r5");

(reg["r3"] as SoReg).In((reg["r3"] as SoReg).val + (reg["r5"] as SoReg).val);

Visual.VisualCommandFMul("r4", "r3");

(reg["r4"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r3"] as SoReg).val);

Visual.VisualCommandFAdd(0, "r3", "r5");

(reg["r3"] as SoReg).In((reg["r3"] as SoReg).val + (reg["r5"] as SoReg).val);

Visual.VisualCommandFMul("r4", "r3");

(reg["r4"] as SoReg).In((reg["r4"] as SoReg).val \* (reg["r3"] as SoReg).val);

Console.WriteLine("\nCount: S{0} \n", n);

Visual.VisualCommandFDiv("r2", "r4");

(reg["r6"] as SoReg).In((reg["r2"] as SoReg).val / (reg["r4"] as SoReg).val);

Console.WriteLine("\nCount: S += S{0} \n", n);

int v = 1;

if (sign)

v = 1;

else

v = -1;

byte z = 0;

if ((reg["r0"] as SoReg).val + v \* (reg["r6"] as SoReg).val < 0)

z = 1;

Visual.VisualCommandFAdd(z, "r0", "r6");

(reg["r0"] as SoReg).In((reg["r0"] as SoReg).val + v \* (reg["r6"] as SoReg).val);

if (n == 1000)

{

MExit = false;

return "Answer does not exist!"; ;

}

}

MExit = false;

string f = String.Format("{0: 0.0000}", Math.Round((reg["r0"] as SoReg).val, 5));

string s = String.Format("{0: 0.0000}", y);

//return (reg["r0"] as SoReg).val.ToString() + " " + y.ToString();

return String.Format("{0, -6} = {1} | {4, -6} = {2} | Steps n = {3}", "S(" + x + ")", (reg["r0"] as SoReg).val > 0 ? " " + f : f, y > 0 ? " " + s : s, n, "Y(" + x + ")");

}