Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

Лабораторная работа

**«Решение нелинейного уравнения разными методами»**

Выполнила:

студентка группы РИС-23-2б

Виноградова Юлия Дмитриевна

Проверила:

доцент кафедры ИТАС

О.А. Полякова

2023

**Постановка задачи:**

Разработать алгоритм и написать код на языке C++ для решения данного уравнения: .

**Метод половинного деления**

**Словесный алгоритм:**

1. В условии задачи уточняется отрезок [a; b], на котором функция имеет корень, сам вид функции , а также требуемая точность epsilon;
2. С помощью теоремы1 определяется первый корень, который находится как ;
3. Далее запускается цикл while, который продолжает работу, пока модуль разности между концами отрезка не станет меньше или равен заданной точности ;
   1. Eсли , то значение b заменяется на x, иначе значение a заменяется на x;
4. В результате точное значение корня выводится в виде .

1Теорема: «Если непрерывная функция на концах некоторого интервала имеет значения разных знаков, то внутри этого интервала у нее есть корень».

**Смысловые значения переменных:**

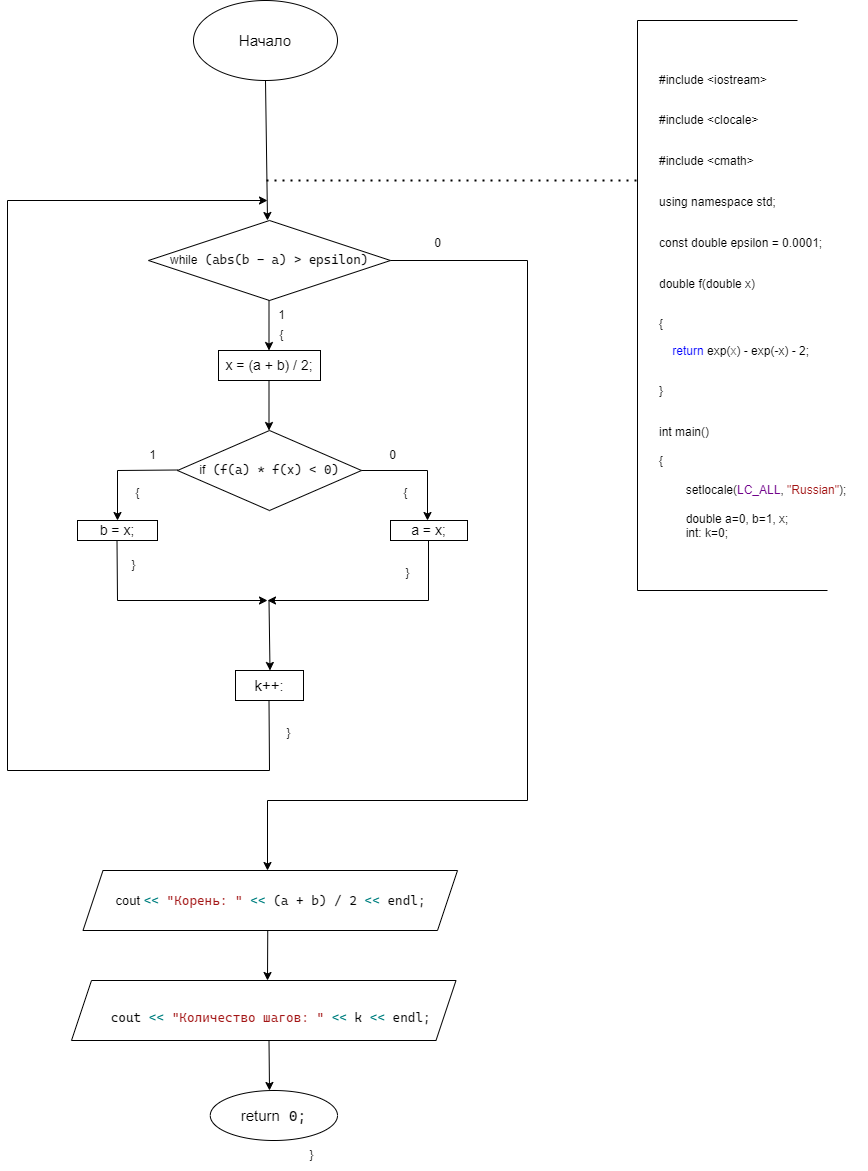
a – левый конец отрезка;

b – правый конец отрезка;

x – корень уравнения;

epsilon – точность.

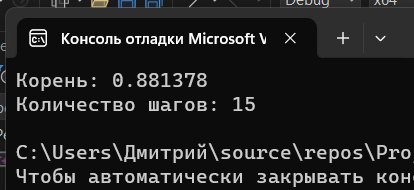
**Блок-схема**



**Код**

|  |
| --- |
| #include <iostream>  #include <clocale>  #include <cmath>  using namespace std;  const double epsilon = 0.0001; // точность  double f(double x)  {  return exp(x) - exp(-x) - 2;  }  int main()  {  setlocale(LC\_ALL, "Russian");  double a, b, x;  a = 0; // левый конец отрезка  b = 1; // правый конец отрезка  int k = 0; // шаги  while (abs(b - a) > epsilon)  {  x = (a + b) / 2;  if (f(a) \* f(x) < 0)  {  b = x;  }  else  {  a = x;  }  k = k + 1;  }  cout << "Корень: " << (a + b) / 2 << endl;  cout << "Количество шагов: " << k + 1 << endl;  return 0;  } |

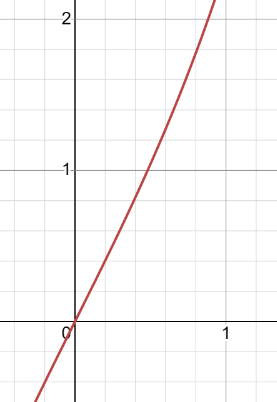
**Работа программы:**

****

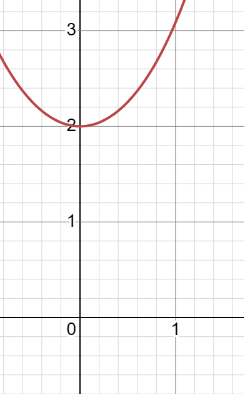
**Метод Ньютона**

**Условия применения алгоритма:**

Данный алгоритм требует, чтобы функция была монотонной и непрерывной, что подтверждается графической проверкой существования второй производной на интервале [0, 1]:



Также необходимо удостовериться, что первая производная не равна нулю для всех x на интервале, что также подтверждается графически:



Наконец, нужно убедиться, что знаки производных f’(x) и f’’(x) остаются постоянными для всех значений x на данном интервале, что также проверяется графически.

**Словесный алгоритм:**

1. В условии задачи уточняется отрезок [a; b], на котором функция имеет корень, сам вид функции , и необходимая точность epsilon;
2. Алгоритм начинается с выбора начального приближения x\_0. Значение x\_0 определяется в зависимости от характеристик функции: если , то x\_0 = a, если , то x\_0 = b;
3. Затем находится x\_1 по формуле ;
4. После этого запускается цикл, который работает до тех пор, пока модуль разности x\_0 и x\_1 не станет меньше или равен заданной точности;
   1. Значение x\_0 менется на x\_1;
   2. Повторяется третий шаг;
5. Выводится ответ.

**Смысловые значения переменных:**

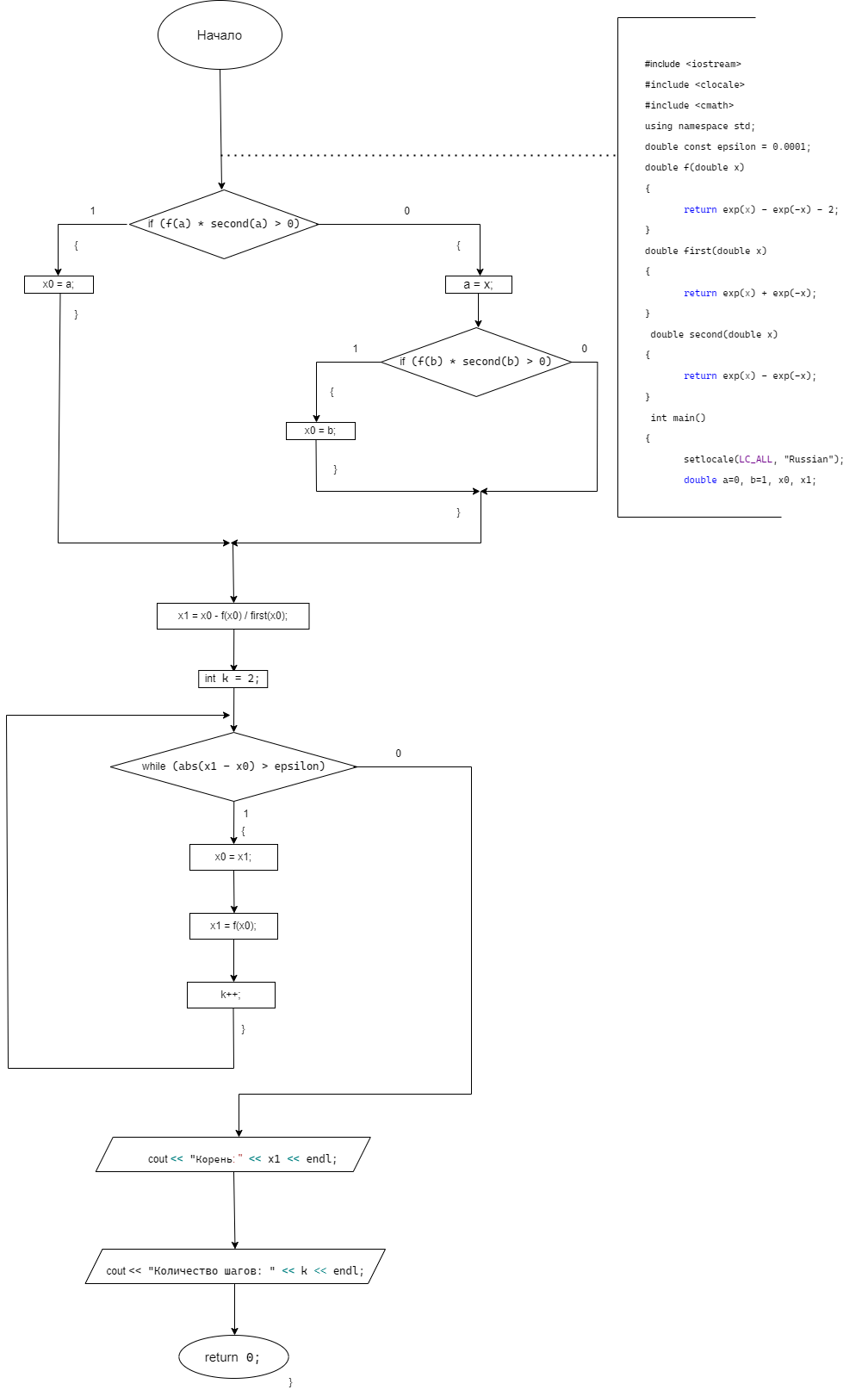
a – левый конец отрезка;

b – правый конец отрезка;

x\_0, x\_1 – точки пересечения касательной и оси абсцисс, корни уравнения;

epsilon – точность.

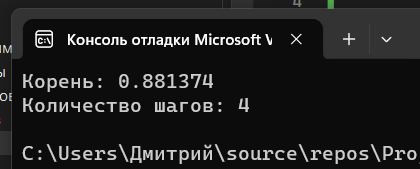
**Блок-схема**

****

**Код**

|  |
| --- |
| #include <iostream>  #include <clocale>  #include <cmath>  using namespace std;  double const epsilon = 0.0001;  double f(double x)  {  return exp(x) - exp(-x) - 2;  }  double first(double x)  {  return exp(x) + exp(-x);  }  double second(double x)  {  return exp(x) - exp(-x);  }  int main()  {  setlocale(LC\_ALL, "Russian");  double a, b, x0, x1;  a = 0;  b = 1;  if (f(a) \* second(a) > 0)  {  x0 = a;  }  else  {  if (f(b) \* second(b) > 0)  {  x0 = b;  }  }  x1 = x0 - f(x0) / first(x0);  int k = 2;  while (abs(x1 - x0) > epsilon)  {  x0 = x1;  x1 = x0 - f(x0) / first(x0);  k = k + 1;  }  cout << "Корень: " << x1 << endl;  cout << "Количество шагов: " << k << endl;  return 0;  } |

**Работы программы:**

****

**Метод итераций**

**Условия применения алгоритма:**

Этот метод может быть использован в том случае, если известен интервал, в котором находится корень уравнения (x ∈ [a; b]), данный интервал уточнен в условии задачи (x ∈ [0; 1]). Важным требованием также является то, что абсолютное значение производной новой функции должно быть меньше 1: .

Для новой функции , производная которой равна , абсолютное значение производной в точке x = 0.5 составляет примерно -0.232. Это значение удовлетворяет указанному требованию.

**Словесный алгоритм:**

1. Уравнение f(x)=0 переписывается в виде x=φ(x) для дальнейших вычислений.
2. На заданном отрезке [a; b] устанавливается начальное приближение х\_0 (x\_0 ∈ [0; 1]), ;
3. Следующее приближение вычисляется по формуле )
4. Этот процесс повторяется, пока . Как только , цикл завершается и становится искомым значением.
5. Ответ выводится на экран.

**Смысловые значения переменных:**

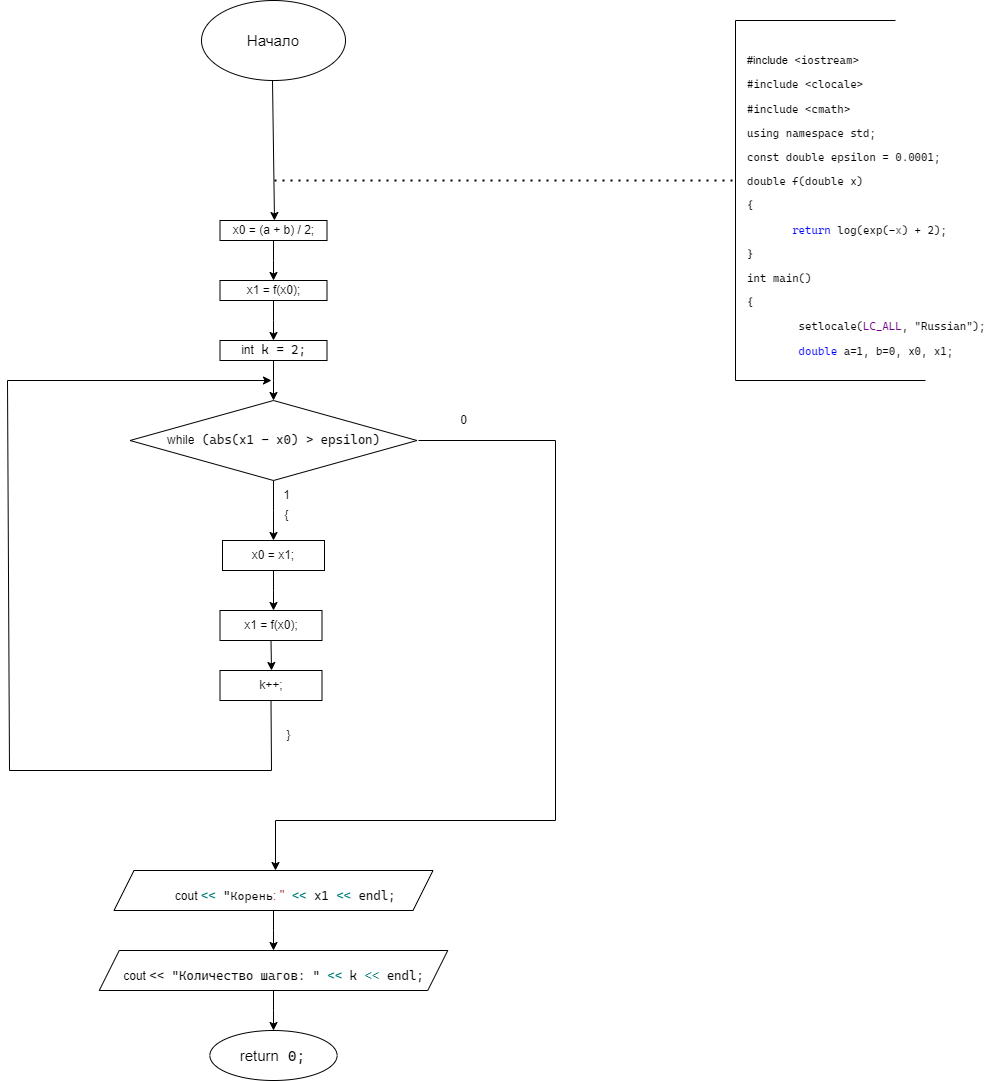
a – левый конец отрезка;

b – правый конец отрезка;

x\_0, x\_1 – точки пересечения касательной и оси абсцисс, корни уравнения;

epsilon – точность.

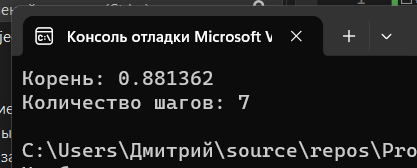
**Блок-схема**

****

**Код**

|  |
| --- |
| #include <iostream>  #include <clocale>  #include <cmath>  using namespace std;  const double epsilon = 0.0001;  double f(double x)  {  return log(exp(-x) + 2);  }  int main()  {  setlocale(LC\_ALL, "Russian");  double a, b, x0, x1;  a = 0;  b = 1;  x0 = (a + b) / 2;  x1 = f(x0);  int k = 2;  while (abs(x1 - x0) > epsilon)  {  x0 = x1;  x1 = f(x0);  k = k + 1;  }  cout << "Корень: " << x1 << endl;  cout << "Количество шагов: " << k << endl;  return 0;  } |

**Работа программы:**

****

**Скриншот из гита**

**Вывод**

В ходе выполнения лабораторной работы были исследованы различные методы решения нелинейных уравнений, такие как метод половинного деления, метод Ньютона и метод простой итерации.

Лабораторная работа позволила углубленно изучить методы их решения, провести сравнительный анализ и получить представление о применимости различных методов в реальных задачах.