**ВАРИАНТ 25**

**~~Написать отчет~~**~~, в котором должны присутствовать:~~

~~- условие задачи;~~

~~- условия заданного варианта задания, две функции и названия методов;~~

*~~- для каждого метода:~~*

~~-- описание заданного численного метода;~~

~~--~~ **~~блок-схему~~** ~~алгоритма подзадачи вычисления корня;~~

~~-- программный код функции вычисления корня, в которую передается~~ **~~указатель на нужную функцию;~~**

*~~- сравнение методов:~~*

~~-- результаты вычислений значения корня для обеих функций для заданных 10 вариантов погрешности и количество выполненных итераций (см. таблицу выше);~~

~~--~~ *~~вывод~~*~~: какой метод и чем считаете лучше и почему?~~

**Задание:**

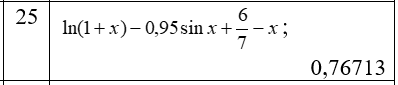
Двумя методами (см. табл.1) **для *N*** значений погрешности *ε* (0,1; 0,01;0,001;..1*e*-*N*, 1≤*N*≤10) вычислить значение корня **для двух** заданных функций **(№** и **№+3** по табл.2) на отрезке [*A*, *B*] ⊂ (0,2) и вывести их **в виде таблицы** (для каждой функции отдельную):

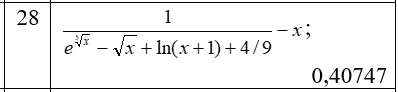
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер функции  Погрешность | Название метода | | Название метода | |
| Корень | Итераций | Корень | Итераций |
| 0,1 |  |  |  |  |
| … |  |  |  |  |
| 0,0…1 |  |  |  |  |

**При вводе значений анализировать аномалии**: *A*, *B*, *N* – числа и 0<*A*<*B*<2, 1≤*N*≤10.

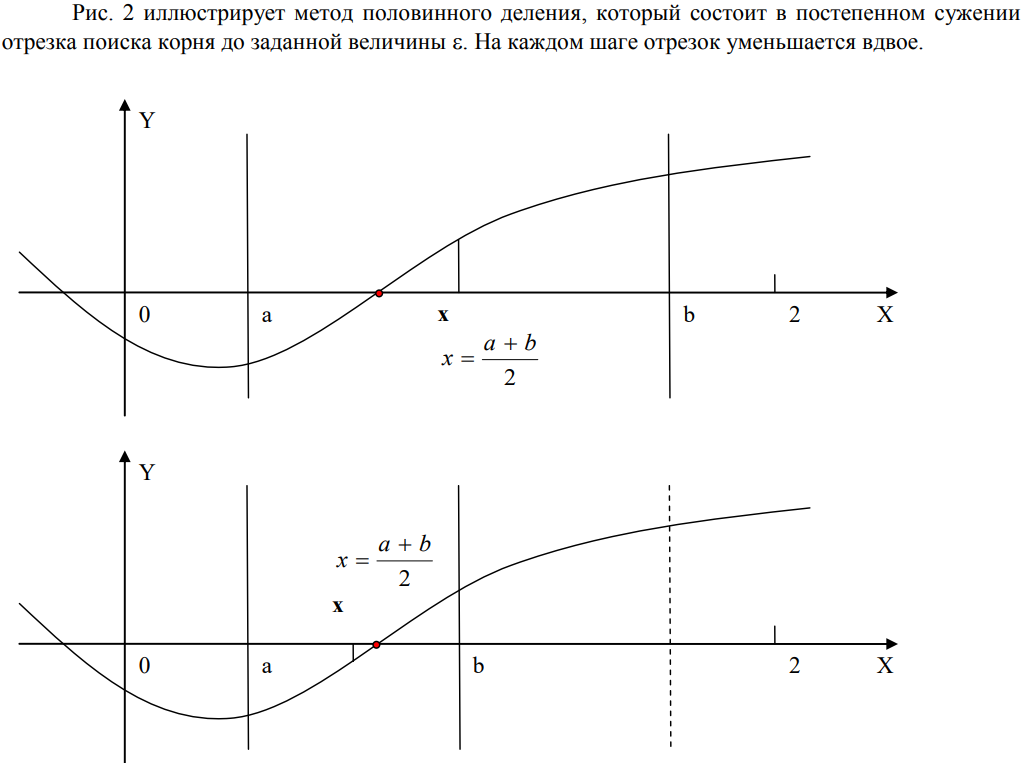
Перед поиском корня обязательно **проверять наличие корня** на отрезке [A, B] (разные знаки значения функции на концах отрезка), и в случае отсутствия выдать сообщение «Корня на отрезке AB нет».

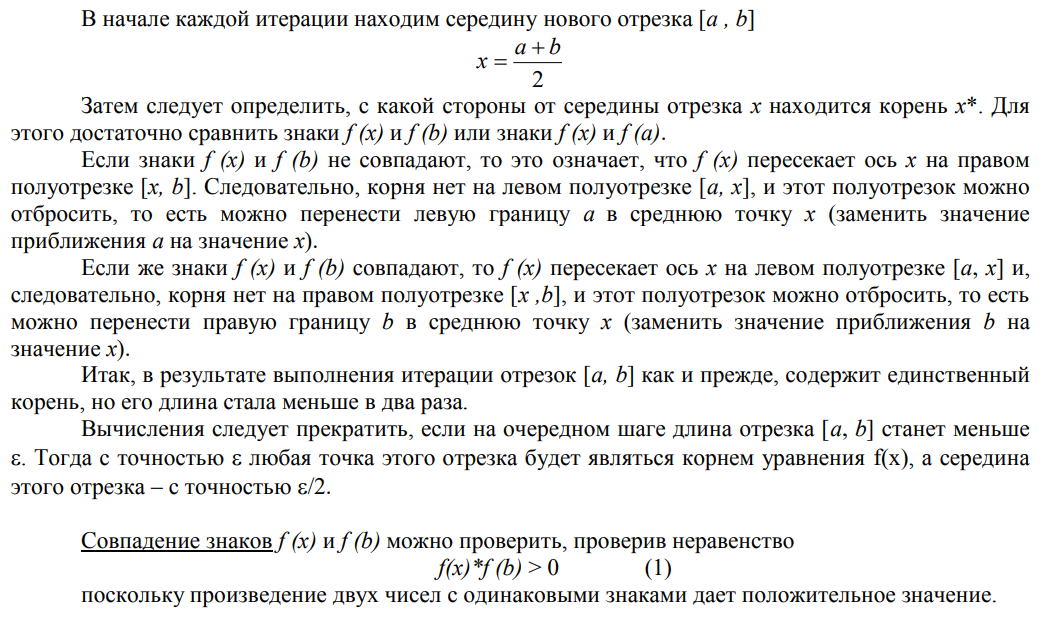
|  |  |
| --- | --- |
| **М**  **Е**  **Т**  **О**  **Д**  **Ы** | **1)Метод деления отрезка пополам**  **2) метод простых итераций** |

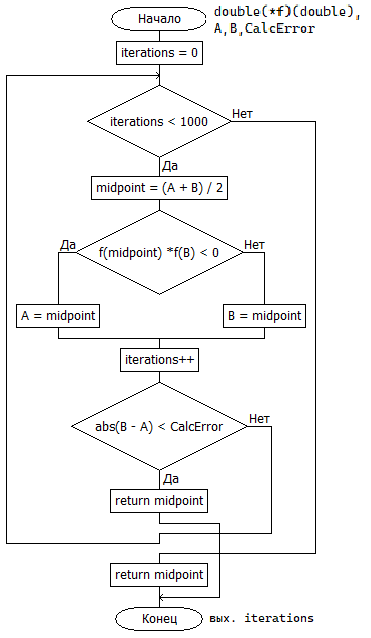




1. **МЕТОД ДЕЛЕНИЯ ОТРЕЗКА ПОПОЛАМ**







double BisectionMethod(double(\*f)(double), double A, double B, double CalcError, int&iterations)

{

double midpoint;

iterations = 0;

while(iterations < 1000)

{

midpoint = (A + B) / 2;

if (f(midpoint) \* f(B) < 0) A = midpoint;

else B = midpoint; iterations++;

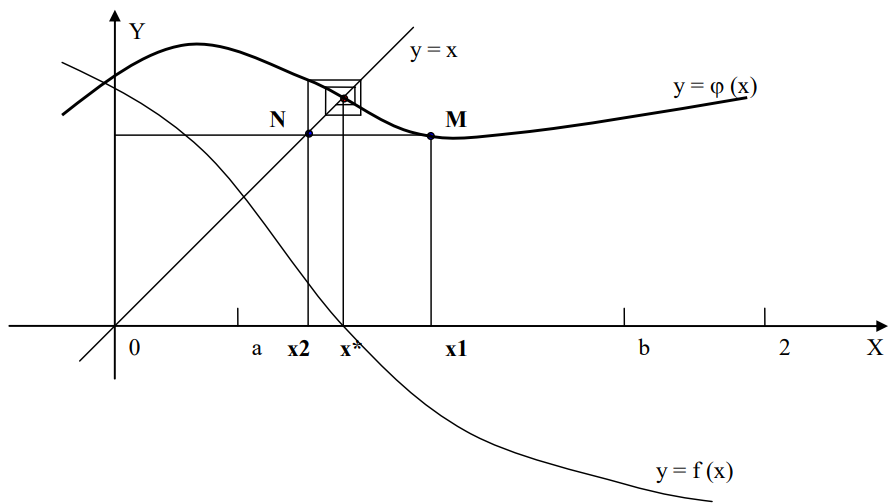
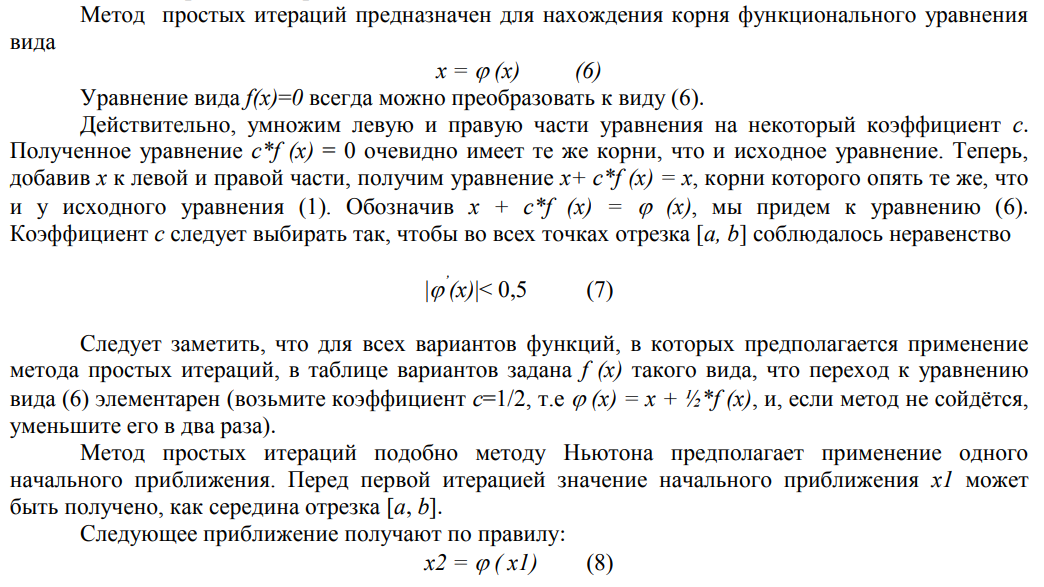
if (abs(B - A) < CalcError) return midpoint;

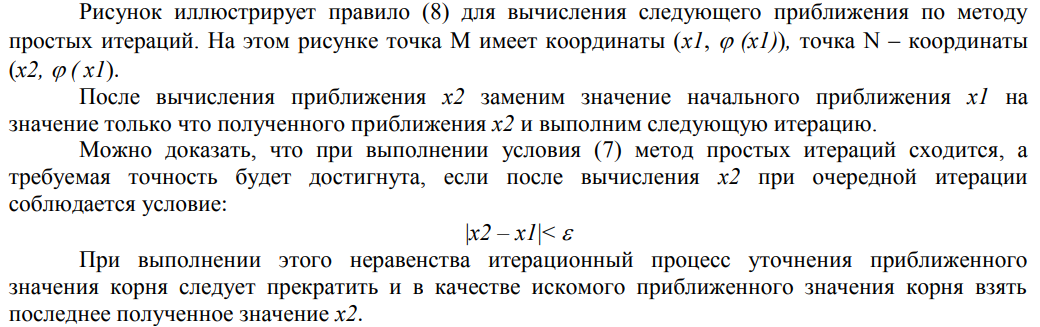
}

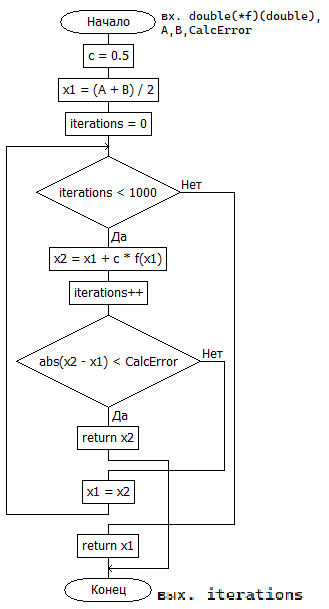
return midpoint;

}}

1. **МЕТОД ПРОСТЫХ ИТЕРАЦИЙ**







double SimpleIterationMethod(double(\*f)(double), double A, double B, double CalcError, int& iterations)

{

double c = 0.5;

double x1 = (A + B) / 2;

double x2;

iterations = 0;

while(iterations < 1000)

{

x2 = x1 + c \* f(x1);

iterations++;

if (abs(x2 - x1) < CalcError) return x2;

x1 = x2;

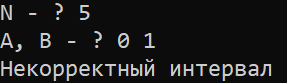
}

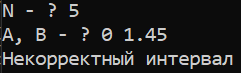
return x1;

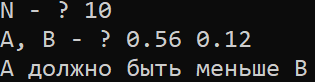
}

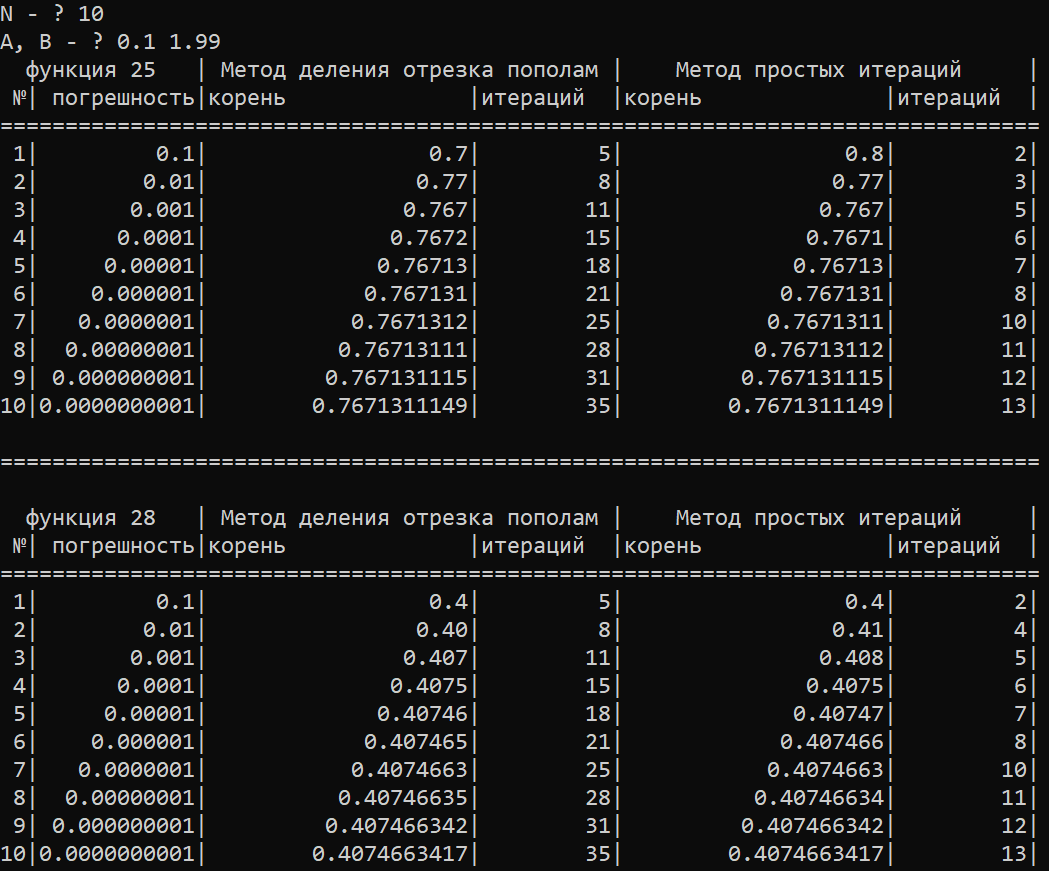
**СРАВНЕНИЕ МЕТОДОВ:**

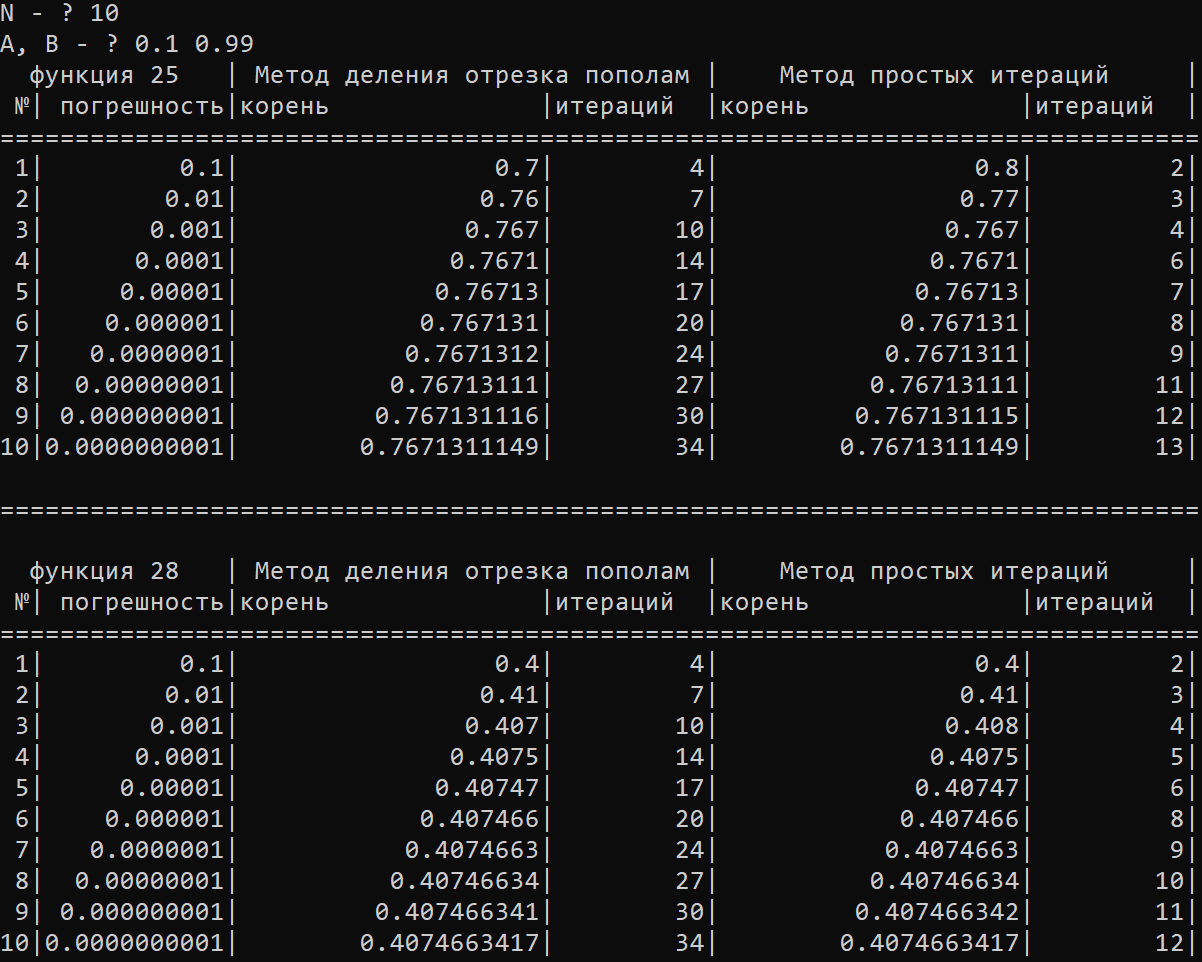
Тесты:

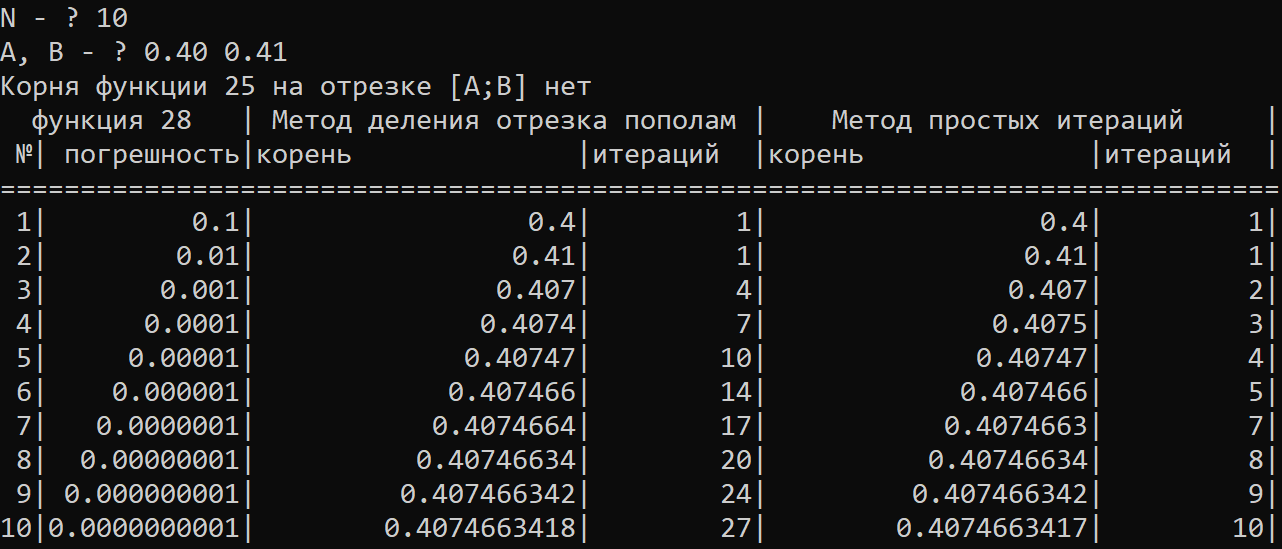




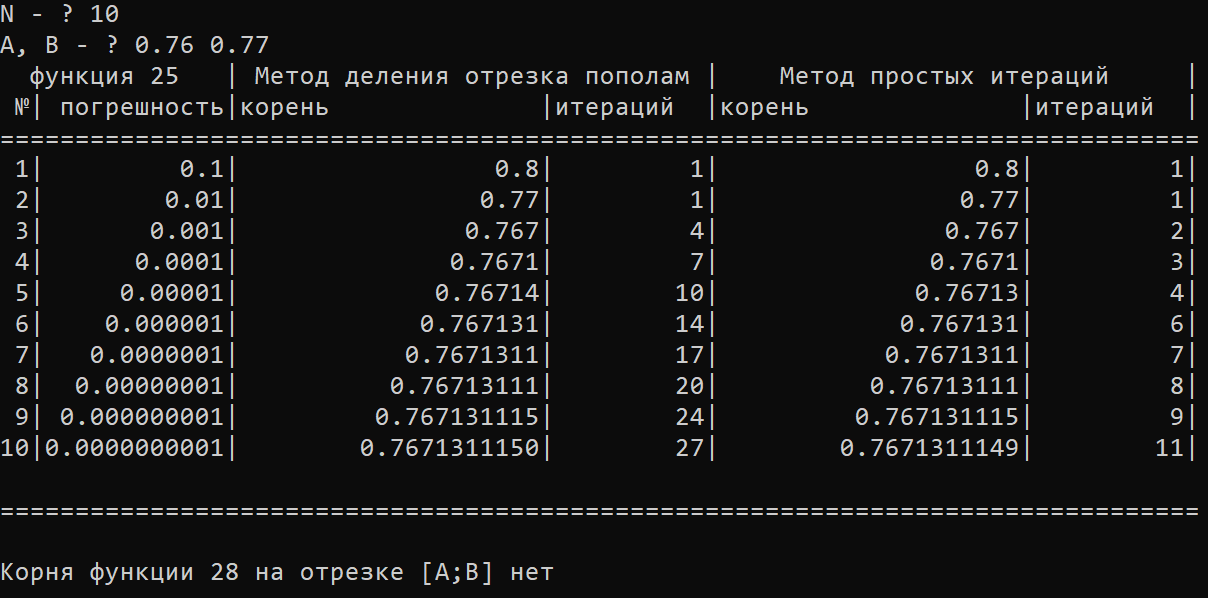


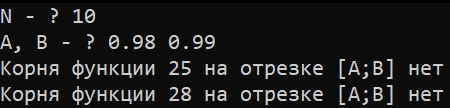






==========================================================





Вывод:

Я считаю метод деления отрезка пополам более стабильным. При заданном интервале он находит корень с заданной точностью всегда за фиксированное число итераций, в то время как метод простых итераций может оказаться как более эффективным, так и менее эффективным методом вычисления корня (зависит от коэффициента c), а может вообще не сойтись.