Подпрограммы – параметры других подпрограмм. Указатели на функции в Си

лекция №5

В каких задачах используются подпрограммы-параметры (в Си функции-параметры)?

- Когда некоторый алгоритм, описанный как подпрограмма, применим к множеству алгоритмов, каждый из которых также задается подпрограммой.
- **Классические примеры** таких ситуаций дают численные методы. В подпрограммах численных методов (вычисления определенного интеграла, нахождения экстремумов и нулей функций, вывода графиков, линий уровня, таблиц функций) обрабатываемые функции задаются как параметры.
- Возможности использования параметров-подпрограмм имеются во всех алгоритмических языках, предназначенных для решения вычислительных задач (СИ, Фортран, Паскаль, Матлаб, …).

Средства СИ для работы с подпрограммами-параметрами: указатели на функцию

Скобки обязательны, чтобы * не относилась к типу функции. Допустимо: тип* (**имя_функции*(*с_ф_п*)

- Указатель на функцию:
  тип (**имя_функции*)(*список формальных параметров*)

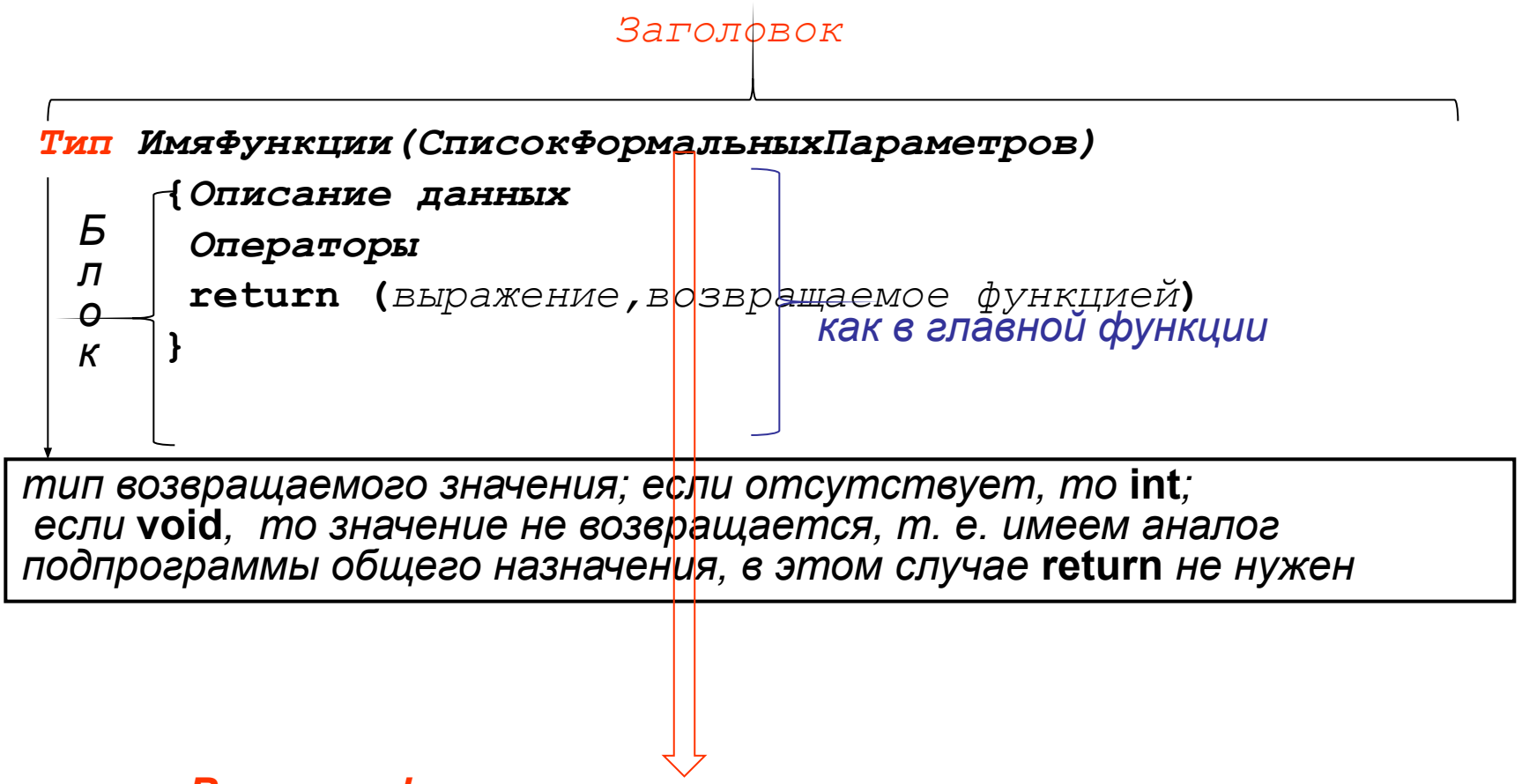
По имени функции определяется адрес ее начала (точки входа) как указатель на функцию.

- В списке формальных параметров основной функции приводится полный заголовок указателя на формальную функцию (возможно, без имен формальных параметров):
  тип (**имя_формальной_функции*)(*список формальных параметров*)

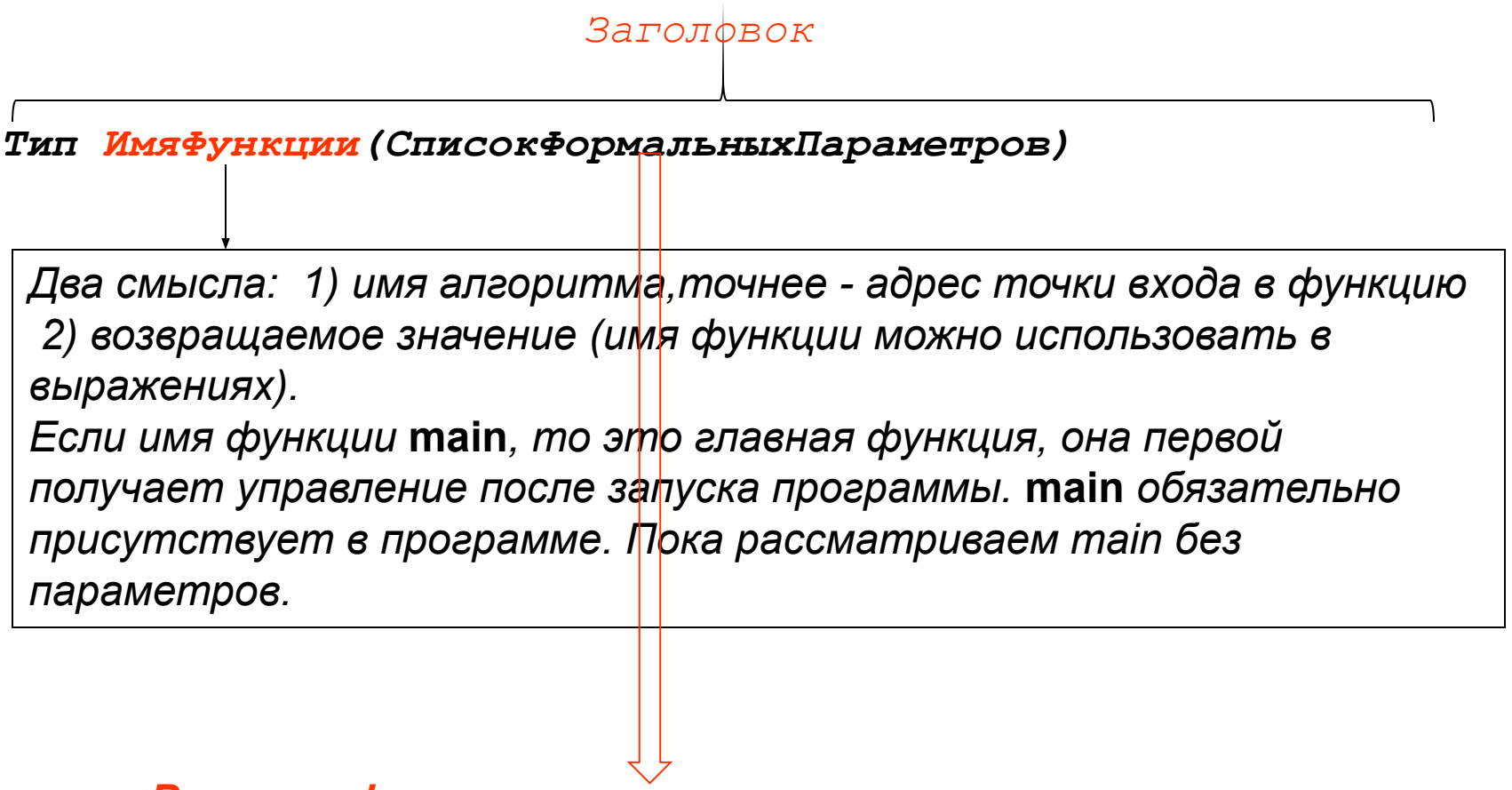
В теле основной функции формальная функция вызывается так:
(**имя_формальной_функции)(список фактических параметров)*

- В список фактических параметров подставляется указатель **имя_фактической_функции.*
  Заголовок фактической функции должна совпадать с формальным указателем на функцию с точностью до обозначений (т. е. типы функций и формальных параметров должны быть одинаковыми.

ОПИСАНИЕ ФУНКЦИИ В СИ



ОПИСАНИЕ ФУНКЦИИ В СИ



Пример 1. Решение **двух** уравнений (**в одной** программе) на отрезке [0.1, 2] с погрешностью 0.0001 (задача 1.8.N,N+1 – таблица 1).

fx1(x)

$$x^2 - 1 = 0$$

fx2(x)

$$\frac{\ln(x+1)}{0{,}001 + \sqrt[4]{x}\sin^2 x} - \frac{1}{\pi x\sqrt[3]{x}} - e^{x/7} = 0$$

Си-программа

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>
#include <locale.h>
/*fx1 и fx2 - функции, задающие левую часть
уравнений, их заголовки соответствуют указателю
на функцию вызывающей функции root */
double fx1(double x)
 {return (x*x-1);
 }
double fx2(double x)
 {return
(log(x+1)/(0.001+pow(x,1.0/4)*pow(sin(x),2))-1/(M_PI*x
*pow(x,1.0/3))-exp(x/7));
 }
```

Си-программа (продолжение)

```
/*root - функция вычисления корня уравнения f(x)=0
на отрезке [a,b] методом дихотомии*/
/*с точностью e*/
double root(double(*f)(double),double a,double b,
double e)
 {double x;

  while(fabs(b-a)>e)
  {x=(a+b)/2.0;
   if ((*f)(a)*(*f)(x)>0)
         a=x;
   else
          b=x;
   }
  x=(a+b)/2;
  return x;
 }
```

указатель на функцию, задающую левую часть уравнения

вызов формальной функции

Си-программа (продолжение)

```
void main()
 {double r1,r2; /*значения корней*/
  setlocale(LC_ALL, "");

  r1=root(*fx1,0.1,2,1e-4);

  r2=root(*fx2,0.1,2,1e-4);

  printf("корень первого уравнения=%7.4f f(r1)=%8.5f \n"
     "корень второго уравнения=%7.4f
    f(r2)=%8.5f\n",r1,fx1(r1),r2,fx2(r2));
  _getch();
 }
```

Подстановка указателя на фактическую функцию

Приближенное решение уравнения на отрезке

Известно, что уравнение

$$F(x)=0 \qquad (*)$$

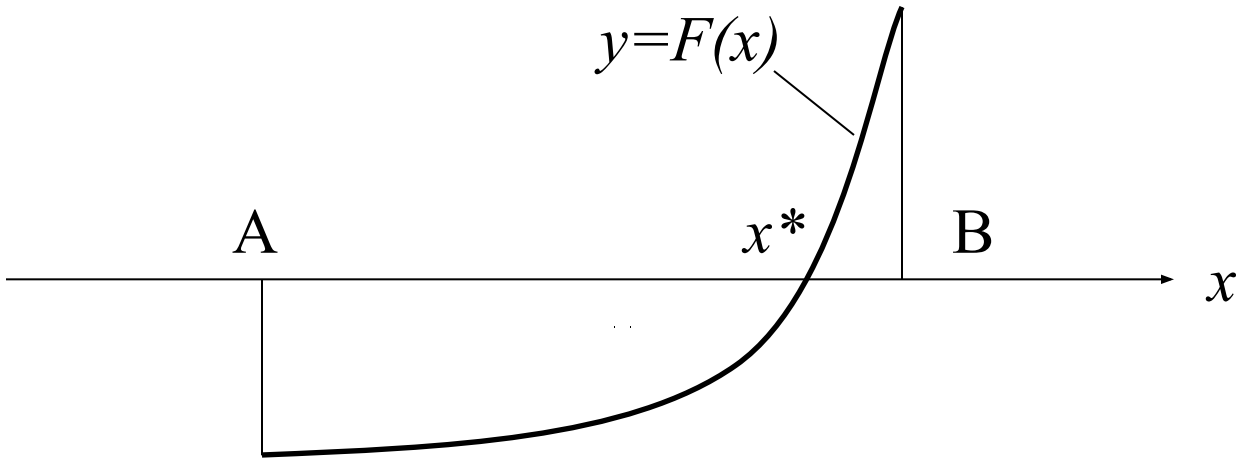
на отрезке [A,B] имеет ровно один корень.

Требуется найти приближенное значение корня с точностью ε:

$$|x^*-x_{пр}|< \varepsilon,$$

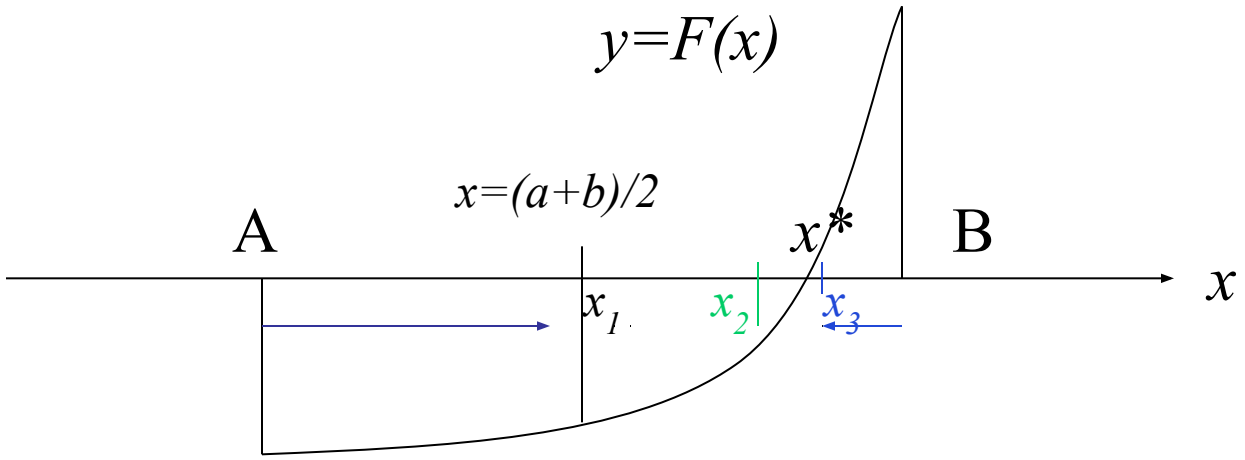
где x^* - точное значение корня,
$x_{пр}$ – приближенное значение корня.

Приближенное решение уравнения на отрезке



Если уравнение (*) имеет на отрезке [A,B] ровно один корень, то ***F(A)*F(B)≤0***.

Метод деления отрезка пополам (дихотомии)



Если $F(x)*F(A)>0$, то $x^* \notin [A,x] \Rightarrow$ корень надо искать на правой половине отрезка $x^* \in [x,B]$: **A=*x***;

иначе $x^* \in [A,x] \Rightarrow$ корень надо искать на левой половине отрезка: **B=*x*.**

Далее деление пополам нового отрезка.

Метод деления отрезка пополам (дихотомии)

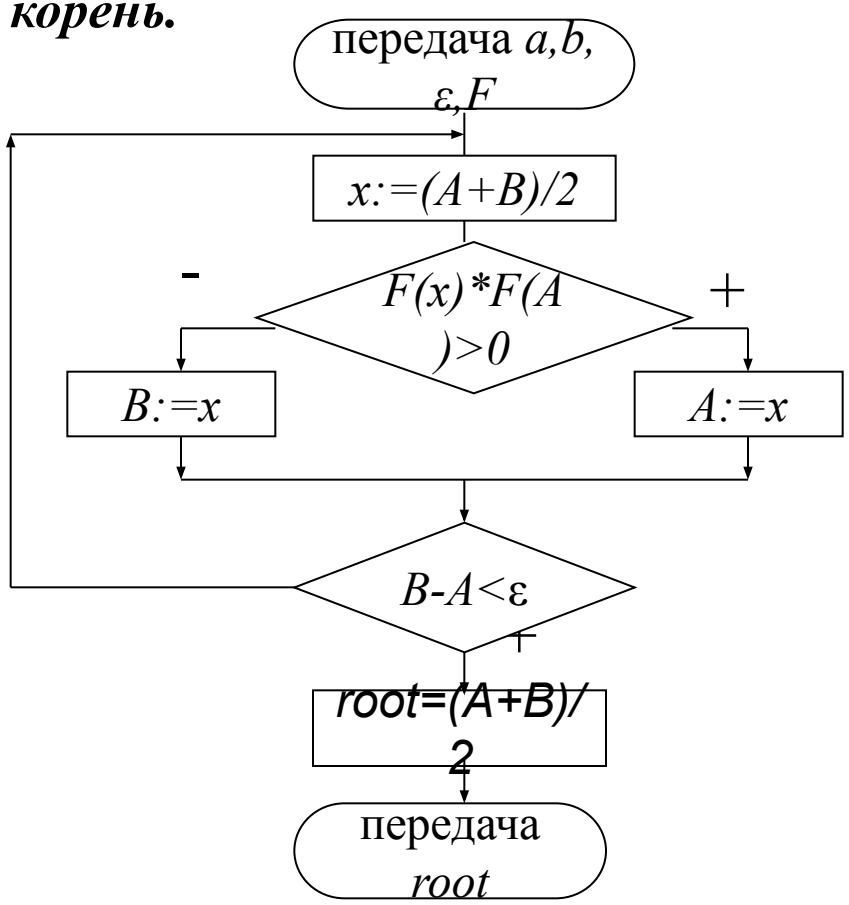
i-ая итерация (цикл): вычисление x_i - середины i-го отрезка и выбор его левой или правой половины.

$$\{x_i\} \to x^* \text{ при } i \to \infty.$$

Условие продолжения цикла: $B-A>\varepsilon$.

Метод деления отрезка пополам (дихотомии) – блок-схема функции root

Алгоритм для идеального случая: на [A,B] ровно один корень.



Можно определить число N итераций (циклов), необходимых для обеспечения погрешности ε. В конце N-го цикла длина отрезка, накрывающего корень, равна:

$$l = \frac{B - A}{2^N}.$$

Число итераций можно вычислить из соотношения:

$$l \le \varepsilon.$$

Откуда:

$$\log_2(B-A) - N \le \log_2(\varepsilon),$$

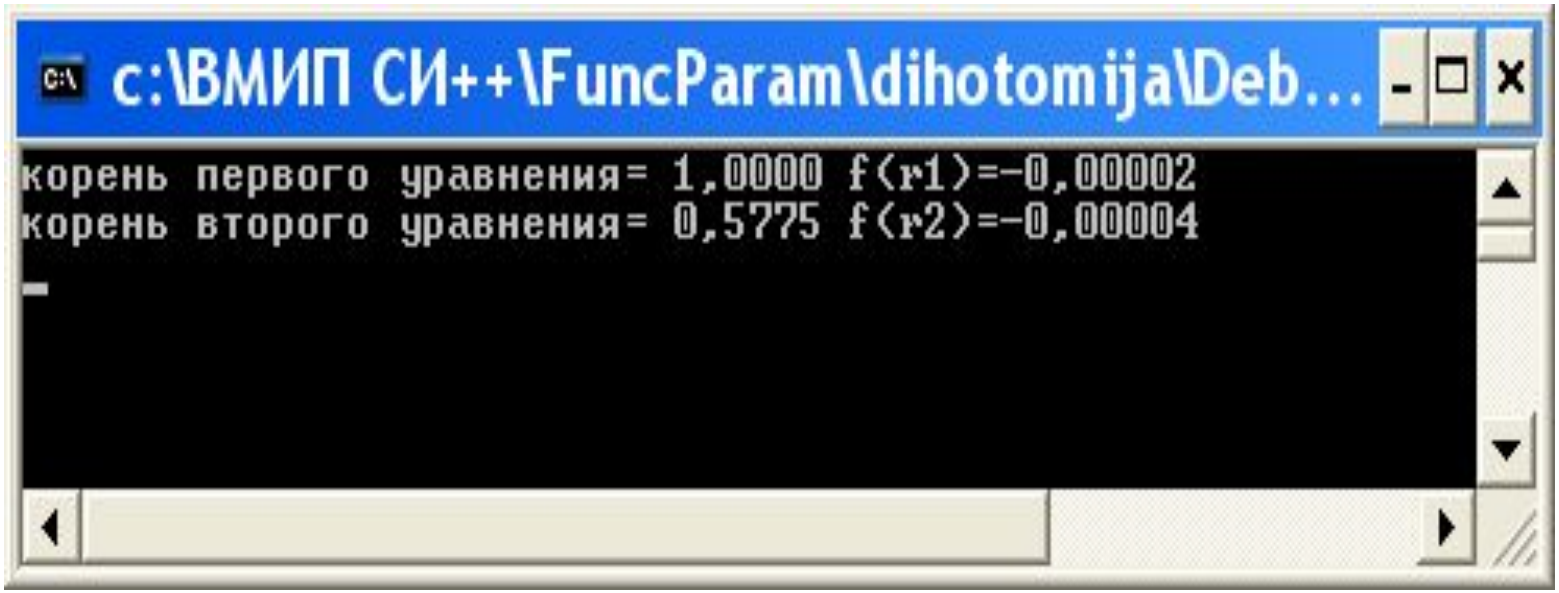
и, следовательно,

$$N = \lceil \log_2(B-A) - \log_2\varepsilon \rceil,$$

где ⌈ .⌉ - *ближайшее максимальное целое.*

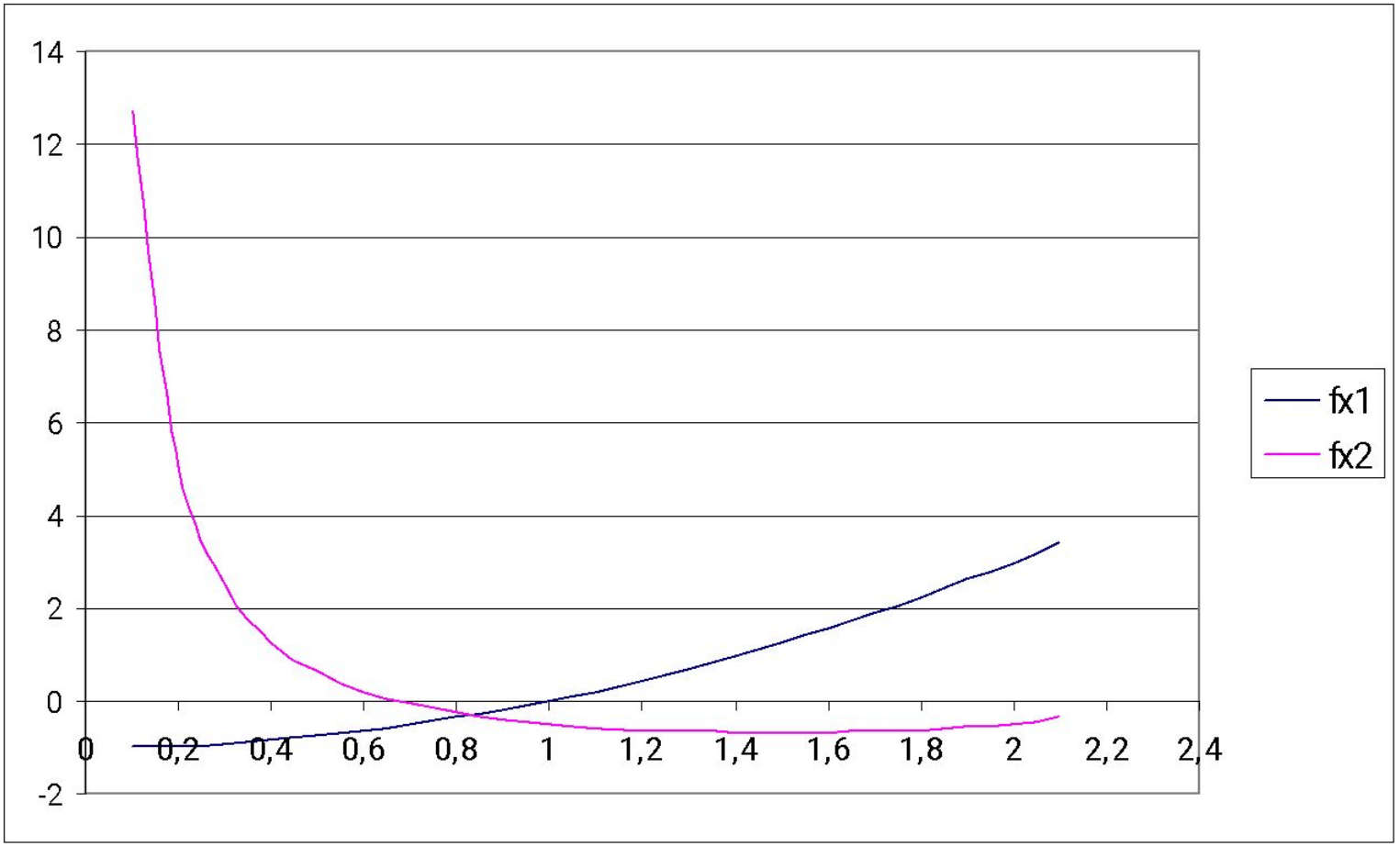
Как протестировать программу?

1. Вывести не только r1, r2, но и fx1(r1), fx2(r2). Эти значения функций должны быть близкими к нулю. Если они сильно отличаются от нуля, то программа работает неправильно. Однако их близость к нулю не гарантирует правильность программы.



Как протестировать программу?

2. Построить графики функций или решить уравнение в другой вычислительной среде



Как еще можно использовать указатели на функции

- Описывается шаблон указателя на функцию:

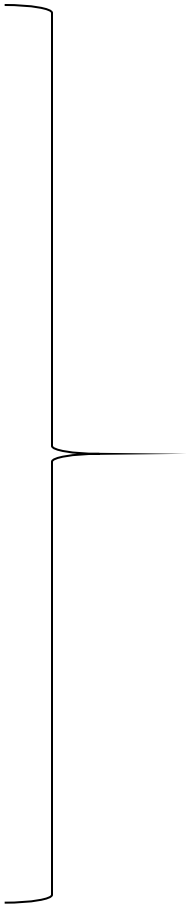
тип (**имя_функции*)(*список формальных параметров*);
/*такой функции не существует, просто объявлен шаблон*/

- *имя_функции= имя_функции_существующей*;
- Далее, когда пишется имя функции шаблона, вызывается существующая функция.

Пример 2

```
#include <conio.h>
#include <stdio.h>

int add(int a, int b)
{return (a+b);
}
int substruct(int a, int b)
{return (a-b);
}
int multiplicate(int a, int b)
{return (a*b);
}
int divide(int a, int b)
{return (a/b);
}
```



Объявлены реальные функции, соответствующие одному шаблону

Продолжение примера 2

```
void main()
{int a,b; int operation;
 int (*f)(int, int);  //объявляется шаблон
 printf("input a,b, operation\n");
 scanf_s("%d%d%d",&a,&b,&operation);
 printf("op=%d\n", operation);

 switch (operation)
 {
    case 1: f=add; break;
       case 2: f=substruct; break;
       case 3: f=multiplicate; break;
       case 4: f=divide; break;
       default:
              puts("no such operation");
 }
 printf("f(a,b)=%d\n", f(a,b));    //вызывается выбранная функция
 _getch();
}
```