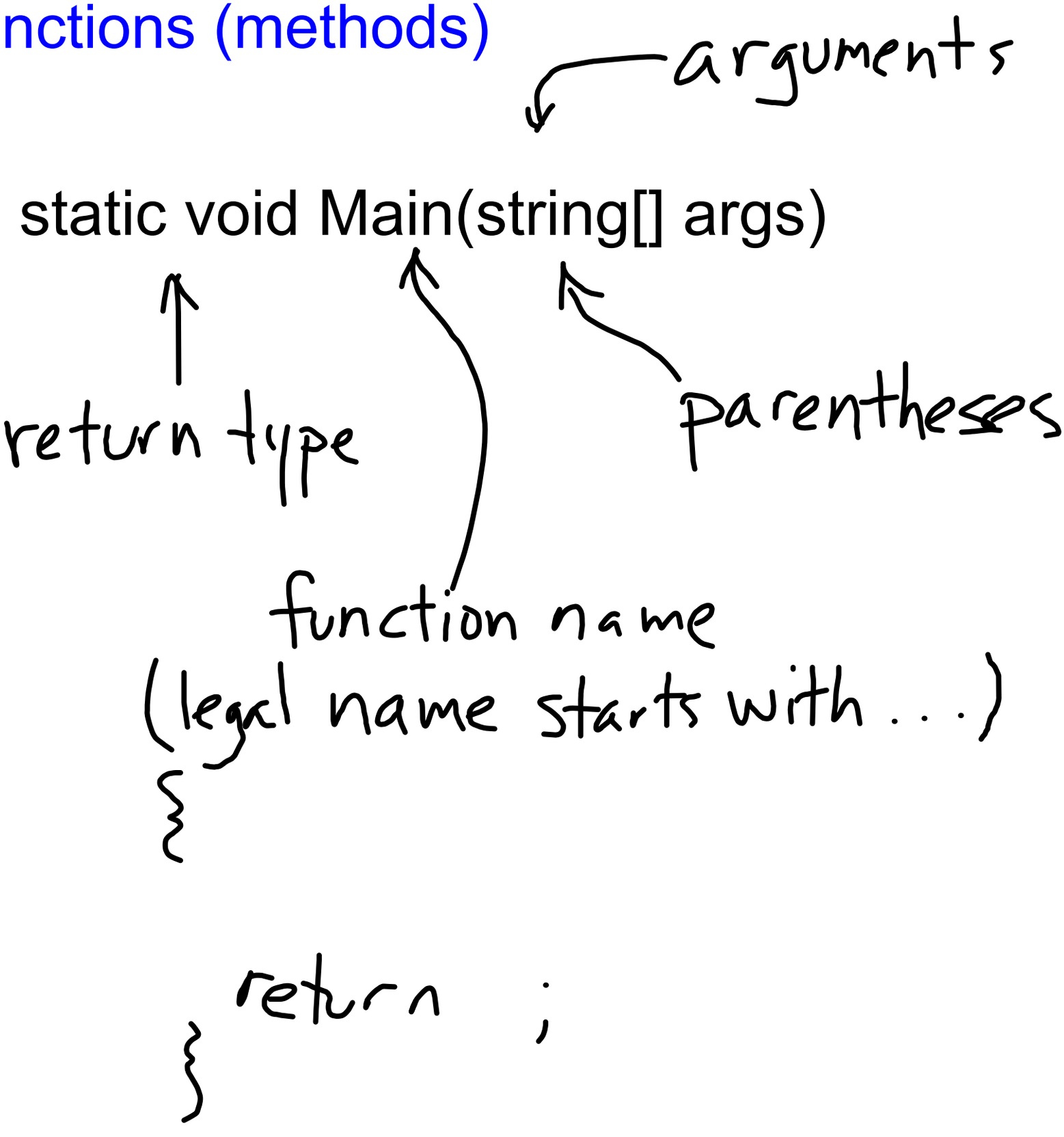
Functions (methods)



Why use functions

1) Break problem into smaller pieces
2) Modularization — reuse
3) specific tasks
4) (save keystrokes)

Functions

```
static void Main(string[] args)
 {
        float  ans, a, b, c .... j
          :// a = 3.0, b = 1.5;
        ans = fn1(a, b); //fn call
          :
        return ;
 }

static float fn1(float a1, float a2)
     { return a1/a2; } // fn definition
```

Functions

```
static float f1(float a1, float a2)
{
    float intermediate; //local var
    intermediate = a1 * 3.0;
    intermediate += a2;
    a2 = 6;
    return intermediate;
}
```

Functions

call functions from other fns

every program must have a main()

not every fn has arguments

every fn must have a return type

→ void, float, int, bool, double, etc

needs no return

normally do a specific job

→ usually used more than once

Functions

Scope of variables/constants

variables/constants declared in a fn are local

identifiers declared outside a fn are global → stay away from ↓ poor prog practice

Functions

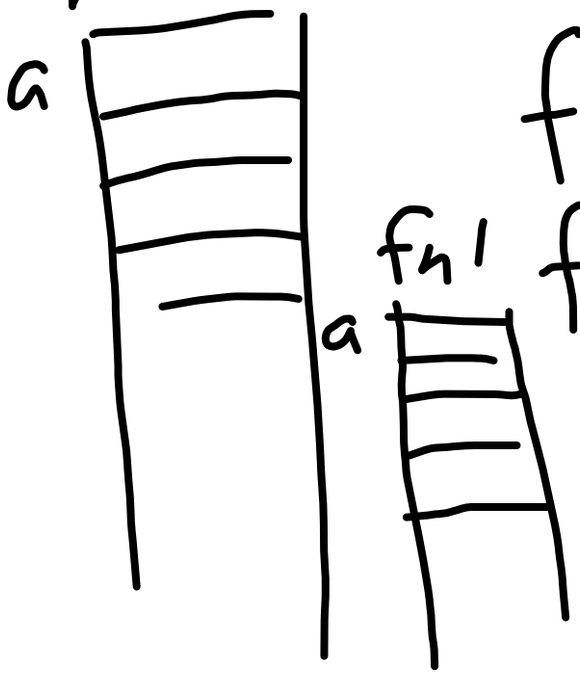
```
static void directions( )
{
    printf("This program calculates");
                    .
                    .
                    .
    printf("    .....    ");
}
```

Functions

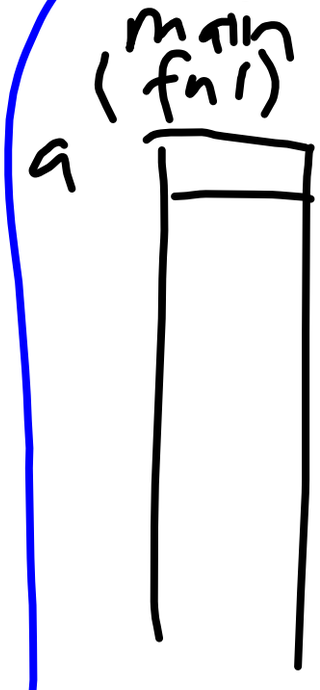
passing data

- by value
- by reference (allows var to change)

```
int fn1(int a);
fn1(3);
fn1(a);
```




```
int fn1(ref int a);
fn1(3);
fn1(ref a);
```



Functions

```
namespace FunctionExample;
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, World!");
        int a = 4;
        Console.WriteLine($"a = {a}");
        int ans = fn1(ref a);
        Console.Write($"a = {a}\nans = {ans}");
        Console.Write("\n\nPress the any key to continue...");
        Console.ReadKey();
    }
    static int fn1(ref int a)
    {
        a = a * 3;
        int b = a - 7;
        return b;
    }
}
```