Memory allocation

Value and reference types in c#

EVERYLOOP

www.everyloop.com

Common type system (CTS)

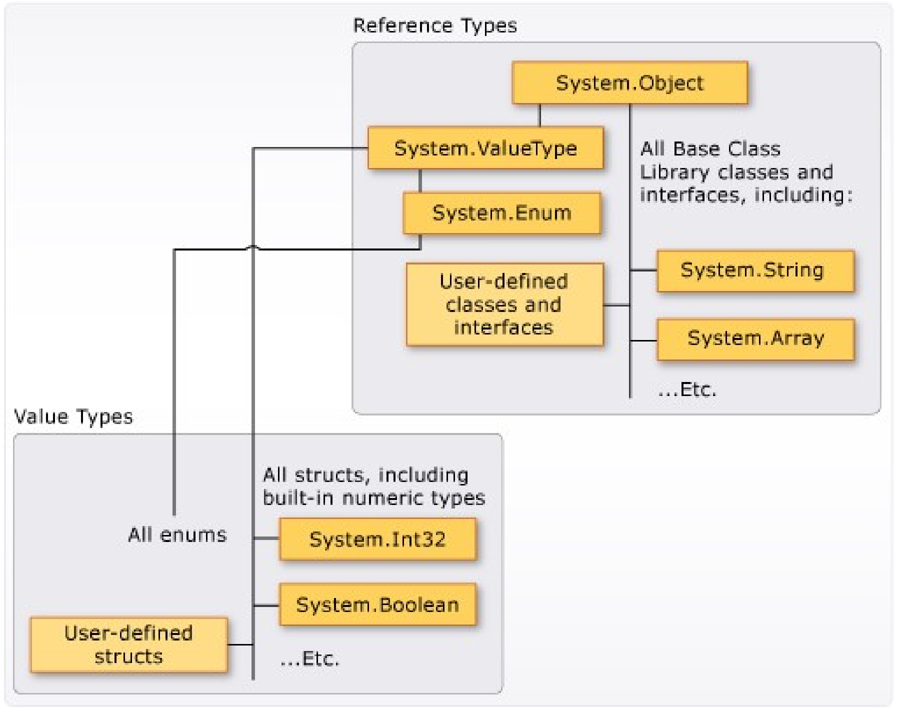
defines how types are declared, used, and managed in the common language runtime

All types in .NET are either value types or reference types.

Value types are data types whose objects are represented by the object's actual value. If an instance of a value type is assigned to a variable, that variable is given a fresh copy of the value.

Reference types are data types whose objects are represented by a reference to the object's actual value. If a reference type is assigned to a variable, that variable references (points to) the original value. No copy is made.

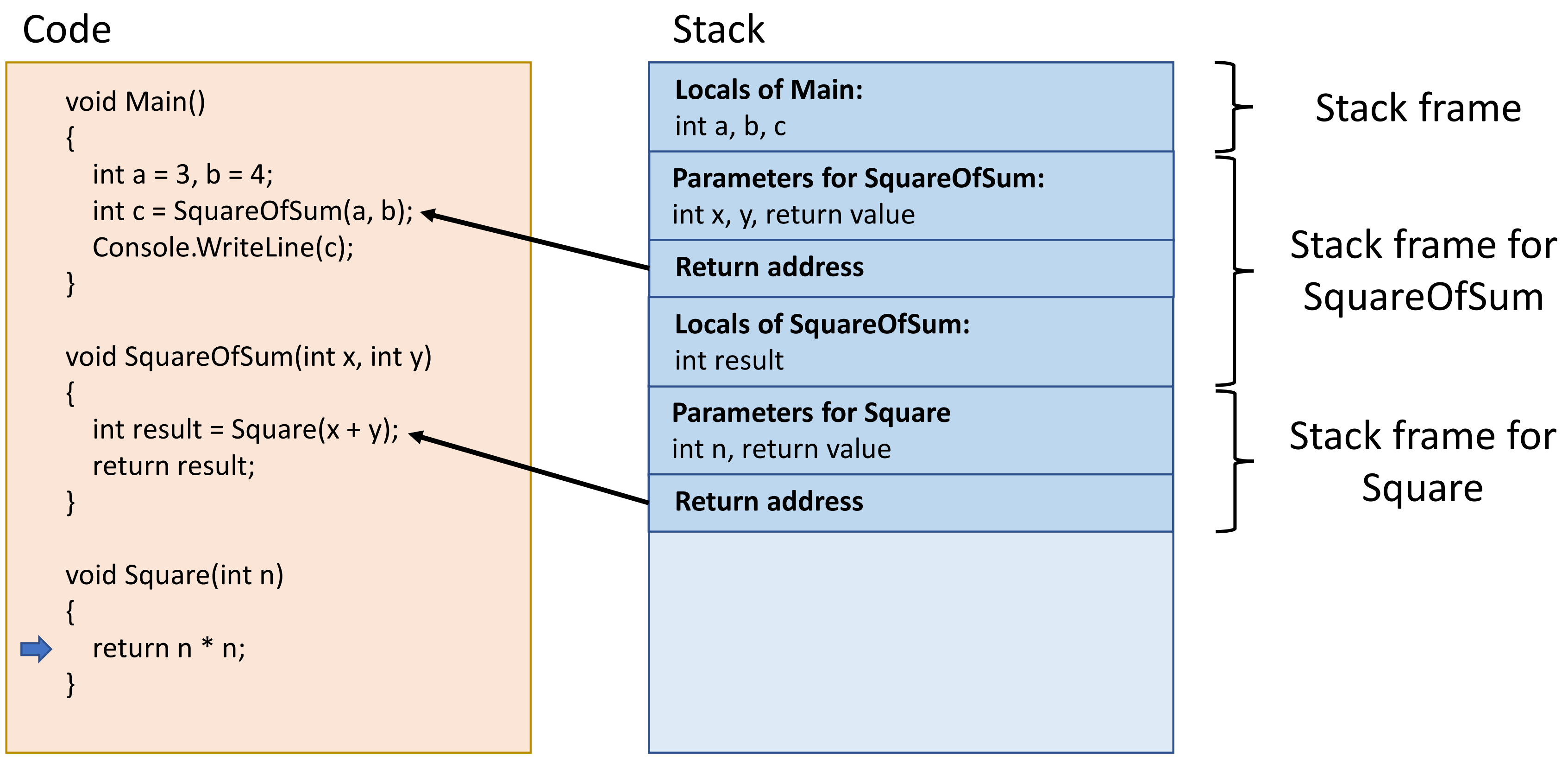
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/



Memory allocation

The process of reserving portions of computer memory for execution of a program

Static memory



Code

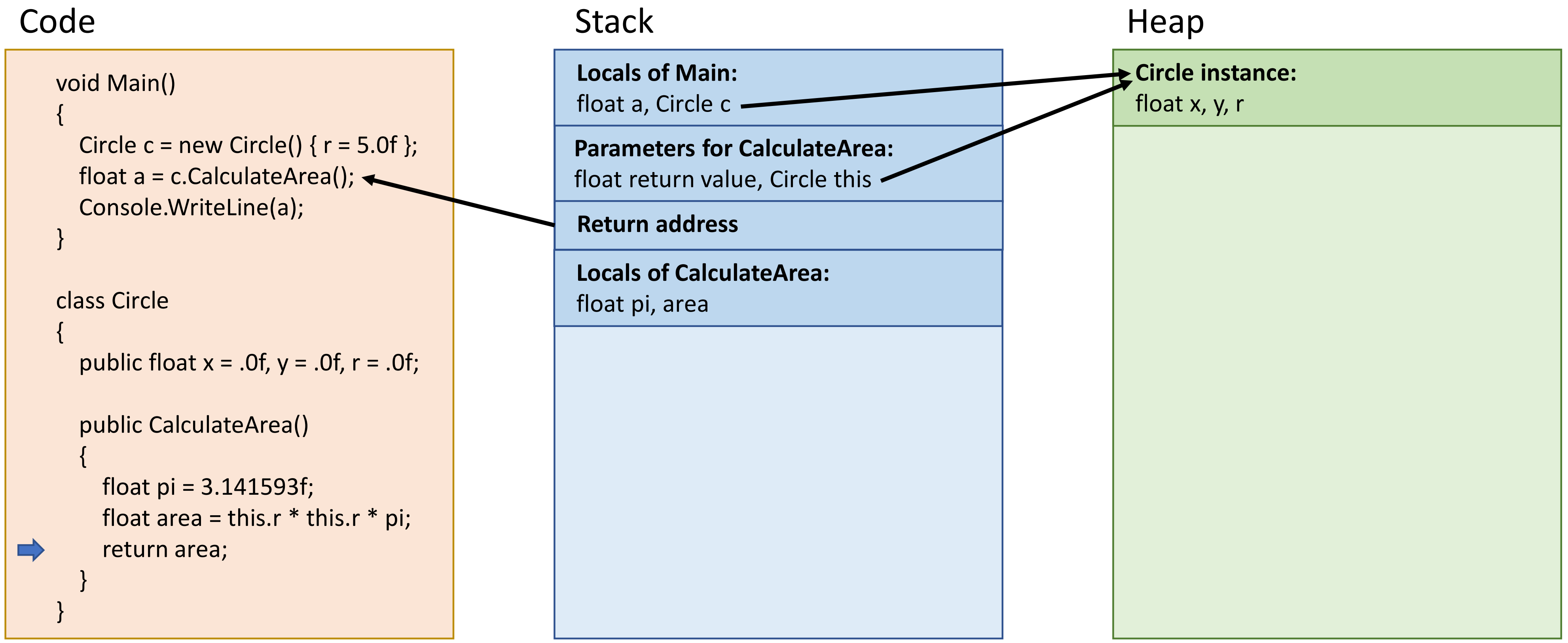
```
void Main()
{
   int a = 3, b = 4;
   int c = SquareOfSum(a, b);
   Console.WriteLine(c);
}

void SquareOfSum(int x, int y)
{
   int result = Square(x + y);
   return result;
}

void Square(int n)
{
   return n * n;
}
```

Memory allocation

The process of reserving portions of computer memory for execution of a program

Static memory

Dynamic memory



Code

```
void Main()
{
   Circle c = new Circle() { r = 5.0f };
   float a = c.CalculateArea();
   Console.WriteLine(a);
}

class Circle
{
   public float x = .0f, y = .0f, r = .0f;

   public CalculateArea()
   {
      float pi = 3.141593f;
      float area = this.r * this.r * pi;
      return area;
   }
}
```

Passing parameters

Pass by value

The default way to pass parameters in C#

Creates a *copy* of the passed parameter
that will be used in the called method.

If the called method modifies the parameter,
the value of the original variable in the
calling method remains unchanged.

When passing a value type, the value is
copied to the parameter of called method.

When passing a reference type, the reference
to the instance of the type is copied to the
parameter of the called method.

Pass by reference

Only when the ***ref*** or ***out*** keywords are specified.

Creates a *reference* to the passed parameter
that will be used in the called method.

If the called method modifies the parameter,
the value of the original variable in the
calling method will change as well.

When passing a value type, a reference to
the original variable is used in the called method.

When passing a reference type, a reference to
the original reference is used in the called method.