

4. Non-Primitive Types

5. Enums

6. Value Types

7. Reference Types

Enum

A set of name/value pairs (constants).

Use Enums where we need to define related constants.



```
const int RegularAirMail = 1;  
const int RegisteredAirMail = 2;  
const int Express = 3;
```

```
public enum ShippingMethod  
{  
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3;  
}
```

```
const int RegularAirMail = 1;
const int RegisteredAirMail = 2;
const int Express = 3;

public enum ShippingMethod
{
    RegularAirMail = 1,
    RegisteredAirMail = 2,
    Express = 3;
}

var method = ShippingMethod.Express;
```

```
const int RegularAirMail = 1;  
const int RegisteredAirMail = 2;  
const int Express = 3;
```

```
public enum ShippingMethod : byte  
{  
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3;  
}
```

```
var method = ShippingMethod.Express;
```

By default enums
are integers

```
namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail,
        RegisteredAirMail,
        Express
    }
}

class Program
{
    static void Main(string[] args)
    {
    }
}
```

As enum is a new kind of user define data type – we need to define it at namespace level.

If we do not assign any value to enum at the time of Declaration – first one be 1 and rest will be incremented by one.

Program.cs

CSharpFundamentals.Program

```
namespace CSharpFundamentals
{
    public enum ShippingMethod
    {
        RegularAirMail = 1,
        RegisteredAirMail = 2,
        Express = 3
    }

    class Program
    {
        static void Main(string[] args)
        {
            var method = ShippingMethod.Express;
        }
    }
}
```

Program.cs • X

CSharpFundamentals.Program

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    public enum ShippingMethod
```

```
    {
```

```
        RegularAirMail = 1,
```

```
        RegisteredAirMail = 2,
```

```
        Express = 3
```

```
    }
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

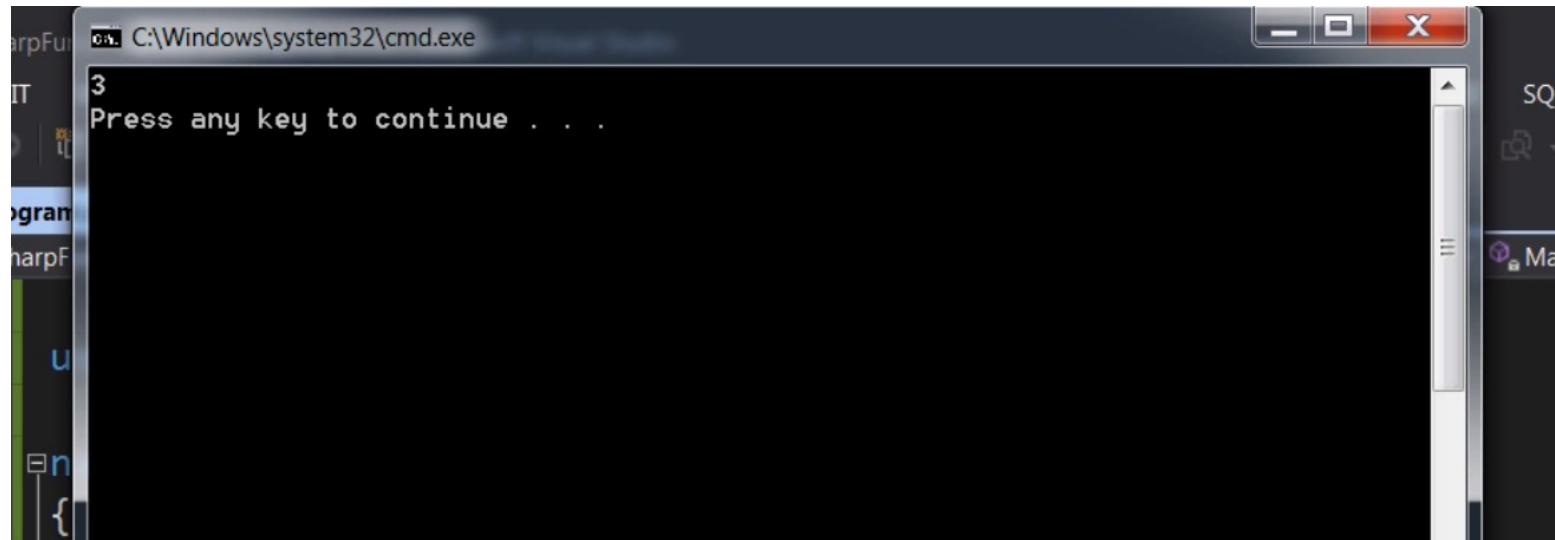
```
            var method = ShippingMethod.Express;
```

```
            Console.WriteLine((int)method);
```

```
        }
```

```
    }
```

```
}
```

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    public enum ShippingMethod
```

```
    {
```

```
        RegularAirMail = 1,
```

```
        RegisteredAirMail = 2,
```

```
        Express = 3
```

```
    }
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            var method = ShippingMethod.Express;
```

```
            Console.WriteLine((int)method);
```

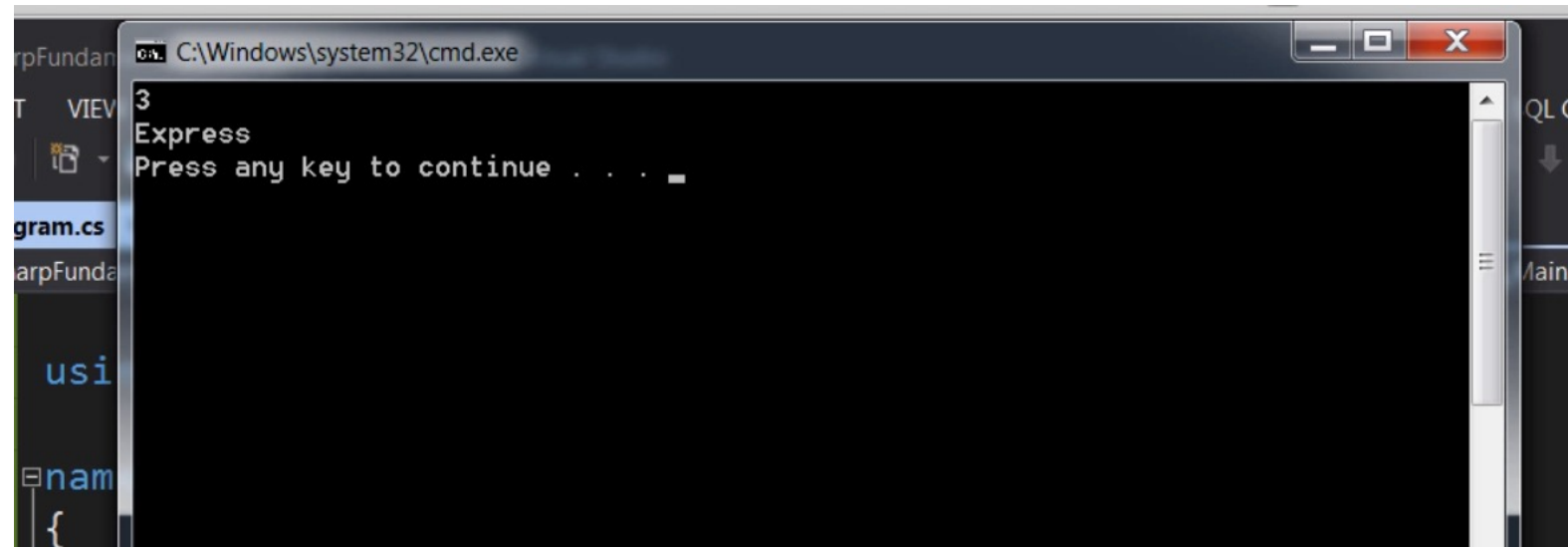
```
            var methodId = 3;
```

```
            Console.WriteLine((ShippingMethod)methodId);
```

```
        }
```

```
    }
```

```
}
```



```
C:\Windows\system32\cmd.exe
```

```
3  
Express  
Press any key to continue . . . _
```

```
rpFundan  
T VIEW  
gram.cs  
arpFundan  
usi  
nam  
{
```

```
QL CO  
↓  
Main(s
```

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    public enum ShippingMethod
```

```
    {
```

```
        RegularAirMail = 1,
```

```
        RegisteredAirMail = 2,
```

```
        Express = 3
```

```
    }
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            var method = ShippingMethod.Express;
```

```
            Console.WriteLine((int)method);
```

```
            var methodId = 3;
```

```
            Console.WriteLine((ShippingMethod)methodId);
```

```
            Console.WriteLine(method.ToString());
```

```
        }
```

```
    }
```

```
}
```

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    public enum ShippingMethod
```

```
{
```

```
        RegularAirMail = 1,
```

```
        RegisteredAirMail = 2,
```

```
        Express = 3
```

```
}
```

```
    class Program
```

```
{
```

```
        static void Main(string[] args)
```

```
{
```

```
            var method = ShippingMethod.Express;
```

```
            Console.WriteLine((int)method);
```

```
            var methodId = 3;
```

```
            Console.WriteLine((ShippingMethod)methodId);
```

```
            Console.WriteLine(method);
```

```
}
```

```
}
```

```
}
```

```
ace CSharpFundamentals
```

```
blic enum ShippingMethod
```

```
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3
```

```
ass Program
```

```
static void Main  
{
```

```
    var method  
    Console.WriteLine
```

```
    var method  
    Console.WriteLine
```

```
    Console.WriteLine
```

```
    var method  
    Enum.Parse()
```

([NotNull] Type enumType, [NotNull] string value):object

Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

enumType: An enumeration type.

([NotNull] Type enumType, [NotNull] string value, bool ignoreCase):object

global

(?) ignoreCase:

int

long

method

methodId

methodName

new

null

object

out

Local variable string methodName


```
namespace CSharpFundamentals
```

```
public enum ShippingMethod
```

```
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3
```

```
class Program
```

```
static void Main  
{
```

```
    var method = ShippingMethod.Express;  
    Console.WriteLine((int)method);
```

```
    var methodId = 3;  
    Console.WriteLine((ShippingMethod)methodId);
```

```
    Console.WriteLine(method.ToString());
```

```
    var methodName = "Express";  
    Enum.Parse(typeof(ShippingMethod), methodName, true);
```

([NotNull] Type enumType, [NotNull] string value):object

Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

value: A string containing the name or value to convert.

([NotNull] Type enumType, [NotNull] string value, bool ignoreCase):object

global
(ignoreCase:
int
long
method
methodId
methodName
new
null
object
out

Local variable string methodName

```
namespace CSharpFundamentals
```

```
public enum ShippingMethod
```

```
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3
```

```
class Program
```

```
    static void Main(string[] args)  
    {  
        var method = ShippingMethod.Express;  
        Console.WriteLine((int)method);  
  
        var methodId = 3;  
        Console.WriteLine((ShippingMethod)methodId);
```

type enumType, [NotNull] string value):object

the string representation of the name or numeric value of one or
generated constants to an equivalent enumerated object.
string containing the name or value to convert.

type enumType, [NotNull] string value, bool ignoreCase):object

```
Enum.Parse(typeof(ShippingMethod), methodName)
```

```
String());
```



```
ace CSharpFundamentals
```

```
public enum ShippingMethod
```

```
    RegularAirMail = 1,  
    RegisteredAirMail = 2,  
    Express = 3
```

```
class Program
```

```
    static void Main(string[] args)  
    {
```

```
        var method = ShippingMethod.Express;  
        Console.WriteLine((int)method);
```

```
        var methodId = 3;  
        Console.WriteLine((ShippingMethod)methodId);
```

```
        Console.WriteLine(method.ToString());
```

```
        var methodName = "Express";  
        (ShippingMethod)Enum.Parse(typeof(ShippingMethod), methodName)
```

```
RegularAirMail = 1,  
RegisteredAirMail = 2,  
Express = 3
```

class Program

```
static void Main(string[] args)  
{  
    var method = ShippingMethod.Express;  
    Console.WriteLine((int)method);  
  
    var methodId = 3;  
    Console.WriteLine((ShippingMethod)methodId);  
  
    Console.WriteLine(method.ToString());  
  
    var methodName = "Express";  
    var shippingMethod = (ShippingMethod) Enum.Parse(typeof (ShippingMethod), methodName);  
    enum CSharpFundamentals.ShippingMethod  
  
}
```

Types

- int
- char
- float
- bool
- classes
- structures
- arrays (`System.Array`)
- strings (`System.String`)

Types

Structures

- Primitive types
- Custom structures

Classes

- Arrays
- Strings
- Custom classes

Value Types

Structures

- Allocated on stack
- Memory allocation done automatically
- Immediately removed when out of scope

Reference Types

Classes

- You need to allocate memory
- Memory allocated on heap
- Garbage collected by CLR

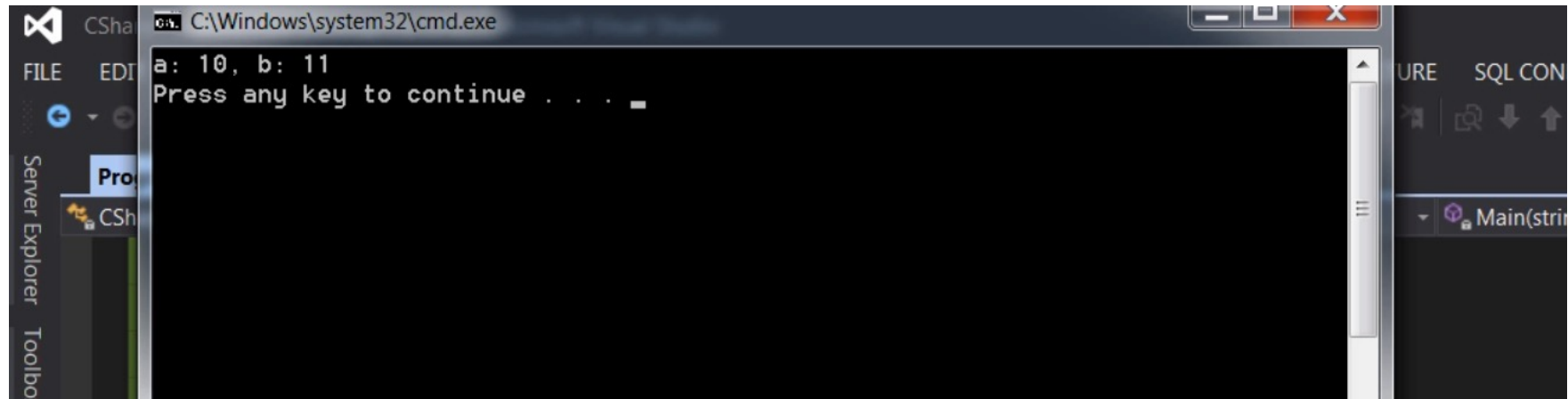
Copying value types and Reference Types

```
var anotherObject = someObject;
```

```
Program.cs • X
CSharpFundamentals.Program Main(string[] args)

using System;

namespace CSharpFundamentals
{
    class Program
    {
        static void Main(string[] args)
        {
            var a = 10;
            var b = a;
            b++;
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));
        }
    }
}
```



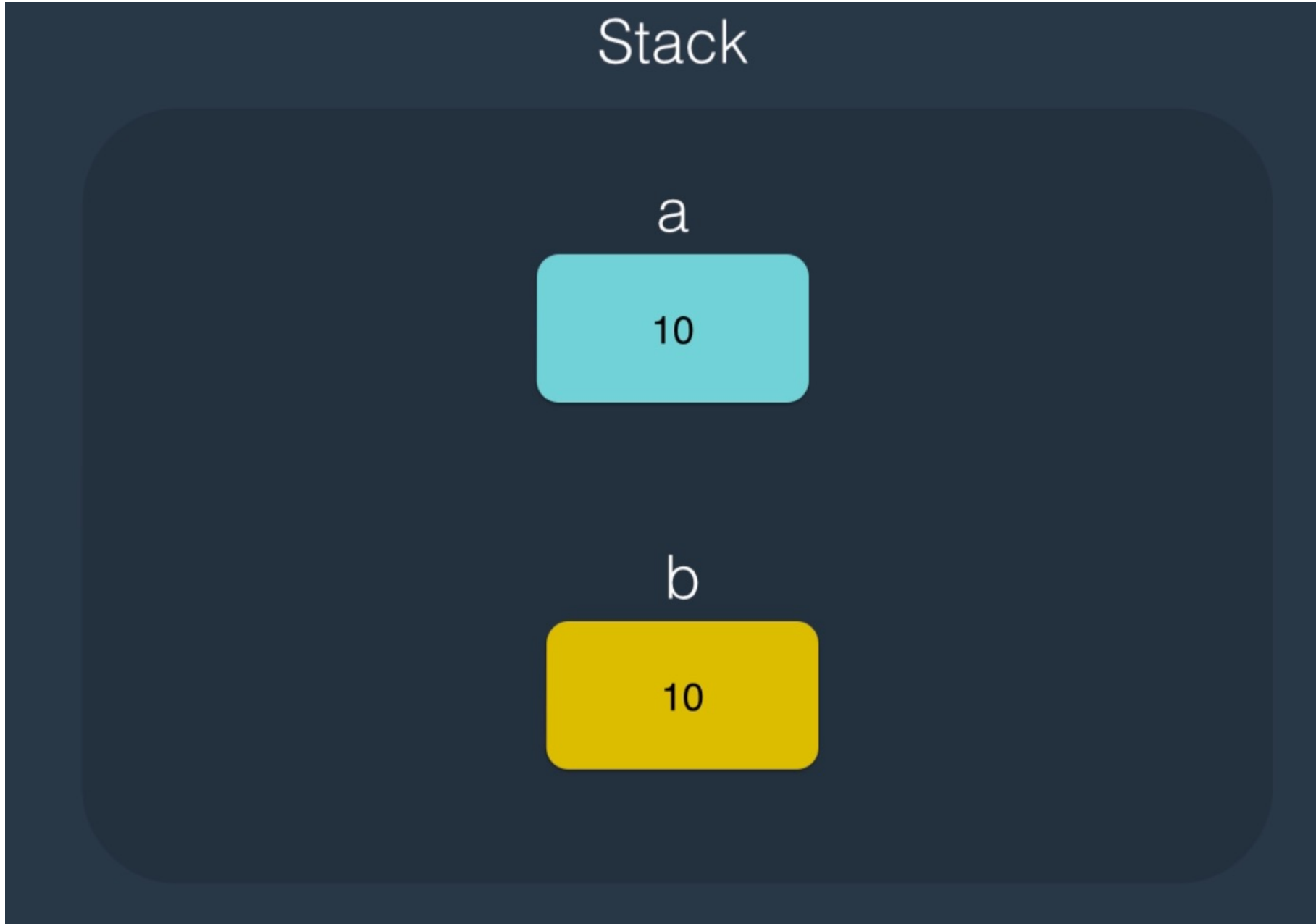
Stack

a

10

b

10



Program.cs

CSharpFundamentals.Program

Main(string[] args)

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            var a = 10;
```

```
            var b = a;
```

```
            b++;
```

```
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));
```

```
            var array1 = new int[3] {1, 2, 3};
```

```
            var array2 = array1;
```

```
            array2[0] = 0;
```

```
        }
```

```
    }
```

```
}
```

Stack

Heap

▶ 0x00416A

1	2	3
---	---	---

Stack

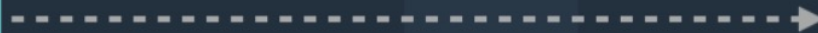
array1

0x00416A

Heap

0x00416A

1	2	3
---	---	---



Stack

array1

0x00416A

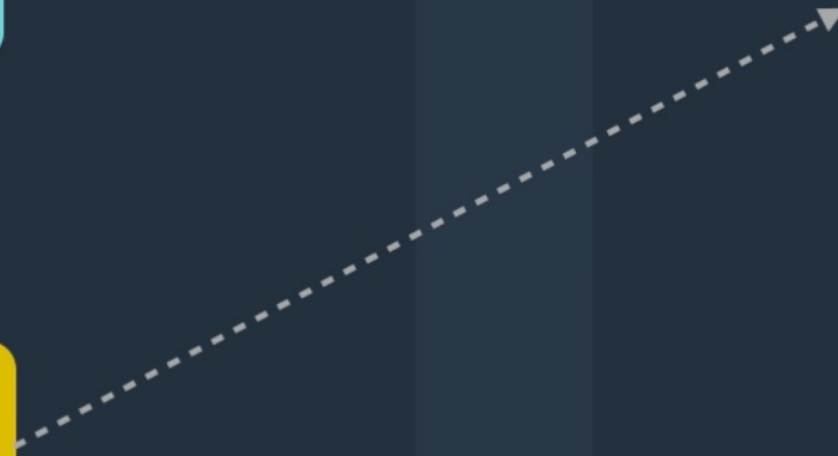
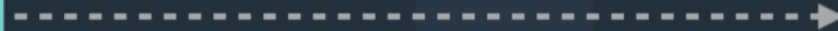
array2

0x00416A

Heap

0x00416A

1	2	3
---	---	---



Program.cs

CSharpFundamentals.Program

Main(string[] args)

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            var a = 10;
```

```
            var b = a;
```

```
            b++;
```

```
            Console.WriteLine(string.Format("a: {0}, b: {1}", a, b));
```

```
            var array1 = new int[3] {1, 2, 3};
```

```
            var array2 = array1;
```

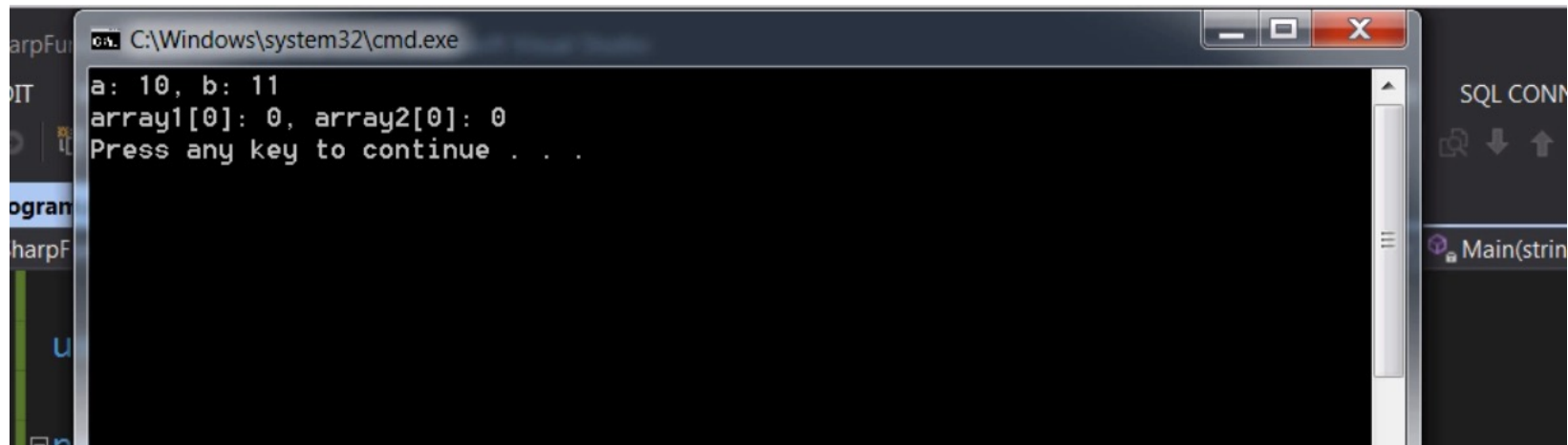
```
            array2[0] = 0;
```

```
            Console.WriteLine(string.Format("array1[0]: {0}, array2[0]: {1}", array1[0], array2[0]));
```

```
        }
```

```
    }
```

```
}
```



The image shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a dark background and a light gray title bar with standard Windows window controls (minimize, maximize, close). The output text is as follows:

```
a: 10, b: 11  
array1[0]: 0, array2[0]: 0  
Press any key to continue . . .
```

On the left side of the image, parts of other application windows are visible, including "arpFun", "IT", "ogram", "harpF", and "U". On the right side, a portion of a dark-themed application window is visible, showing the text "SQL CONN" and "Main(strin" with some navigation icons.

```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
}
```



```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
}
```

```
namespace CSharpFundamentals
{
    public class Person
    {
        public int Age;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var number = 1;
            Increment(number);
        }

        public static void Increment(int number)
        {
            number += 10;
        }

        public static void MakeOld(Person person)
        {
            person.Age += 10;
        }
    }
}
```

```
using System;
```

```
namespace CSharpFundamentals
```

```
{
```

```
    public class Person
```

```
    {
```

```
        public int Age;
```

```
    }
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            var number = 1;
```

```
            Increment(number);
```

```
            Console.WriteLine(number);
```

```
        }
```

```
        public static void Increment(int number)
```

```
        {
```

```
            number += 10;
```

```
        }
```

```
        public static void MakeOld(Person person)
```

```
        {
```

```
            person.Age += 10;
```



Program.cs • X

CSharpFundamentals.Program

```
static void Main(string[] args)
{
    var number = 1;
    Increment(number);
    Console.WriteLine(number);

    var person = new Person() {Age = 20};
}

public static void Increment(int number)
{
    number += 10;
}

public static void MakeOld(Person person)
{
    person.Age += 10;
}
} // Program
```

Program.cs

CSharpFundamentals.Program

```
static void Main(string[] args)
{
    var number = 1;
    Increment(number);
    Console.WriteLine(number);

    var person = new Person() {Age = 20};
}

public static void Increment(int number)
{
    number += 10;
}

public static void MakeOld(Person person)
{
    person.Age += 10;
}
}
```

Program

Program.cs

CSharpFundamentals.Program

```
static void Main(string[] args)
{
    var number = 1;
    Increment(number);
    Console.WriteLine(number);

    var person = new Person() {Age = 20};
}

public static void Increment(int number)
{
    number += 10;
}

public static void MakeOld(Person person)
{
    person.Age += 10;
}
}
Program
```

Program.cs

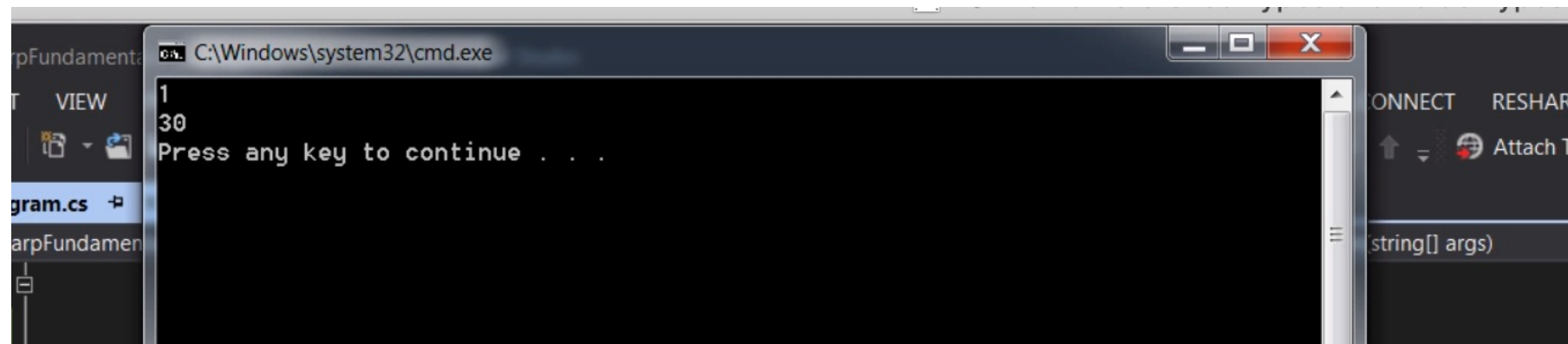
CSharpFundamentals.Program

```
static void Main(string[] args)
{
    var number = 1;
    Increment(number);
    Console.WriteLine(number);

    var person = new Person() {Age = 20};
    MakeOld(person);
    Console.WriteLine(person.Age);
}

public static void Increment(int number)
{
    number += 10;
}

public static void MakeOld(Person person)
{
    person.Age += 10;
}
}
```

Classes

Strings
Arrays
Custom Classes

Structures

Primitive Types
Custom Structures