Parameters (on stack)

(by value, by reference, by reference out)

Stack
(local variables)

Heap
(objects & arrays)

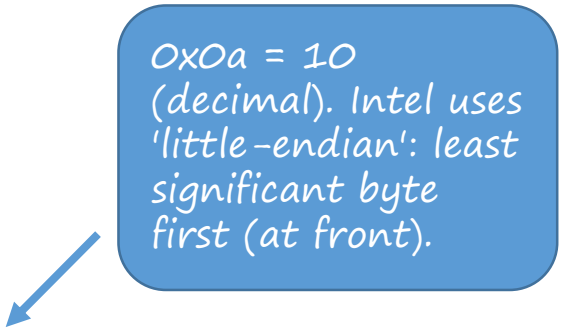
Memory space is allocated for an integer (32 bits) variable ('age') on the stack.

```
0xBE8A0D40    0x00000000    (age)
```

```
static void Main(string[] args)
{
    int age;
}
```

Stack
(local variables)

Heap
(objects & arrays)



0xBE8A0D40 0x0a000000 (age)

```
static void Main(string[] args)
{
    int age;

    age = 10;
}
```

Stack
(local variables)

Heap
(objects & arrays)

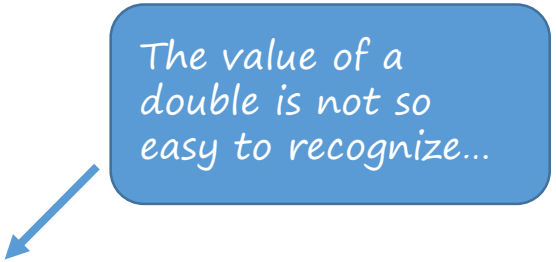
A double is 64 bits, twice as 'big' as an integer variable.

```
0xBE8A0D48    0x00000000    (length)
0xBE8A0D44    0x00000000
0xBE8A0D40    0x00000000    (age)
```

```
static void Main(string[] args)
{
    int age;
    double length;
}
```

Stack
(local variables)

Heap
(objects & arrays)




```
0xBE8A0D48    0xae47e17a    (length)
0xBE8A0D44    0x14aef73f
0xBE8A0D40    0x0a000000    (age)
```

```
static void Main(string[] args)
{
    int age;
    double length;

    age = 10;
    length = 1.48;
}
```

Stack
(local variables)

Heap
(objects & arrays)

0xBE8A0D40 0x0a000000 (age)

```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(int a)
{
    a = a + 1;
}
```

← BY VALUE

Now we'll take a look what happens with a 'by value'-parameter of a method.

Stack
(local variables)

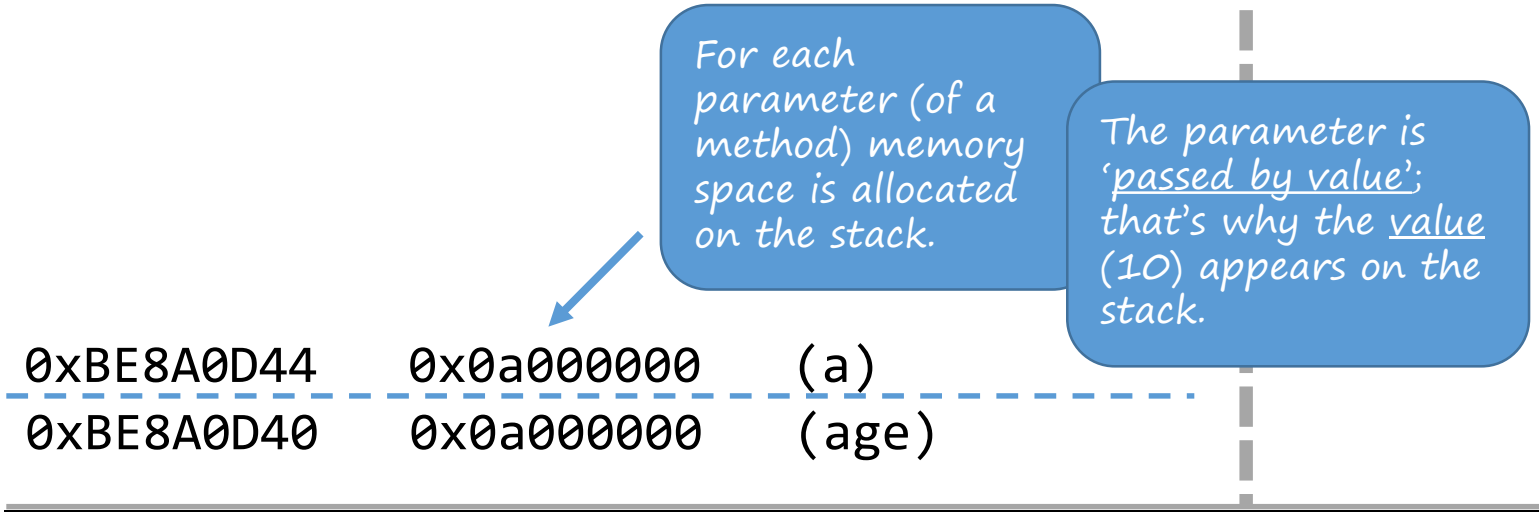
Heap
(objects & arrays)

0xBE8A0D40 0x0a000000 (age)

```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(int a)
{
    a = a + 1;
}
```

Stack
(local variables)

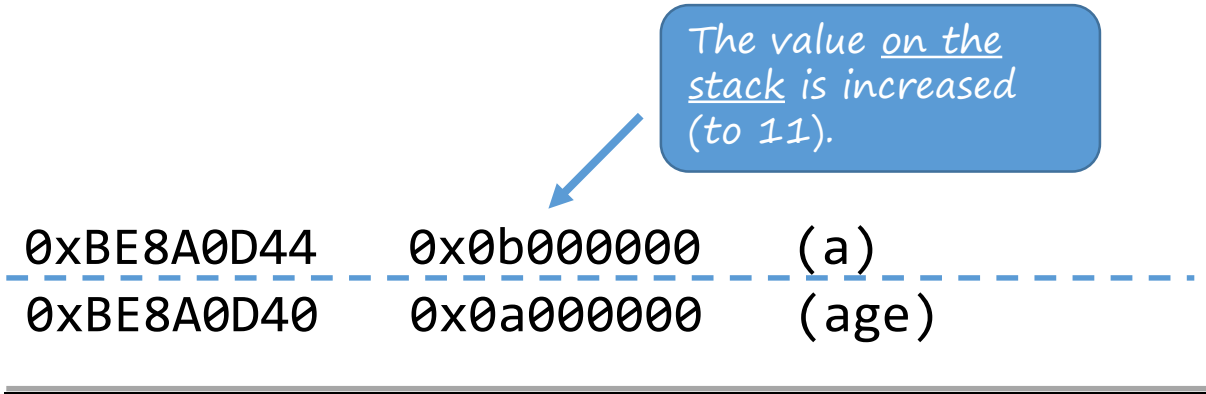
Heap
(objects & arrays)




```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)



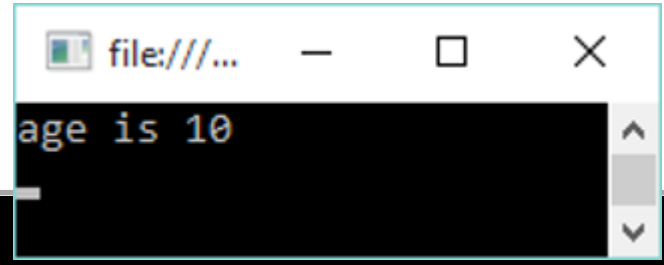

```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)

The parameter is removed from stack when the method is done.

0xBE8A0D40 0x0a000000 (age)




```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)

0xBE8A0D40 0x0a000000 (age)

```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(ref age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(ref int a)
{
    a = a + 1;
}
```

← BY REFERENCE

Now we'll take a look what happens with a 'by reference'-parameter of a method.

Stack
(local variables)

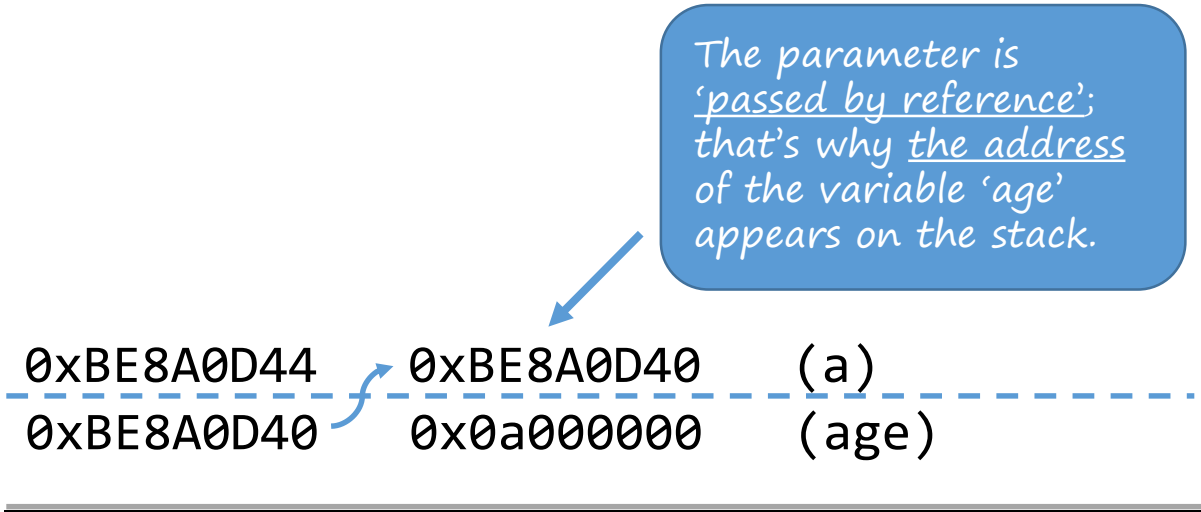
Heap
(objects & arrays)

0xBE8A0D40 0x0a000000 (age)

```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(ref age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(ref int a)
{
    a = a + 1;
}
```

Stack
(local variables)

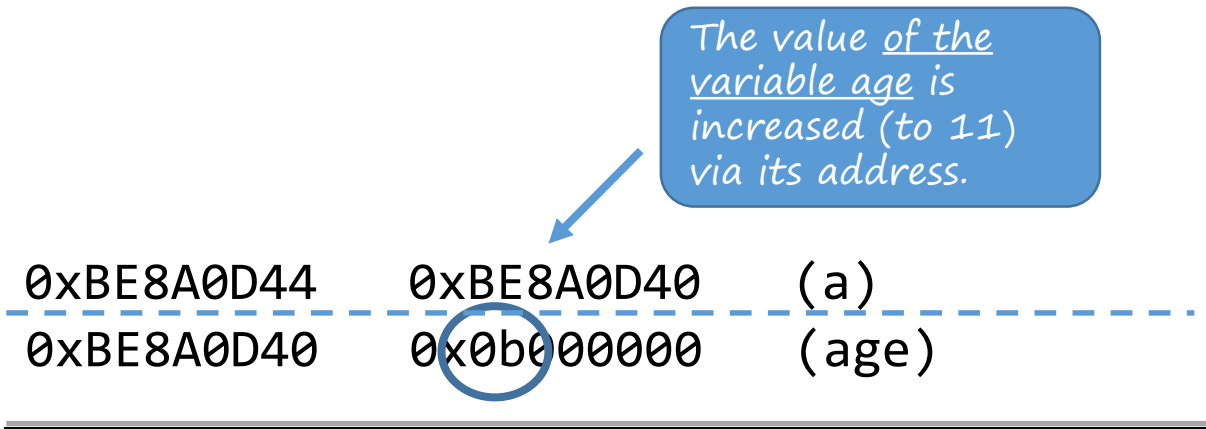
Heap
(objects & arrays)




```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(ref age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(ref int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)

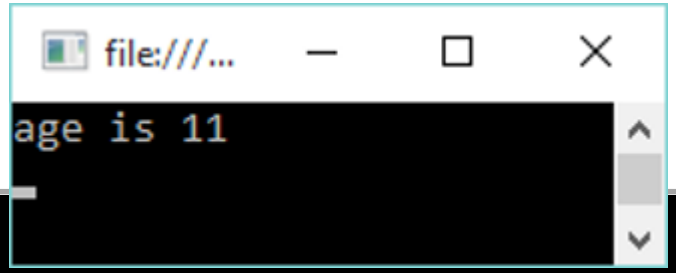



```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(ref age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(ref int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)

0xBE8A0D40 0x0b000000 (age)




```
static void Main(string[] args)
{
    int age = 10;
    CelebrateBirthday(ref age);
    Console.WriteLine("age = {0}", age);
}

static void CelebrateBirthday(ref int a)
{
    a = a + 1;
}
```

Stack
(local variables)

Heap
(objects & arrays)

0xBE8A0D40 0x00000000 (age)

```
static void Main(string[] args)
{
    int age;
    SetAge(out age);
    Console.WriteLine("age = {0}", age);
}

static void SetAge(out int a)
{
    a = 28;
}
```

← BY REFERENCE OUT

Now we'll take a look what happens with a 'by reference out'-parameter of a method.

Stack
(local variables)

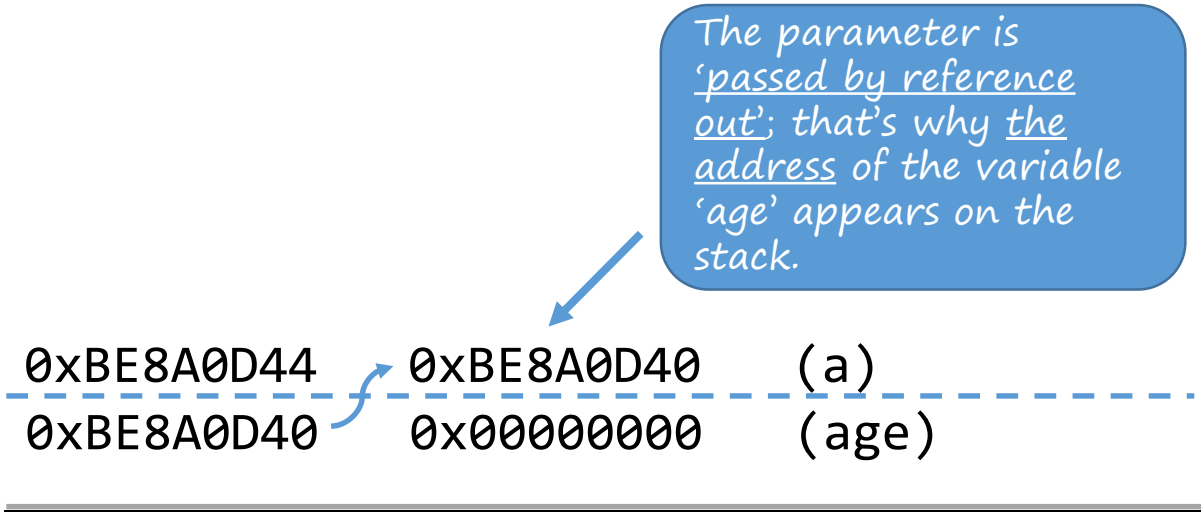
Heap
(objects & arrays)

0xBE8A0D40 0x00000000 (age)

```
static void Main(string[] args)
{
    int age;
    SetAge(out age);
    Console.WriteLine("age = {0}", age);
}

static void SetAge(out int a)
{
    a = 28;
}
```

Stack
(local variables)

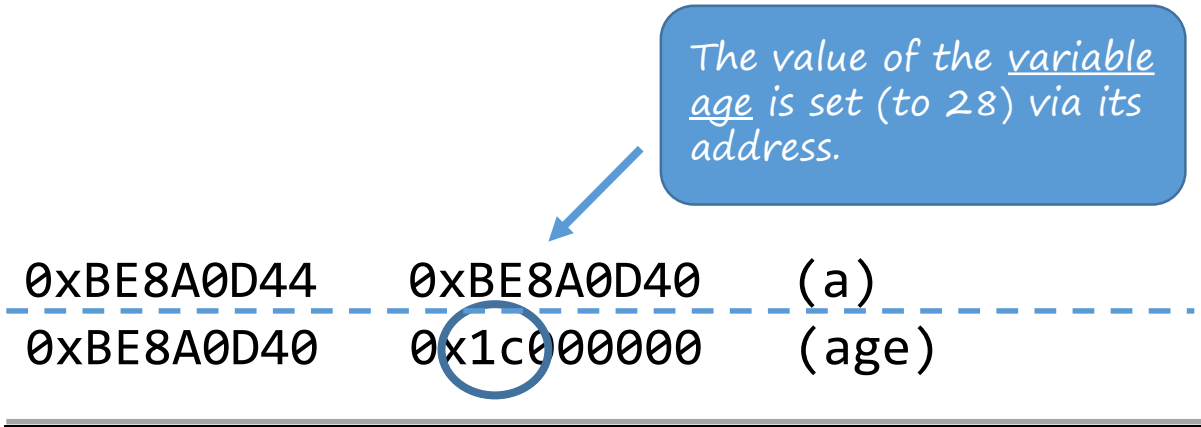
Heap
(objects & arrays)




```
static void Main(string[] args)
{
    int age;
    SetAge(out age);
    Console.WriteLine("age = {0}", age);
}

static void SetAge(out int a)
{
    a = 28;
}
```

Stack
(local variables)

Heap
(objects & arrays)

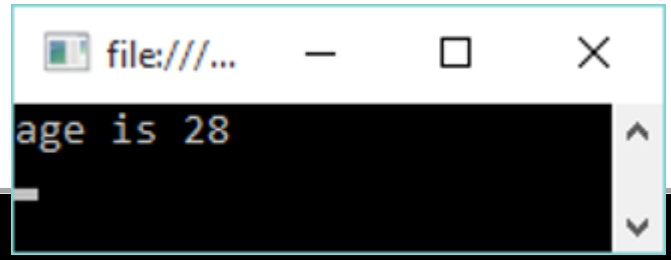



```
static void Main(string[] args)
{
    int age;
    SetAge(out age);
    Console.WriteLine("age = {0}", age);
}

static void SetAge(out int a)
{
    a = 28;
}
```

Stack
(local variables)

Heap
(objects & arrays)

0xBE8A0D40 0x1c000000 (age)




```
static void Main(string[] args)
{
    int age;
    SetAge(out age);
    Console.WriteLine("age = {0}", age);
}

static void SetAge(out int a)
{
    a = 28;
}
```