

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

عنوان: متغیرهای مقدار محور (Value type) و

مرجع محور (Reference type)

استاد: میثاق یاریان

نام دانشجو: سید محمد موسوی مطلق

درس: برنامه نویسی سمت سرور

نیمسال: اول ۱۴۰۲

متغیرهای مقدار محور (Value type) و مرجع محور (Reference type)

متغیرها در #C شامل انواع زیر است:

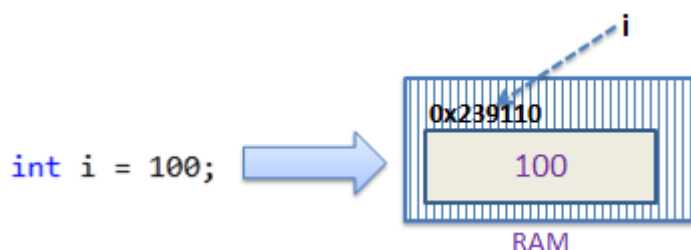
۱. مقدار محور Value type
۲. مرجع محور Reference type

مقدار محور: Value type

یک نوع داده، مقدار محور است اگر مقدار داده در فضای حافظه خود داشته باشد. این به بدان معنا است که متغیرهای این نوع داده‌ها مستقیماً دارای مقدارهایشان هستند.

برای مثال متغیر عدد صحیح `int i = 100` را در نظر بگیرید.

سیستم عدد ۱۰۰ را در فضای حافظه اختصاص داده شده برای متغیر 'i' ذخیره می‌کند. تصویر زیر نشان می‌دهد که چگونه عدد ۱۰۰ در حافظه فرضی (۰x239110) برای 'i' ذخیره می‌شود:



متغیر مقدار محور

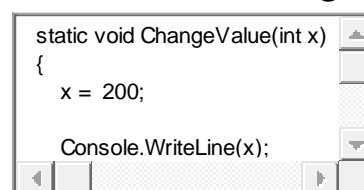
نوع داده‌های زیر مقدار محور هستند:

- bool
- byte
- char
- decimal
- double
- enum

- float .
- int .
- long .
- sbyte .
- short .
- struct .
- uint .
- ulong .
- ushort .

هنگامی که متغیر مقدار محور را از یک متد به متد دیگر منتقل می کنید، سیستم یک نسخه جداگانه از یک متغیر را در متد دیگری ایجاد می کند، به طوری که اگر مقدار در یک متد تغییر کند، این در متغیر متد دیگری تاثیر نمی گذارد.

مثال :



```
1 static void ChangeValue(int x)
2     {
3         x = 200;
4
5         Console.WriteLine(x);
6     }
7
8 static void Main(string[] args)
9     {
10         int i = 100;
11
12         Console.WriteLine(i);
13
14         ChangeValue(i);
15
16         Console.WriteLine(i);
17     }
```

با اجراء كدهای بالا ابتدا عدد ۱۰۰ سپس عدد ۲۰۰ و بعد از آن دوباره عدد ۱۰۰ در خروجی نمایش داده می شوند.

در مثال فوق، متغیر i در Main، حتی پس از انتقال آن به متد ChangeValue و تغییر آن در آنجا نیز تغییری پیدا نمی کند.

مرجع محور: Reference type

بر خلاف انواع مقدار محور، نوع مرجع محور مقدار را به طور مستقیم ذخیره نمی کند. در عوض، آدرس متغیری که دارای مقدار مورد نظر است در آن ذخیره می شود. به عبارت دیگر، نوع مرجع محور شامل یک اشاره گر به مکان حافظه دیگری است که داده ها را نگه می دارد.

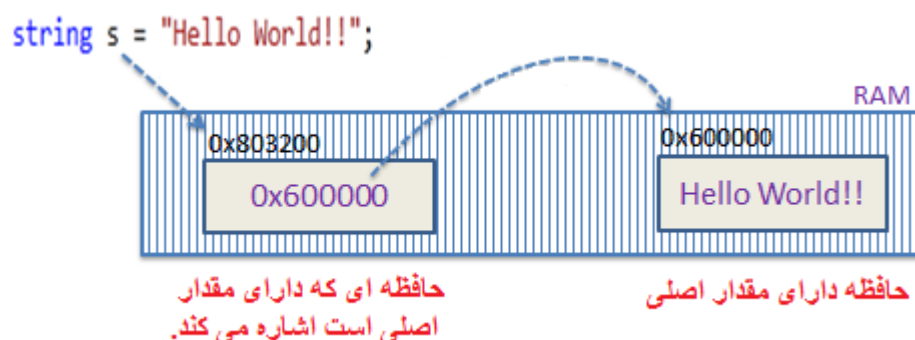
برای مثال، متغیر رشته ای فوق را در نظر بگیرید:

مثال :

```
string s = "Hello World!";
```

1 string s = "Hello World!";

تصویر زیر نشان می دهد که چگونه سیستم حافظه را برای متغیر رشته ای فوق اختصاص می دهد.



متغیر مرجع محور

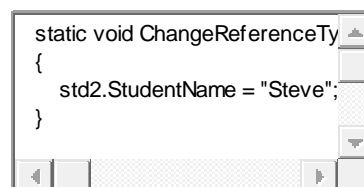
همانطور که می توانید در تصویر بالا مشاهده کنید، سیستم یک مکان تصادفی در حافظه (0x803200) را برای متغیر s انتخاب می کند. مقدار این متغیر 0x600000 است که آدرس حافظه ای که دارای مقدار واقعی داده است را نشان می دهد. بنابراین، نوع مرجع محور آدرس محل نگهداری مقدار را ذخیره می کند

انواع داده های زیر از نوع مرجع محور هستند:

- String
- All arrays
- Class
- Delegates

وقتی یک متغیر نوع مرجع محور را از یک متد به دیگری انتقال می دهید، یک کپی جدید ایجاد نمی کند؛ در عوض، آدرس متغیر را می گذارد. اگر ما اکنون مقدار متغیر را در یک متد تغییر دهیم، در متدی که فراخوانی میشود نیز منعکس خواهد شد.

مثال :



```
1 static void ChangeReferenceType(Student std2)
2     {
3         std2.StudentName = "Steve";
4     }
5
6     static void Main(string[] args)
7     {
8         Student std1 = new Student();
9         std1.StudentName = "Bill";
10
11         ChangeReferenceType(std1);
12
```

```

13 Console.WriteLine(std1.StudentName);
14 }

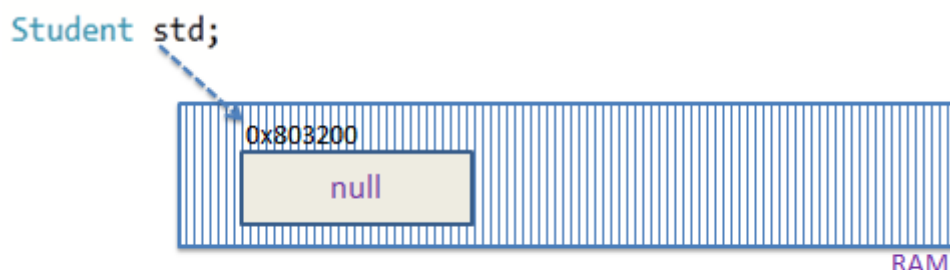
```

خروجی کدهای زیر “Steve” خواهد بود.

در مثال بالا، از آنجا که Student یک شی است، وقتی شی ساخته شد std1 را به متد ChangeReferenceType ارسال میکنیم، در واقع آدرس واقعی حافظه را ارسال می کنیم . بنابراین، هنگامی که متد ChangeReferenceType ستون نام شی std2 را تغییر می دهد، در واقع ستون نام شی std1 را تغییر داده است ، زیرا std1 و std2 هر دو به یک آدرس در حافظه اشاره می کنند.

مقدار: NULL

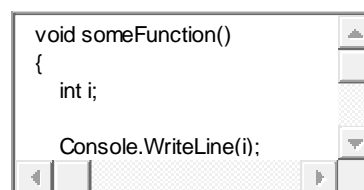
به طور پیش فرض، زمانی که متغیرهای مرجع محور دارای مقدار اولیه نیستند مقدار NULL (خالی) را در خود نگهداری می کنند، این بدان معنا است که آنها به هیچ مکان حافظه دیگری اشاره نمی کنند، زیرا هنوز مقداری ندارند.



متغیر دارای مقدار NULL

یک متغیر مقدار محور نمی تواند null باشد، زیرا دارای یک مقدار است نه یک آدرس حافظه. در هر حال، متغیرهای نوع مقدار محور باید قبل از استفاده، مقدار خاصی را دریافت کنند . و در غیر اینصورت کامپایلر خطا برمی گرداند.

مثال :

A screenshot of a code editor window. The code inside is:

```
void someFunction()
{
    int i;

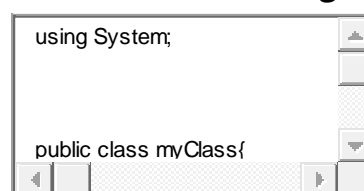
    Console.WriteLine(i);
}
```

 The editor has a standard Windows-style interface with a title bar, menu bar, and toolbar.

```
1 void someFunction()
2     {
3         int i;
4
5     Console.WriteLine(i);
6     }
```

با این حال، اگر به فیلد مقدار محور در یک کلاس مقدار پیشفرض داده نشود می تواند بدون مقدارهی اولیه استفاده شود.، به عنوان مثال، `int` دارای مقدار `۰` می شود، `Bool` دارای مقدار `False` و ... خواهد شد.

مثال :

A screenshot of a code editor window. The code inside is:

```
using System;

public class myClass{
}
```

 The editor has a standard Windows-style interface with a title bar, menu bar, and toolbar.

```
1         using System;
2
3
4
5     public class myClass{
6
7         public int i;
8     }
9
10    public class Program
11    {
12        public static void Main()
13        {
14            myClass mcls = new myClass();
15            Console.WriteLine(mcls.i);
16        }
17    }
```

16	}
17	}
18	
19	
20	

خروجی کدهای بالا خط نبوده و مقدار ۰ است.

نکات مهم:

۱. متغیرهای نوع مقدار محور در حافظه مقدار خود را ذخیره می کنند، در حالی که متغیرهای نوع مرجع محور آدرس محل ذخیره شدن مقدار را نگهداری می کنند.
۲. انواع متغیرهای داده اولیه و ساختاری از نوع مقدار محور هستند و انواع اشیاء در کلاس ها، رشته ها، آرایه ها و ... نیز از نوع مرجع محور هستند.
۳. نوع مقدار محور به طور پیش فرض به صورت `byval` و نوع مرجع محور به طور پیش فرض بصورت `byref` استفاده می شود.
۴. انواع متغیرهای مقدار محور و مرجع محور در پشته ذخیره می شوند و پشته در حافظه بستگی به دامنه متغیر دارد