

# برنامه نویسی با زبان C آدرس و اشاره گر

مجتبی اجمی

استادیار دانشگاه آزاد اسلامی واحد زنجان

# مقدمه: مبنای ۱۶ (Hexadecimal)

- یکی از مبناهای متداول برای نمایش اطلاعات در کامپیوتر مبنای ۱۶ (Hexadecimal) می باشد.
- می دانیم که اگر عددی در مبنای  $m$  نمایش داده شود، هر کدام از رقم های آن می تواند یکی از اعداد ۰ تا  $m-1$  باشد.
- رقم های یک عدد Hexadecimal می تواند یکی از ارقام زیر باشد:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

## مقدمه: مبنای ۱۶ (Hexadecimal)

• برای تبدیل یک عدد از مبنای ۱۰ به مبنای ۱۶ از روش تقسیمات متوالی استفاده می شود.

$$\begin{array}{r} 450 \div 16 = 28 \text{ remainder } 2 \\ 28 \div 16 = 1 \text{ remainder } 12 \\ 1 \div 16 = 0 \text{ remainder } 1 \end{array}$$

Division	Result	Remainder
450/16	28	2
28/16	1	12(C)
1/16	0	1

$$(450)_{10} = (1C2)_{16}$$

# مقدمه: مبنای ۱۶ (Hexadecimal)

- برای تبدیل یک عدد از مبنای ۱۶ به مبنای ۱۰ از روش ارزش رقم ها استفاده می شود.

Hexadecimal Value = 2A5

$$\begin{array}{ccc} 2 & A & 5 \\ 16^2 & 16^1 & 16^0 \end{array}$$


$$256 \times 2 = 512 \quad 16 \times 10 = 160 \quad 1 \times 5 = 5$$

$$512 + 160 + 5$$

677

$$(2A5)_{16} = (677)_{10}$$

# چرا مبنای ۱۶؟

• می دانیم که اطلاعات در کامپیوتر به صورت 0,1 هستند.  ولی استفاده از مبنای ۱۶، مزایایی دارد:

• خوانایی: هگزادسیمال از اعدادی استفاده می کند که شباهت بیشتری به سیستم شمارش پایه ۱۰ معمول ما دارند و بنابراین تصمیم گیری در یک نگاه ساده تر است. برای مثال عددی مانند E7 راحت تر از 1110011 خوانده می شود.

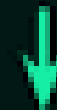
• تراکم اطلاعات بالاتر: با دو رقم هگزادسیمال، می توانیم هر عددی را از ۰ تا ۲۵۵ بیان کنیم. برای انجام همین کار در باینری، به ۸ رقم نیاز داریم. هرچه اعداد بزرگتر و بزرگتر می شوند، نیاز به ارقام بیشتر و بیشتر می شود و استفاده از باینری سخت تر می شود.

# تبدیل از باینری به هگزادسیمال

HEXADECIMAL NUMBER SYSTEM TABLE

Decimal Numbers	4-bit Binary Number	Hexadecimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

1100 1110 1001 1010



C



E



9



A

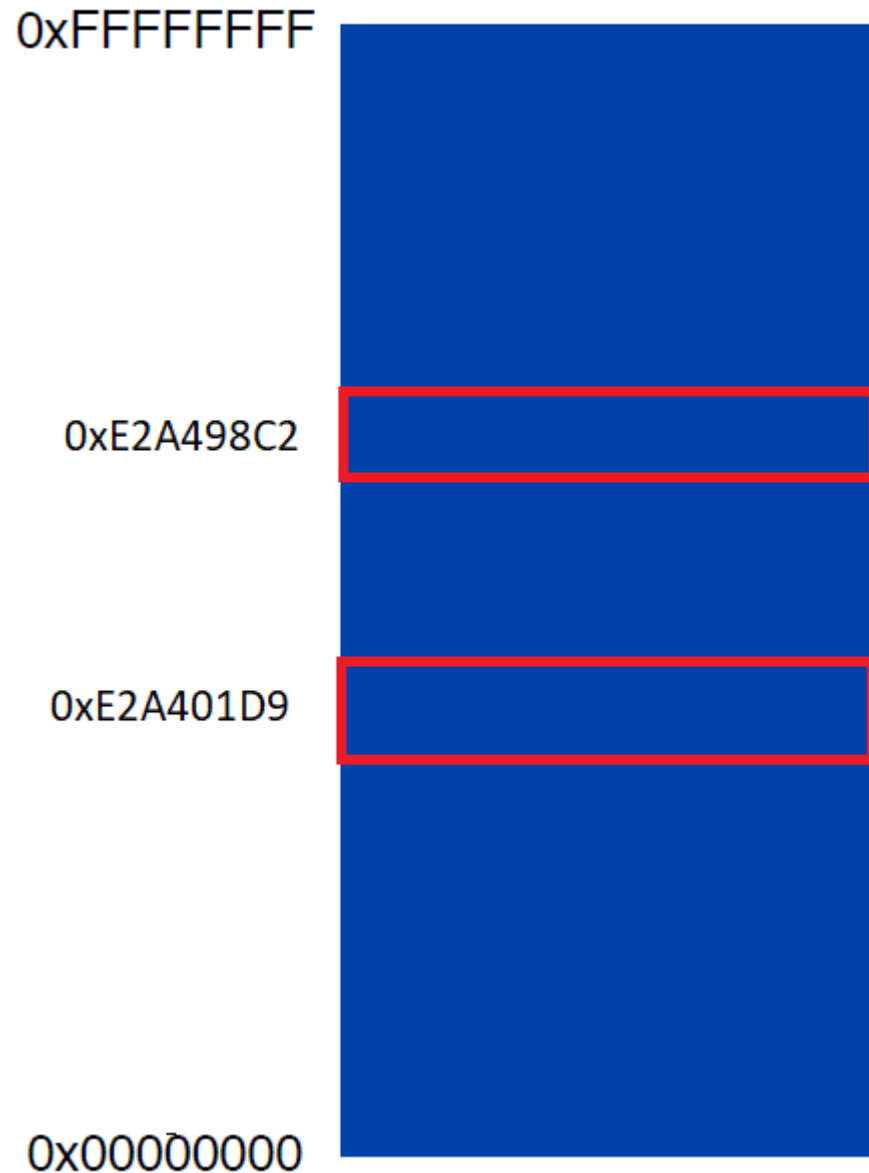
(CE9A)<sub>16</sub>

# فضای آدرس (Address Space)



- سیستم عامل برای هر برنامه ای که اجرا می شود در حافظه فضایی اختصاص می دهد که به آن، **فضای آدرس** برنامه گوییم.

# آدرس حافظه



- حافظه به بخش هایی تقسیم می شود.

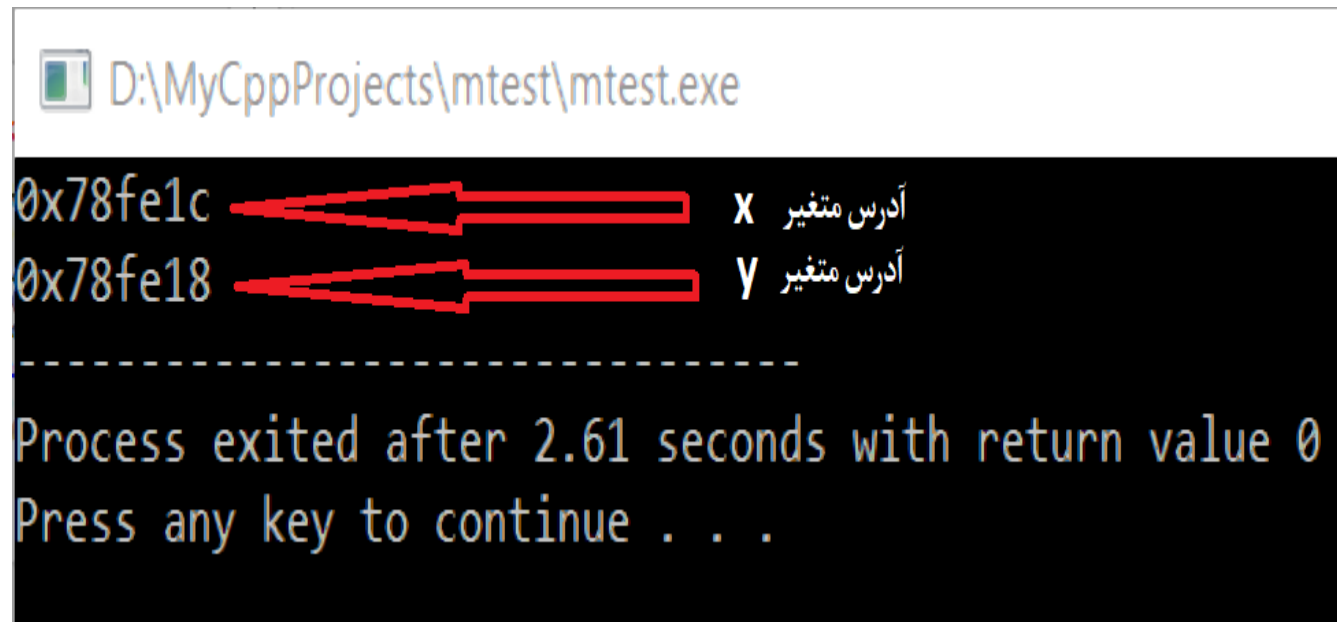
- هر بخش با شماره ای مشخص می شود که به آن، **آدرس** آن بخش گوییم.



# آدرس در زبان C

• در زبان C، عملگر **&** آدرس یک متغیر را به ما می دهد.

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int x = 10;
6     float y = 3.14;
7     cout<<&x<<"\n"<<&y;
8 }
```



The screenshot shows a Windows command prompt window titled "D:\MyCppProjects\mtest\mtest.exe". The output of the program is as follows:

```
0x78fe1c  آدرس متغیر x
0x78fe18  آدرس متغیر y
-----
Process exited after 2.61 seconds with return value 0
Press any key to continue . . .
```

Red arrows point from the memory addresses to the variable names in the output.

# آدرس در زبان C

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int A[3] = {15, 20, 130};
6     cout << &A[0] << "\n" << &A[1] << "\n" << &A[2];
7 }
```

0x78fe14        A[0]    آدرس  
0x78fe18        A[1]    آدرس  
0x78fe1c        A[2]    آدرس

-----  
Process exited after 0.3817 seconds with return value 0  
Press any key to continue . . .

# اشاره گر (Pointer)

• اشاره گر متغیری است که می توان آدرس محلی از حافظه را در آن ذخیره کرد.

• نحوه تعریف اشاره گر:

- `type *name;`
- `type *name = address;`

# اشاره گر: مثال ۱

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int x = 120;
6     int *p;
7     p = &x;
8     cout<<"x is "<< x <<"\n";
9     cout<<"&x(address of x) is "<< &x <<"\n";
10    cout<<"p is "<< p <<"\n";
11 }
```

```
x is 120
&x(address of x) is 0x78fe14
p is 0x78fe14
```

## اشاره گر: مثال ۲

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int x,y;
6     int *p,*q;
7     p = &x;
8     q = &y;
9     if (p==q){
10         cout<<"p and q are pointing to the same location";
11     }
12     else{
13         cout<<"p and q are pointing to different locations";
14     }
15 }
```

p and q are pointing to different locations

-----

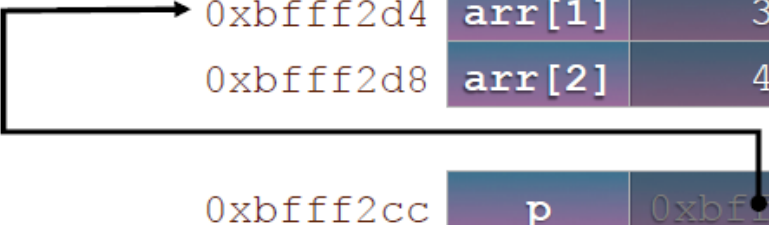
Process exited after 0.1901 seconds with return value 0

Press any key to continue . . .

## اشاره گر: مثال ۳

```
int main() {  
    int x = 1;  
    int arr[3] = {2, 3, 4};  
    int *p = &arr[1];  
}
```

address	name	value
0xbfff2dc	x	1
0xbfff2d0	arr[0]	2
0xbfff2d4	arr[1]	3
0xbfff2d8	arr[2]	4
0xbfff2cc	p	0xbfff2d4



# ارجاع از طریق اشاره گر

• ارجاع: دسترسی به محلی از حافظه که اشاره گر به آن محل اشاره می کند.

• نحوه ارجاع:

- `*pointer`
- `*pointer = value;`

# ارجاع از طریق اشاره گر: مثال

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int x = 42;
6     int *p;
7     p = &x;
8     cout<<"x is "<< x <<"\n";
9     *p = 99;
10    cout<<"x is "<< x <<"\n";
11 }
```

 D:\MyCppProjects\mtest\mtest.exe

x is 42

x is 99

-----  
Process exited after 0.2882 seconds with return value 0  
Press any key to continue . . .



## & and \*

- `&foo` آدرس `foo`
- `*pointer` ارجاع به یک اشاره گر
- `*pointer = value;` قرار دادن یک مقدار در محل ارجاع شده توسط یک اشاره گر