اشاره‌گرها ۱

مبانی برنامه‌نویسی

(۱۳۹۹-۱۳-۱۱)

جلسه‌ی بیست‌وهفتم

دانشگاه شهید بهشتی

پاییز ۱۳۹۹

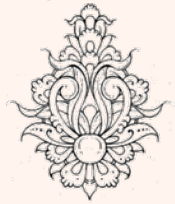
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

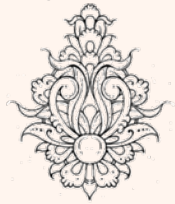
فهرست مطالب

- اشاره‌گر
  - عملگرهای حسابی
- اشاره‌گر و آرایه
  - اشاره‌گر و رشته



اشاره‌گر–مفاهیم پایه

- هر بایت از حافظه دارای آدرسی منحصر به فرد است.
- آدرس هر متغیر آدرس اولین بایتی از حافظه است که به آن متغیر اختصاص داده شده است.
- به وسیله‌ی اپراتور «&» می‌توان به آدرس یک متغیر دست یافت.
- برای نگهداری یک آدرس، متغیر از جنس اشاره‌گر تعریف می‌شود.

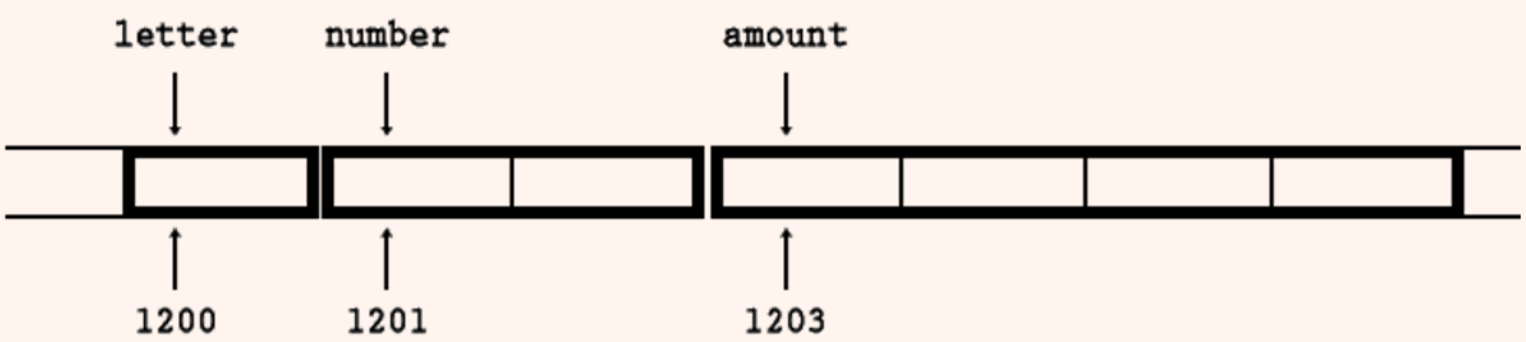


مفاهیم (ادامه...)

- اگر در نظر داشته باشید:

```
char letter;
short number;
float amount;
```

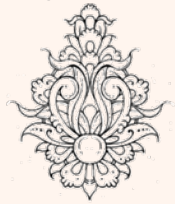
- خواهیم داشت:



مثال

```
int main()
{
short x = 25;
cout << "The address of x is " <<&x<< endl;
cout << "The size of x is " << sizeof(x) << " bytes\n";
cout << "The value in x is " << x << endl;
return 0;
}
```

```
The address of x is 0012FF28
The size of x is 2 bytes
The value in x is 25
```



مفاهیم پایه (ادامه...)

- اشاره‌گرها سرعت پردازش را زیاد می‌کنند و کدنویسی را کم.
- با استفاده از اشاره‌گرها می‌توان به بهترین شکل از حافظه استفاده کرد.
- با به کارگیری اشاره‌گرها می‌توان اشیایی پیچیده‌تر و کارآمدتر ساخت.
- اشاره‌گرها به متغیرهایی گفته می‌شود که آدرس یک متغیر یا یک تابع و یا مکانی از حافظه را در خود نگه می‌دارد.
- اشاره‌گرها انواع متفاوتی دارند.
  - اشاره‌گر از نوع int (آدرس یک int را در خود نگاه می‌دارد)
  - اشاره‌گر از نوع char
  - اشاره‌گر از نوع double
  - .....

نکته: اشاره‌گر حاوی آدرس اولین بایت یک نوع داده و یا بخشی از برنامه است. از این جهت اشاره‌گر از هر نوعی که باشد، طول یکسانی خواهد داشت.

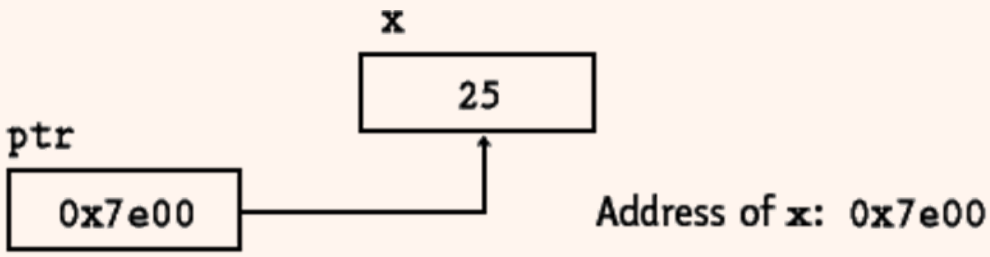
int *ptr;

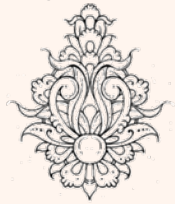
- ptr متغیری از جنس اشاره‌گر است که می‌تواند آدرس یک int را نگاه‌داری کند.

```
int main()
 {
 int x = 25;
 int *ptr;

 ptr = &x; // Store the address of x in ptr
 cout << "The value in x is " << x << endl;
 cout << "The address of x is " << ptr << endl;
 return 0;
 }
```

```
The value in x is 25
The address of x is 0012FF28
```

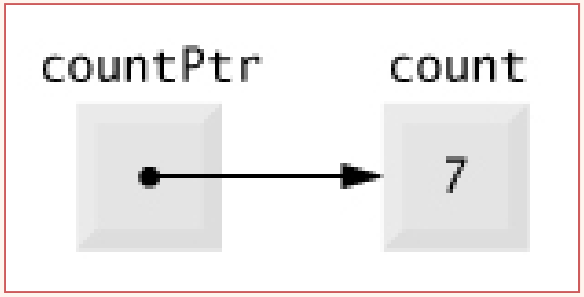
```
ptr = &x;
```





نحوه‌ی تعریف اشاره‌گرها

- با در نظر گرفتن شکل روبرو می‌توان در نظر داشت متغیر countPtr به صورت غیرمستقیم به متغیر count که مقدار ۷ را در خود جای داده اشاره می‌نماید.




```
int *countPtr, count;
```

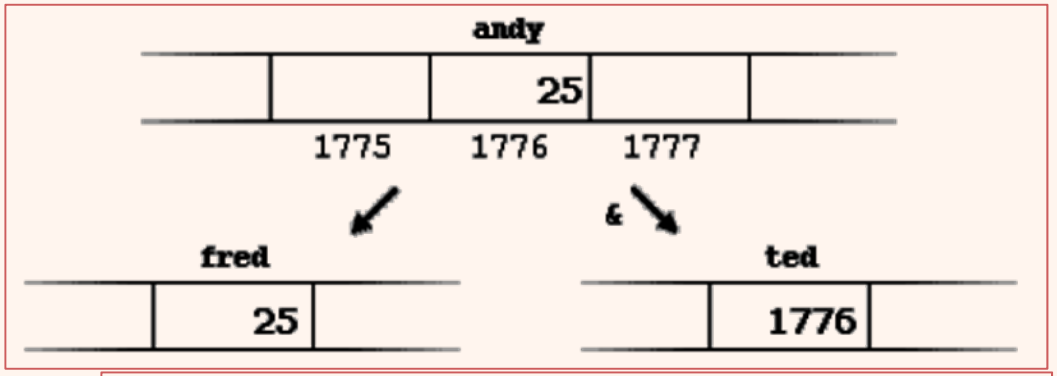
```
int y = 5; // declare variable y
int *yPtr; // declare pointer variable yPtr
```

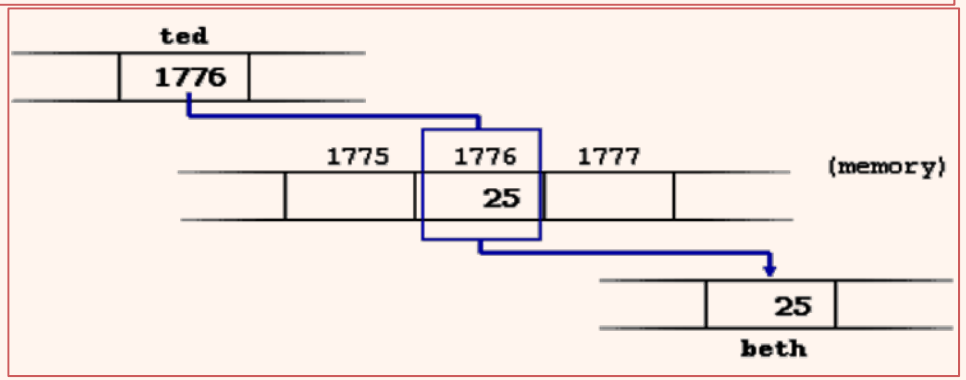
```
yPtr = &y; // assign address of y to yPtr
```

عملگرها

- دو عملگر مورد استفاده اشاره‌گرها می‌باشد.
  - عملگر آدرس **&** : یک عملگر یکانی است که آدرس عملوند خود را تعیین می‌نماید. **reference operator**
  - عملگر محتوا *: یک عملگر یکانی است که محتویات عملوند خود را تعیین می‌نماید. **dereference operator**

```
andy = 25;
fred = andy;
ted = &andy;
```



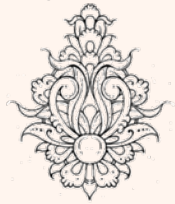



```
beth = *ted;
```

مثال

```
int main()
 {
 int x = 25;
 int *ptr;

 ptr = &x; // Store the address of x in ptr
 cout << "Here is the value in x, printed twice:\n";
 cout << x << " " << *ptr << endl;
 *ptr = 100;
 cout << "Once again, here is the value in x:\n";
 cout << x << " " << *ptr << endl;
 return 0;
 }
```

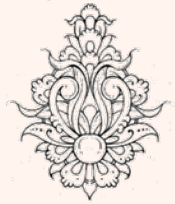
```
Here is the value in x, printed twice:
25 25
Once again, here is the value in x:
100 100
```



مثال

```
// my first pointer
#include <iostream>
using namespace std;

int main ()
{
  int firstvalue, secondvalue;
  int * mypointer;

  mypointer = &firstvalue;
  *mypointer = 10;
  mypointer = &secondvalue;
  *mypointer = 20;
  cout << "firstvalue is " << firstvalue << endl;
  cout << "secondvalue is " << secondvalue << endl;
  return 0;
}
```

```
firstvalue is 10
secondvalue is 20
```



مثال

```
int main()
 {
 int x = 25, y = 50, z = 75;
 int *ptr;

 cout << "Here are the values of x, y, and z:\n";
 cout << x << " " << y << " " << z << endl;
 ptr = &x;
 *ptr *= 2;
 ptr = &y;
 *ptr *= 2;
 ptr = &z;
 *ptr *= 2;
 cout << "Once again, here are the values of x, y, and z:\n";
 cout << x << " " << y << " " << z << endl;
 return 0;
 }
```

```
Here are the values of x, y, and z:
25 50 75
Once again, here are the values of x, y, and z:
50 100 150
```

مثال

```
// more pointers
#include <iostream>
using namespace std;

int main ()
{
  int firstvalue = 5, secondvalue = 15;
  int * p1, * p2;

  p1 = &firstvalue;  // p1 = address of firstvalue
  p2 = &secondvalue; // p2 = address of secondvalue
  *p1 = 10;          // value pointed by p1 = 10
  *p2 = *p1;         // value pointed by p2 = value pointed by p1
  p1 = p2;           // p1 = p2 (value of pointer is copied)
  *p1 = 20;          // value pointed by p1 = 20

  cout << "firstvalue is " << firstvalue << endl;
  cout << "secondvalue is " << secondvalue << endl;
  return 0;
}
```

```
firstvalue is 10
secondvalue is 20
```

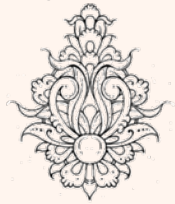
عملگرهای حسابی

- تنها عملگرهای جمع و تفریق به روی اشاره‌گرها قابل تعریف می‌باشند.
- نحوه‌ی عملکرد رابطه تنگاتنگی با نوع داده تعریف شده دارد.

```
char *mychar;
short *myshort;
long *mylong;
```

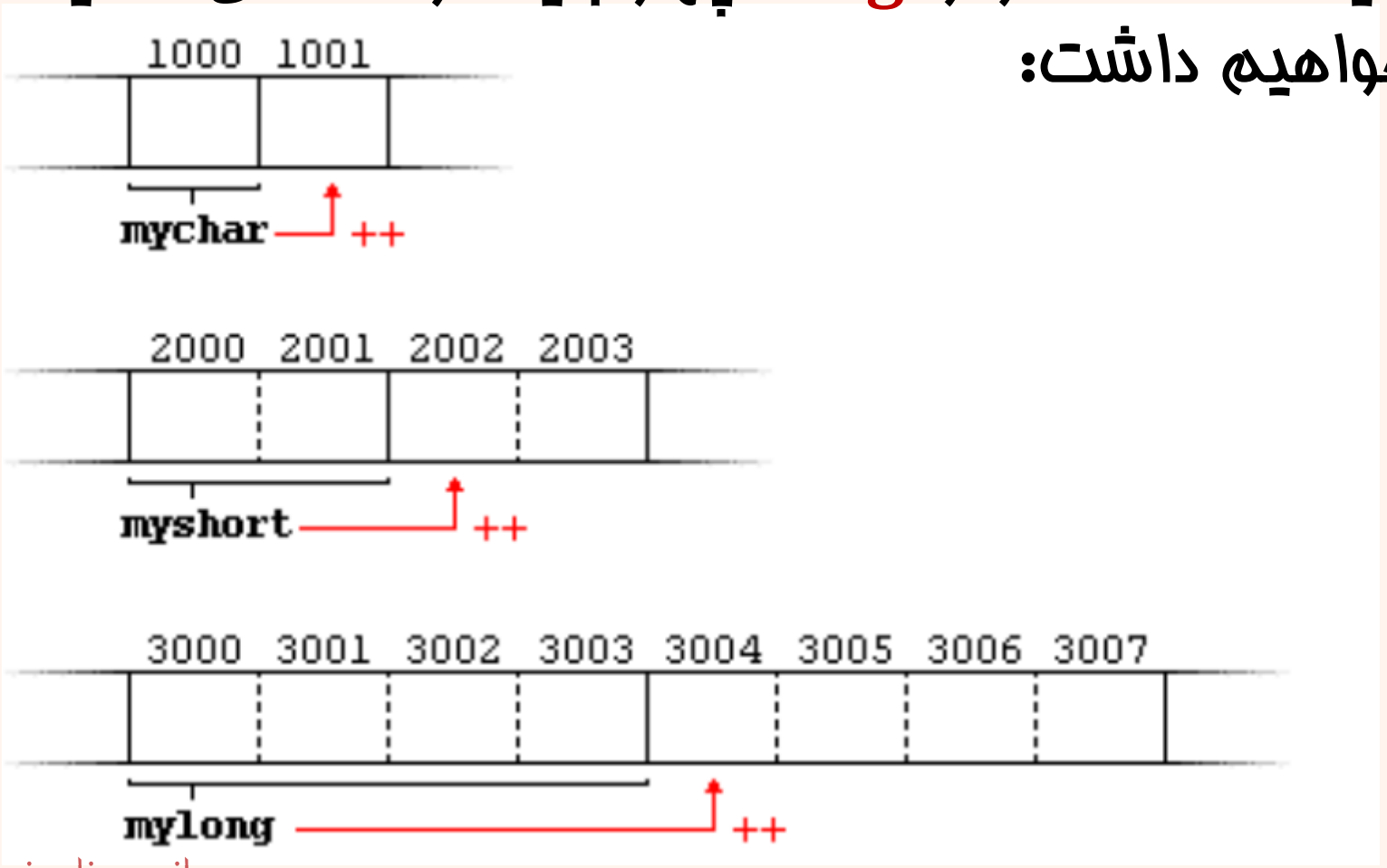
- اگر به ترتیب به مکان‌های حافظه با آدرس‌های ۱۰۰۰ و ۲۰۰۰ و همچنین ۳۰۰۰ اشاره کنند و داشته باشیم:

```
mychar++;
myshort++;
mylong++;
```



عملگرهای حسابی (ادامه...)

- اگر فرض کنیم برای سیستم مشخصی **char** یک بایت، **short** دو و **long** چهار بایت را اشغال نماید خواهیم داشت:



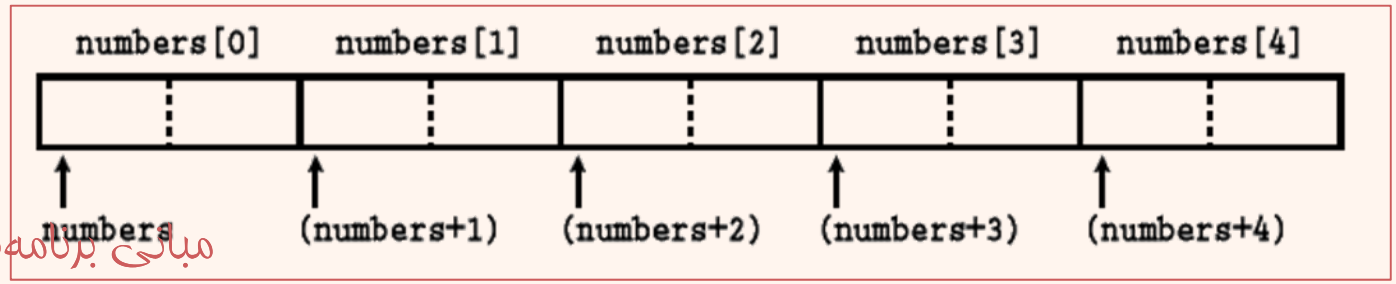
عملگرهای حسابی (ادامه...)

- اشاره‌گرها با متغیرهای معمولی تفاوت‌هایی دارند.
- هنگامی که یک عدد با اشاره‌گر جمع می‌شود در حقیقت sizeof نوعی که اشاره‌گر به آن اشاره می‌کند اهمیت دارد.
- خواهیم داشت:

short * numbers

```
*(numbers + 1)  is actually  *(numbers + 1 * 2)
*(numbers + 2)  is actually  *(numbers + 2 * 2)
*(numbers + 3)  is actually  *(numbers + 3 * 2)
```

Sizeof(short)



اشاره‌گرها و آرایه‌ها

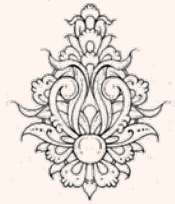
- بین اشاره‌گرها و آرایه‌ها ارتباط بسیار نزدیکی وجود دارد.
- اشاره‌گرها حاوی یک آدرس و نام آرایه نیز نشان‌گر یک آدرس است.
- نام آرایه، آدرس اولین عنصر آرایه را مشخص می‌سازد.
- فرآیندی که به وسیله‌ی اشاره‌گرها صورت می‌گیرد سریع‌تر ولی در بسیاری موارد درک آن دشوارتر است.
- اگر داشته باشیم:

```
int numbers [20];
int * p;
```

- از آنجا که نام آرایه آدرس اولین عنصر آرایه نیز هست می توانیم داشته باشیم:

```
p = numbers;
```

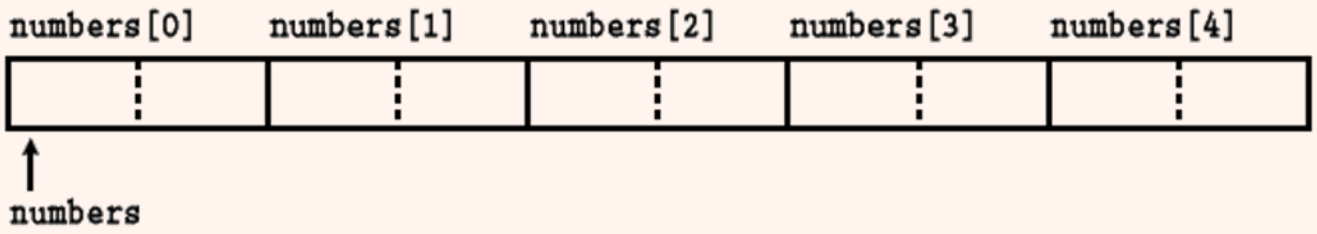
- در این حالت هر دو به یک مکان اشاره می‌کنند.



- نام یک آرایه می‌تواند به عنوان اشاره‌گر ثابت در نظر گرفته شود.

```
int main()
  {
  short numbers[] = {10, 20, 30, 40, 50};

  cout << "The first element of the array is ";
  cout << *numbers << endl;
  return 0;
  }
```

```
The first element of the array is 10
```

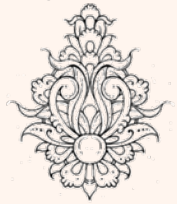


اشاره‌گرها و آرایه‌ها

- هر دو هم p و هم number دارای خصوصیاتی یکسان هستند.
- تفاوت
  - تفاوت در این است که مقدار p می‌تواند تغییر یابد ولی number همواره به اولین عنصر اشاره می‌کند.
  - می‌توان در نظر گرفت p یک اشاره‌گر معمولی است.
  - اما number یک اشاره‌گر ثابت در نظر گرفته می‌شود.

```
numbers = p;
```

- عبارت بالا صحیح نمی‌باشد.



اشاره‌گرها و آرایه‌ها

```
Int *p;

Int a [5] = {10, 20, 30, 40, 50};

p=a;
```

a[0] ⟶ *(a+0) ⟶ *p ⟶ 10

a[1] ⟶ *(a+1) ⟶ *(p+1) ⟶ 20

a[2] ⟶ *(a+2) ⟶ *(p+2) ⟶ 30

a[3] ⟶ *(a+3) ⟶ *(p+3) ⟶ 40

a[4] ⟶ *(a+4) ⟶ *(p+4) ⟶ 50

مثال

```
int main()
 {
 const int SIZE = 5;
 int numbers[SIZE];
 int count;
 cout << "Enter " << SIZE << " numbers: ";
 for (count = 0; count < SIZE; count++)
 cin >> *(numbers + count);
 cout << "Here are the numbers you entered:\n";
 for (count = 0; count < SIZE; count++)
 cout << *(numbers + count)<< " ";
 cout << endl;
 return 0;
 }
```

```
Enter 5 numbers: 10
20
30
40
50
Here are the numbers you entered:
10 20 30 40 50
```

array[index] is equivalent to *(array + index)

کنترلی به روی مرزهای آرایه ندارد، هنگامی که از اشاره‌گر استفاده C++ می‌شود ممکن است به آدرسی خارج از محدوده‌ی آرایه اشاره شود.

مثال

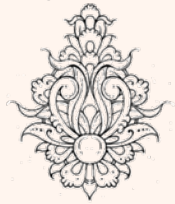
```
int main()
 {
 const int NUM_COINS = 5;
 double coins[NUM_COINS] = {0.05, 0.1, 0.25, 0.5, 1.0};
 double *doublePtr; // Pointer to a double
 int count; // Array index

 doublePtr = coins; // doublePtr now points to coins array
 cout << "Here are the values in the coins array:\n";
 for (count = 0; count < NUM_COINS; count++)
    cout << doublePtr[count] << " ";
 cout << "\nAnd here they are again:\n";
 for (count = 0; count < NUM_COINS; count++)
    cout << *(coins + count) << " ";
 cout << endl;
 return 0;
 }
```

```
Here are the values in the coins array:
0.05 0.1 0.25 0.5 1
And here they are again:
0.05 0.1 0.25 0.5 1
```

مثال

```
int main()
{
 const int NUM_COINS = 5;
 double coins[NUM_COINS] = {0.05, 0.1, 0.25, 0.5, 1.0};
 double *doublePtr; // Pointer to a double
 int count; // Array index

 cout << "Here are the values in the coins array:\n";
 for (count = 0; count < NUM_COINS; count++)
     {
         doublePtr = &coins[count];
         cout << *doublePtr << " ";
     }
 cout << endl;
 return 0;
 }
```

```
Here are the values in the coins array:
0.05 0.1 0.25 0.5 1
```



آرایه‌ها و اشاره‌گرها (ادامه...)

- تفاوت میان نام آرایه و اشاره‌گر در این است که نام آرایه اشاره‌گری ثابت است و تغییر نمی‌کند ولی اشاره‌گر معمولی قابلیت تغییر خواهد داشت.

```
double readings[20], totals[20];
double *dptr;

dptr = readings; // Make dptr point to readings.
dptr = totals;   // Make dptr point to totals.

readings = totals;      // ILLEGAL! Cannot change readings
totals = dptr;          // ILLEGAL! Cannot change totals
```

آدرس

- اگر n یک متغیر باشد، «&n» آدرس آن متغیر است. از طرفی با استفاده از عملگر «*» می‌توان مقداری که در آدرس n قرار گرفته را به دست آورد.

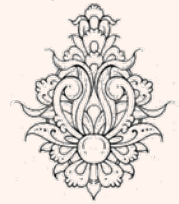
آدرس

- M=&n (M receives the address of n)

- «*» مکمل«&» می‌باشد.

- **Q=*M (Q receive the value at address M)**

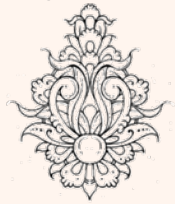
به آن اشاره می‌کند M جایی که



مقداریابی

- اگر M یک اشاره‌گر باشد، *M نشان‌گر مقداری است که M به آن اشاره دارد. از طرفی با استفاده از عملگر & می‌توانیم آدرس آن‌چیز که در *M قرار گرفته را به دست آوریم. پس &*M برابر با M خواهد بود.

می‌توان گفت &n* برابر با خود n خواهد بود.



```cpp
#include <iostream>
 using namespace std;
int main()
   {
     int a; // a is an integer
     int *aPtr; // aPtr is an int * -- pointer to an integer

     a = 7; // assigned 7 to a
     aPtr = &a; // assign the address of a to aPtr

    cout << "The address of a is " << &a
       << "\nThe value of aPtr is " << aPtr;
     cout << "\n\nThe value of a is " << a
       << "\nThe value of *aPtr is " << *aPtr;
     cout << "\n\nShowing that * and & are inverses of "
       << "each other.\n&*aPtr = " << &*aPtr
       << "\n*&aPtr = " << *&aPtr << endl;
     return 0; // indicates successful termination
  } // end main
```

```
The address of a is 0012FF28
The value of aPtr is 0012FF28

The value of a is 7
The value of *aPtr is 7

Showing that * and & are inverses of each other.
&*aPtr = 0012FF28
*&aPtr = 0012FF28
```

مثال

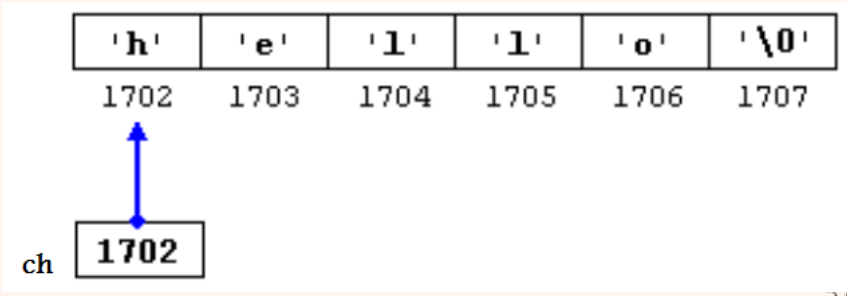
```
// more pointers
#include <iostream>
using namespace std;

int main ()
{
  int numbers[5];
  int * p;
  p = numbers;
  *p = 10;
  p++;
  *p = 20;
  p = &numbers[2];
  *p = 30;
  p = numbers + 3;
  *p = 40;
  p = numbers;
  *(p+4) = 50;
  for (int n=0; n<5; n++)
    cout << numbers[n] << ", ";
  cout<< "\n";
  for (int n=0; n<5; n++)
    cout << *p++ << ", ";
  return 0;
}
```

```
10, 20, 30, 40, 50,
10, 20, 30, 40, 50,
```

اشاره‌گرها و رشته‌ها

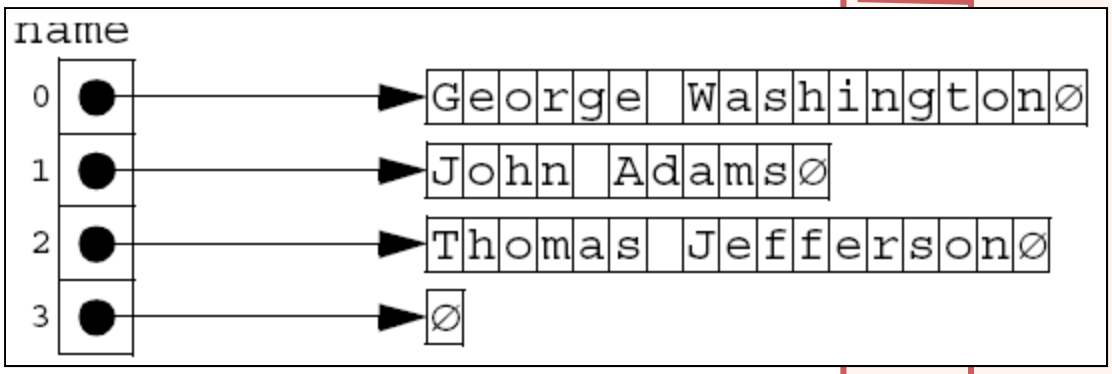
- رشته‌ها آرایه‌ای از جنس کاراکتر هستند.
- ارتباطی که میان آرایه و اشاره‌گرها وجود دارد برای رشته‌ها نیز صادق است.

char * ch=“hello”;



char * str[5]={“amir”, “maryam”, “ziba”, “ali”, “reza”};

```
#include <iostream>

using namespace std;
int main()
{
char buffer[80];
cin.getline(buffer,80,'$');
char* name[4];
name[0] = buffer;
int count = 0;
for (char* p=buffer; *p != '\0'; p++)
     if (*p == '\n')
        { *p = '\0'; // end name[count]
            name[++count] = p+1; // begin next name
         }
cout << "The names are:\n";
   for (int i=0; i<count; i++)
       cout << "\t" << i << ". [" << name[i] << "]" << endl;
}
```




```
George Washington
John Adams
Thomas Jefferson
$
The names are:
        0. [George Washington]
        1. [John Adams]
        2. [Thomas Jefferson]
```