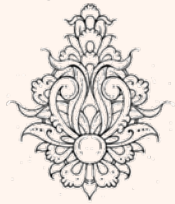
مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی بیست‌وهشتم

راشاره‌گرها ۲

دانشگاه شهید بهشتی
پاییز ۱۳۹۹
دانشکده‌ی مهندسی برق و کامپیوتر
احمد محمودی ازناوه

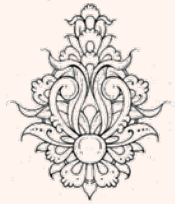
فهرست مطالب

- ارجاع و اشاره‌گر
  - ارسال پارامتر از طریق اشاره‌گر
- نوع بازگشتی از جنس اشاره‌گر و ارجاع
- اشاره‌گر به اشاره‌گر
- اشاره‌گر void



انواع فراخوانی تابع

- در C++ به سه روش می‌توان آرگومان‌ها را به توابع ارسال نمود:
  - فراخوانی با **«مقدار»**
  - فراخوانی با ارجاع به وسیله‌ی **«آرگومان‌های ارجاعی»**
  - فراخوانی با ارجاع به وسیله‌ی **«اشاره‌گر»**



اشاره‌گر و ارجاع

Pointers

```
int i;
int *pi = &i;
```

References

```
int i;
int &ri = i;
```

C++

```
*pi = 4;
```

باید dereference شود

```
ri = 4;
```

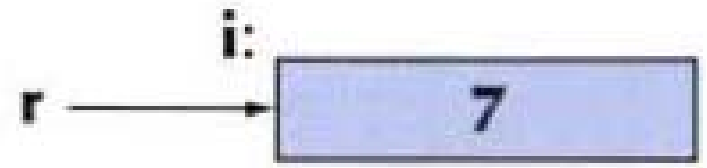
به‌وسیله‌ی کامپایلر اتوماتیک dereference می‌گردد

مرجع هم مانند اشاره‌گر می‌تواند حاوی آدرس یک متغیر باشد،

اشاره‌گر یک متغیر است و می‌تواند به متغیرهای متفاوتی اشاره کند، اما مرجع چنین نیست

```
int main()
    {
int i = 7;
int&r = i;
r = 9;
i = 10;
cout << r <<' '<< i <<endl;
    }
```

```
10 10
```



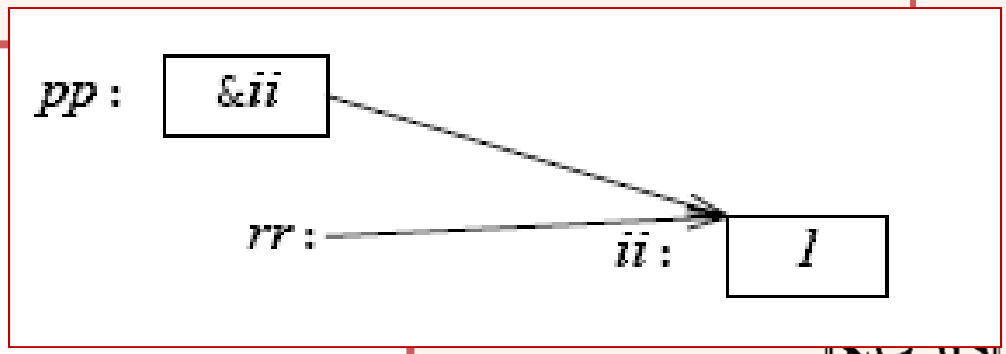
- نام دیگری برای یک Object است.
  - زمان به وجود آمدن باید مقداردهی گردد.
  - هنگامی که initialized گردد، نمی‌تواند مجدداً به Object دیگری اشاره کند.

ارجاع (ادامه...)

```
int main()
{
     int i = 1 ;
     int &r = i ;
     int &r2 ;
     cout << r <<' '<< i <<endl;
}
```

```
: error C2530: 'r2' : references must be initialized
```

```
int main()
{
     int ii = 0 ;
     int & rr = ii ;
     rr ++;
     int * pp = &rr ;
     *pp = 12;
     cout << rr <<' '<< ii << ' ' << *pp << endl;
}
```




```
12 12 12
```

ارجاع ثابت

یک Object را از طریق Const Reference نمی‌توان تغییر داد.

```
int main()
    {
int i = 7;
int& r = i;
r = 9;
const int& cr = i;
cr = 7;
i = 11;
cout << cr << endl;
cout << r << endl;
cout << i << endl;
    }
```

```
: error C3892: 'cr' : you cannot assign to a variable that is const
```

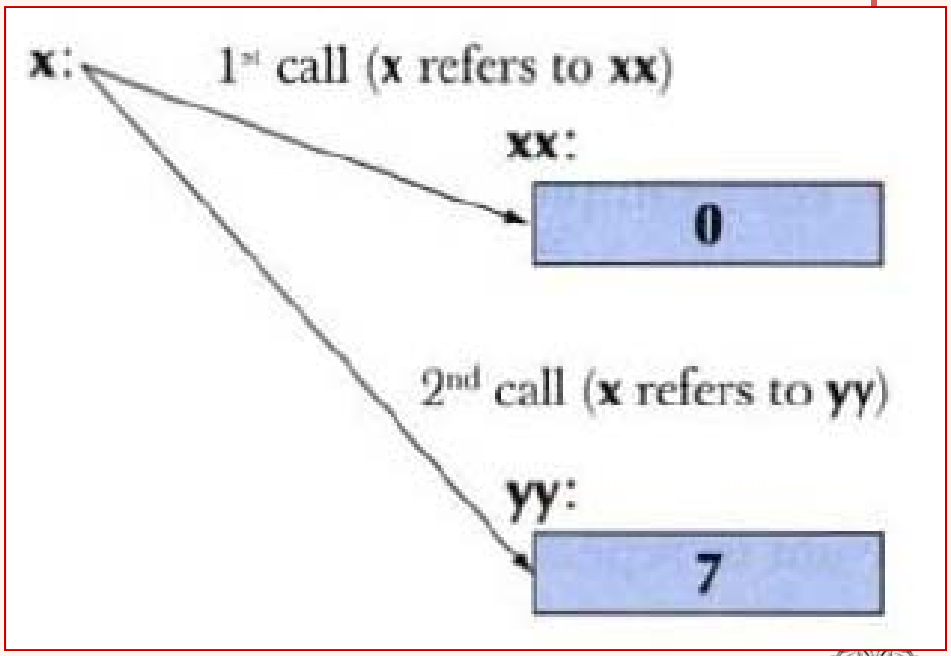
```
int main()
    {
int i = 7;
int& r = i;
r = 9;
const int& cr = i;
i = 11;
cout << cr << endl;
cout << r << endl;
cout << i << endl;
    }
```

```
11
11
11
```

فراخوانی با ارجاع

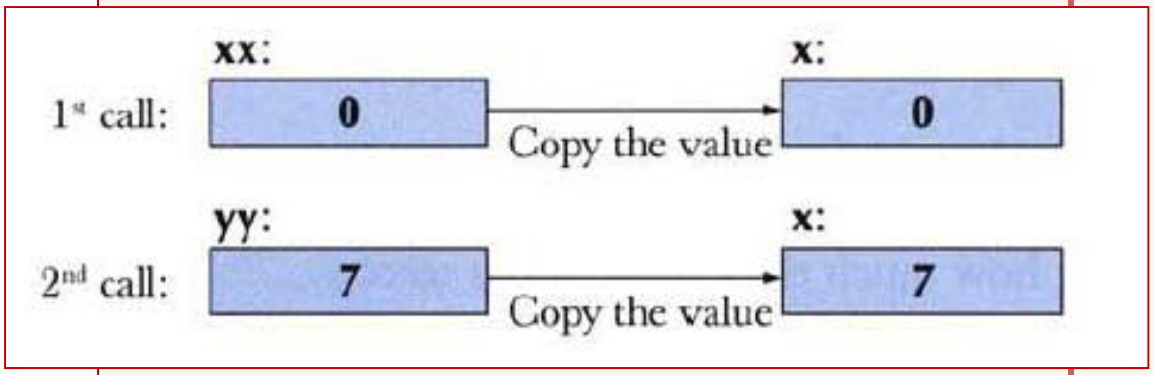
```
int f(int& x)
{
  x = x+1;
  return x;
}
int main()
    {
int xx=0;
cout << f(xx)<<endl;
cout << xx << endl;
int yy = 7;
cout << f(yy)<< endl;
cout << yy << endl;
}
```

```
1
1
8
8
```

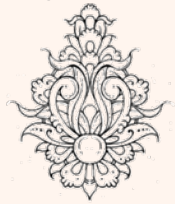


فراخوانی با مقدار

```
int f(int x)
{
   x = x+1;
   return x;
}
int main()
     {
int xx=0;
cout << f(xx)<<endl;
cout << xx << endl;
int yy = 7;
cout << f(yy)<< endl;
cout << yy << endl;
}
```




```
1
0
8
7
```



مثال

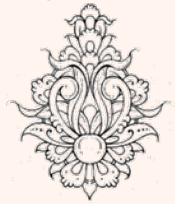
```
void f(int a, int& r, const int& cr){
   ++a; ++r; ++cr;
}
void g(int a, int& r, const int& cr){
   ++a; ++r; int x = cr; ++x;
 }

int main(){
   int x = 0;
   int y = 0;
   int z = 0;
   g(x,y,z);
   g(1,2,3);
   g(1,y,3);
}
```

error: cr is const

x==0; y==1; z==0

error: reference argument r needs a variable to refer to

ok: since cr is const we can pass “a temporary”



فراخوانی به وسیله‌ی اشاره‌گر

```
void swapnum(int&,int&);
int main() {
  int a = 10;
  int b = 20;

  swapnum(a,b);
  cout<<" a is:" << a << "\t b is:" <<b <<endl;
  return 0;
}
void swapnum(int& i, int& j)
  int temp;
  temp = i;
  i = j;
  j = temp;
}
```

```
a is:20         b is:10
```

```
void swapnum(int*,int*);
int main() {
  int a = 10;
  int b = 20;

  swapnum(&a,&b);
  cout<<" a is:" << a << "\t b is:" <<b <<endl
  return 0;
}
void swapnum(int *i, int *j) {
  int temp;
  temp = *i;
  *i = *j;
  *j = temp;
}
```

```
a is:20         b is:10
```

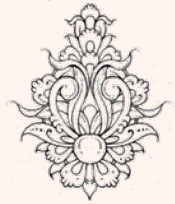
تمرین

تابعی بنویسید که رشته‌ای را به عنوان ورودی گرفته (یک اشاره‌گر به کاراکتر) طول آن را بازگرداند.

```
int strln(char *s)
```

```
using namespace std;
int strln(char *);
int main ()
{
  char a[50];
  cout << "Enter string:";
  cin.getline(a,50);
  cout << "The length of the string is:"<<strln(a)<<endl;
    }
int strln(char *s)
    {
    char *p=s;
    while(*p!='\0')
        p++;
    return p-s;
    }
```

```
Enter string:This is a test
The length of the string is:14
```



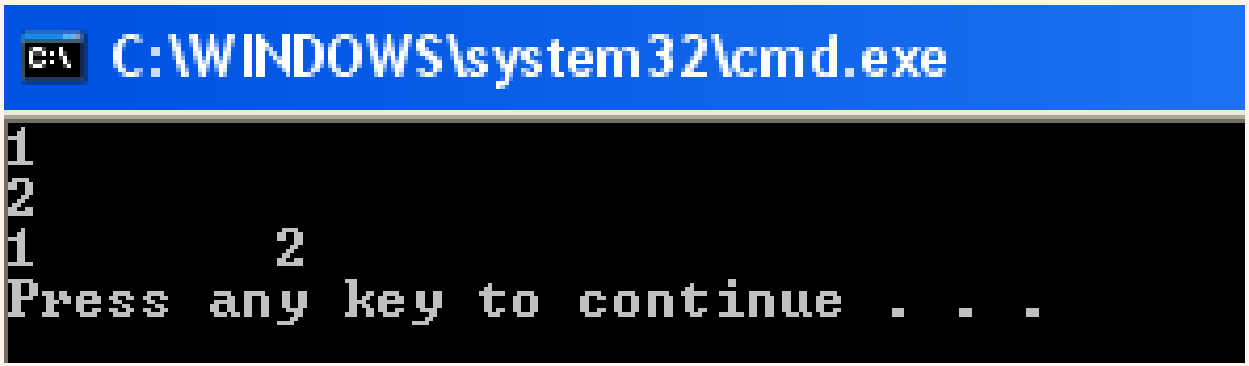
مثال

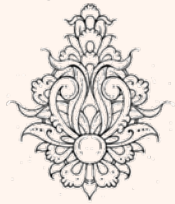
```
#include <iostream>
 using namespace std;


int main(void) {
    int n = 108;                    // an int
    int* p = &n;                   // a pointer to an int
    cout<< "The address of p is:"<< p <<"and the value is:"<<*p <<endl;
    ++(*p);                        // OK: increments int *p
    cout<< "The value of *p is:"<<*p <<endl;
    ++p;                            // OK: increments of address p
    cout<< "The address of p is:"<< p <<"and the value is:"<<*p <<endl;
        }//end main
```

```
The address of p is:0012FF28and the value is:108
The value of *p is:109
The address of p is:0012FF2Cand the value is:-858993460
```

مثال

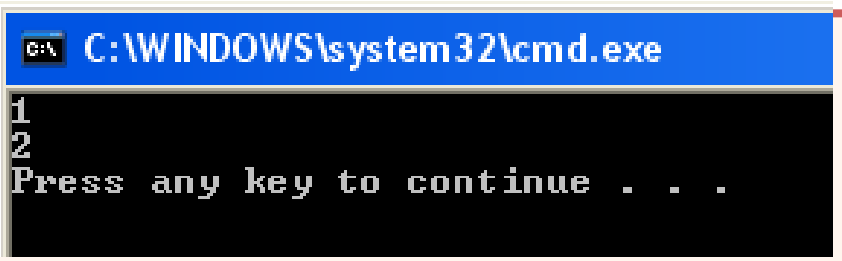
```
int main()
{
    int n[]={1,2,3,4};
    int *p;
    p=n;
    cout << *p++ <<endl;
    cout << *p <<endl;
    cout<< n[0]<<'\t'<<n[1]<<endl;
}
```

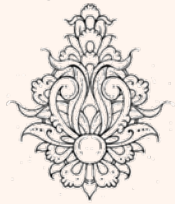




مثال

```
#include <iostream>
using namespace std;
int* f(int* x) {
     (*x)++;
     return x;
}
int& g(int& x) {
     x++; // Same effect as in f()
     return x;
}
int main() {
     int a = 0;
     cout<<*f(&a)<<endl; // Ugly (but explicit)
     cout<<g(a)<<endl; // Clean (but hidden)
}
```

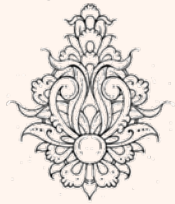




مثال

```
int* f(int* x) {
     (*x)++;
     return x;
}
int& g(int& x) {
     x++; // Same effect as in f()
     return x;
}
int main() {
     int a = 0;
     cout<<*f(&a)<<endl; // Ugly (but explicit)
     cout<<(g(a)=10)<<endl; // Clean (but hidden)
     g(a);
     cout<<a<<endl;
}
```

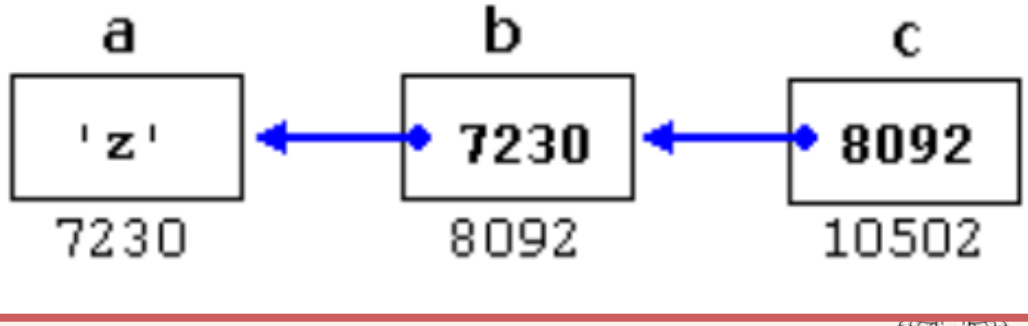
```
C:\WINDOWS\system32\cmd.exe
1
10
11
Press any key to continue . . .
```



اشاره‌گر به اشاره‌گر

- اگر اشاره‌گری آدرس اشاره‌گر دیگری را در خود ذخیره نماید، به آن «اشاره‌گر به اشاره‌گر» می‌گویند.
- ساختار تعریف اشاره‌گر به اشاره‌گر به صورت زیر است:

<نوع> ** نام متغیر;



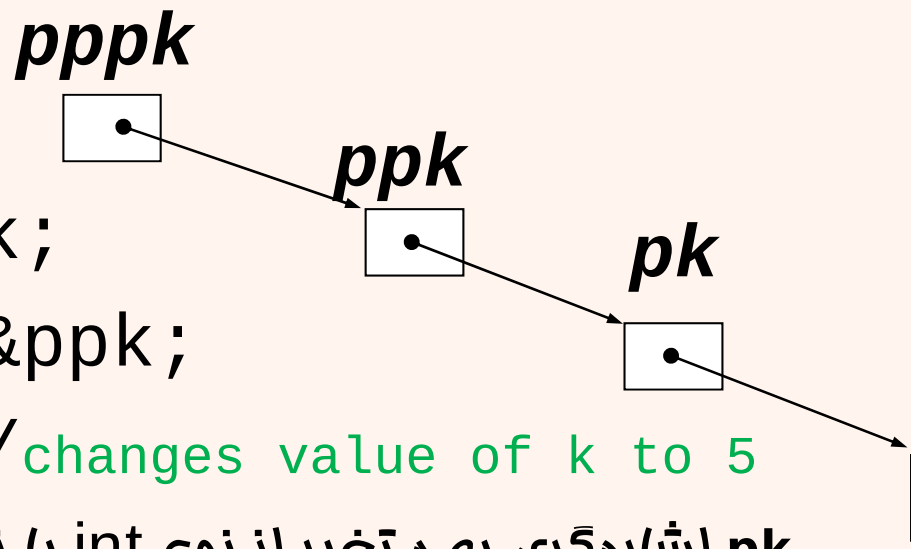

```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```

- c has type char** and a value of 8092
- *c has type char* and a value of 7230
- **c has type char and a value of 'z'

اشاره‌گر به اشاره‌گر

- **یک اشاره‌گر می‌تواند به اشاره‌گر دیگری اشاره کند.**
- **مثال:**

```
int k = 3;
int* pk = &k;
int** ppk = &pk;
int*** pppk = &ppk;
***pppk = 5; //changes value of k to 5
```



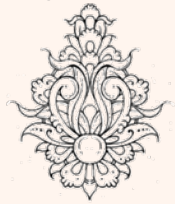
- **pk** اشاره‌گری به متغیر از نوع int با نام k است.
- ppk اشاره‌گری به اشاره‌گر pk است و اشاره‌گر pppk هم به اشاره‌گر ppk اشاره دارد.

مثال

- نتیجه‌ی کد زیر چیست؟

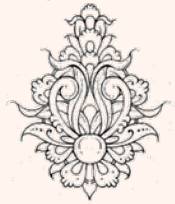
```
#include <iostream>
  using namespace std;
int main()
    {
      float x,y;
      int *p;
      x=3.1415;
      p = &x;
      y = *p;

    cout << "The address of x is " << p<<endl;
    cout<< "\nThe value of x is " << y <<endl;

      return 0; // indicates successful termination
   } // end main*/
```

```
1>e:\ac++\code\pointers\pointers\5th.cpp(7) : warning C4305: '=' : truncation from 'double' to 'float'
1>e:\ac++\code\pointers\pointers\5th.cpp(8) : error C2440: '=' : cannot convert from 'float *__w64 ' to 'int *'
1>        Types pointed to are unrelated; conversion requires reinterpret_cast, C-style cast or function-style cas
1>e:\ac++\code\pointers\pointers\5th.cpp(9) : warning C4244: '=' : conversion from 'int' to 'float', possible loss
```



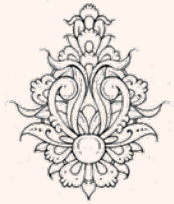
void*

- اشاره‌گری است که «عدم وجود نوعی خاص» را نشان می‌دهد.
- به وسیله‌ی این نوع اشاره‌گر می‌توانیم به هر نوع از داده از قبیل int, float , ... اشاره نماییم.
- محدودیت بزرگی که وجود دارد این است که در این حالت فرایندهایی همانند dereference را به علت مشخص نبودن نوع نمی‌توان صورت داد.
- فرایند مذبور با استفاده از casting صورت می‌پذیرد.



void*

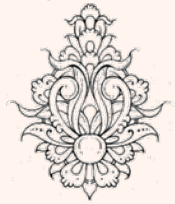
```
void f (int* pi )
{
void * pv = pi ; ok: implicit conversion of int* to void*
*pv ;   error: can't dereference void*
pv ++;    error: can't increment void*
int *pi2 =(int *)(pv); explicit conversion back to int*
double * pd1 = pv ;  error
double *pd2 =pi;  error
double *pd3 =(double *)(pv);unsafe
}

void main() {
 int x=10;
   f(&x);
    }
```



void*

```
// increaser
#include <iostream>
using namespace std;

void increase (void* data, int psize)
{
  if ( psize == sizeof(char) )
  { char* pchar; pchar=(char*)data; ++(*pchar); }
  else if (psize == sizeof(int) )
  { int* pint; pint=(int*)data; ++(*pint); }
}

int main ()
{
  char a = 'x';
  int b = 1602;
  increase (&a,sizeof(a));
  increase (&b,sizeof(b));
  cout << a << ", " << b << endl;
  return 0;
}
```

y, 1603



- **وقتی یک اشاره‌گر داشته باشیم:**
- int* p; // p is a pointer to a int
- p یک آدرس است.
- در این حالت p ایجاد شده است اما به جایی (اختصاص داده شده) اشاره نمی‌کند.
  - زیرا هنوز آدرسی معتبر درون آن قرار نگرفته است.
    - به چنین اشاره‌گری «***اشاره‌گر سرگردان***» می‌گویند.

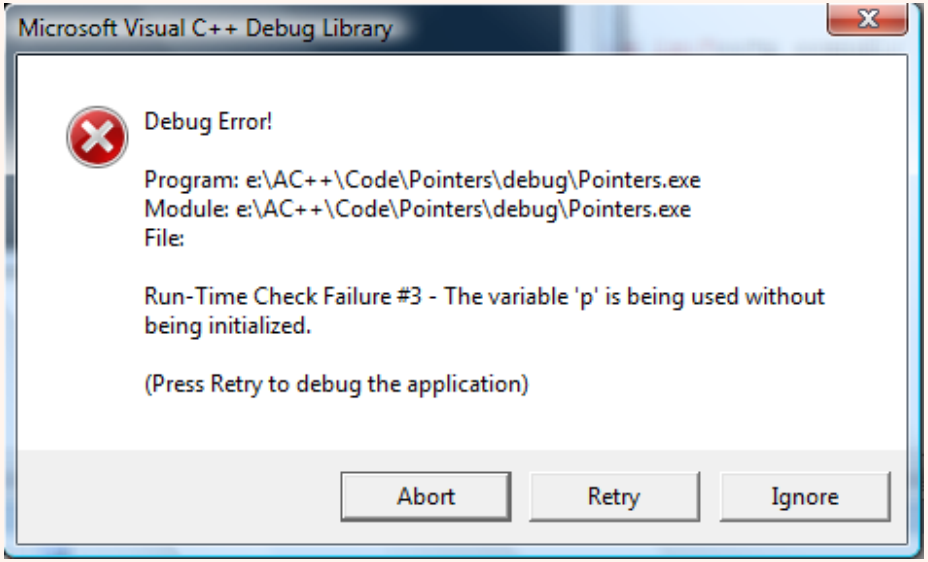
Wild Pointer

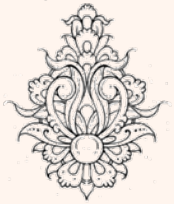
اگر سعی کنیم یک اشاره‌گر سرگردان را مقداریابی یا ارجاع کنیم با مشکل مواجه می‌شویم.

مثال

```
#include <iostream>
 using namespace std;

int main(void)
    {
     int* p ;          // a pointer
     *p = 3;
     cout<< "The address of p is:"<< p <<"and the value is:"<<*p <<endl;
           }//end main
```

```
warning C4700: uninitialized local variable 'p' used
```





Visual Studio

مثال

```cpp
#include <iostream>
 using namespace std;

int main(void){
     int* p ;          // a pointer
     *p = 3;
     cout<< "The address of p is:"<< p <<"and the value is:"<<*p <<endl;
}//end main
```

```
28.cpp: In function 'int main()':
28.cpp:6: warning: 'p' is used uninitialized in this function
```

```
The address of p is:0xbf95ca38and the value is:3
```

Linux/g++

این مسأله می‌تواند منجر به تغییرات ناخواسته در حافظه شود، یا موجب بروز خطای **segmentation fualt** در حین اجرا شود. باید به نحوی تشخیص اشاره‌گر سرگردان است یا به جایی معتبر اشاره می‌کند.

راه حل

```
#include <iostream>
 using namespace std;

int main(void)
    {
     int* p ;          // a pointer
     int x=0;
     p=&x; // now p points to x
     *p = 3;// assign new value to the address which p points to
     cout<< "The address of p is:" << p <<"  and the value is:" << *p <<endl;
           }//end main
```

```
The address of p is:0012FF1C  and the value is:3
```

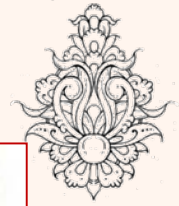
اشاره‌گر null

- **اشاره‌گر null** اشاره‌گری است که به هیچ کجا ازحافظه‌ی مجاز اشاره نمی‌کند.
- هنگامی‌که اشاره‌گری را اعلان می‌کنیم، بهتر است ابتدا آن را **null** کنیم **تا مقدار زباله‌ی آن پاک شود**.
- جهت کنترل هر چه بهتر برنامه استفاده می‌شود.

```
double* p0 = 0;          // the null pointer
```

```
if (p0 != 0)     // consider p0 valid
```

```
if (p0)     // consider p0 valid; equivalent to p0!=0
```



« اشاره‌گر به یک ثابت » با « اشاره‌گر ثابت » متفاوت است ←

اشاره‌گر ثابت

```
int main() {
    int n = 44;
    int* const cp = &n;
    cout<< "The address is:"<< cp <<" and the value is:"<<*cp <<endl;
    ++(*cp);
    cout<< "The value (*cp) is:"<<*cp <<endl;
    ++cp;
    cout<< "The address is:"<< cp <<" and the value is:"<<*cp <<endl;
        }
```

a const pointer to an int

illegal: pointer cp is const

```
1>------ Build started: Project: Pointers, Configuration: Debug Win32 ------
1>Compiling...
1>9th.cpp
1>e:\ac++\code\pointers\pointers\9th.cpp(14) : error C3892: 'cp' : you cannot assign to a variable that is const
```

```
int main() {
    int n = 44;                    // an int
    int* const cp = &n;                    // a const pointer to an int
    cout<< "The address is:"<< cp <<" and the value is:"<<*cp <<endl;
    ++(*cp);
    cout<< "The value (*p) is:"<<*cp <<endl;
        }
```

```
The address of p is:0012FF28and the value is:44
The value of *cp is:45
```

اشاره‌گر به ثابت

```
int main() {
     const int k = 108;
     const int * ptc = &k;
    cout<< "The address  is:"<< ptc <<"and the value is:"<<*ptc <<endl;
    ++(*ptc);
    cout<< "The The new address is:"<<ptc <<endl;
        }
```

a pointer to a const int

illegal: int *ptc is const

```
: error C3892: 'ptc' : you cannot assign to a variable that is const
```

```
int main() {
     const int k = 108;
     const int * ptc = &k;
    cout<< "The address  is:"<< ptc <<"and the value is:"<<*ptc <<endl;
    ++ptc;
    cout<< "The The new address is:"<<ptc <<endl;
        }
```

```
The address of p is:0012FF28and the value is:108
The The new address is:0012FF2C
```