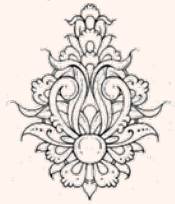
مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی سی‌ام

اشاره‌گر ۳

دانشگاه شهید بهشتی
پاییز ۱۳۹۲
دانشکده‌ی مهندسی برق و کامپیوتر
احمد محمودی ازناوه

فهرست مطالب

- اشاره‌گر به تابع
  - آرایه‌ای از اشاره‌گر به تابع
- تخصیص حافظه‌ی پویا
  - اشاره‌گر معلق
  - نشت حافظه
  - تخصیص حافظه‌ی دوبعدی



مثال

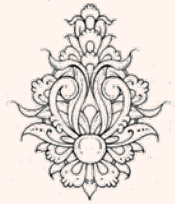
- نتیجه‌ی کد زیر چیست؟

```
#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
```

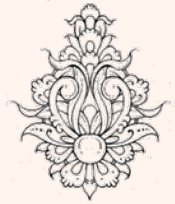
void*

- 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*

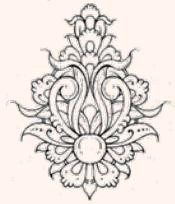
```
void f (int* pi )
{
void * pv = pi ; ok: implicit conversion of int* to void*
*pv ;   error: can't dereference void*
pv ++;    error: can't increment void*
int *pi2 =(int *)(pv); explicit conversion back to int*
double * pd1 = pv ;  error
double *pd2 =pi;  error
double *pd3 =(double *)(pv);unsafe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

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



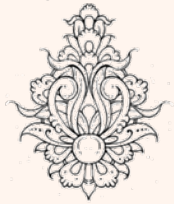
void*

```
// 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

```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
```

این مسئله می‌تواند منجر به تغییرات ناخواسته در حافظه شود، یا موجب بروز خطای **segmentation fualt** در حین اجرا شود

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
```

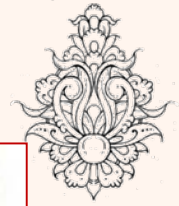
اشاره‌گر null

- **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

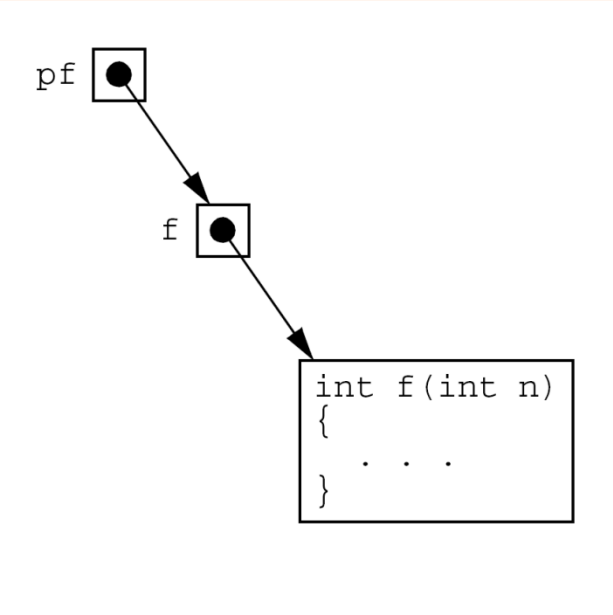
```
: error C3892: 'ptc' : you cannot assign to a variable that is const
```

```
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
```

اشاره‌گر به تابع

- **نام یک تابع** مثل نام یک آرایه، یک **اشاره‌گر ثابت** است.
- نام تابع، آدرسی از حافظه را نشان می‌دهد که کدهای درون تابع در آن قسمت جای گرفته‌اند.
- بنابراین اگر اشاره‌گری به تابع تعریف شود مانند این است که:
  - اشاره‌گری به یک اشاره‌گر ثابت تعریف شده است.

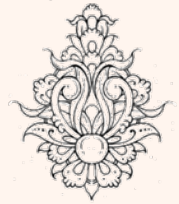


اشاره‌گر به تابع (ادامه...)

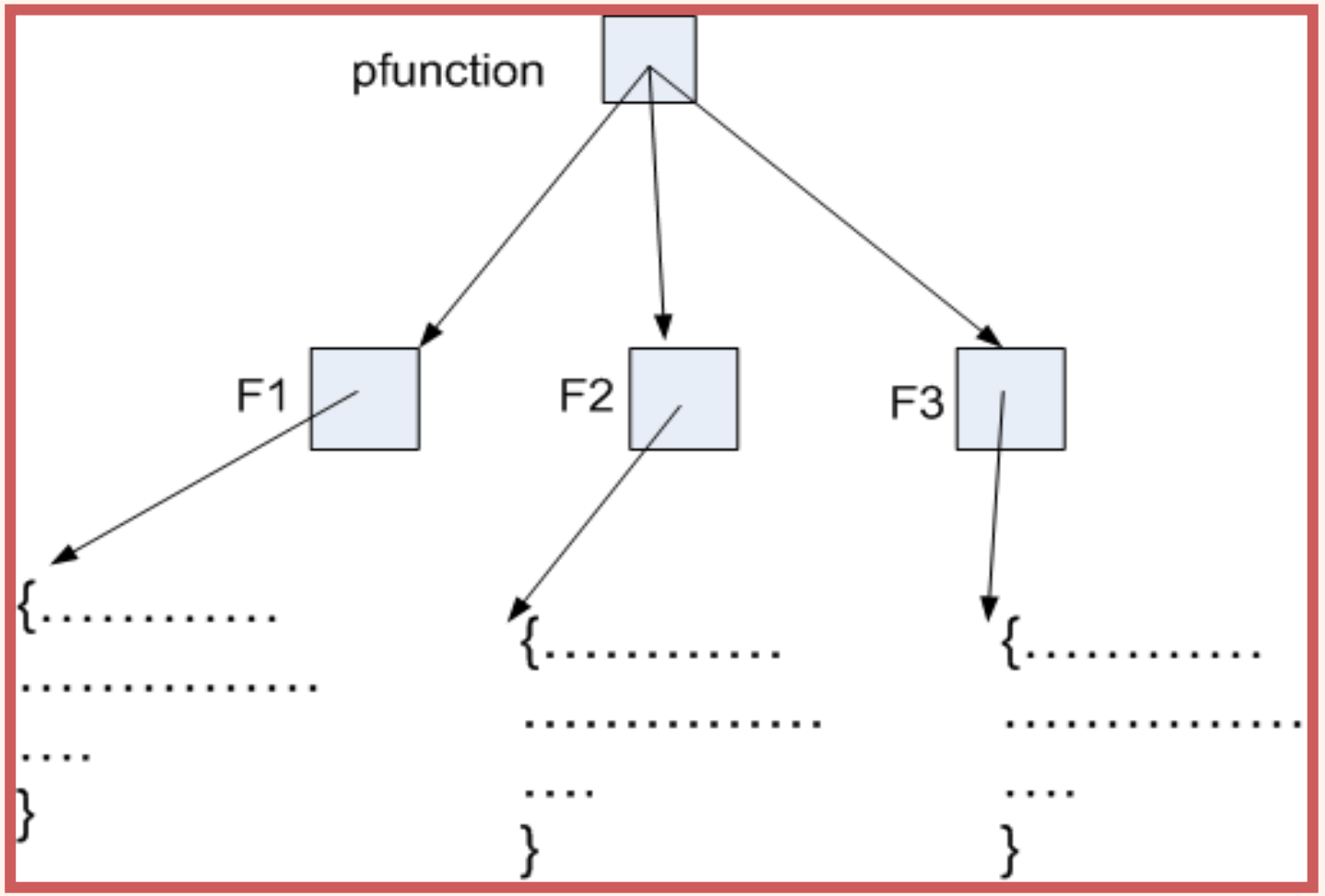
- **ساختار تعریف مانند زیر است:**

```
int function(int);// declares a function
int (*pfunction)(int);// declares pointer to function
pfunction = &function;// assigns address of function
```

- همان‌گونه که مشاهده شد شیوه‌ی تعریف همانند تابع معمولی است تنها نام تابع با علامت «*» و در پرانتز باید نوشته شود.
- از این خاصیت برای ارسال تابع به عنوان آرگومان تابع دیگر است استفاده می‌شود.



اشاره‌گر به تابع



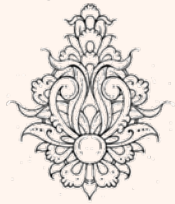
مثال

```
void error (char* s ) { cout << s <<endl; }

int main()
{

void (*f1)(char* ) = &error ; // ok
void (*f2 )(char* ) = error ; // also ok; same meaning as &error
f1 ("f1 function without dereference."); // ok
(*f1)("f1 function with dereference. "); // also ok
f2("f2 function without dereference.");
(*f2)("f2 function with dereference"); // also ok
}
```

```
f1 function without dereference.
f1 function with dereference.
f2 function without dereference.
f2 function with dereference
```

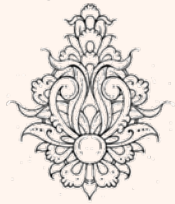
```
// pointer to functions
#include <iostream>
using namespace std;
int addition (int a, int b){
  return (a+b);
}
int subtraction (int a, int b){
  return (a-b);
}
int operation (int x, int y, int (*functocall)(int,int)){
  int g;
  g = (*functocall)(x,y);
  return (g);
}
int main (){
  int m,n;
  int (*minus)(int,int) = &subtraction;

  m = operation (7, 5, &addition);
  n = operation (20, m, minus);
  cout << "The output result is:" << n <<endl;
  return 0;
}
```

```
The output result is:8
```



مثال

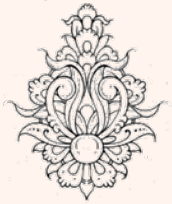
- برنامه‌ای بنویسید که شامل دو تابع p3 و p4 باشد که یک ورودی و یک خروجی از جنس int داشته باشد.
  - P3 ورودی را به توان ۳ رسانده باز می‌گرداند.
  - P4 ورودی را به توان ۴ رسانده باز می‌گرداند.
- به وسیله‌ی تعریف یک تابع دیگر (mixedfunction) و با استفاده از اشاره‌گر به تابع مناسب را در صورت لزوم صدا بزنند.
- mixedfunction دو ورودی دارد یکی از جنس تابع و دیگری int که آرگومان دوم به عنوان ورودی p3 و p4 استفاده می شود.



پاسخ

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

int mixedfunction(int (*)(int), int);
int p3(int);
int p4(int);
int main(){
  cout << (*mixedfunction)(p3,3) << endl; // 1 + 8 + 27
  cout << (*mixedfunction)(p4,3) << endl; //1 + 16 + 81
}
int mixedfunction(int (*pf)(int k), int n){
  // returns the sum f(0) + f(1) + f(2) + ... + f(n-1):
  int s = 0;
  for (int i = 1; i <= n; i++)
    s += (*pf)(i);
  return s;
}
int p4(int k){
  return k*k*k*k;
}
int p3(int k){
  return k*k*k;
}
```

```
36
98
```



آرایه‌ای از اشاره‌گر به توابع

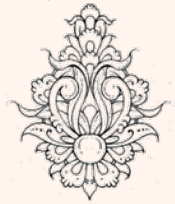
```
void function0( int a ){
  cout << "You entered " << a << " so function0 was called\n\n";
} // end function function0

void function1( int b ){
  cout << "You entered " << b << " so function1 was called\n\n";
} // end function function1

void function2( int c ){
  cout << "You entered " << c << " so function2 was called\n\n";
} // end function function2
```

```
// function prototypes --
#include <iostream>
using namespace std;
void function0( int );
void function1( int );
void function2( int );
int main(){
  void (*f[ 3 ])(int) = {&function0,&function1,&function2 };
  int choice;

  cout << "Enter a number between 0 and 2, 3 to end: ";
  cin >> choice;

  while ( ( choice >= 0 ) && ( choice < 3 ) ) {
    (*f[ choice ])( choice );
    cout << "Enter a number between 0 and 2, 3 to end: ";
    cin >> choice;
  } // end while
  cout << "Program execution completed." << endl;
  return 0; // indicates succ
} // end main
```

```
Enter a number between 0 and 2, 3 to end: 2
You entered 2 so function2 was called

Enter a number between 0 and 2, 3 to end: 3
Program execution completed.
```

تخصیص حافظه پویا

- در زبان c برای تخصیص حافظه پویا یعنی در زمان اجرا از تابع malloc استفاده می‌شود.
  - پس از خروج از یک تابع لازم است حافظه‌ای را که با استفاده از malloc تخصیص داده‌اید با استفاده از تابع free به سیستم بازگردانید.
  - متغیرهایی که در یک تابع تعریف می‌شود در هنگام خروج از تابع به سیستم بازگردانده می‌شوند ولی حافظه‌ای را که با استفاده از malloc تخصیص می‌دهیم در انتها آزاد نمی‌شود.
  - به وسیله‌ی تابع free حافظه‌ی تخصیص داده شده با malloc را آزاد می‌نماییم.
  - اگر حافظه آزاد نشود نشتی حافظه یا memory leak را سبب خواهد شد.

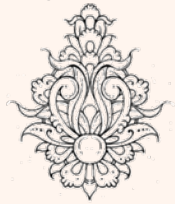


تخصیص حافظه پویا (ادامه...)

- الگوی تابع

```
void * malloc ( size_t size );
```

- در صورت موفقیت اشاره‌گری به ابتدای بلوکی از حافظه تخصیص می‌یابد.
- نوع خروجی از جنس void * است که می‌باید براساس نوع مورد نظر casting صورت پذیرد.
- در غیر این‌صورت خطای compile گزارش داده می‌شود.
- در صورت عدم موفقیت مقدار صفر یا همان NULL را برمی‌گرداند.



تخصیص حافظه پویا (C)

```
int main() {
    int n,i,max;
    int *a;
    cout<< "Enter the size of array:";
    cin >> n;
    a= (int*)malloc( n*sizeof(int));//assign dynamic array
    for(i=0;i<n;i++)
        cin >> a[i];
    max=a[0];
    for(i=1;i<n;i++)
        if(max<a[i])
            max=a[i];
    free(a);//free the allocated ar
    cout << "max=" <<max <<endl;
    }
```

```
Enter the size of array:10
4
7
5
9
3
5
8
12
5
76
max=76
```

تخصیص حافظه پویا

- در زبان c++ برای تخصیص حافظه پویا (در زمان اجرا) از تابع new استفاده می‌شود.

پس از خروج از یک تابع لازم است حافظه‌ای را که تخصیص داده‌اید به سیستم بازگردانید.

متغیرهایی که در یک تابع تعریف می‌شود در هنگام خروج از تابع به سیستم بازگردانده می‌شوند ولی حافظه‌ای را که با استفاده از تخصیص حافظه‌ی پویا در نظر گرفته می شود، در انتها آزاد نمی‌شود.

به وسیله‌ی تابع delete حافظه‌ی تخصیص داده شده را آزاد می‌نماییم.

اگر حافظه آزاد نشود نشتی حافظه یا memory leak را سبب خواهد شد.

تخصیص حافظه پویا

- در زبان c++ برای تخصیص و بازگرداندن حافظه‌ی پویا به سیستم از عملگرهای new و delete استفاده می‌نماییم.

اشاره گر= new < نوع>

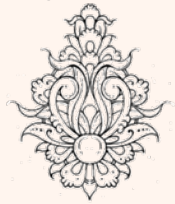
delete اشاره گر

```
int *pn=new int;
delete pn;
```

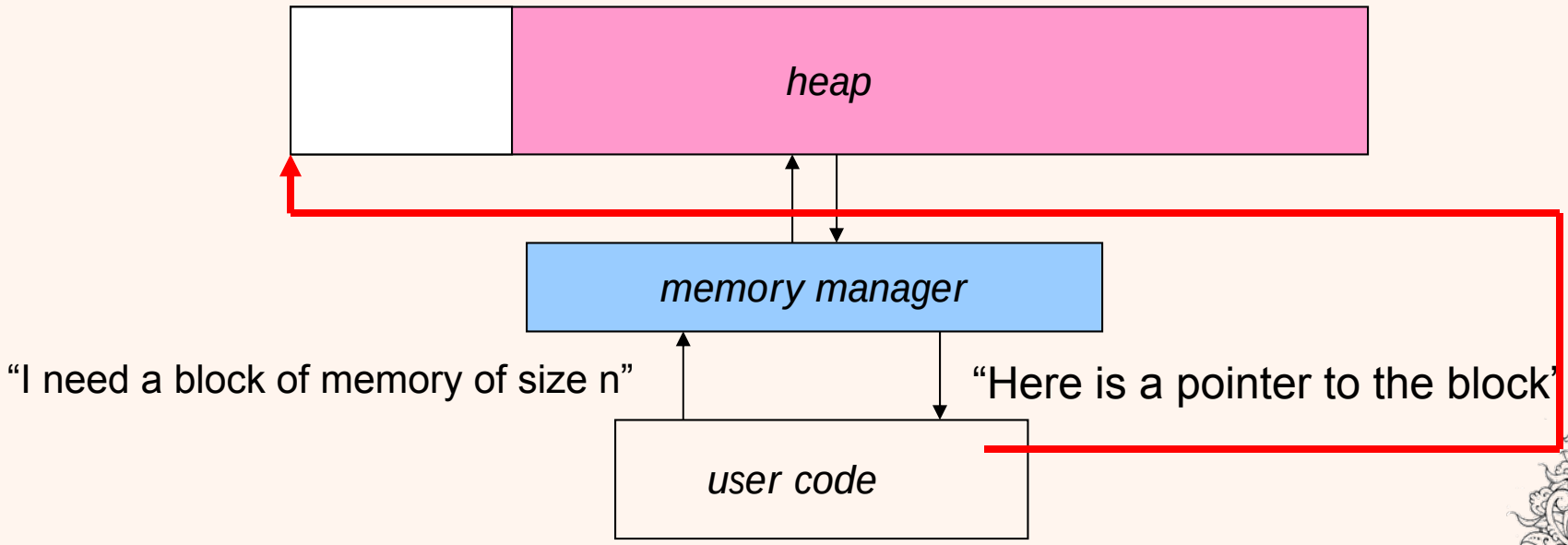
اشاره گر= new < نوع> [طول آرایه]

delete [ ] اشاره گر

```
int *p=new int[20];
delete [ ] p;
```

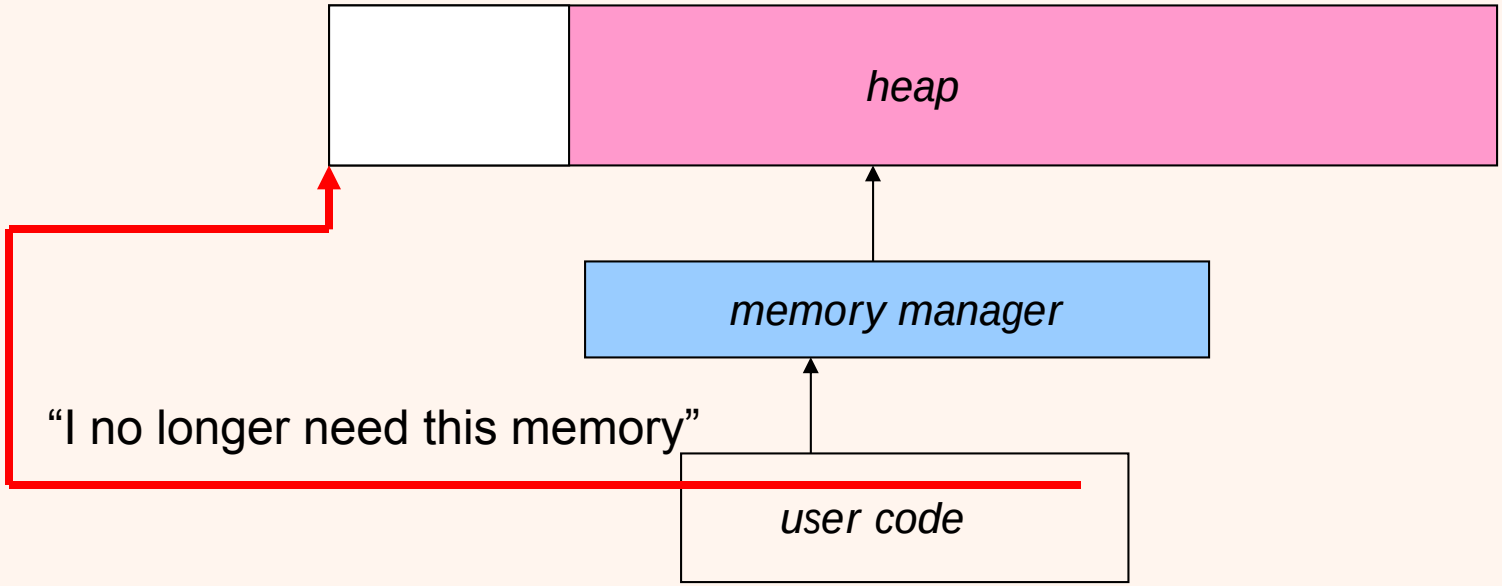


تخصیص حافظه


heap
memory manager
"I need a block of memory of size n"
"Here is a pointer to the block"
user code

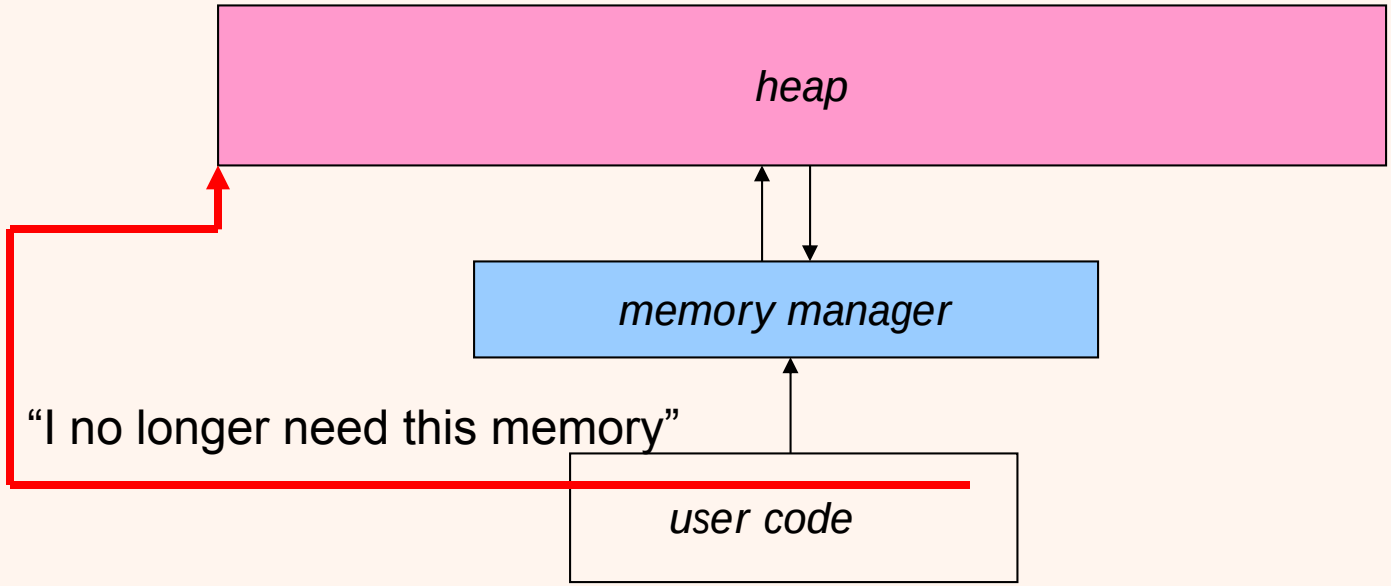
آزادسازی

http://vorlon.case.edu



آزادسازی

http://vorlon.case.edu



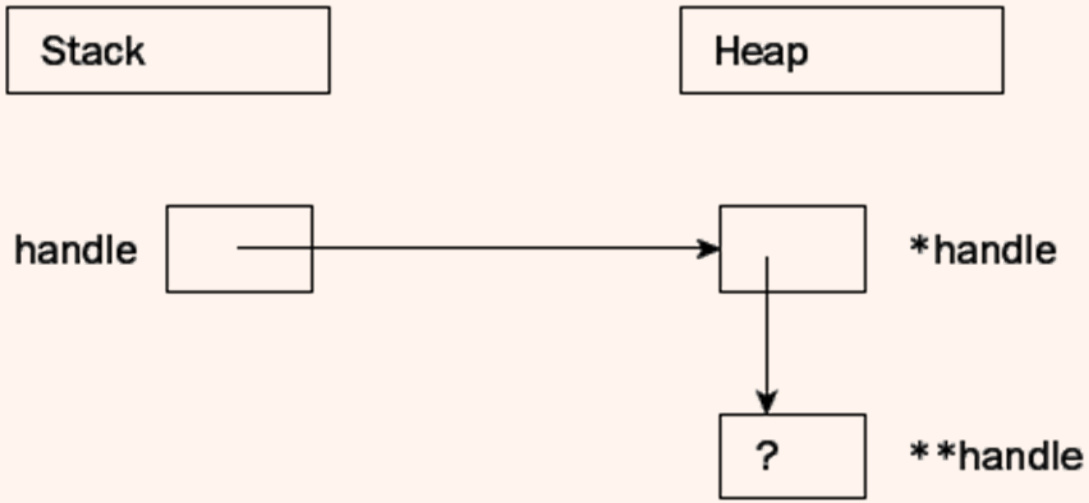
تخصیص حافظه پویا

```
int i=7;
```


```
int *ptr;
ptr = new int;
```

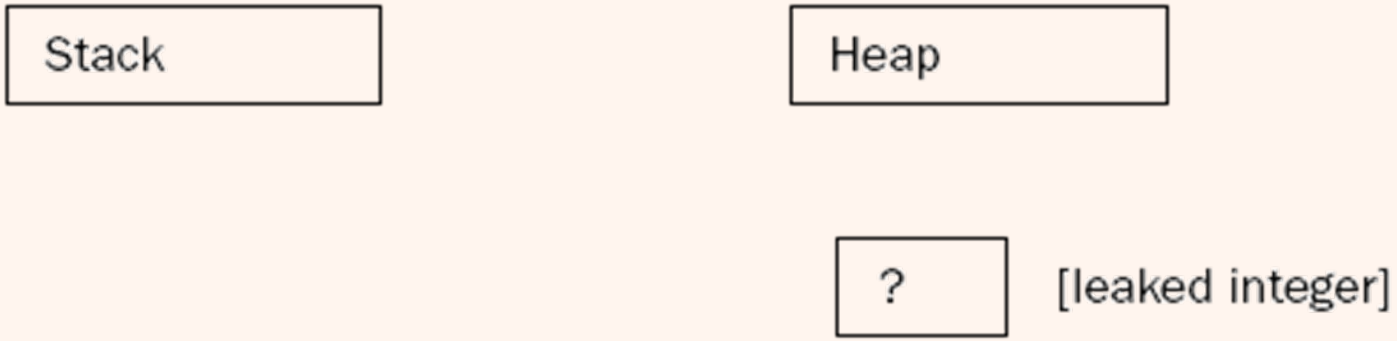
تخصیص حافظه پویا

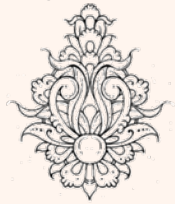
```
int **handle;
handle = new int*;
*handle = new int;
```



تخصیص حافظه پویا

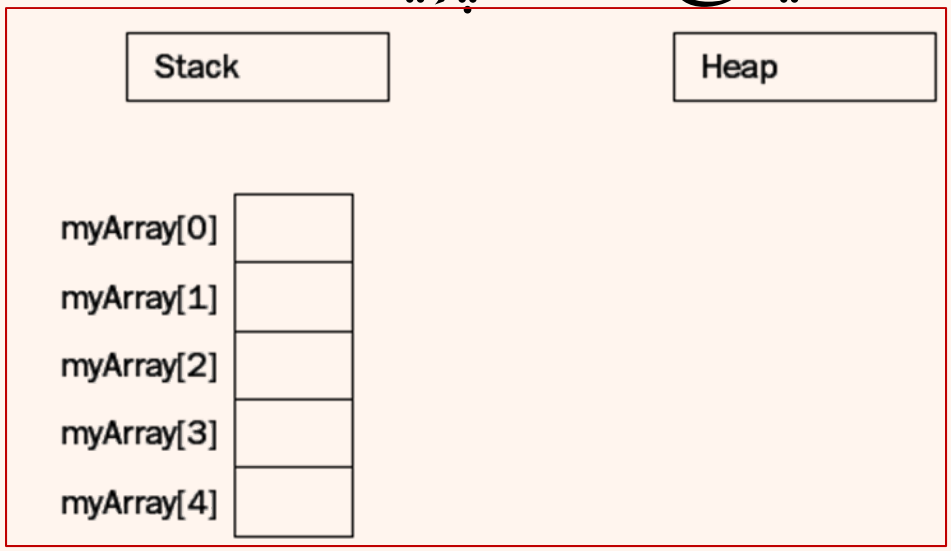
```
void leaky(){
  new int;   //BUG! Orphans memory!
  cout<<“I just a leaked an int!”<<endl;
}
```



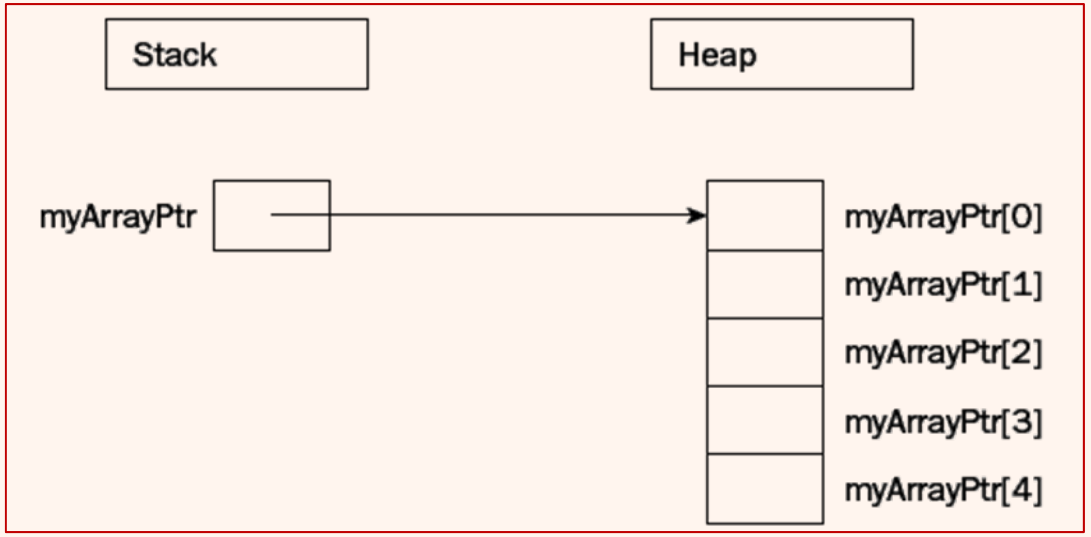


تخصیص حافظه پویا (ادامه...)

```
int myArray[5];
```




```
int* myArrayPtr = new int[5];
```



تخصیص حافظه پویا (C++)

```
int main() {
    int n,i,max;
    int *a;
    cout<< "Enter the size of array:";
    cin >> n;
    a= new int[n];//assign dynamic array
    for(i=0;i<n;i++)
        cin >> a[i];
    max=a[0];
    for(i=1;i<n;i++)
        if(max<a[i])
            max=a[i];
    delete []a;//free the allocated array
    cout << "max=" <<max <<endl;
    }
```

```
Enter the size of array:10
6
8
9
45
34
5
7
4
2
56
max=56
```

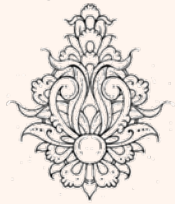
نشت حافظه

```
void fn( ) {
     int* c = new int[10];

   ...

    return;
   }
```

حافظه‌ی اختصاص داده شده به c و به بلوک مورد نظر اشاره می‌کند

حافظه‌ی اختصاص داده شده آزاد نمی‌گردد است

```
void fn( ){
     int* c = new int[10];

   ...

   delete[] c;

    return;
   }
```



مثال

```
int main()
    {
     int* p ;          // a pointer
     float* pf=new float(3.56);
     p=new(int); // allocates storage for 1 int
     *p = 3;// assign new value to the assigned storage
     cout<< "The address (p) is:" << p <<"  and the value is:" << *p <<endl;
     cout<< "The address (pf) is:" << pf <<"  and the value is:" << *pf <<endl;
    }
```

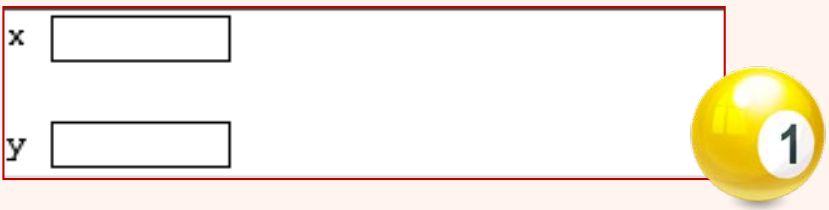
```
The address (p) is:001D1980  and the value is:3
The address (pf) is:001D1950  and the value is:3.56
```

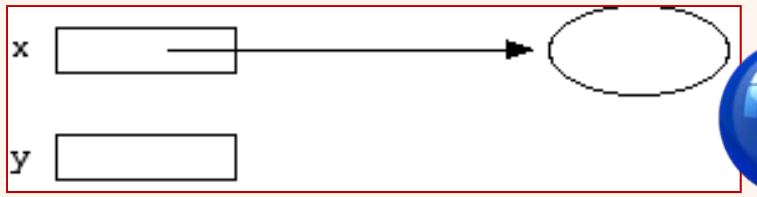
مثال

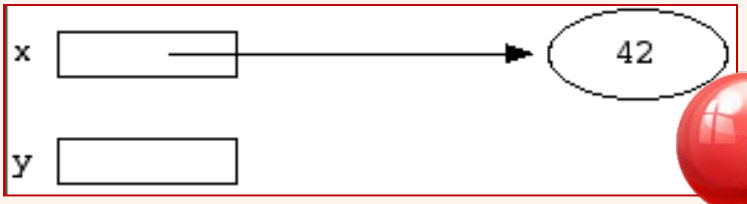
http://cslibrary.stanford.edu

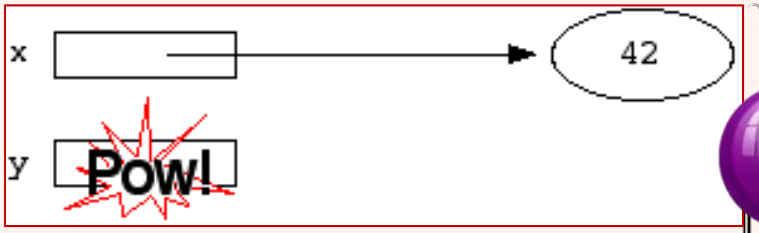
```
int  main() {
  int*     x;
  int*     y;

  x = new int;

  *x = 42;

  *y = 13;

  y = x;

  *y = 13;
}
```

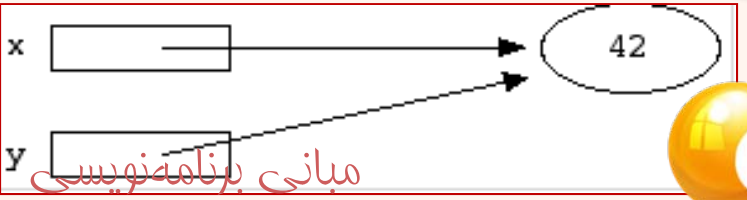
```
// Just Allocate the pointers x and y
//

// Allocate an int pointee,
// and set x to point to it

// Dereference x to store 42 in its pointee

// CRASH -- y does not have a pointee yet

// Pointer assignment sets y to point to x's pointee

// Dereference y to store 13 in its (shared) pointee
```

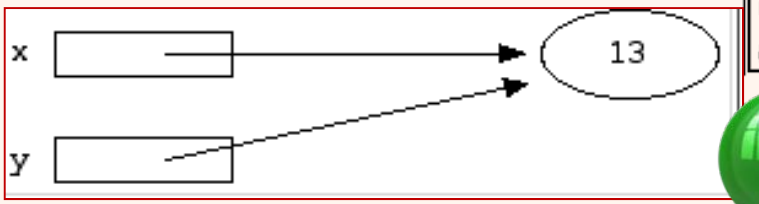












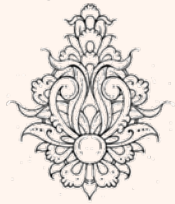
deallocated pointer

dangling Pointer

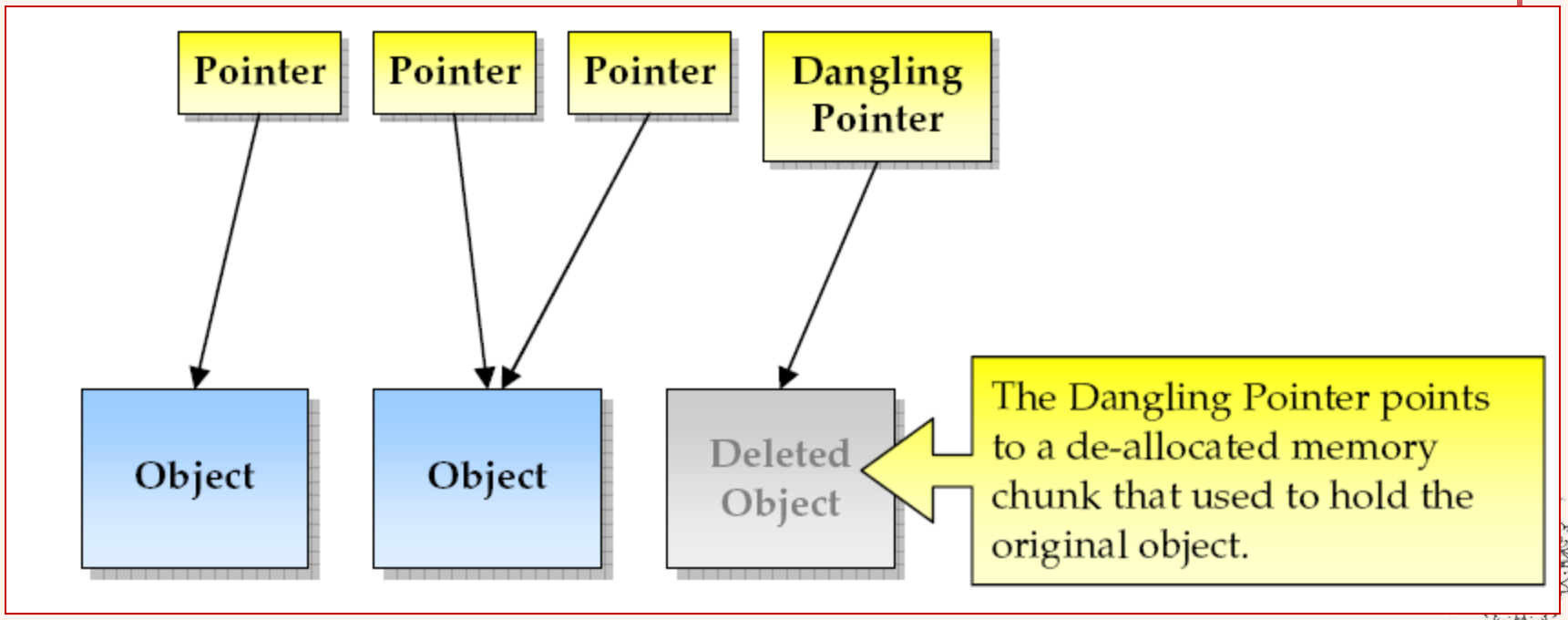
اشاره‌گر معلق

- هنگامی‌که اشاره‌گر به شی تعریف شده باشد و در جریان برنامه شی حذف شود یا حافظه‌ی اختصاص داده شده به اشاره‌گری آزاد شود.
  - در چنین حالتی اشاره‌گر موجود و reference آن وجود ندارد
  - چنین اشاره‌گری را معلق می‌نامند.

```
int main()
{
    char *dp = NULL;
    /* ... */
    {
        char c='a';
        dp = &c;

    } /* c falls out of scope */
      /* dp is now a dangling pointer */
}
```

اشاره‌گر معلق



تخصیص فضا برای آرایه‌ی دوبعدی

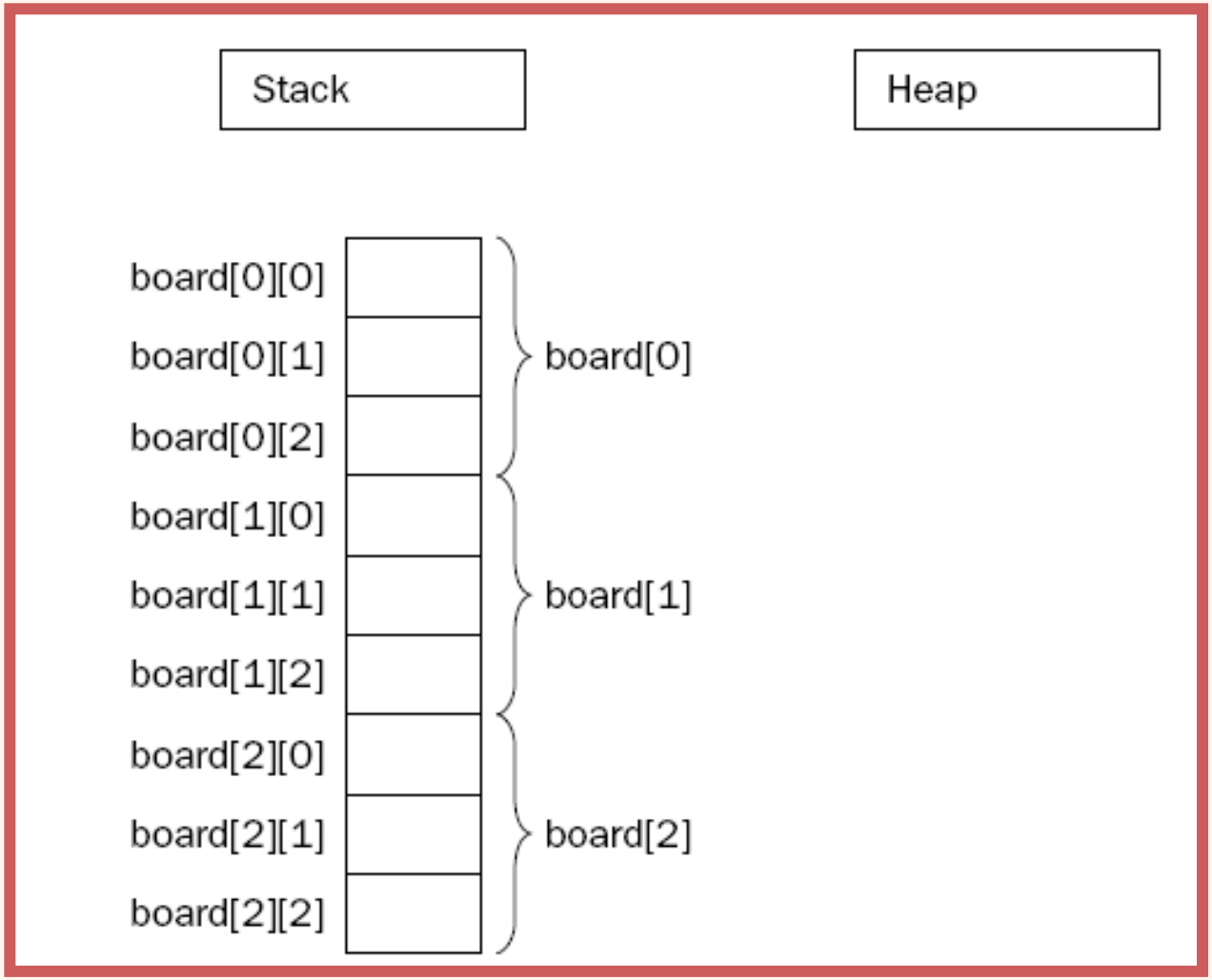
- برای تخصیص حافظه‌ی پویا برای آرایه‌ی پویای ۲ بعدی (متغیر از جنس اشاره‌گر به اشاره‌گر) به طریق روبرو عمل می‌نماییم:

```
int **dArray = 0;
//firstly memory allocated for elements of rows.
          dArray = new int* [ROWS] ;
 //then memory allocated for elements of each column.
        for( int i = 0 ; i < ROWS ; i++ )
              dArray[i] = new int[COLUMNS];
```

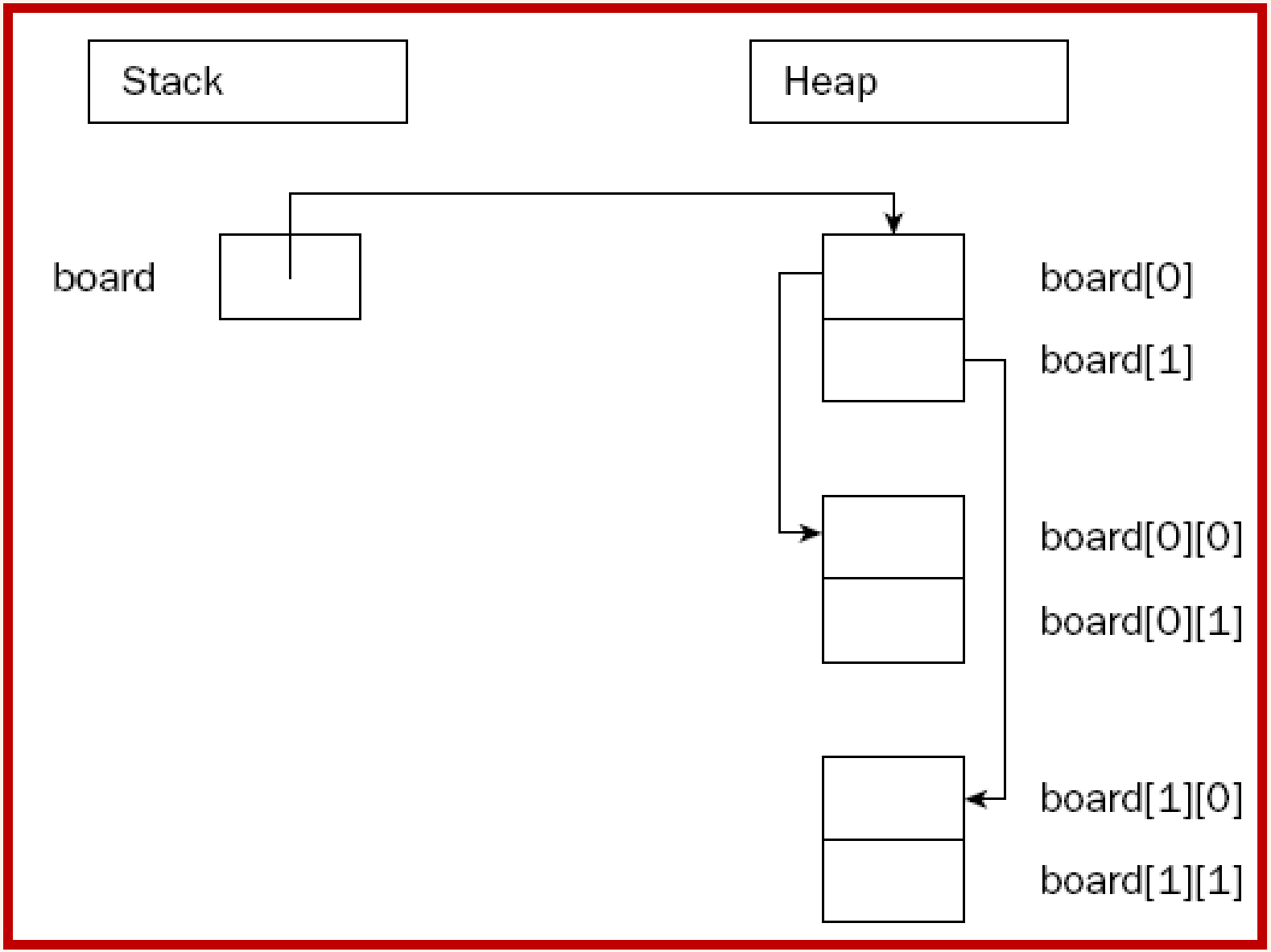
- برای آزادسازی نیز خواهیم داشت:

```
//free the allocated memory
 for( int i = 0 ; i < ROWS ; i++ )
       delete [ ] dArray[i] ;
delete [ ] dArray ;
```

آرایه‌ی دوبعدی

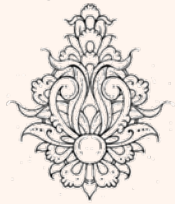


تخصیص فضا برای آرایه‌ی دوبعدی



مثال

- برنامه‌ای بنویسید که با استفاده از آرایه‌ی پویا یک آرایه‌ی دو بعدی را مقداردهی و سپس چاپ نماید.



مثال

```
void main()
{
float **float_values;
float val;
int n,m;
 //allocate memory
 cout << " please enter the value for m row and n col:"<<endl;
 cin>>m;
 cin>>n;
 float_values = new float*[m];
  for(int i=0;i<m;i++){
     float_values[i]=new float[n];
       for(int j=0;j<n;j++)
      {
       cout<<"\n enter element of"<<i<<" th row a
       cin>>val;
       float_values[i][j] = val;
       }
     }
    cout<<"The Matrix is"<<endl;
 for(int i=0; i<m; i++){
        for(int j=0; j<n; j++)
         {
           cout<< (*(*(float_values+i)+j))<<"\t";
         }
       cout<<endl;
        }
    for(int i=0;i<m;i++)//free the allocated memo
          delete [] float_values[i];
    delete [] float_values;
}
```

```
please enter the value for m row and n col:
3
4

 enter element of0 th row and0 th col:11
 enter element of0 th row and1 th col:2
 enter element of0 th row and2 th col:3
 enter element of0 th row and3 th col:45
 enter element of1 th row and0 th col:3
 enter element of1 th row and1 th col:2
 enter element of1 th row and2 th col:55
 enter element of1 th row and3 th col:4
 enter element of2 th row and0 th col:3
 enter element of2 th row and1 th col:4
 enter element of2 th row and2 th col:66
 enter element of2 th row and3 th col:7
The Matrix is
11      2       3       45
3       2       55      4
3       4       66      7
```