

اشاره گر ها

مبانی برنامه نویسی

(۱۱-۱۳۰-۱۳۹)

جلسه بیست و نهم



دانشگاه شهید بهشتی

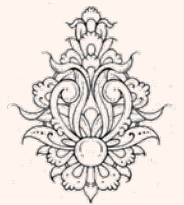
پاییز ۱۳۹۳

دانشکده مهندسی برق و کامپیوتر

احمد محمودی ازناوه

فهرست مطالب

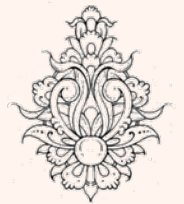
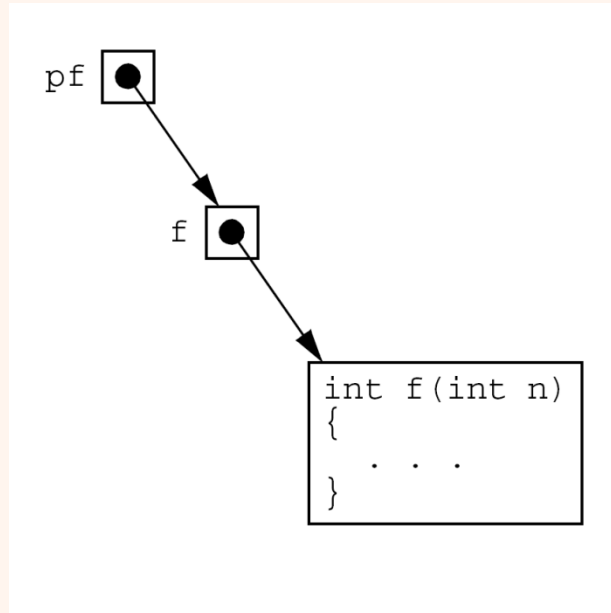
- اشاره‌گر به تابع
- تخصیص حافظه‌ی پویا



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...)

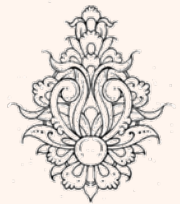
- ساختار تعریف مانند زیر است:

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

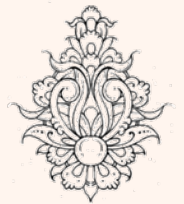
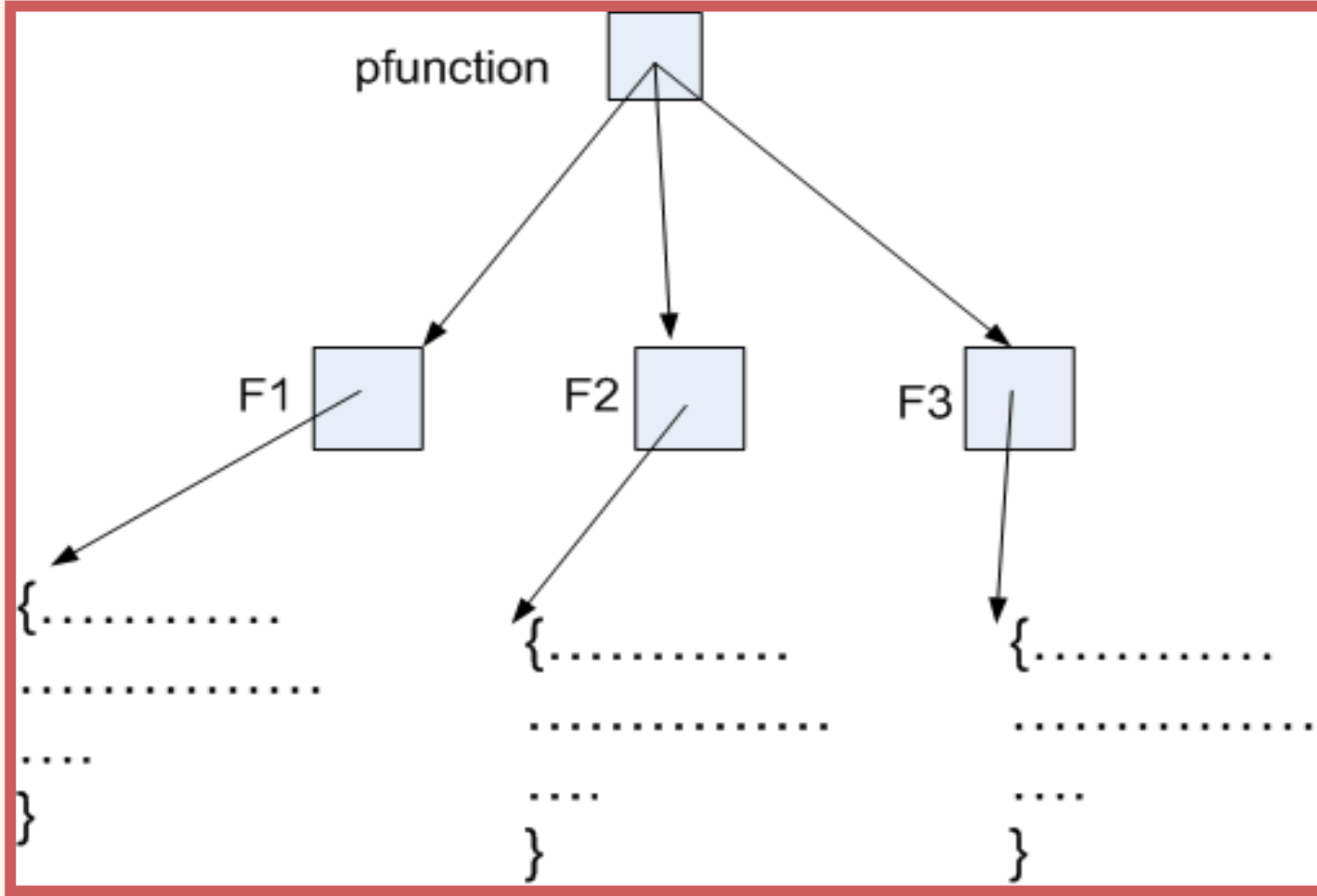
- همان‌گونه که مشاهده شد شیوه‌ی تعریف همانند تابع

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.



اشاره گر به تابع



```

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

int main()
{

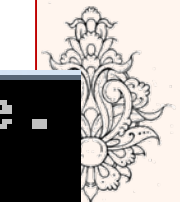
void (*f1) (char* ) = &error ; // ok
void (*f2 ) (char* ) = error ; // also ok; same meaning as &error
f1 ("f1 function without dereference."); // ok
(*f1) ("f1 function with dereference. "); // also ok
f2 ("f2 function without dereference.");
(*f2) ("f2 function with dereference"); // also ok
}

```

```

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

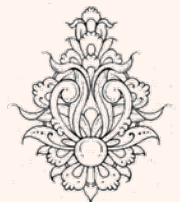
```



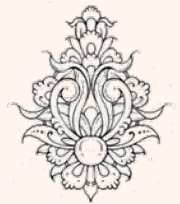
```
// pointer to functions
#include <iostream>
using namespace std;
int addition (int a, int b){
    return (a+b);
}
int subtraction (int a, int b){
    return (a-b);
}
int operation (int x, int y, int (*functocall) (int,int)){
    int g;
    g = (*functocall) (x,y);
    return (g);
}
int main (){
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



- 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 بزند.
- mixedfunction دو ورودی دارد یکی از جنس تابع و دیگری int که آرگومان دوم به عنوان ورودی $p3$ و $p4$ استفاده می‌شود.




```

#include <iostream>
using namespace std;

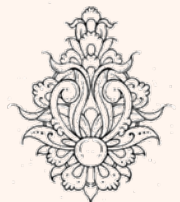
int mixedfunction(int (*)(int), int);
int p3(int);
int p4(int);
int main(){
    cout << (*mixedfunction)(p3,3) << endl; // 1 + 8 + 27
    cout << (*mixedfunction)(p4,3) << endl; //1 + 16 + 81
}
int mixedfunction(int (*pf)(int k), int n){
    // returns the sum f(0) + f(1) + f(2) + ... + f(n-1):
    int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

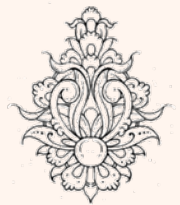
```
// function prototypes --
#include <iostream>
using namespace std;
void function0( int );
void function1( int );
void function2( int );
int main(){
    void (*f[ 3 ])(int) = {&function0,&function1,&function2 };
    int choice;

    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 success
} // end main
```

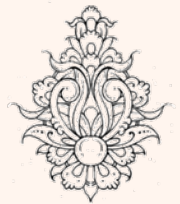
```
Enter a number between 0 and 2, 3 to end: 2
You entered 2 so function2 was called

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 سبب نشتی حافظه یا **memory leak** خواهد شد.

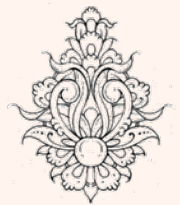


تخصیص حافظه پویا (ادامه...)

- 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 این صورت 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 array
    cout << "max=" <<max <<endl;
}
```

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

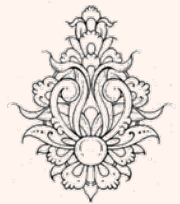
• در زبان C++ برای تخصیص حافظه پویا (در زمان اجرا) از تابع `new` استفاده می‌شود.

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 حافظه پویا در نظر گرفته می‌شود، در انتها آزاد نمی‌شود.

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

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



تخصیص حافظه پویا

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

<نوع > new = اشاره‌گر

اشاره‌گر delete

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

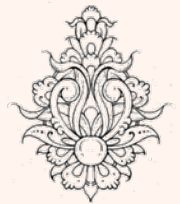
```
delete pn;
```

[طول آرایه] <نوع > new = اشاره‌گر

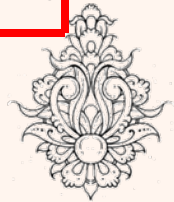
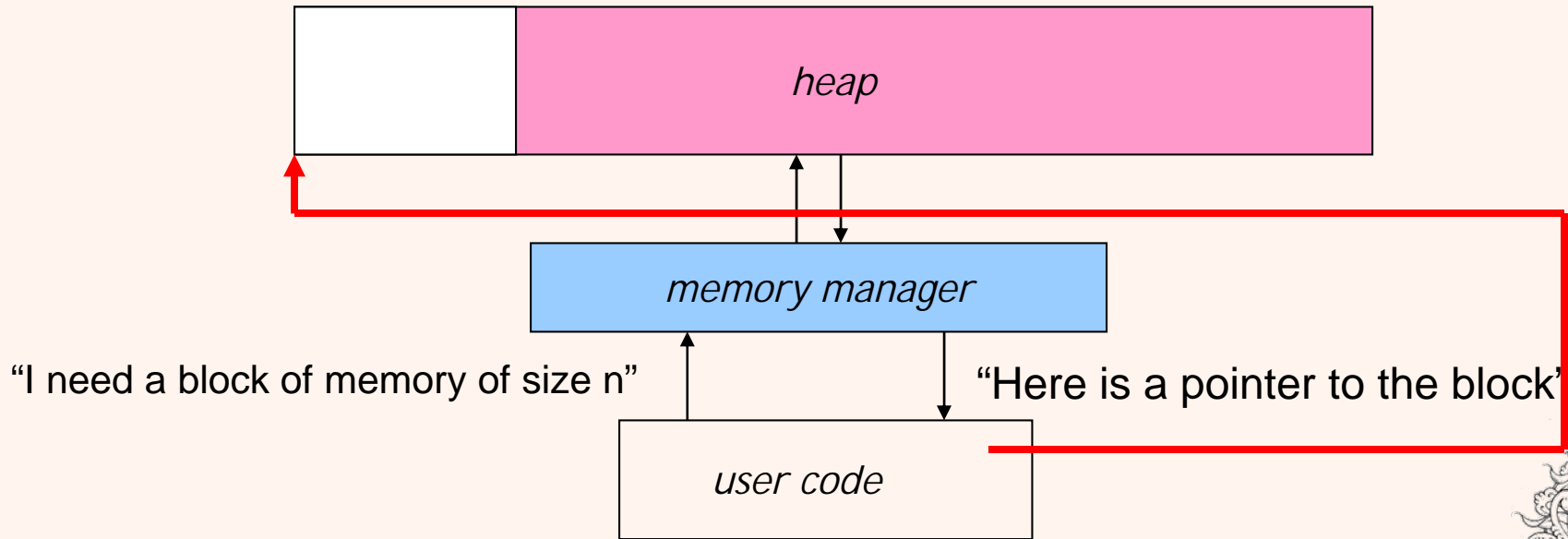
اشاره‌گر [] delete

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

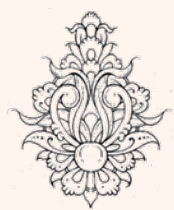
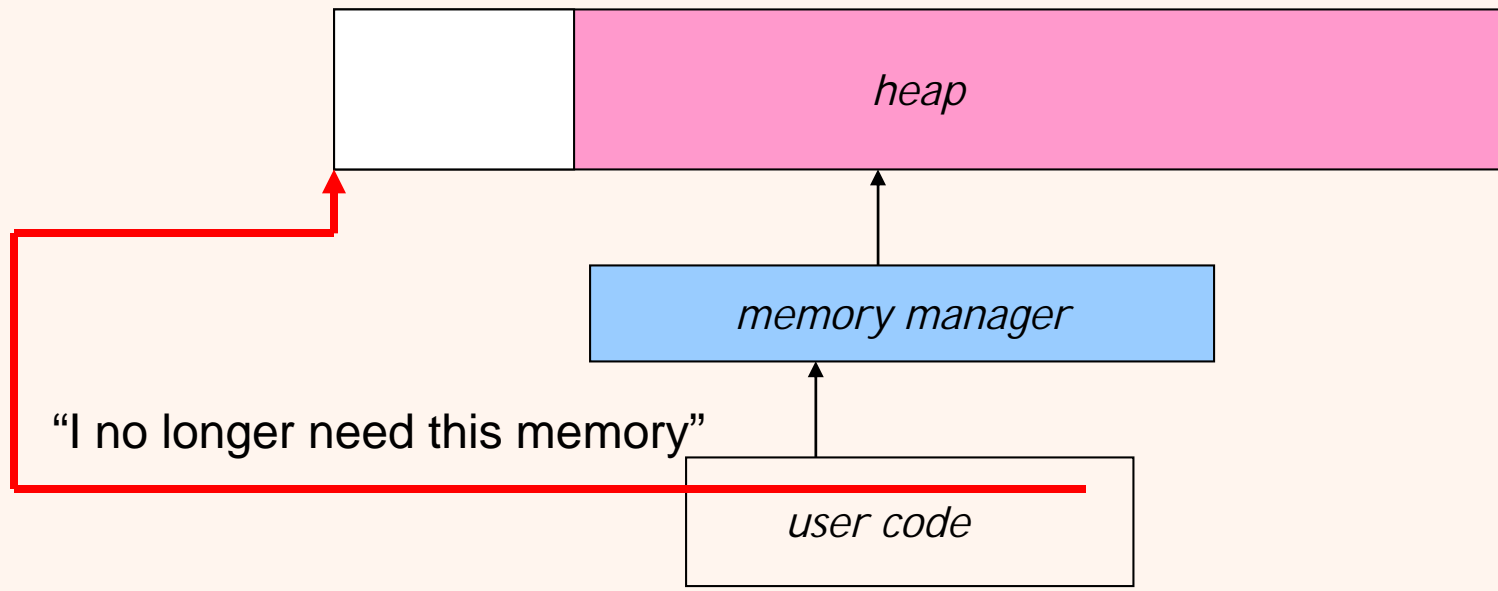
```
delete [ ] p;
```



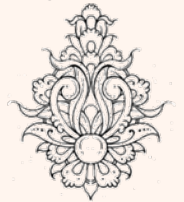
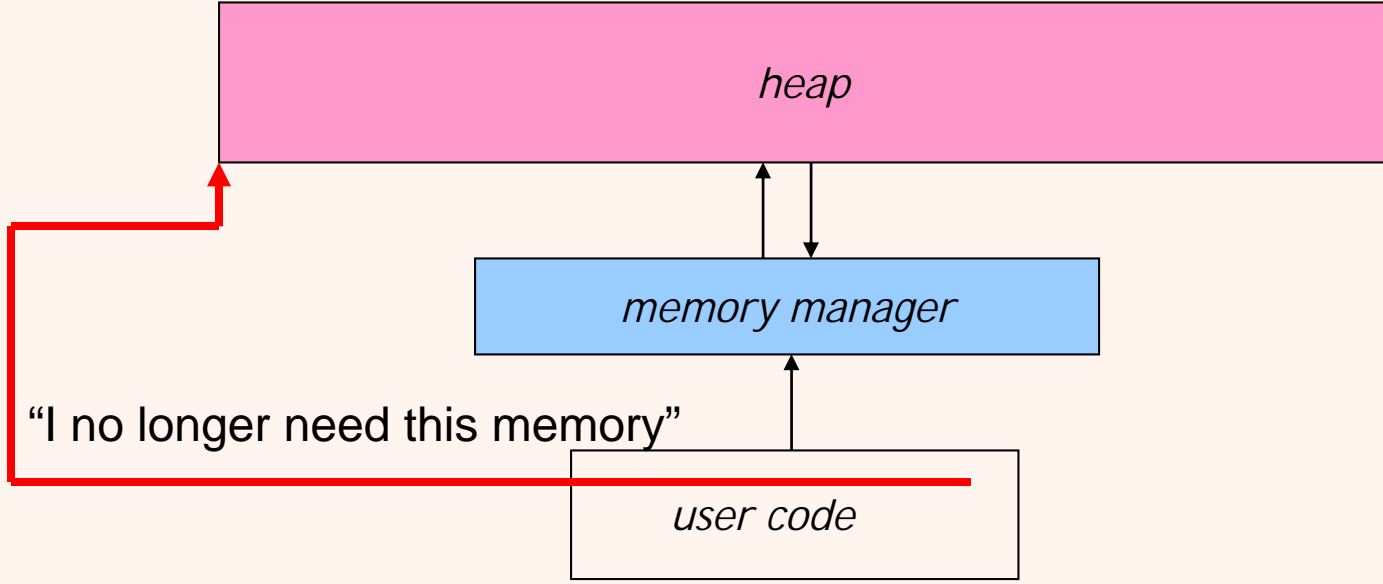
تخصیص حافظه



<http://vorlon.case.edu>

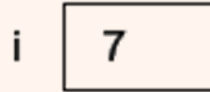
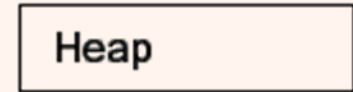
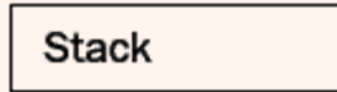


<http://vorlon.case.edu>

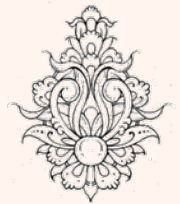
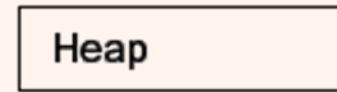
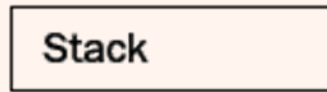


تخصیص حافظه پویا

```
int i=7;
```

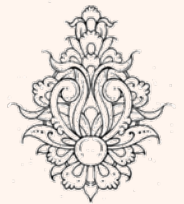
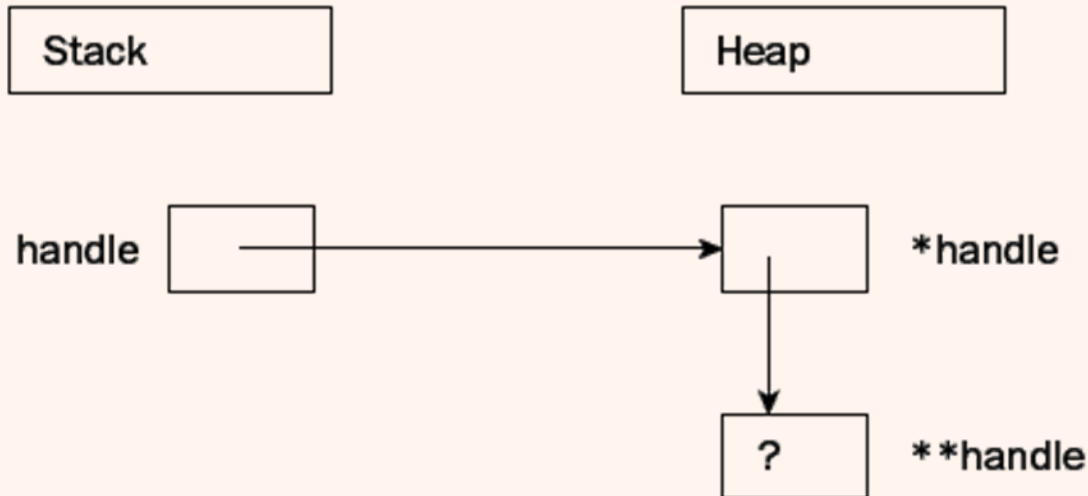


```
int *ptr;  
ptr = new int;
```



تخصیص حافظه پویا

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



تخصیص حافظه پویا



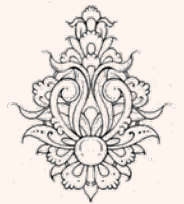
```
void leaky(){  
    new int;    //BUG! Orphans memory!  
    cout<<"I just a leaked an int!"<<endl;  
}
```

Stack

Heap

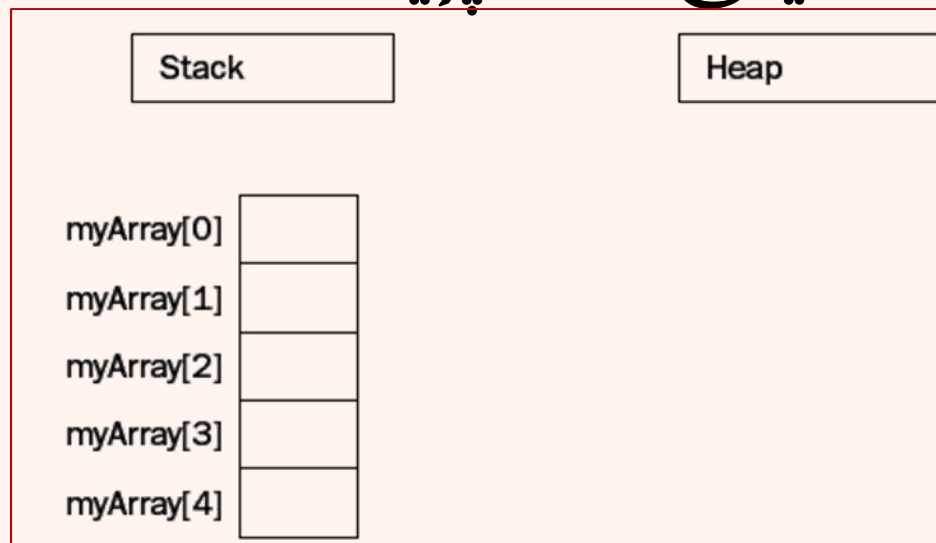
?

[leaked integer]

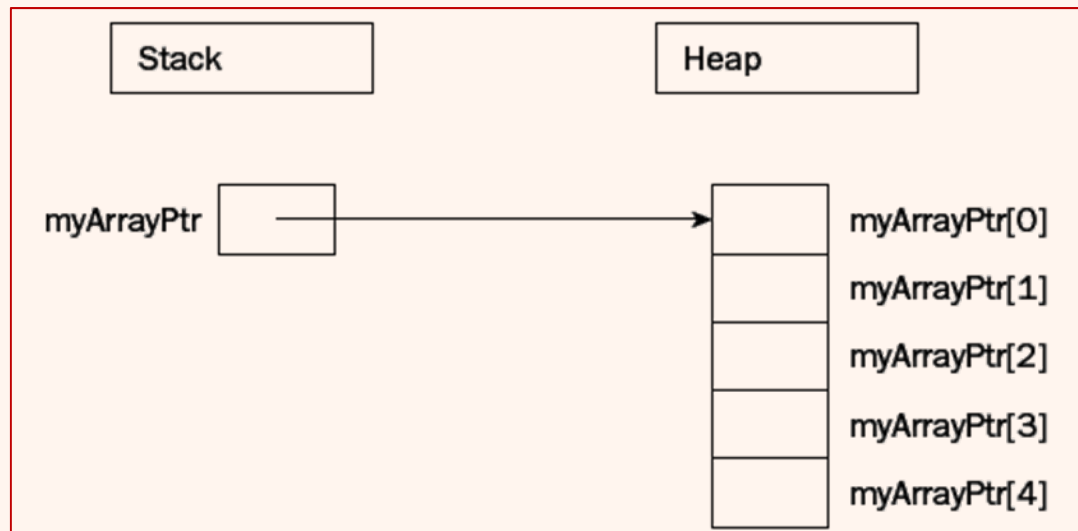


تخصیص حافظه پویا (ادامه...)

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

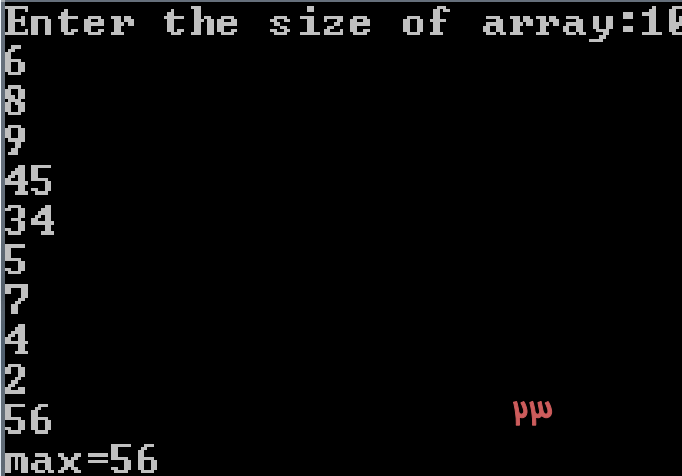


```
int* myArrayPtr = new int[5];
```



تصمیم 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
```





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

حافظه اختصاص داده شده و C به بلوک مورد نظر اشاره می‌کنند


حافظه اختصاص داده شده آزادسازی نگردیده است

```
void fn( ){  
    int* c = new int[10];  
  
    ...  
  
    delete[] c;  
    return;  
}
```





```
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() { // Just Allocate the pointers x and y
  int* x; //
  int* y; //

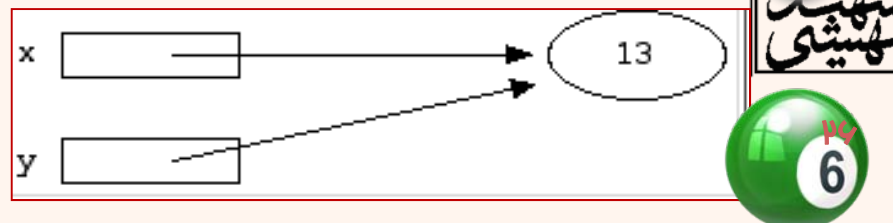
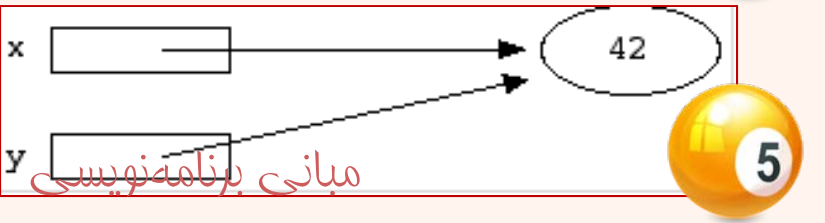
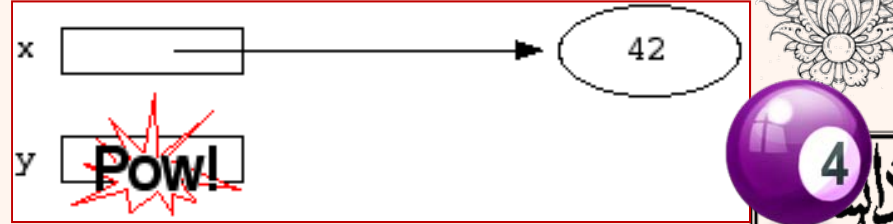
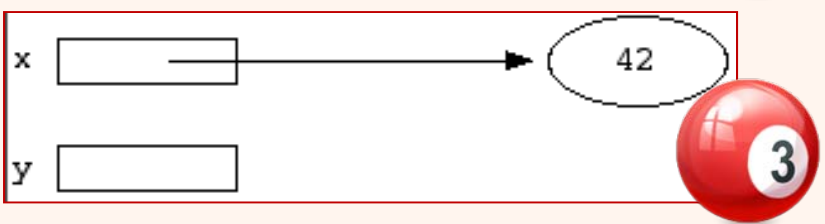
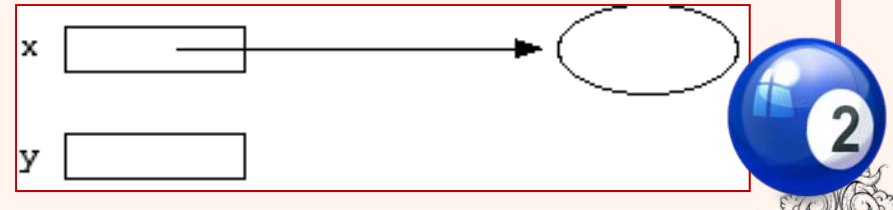
  x = new int; // Allocate an int pointee,
               // and set x to point to it

  *x = 42; // Dereference x to store 42 in its pointee

  *y = 13; // CRASH -- y does not have a pointee yet

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

  *y = 13; // Dereference y to store 13 in its (shared) pointee
}
```



مبانی برنامه نویسی



• هنگامی‌که اشاره‌گر به شی تعریف شده باشد و در جریان برنامه شی حذف شود یا حافظه اختصاص داده شده به اشاره‌گری آزاد شود.

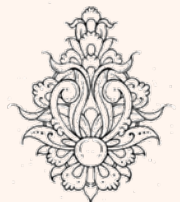
– در چنین حالتی اشاره‌گر موجود و reference آن وجود ندارد

– چنین اشاره‌گری را «معلق» می‌نامند.

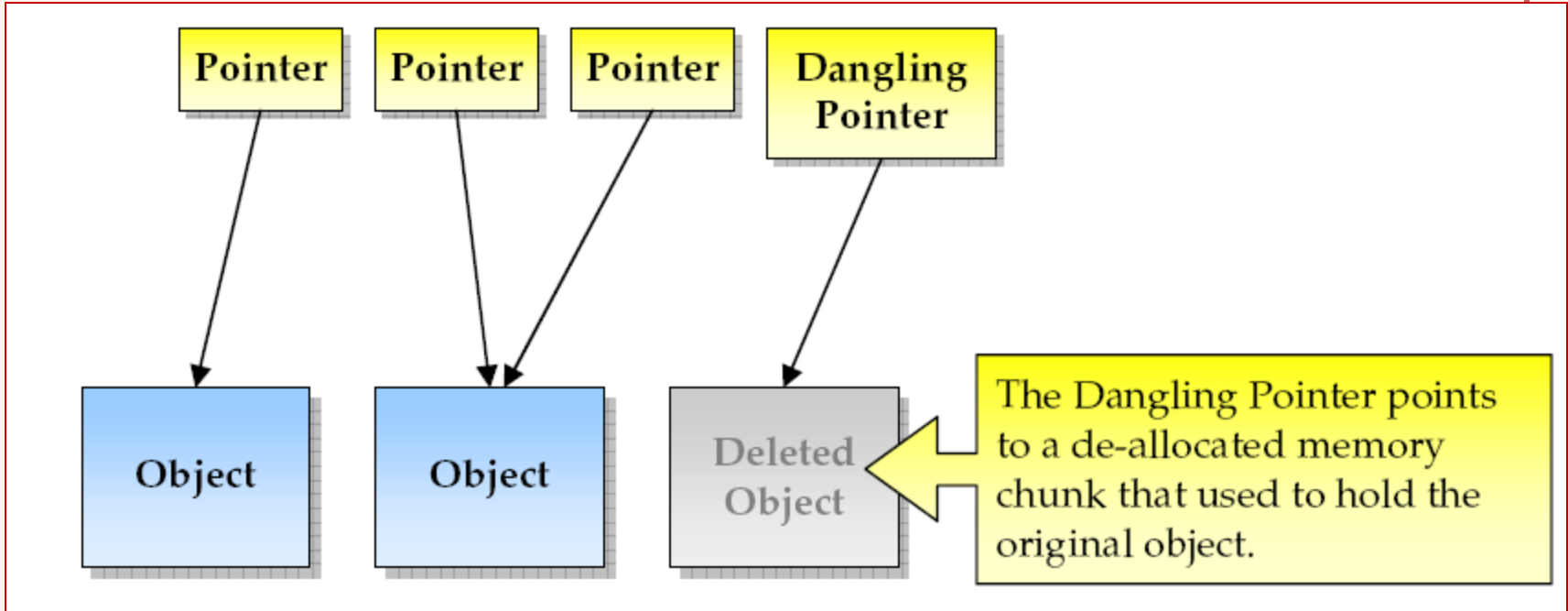
```
int main()
{
    char *dp = NULL;
    /* ... */
    {
        char c='a';
        dp = &c;

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

مبانی برنامه‌نویسی



اشاره گر معلق



تخصیص فضا برای آرایه‌ی دوبعدی

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

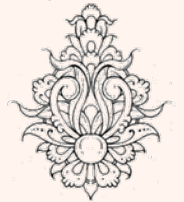
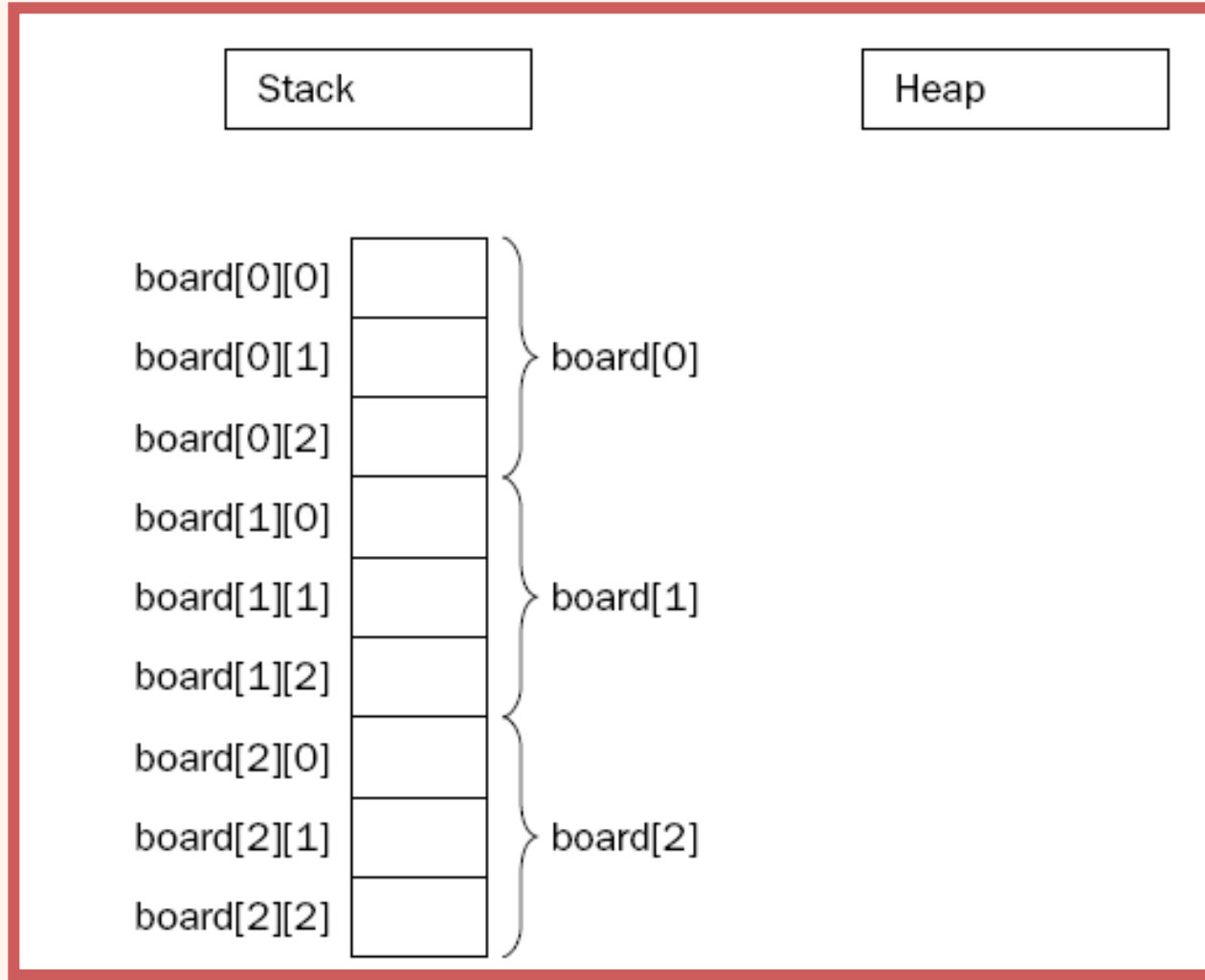
```
int **dArray = 0;
//firstly memory allocated for elements of rows.
dArray = new int* [ROWS] ;
//then memory allocated for elements of each column.
for( int i = 0 ; i < ROWS ; i++ )
    dArray[i] = new int[COLUMNS];
```

- برای آزادسازی نیز خواهیم داشت:

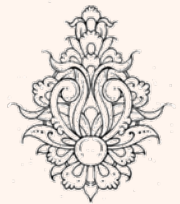
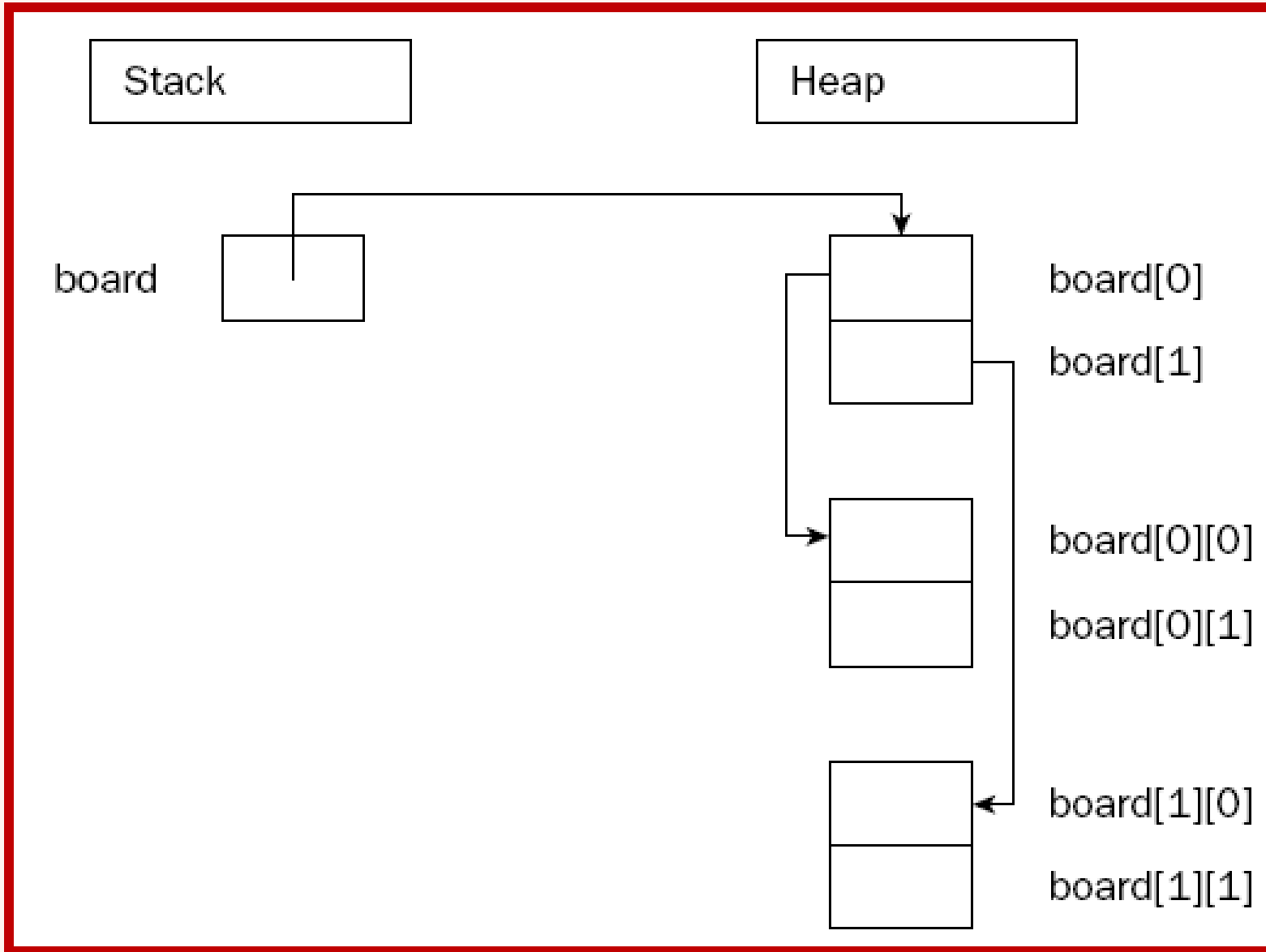
```
//free the allocated memory
for( int i = 0 ; i < ROWS ; i++ )
    delete [ ] dArray[i] ;
delete [ ] dArray ;
```



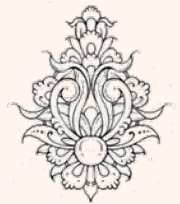
آرایه‌ی دوبعدی



تخصیص فضا برای آرایه‌ی دوبعدی



- 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;
}
```

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

مبارك بن محمد