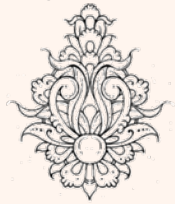
مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی بیست‌وسوم

تابع بازگشتی و آرایه‌ها

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

فهرست مطالب

- مروری بر جلسه‌ی پیش
- حل مسأله به صورت بازگشتی
- آرایه‌ها
- تعریف و مقداردهی اولیه

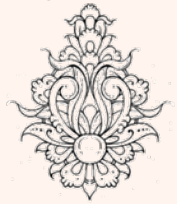


مثال

برنامه‌ای بنویسید که n را به عنوان عددی مثبت از ورودی دریافت کرده و فاکتوریل آن را محاسبه نماید.

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

int factorial(int);

void main(void) {
    int number;

    cout << "Please enter a positive integer: ";
    cin >> number;
    if (number < 0)
        cout << "That is not a positive integer.\n";
    else
        cout << number << " factorial is: " << factorial(number) << endl;
}

int factorial(int number) {
    int temp=1;

    if(number <= 1)
        return 1;
       while (number > 1)
      temp *= number--;
    return temp;
}
```

```
Please enter a positive integer: 9
9 factorial is: 362880
```



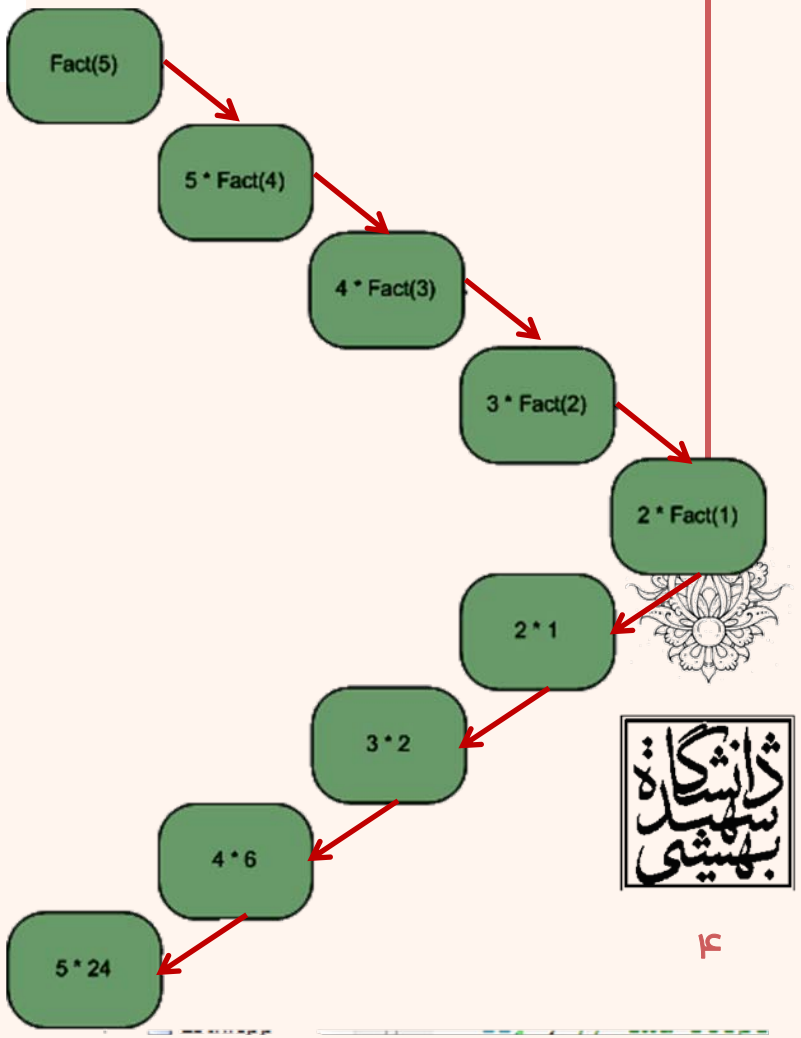
تابع فاکتوریل:

```
n! = n * (n-1) * (n-2) * (n-3) ... * 1
```

```
5! = 5 * 4 * 3 * 2 * 1 = 120
```

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n((n-1)!) & \text{if } n > 0 \end{cases} \qquad \forall n \in \mathbb{N}.$$

```
factorial(5) =
5 * factorial(4)
    4 * factorial(3)
        3 * factorial(2)
            2 * factorial(1)
                1
                = 120
```



فاکتوریل بازگشتی

```
int factorial(int);

void main(void) {
    int number;

    cout << "Please enter a positive integer: ";
    cin >> number;
    if (number < 0)
        cout << "That is not a positive integer.\n";
    else
        cout << number << " factorial is: " << factorial(number) << endl;
}

int factorial(int number) {
    int temp;

    if(number <= 1) return 1;

    temp = number * factorial(number - 1);
    return temp;
}
```

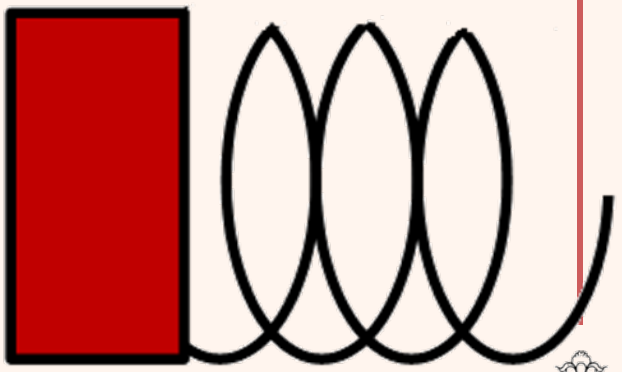
گام اول : شرط پایه

گام دوم : گام استقراء

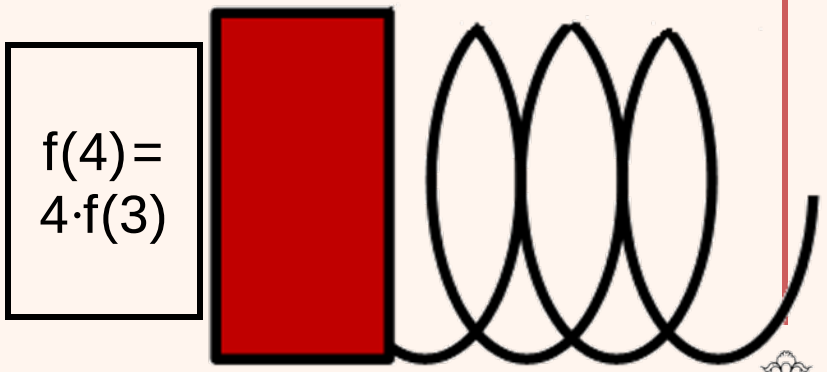
```
Please enter a positive integer: 9
9 factorial is: 362880
```

محاسبه‌ی فاکتوریل

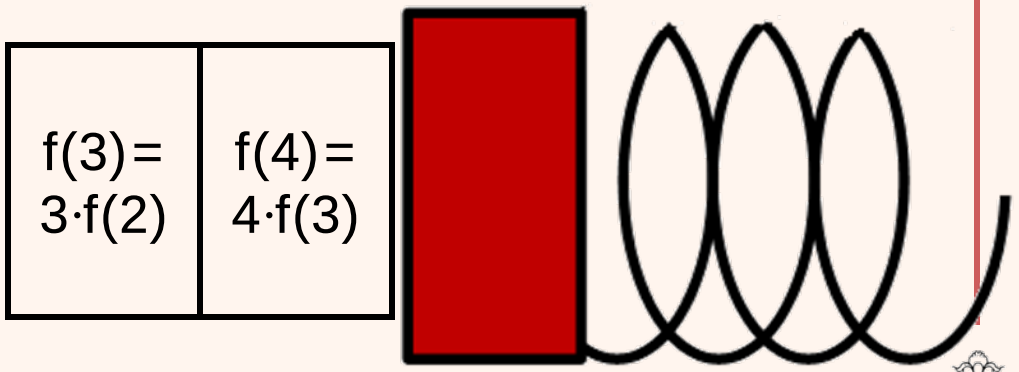
Compute 4!



محاسبه‌ی فاکتوریل

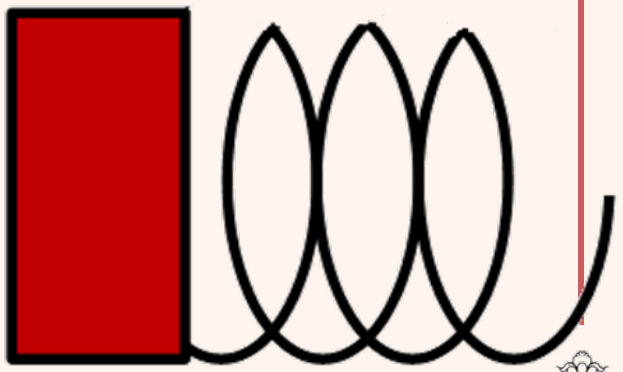


محاسبه‌ی فاکتوریل

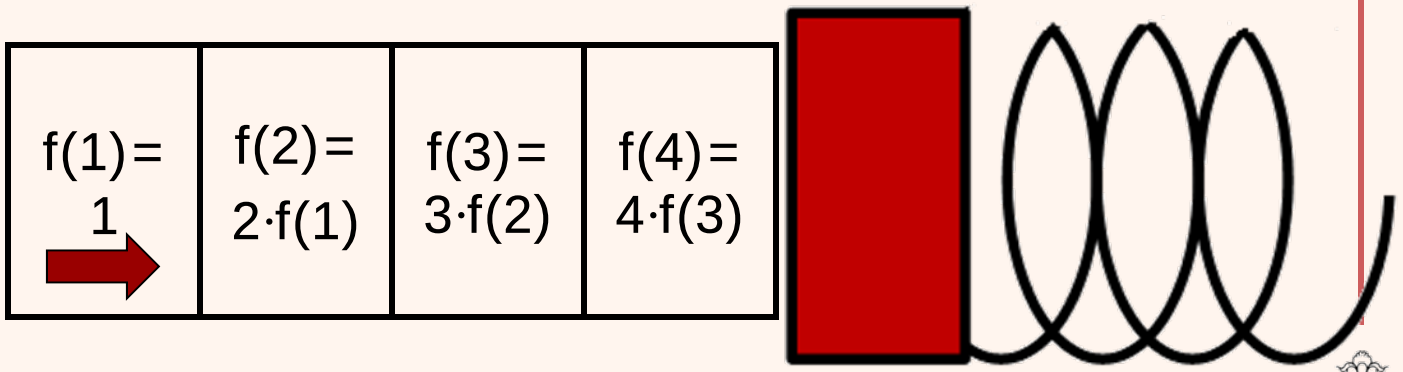


محاسبه‌ی فاکتوریل

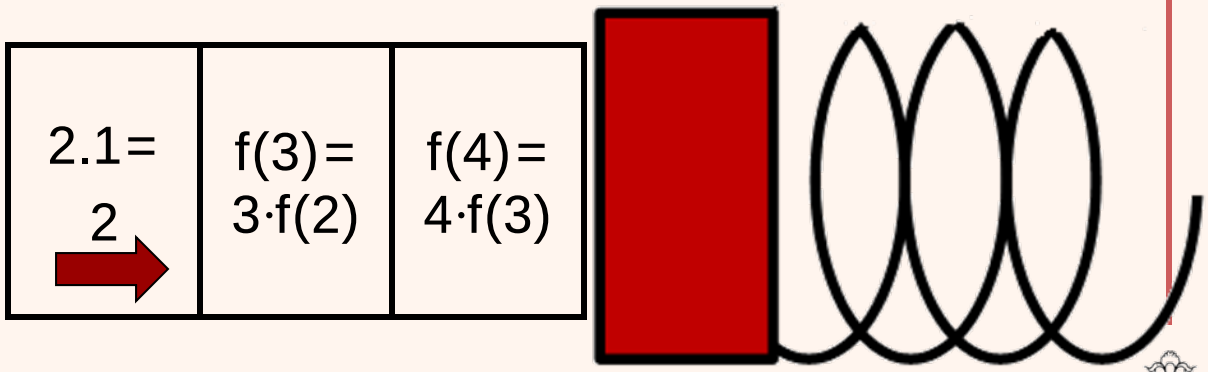
f(2)= 2·f(1)	f(3)= 3·f(2)	f(4)= 4·f(3)



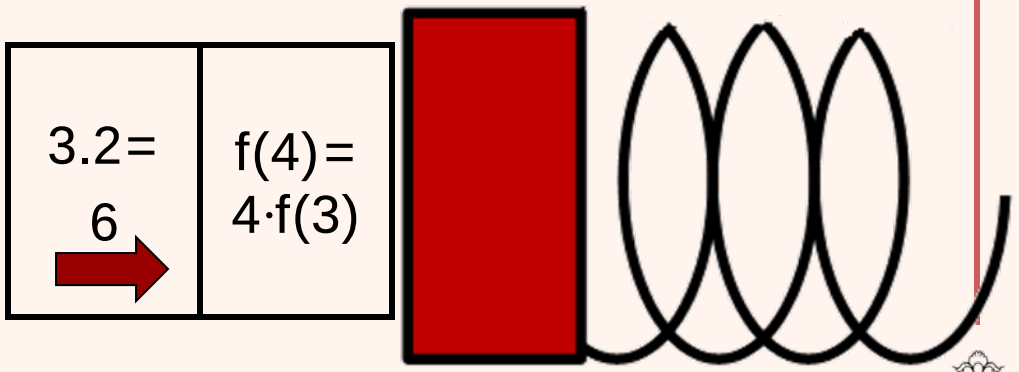
محاسبه‌ی فاکتوریل



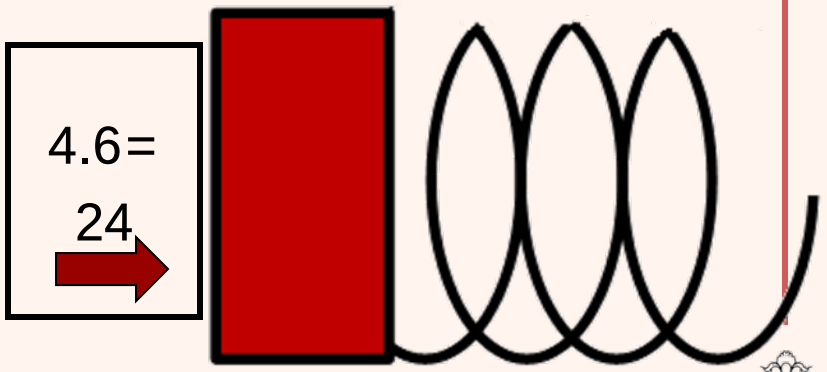
محاسبه‌ی فاکتوریل



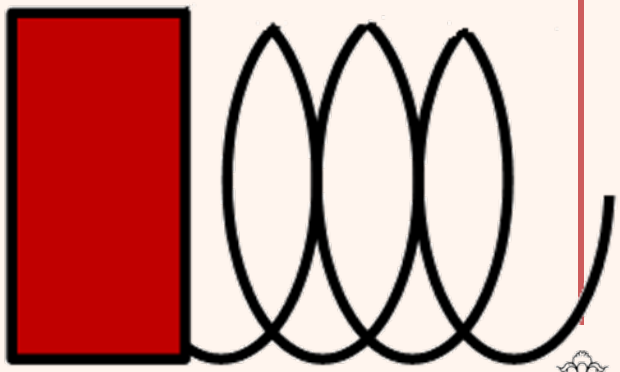
محاسبه‌ی فاکتوریل



محاسبه‌ی فاکتوریل

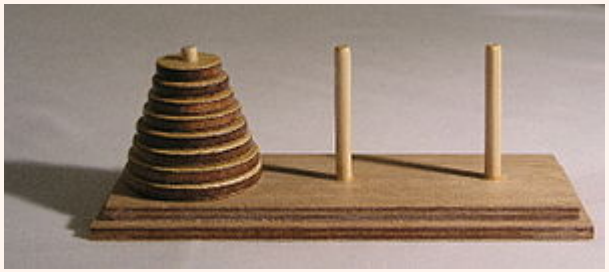


محاسبه‌ی فاکتوریل



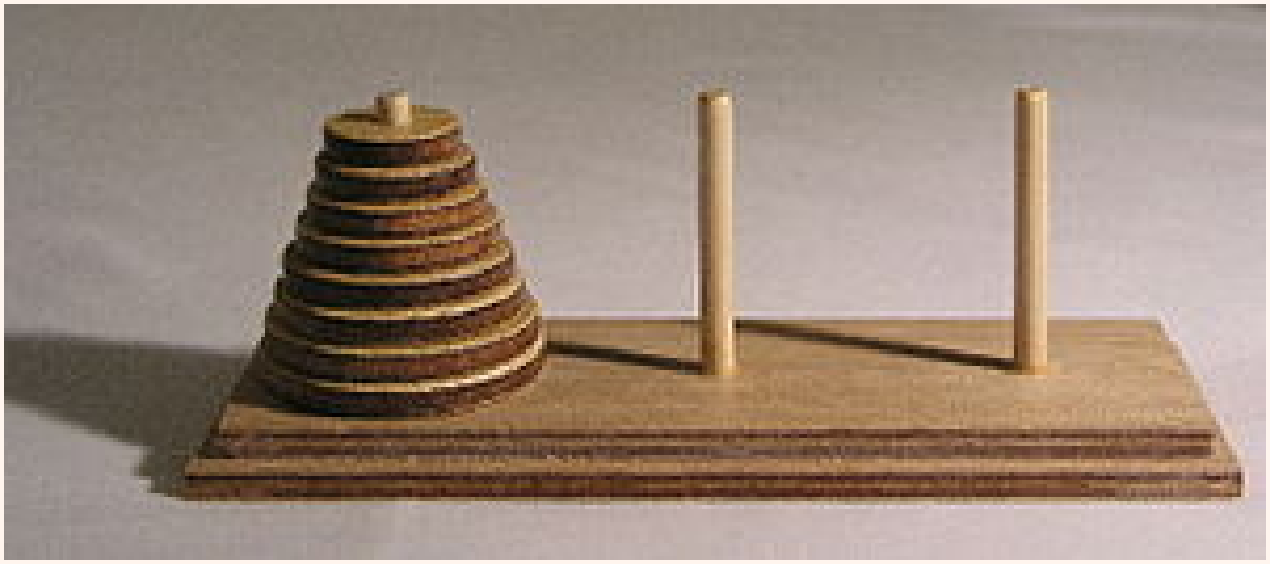
حل به روش بازگشتی

- در اینجا مسئله کلاسیک و معروف برج هانوی را مطرح می‌کنیم:
  - برج هانوی شامل سه حلقه است که بر روی حلقه اول n مهره قرار دارد؛ در حالی که بقیه خالی هستند. مهره‌ها به ترتیب روی هم چیده شده‌اند: بزرگ‌ترین مهره زیر همه و کوچک‌ترین مهره بالای همه مهره‌ها جای می‌گیرد. طبق قوانین بازی در هر نوبت فقط و فقط یک حرکت مجاز است و هیچ مهره‌ای حق قرار گرفتن روی مهره کوچک‌تر از خودش را ندارد. هدف بازی انتقال همه مهره‌ها به میله‌ی سوم است.



مسئله کلاسیک هانوی (ادامه...)

- تابعی بنویسید که پس از گرفتن تعداد دیسک‌ها تعداد جابجایی‌ها را چاپ کند.



- تابعی بنویسید که حرکت‌های دیسک‌ها را چاپ کند.

```
hanoi ( int nDisk, char start, char temp, char finish )
```

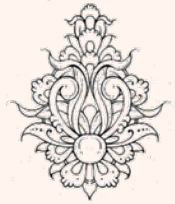
مسئله کلاسیک هانوی (ادامه...)

```
int itr;
```

```
void hanoi ( int nDisk, char start, char temp, char finish ){
  if ( nDisk == 1 )
    cout <<++itr<<")"<< start << " --> " << finish << endl;
  else{
    hanoi ( nDisk - 1, start, finish, temp );
    cout << ++itr<<")"<<start << " --> " << finish << endl;
    hanoi ( nDisk - 1, temp, start, finish );
  }

}
```

```
int main(){
  hanoi(3, 'A', 'B', 'C');
  return 0;
}
```

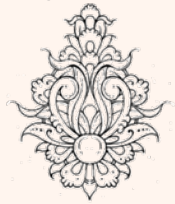
```
1)A --> C
2)A --> B
3)C --> B
4)A --> C
5)B --> A
6)B --> C
7)A --> C
```



مسئله کلاسیک هانوی (ادامه...)

```
void hanoi ( int nDisk, char start, char temp, char finish, bool flg){
    static int itr;
    if(flg)
        itr=0;
    if ( nDisk == 1 ){
        cout << setw(4)<<++itr<<") "<< start << " --> " << finish << endl;
    }
    else{
        hanoi ( nDisk - 1, start, finish, temp );
        cout <<setw(4)<<++itr<<") "<<start << " --> " << finish << endl;
        hanoi ( nDisk - 1, temp, start, finish );
    }
}
```

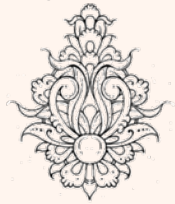
```
#define FIRST_CALL true
void hanoi ( int nDisk, char start, char temp, char finish,bool flg=false);
int main(){
    hanoi(3, 'A', 'B', 'C', FIRST_CALL);
    cout<<endl;
    hanoi(5, 'A', 'B', 'C',FIRST_CALL);
    return 0;
}
```



مسئله کلاسیک هانوی (ادامه...)

```
void hanoi ( int nDisk, char start, char temp, char finish,int num){
    static int itr;
    itr=num;
    if ( nDisk == 1 ){
        cout << setw(4)<<++itr<<") "<< start << " --> " << finish << endl;
    }
    else{
        hanoi ( nDisk - 1, start, finish, temp, itr );
        cout <<setw(4)<<++itr<<") "<<start << " --> " << finish << endl;
        hanoi ( nDisk - 1, temp, start, finish, itr );
    }
}
```

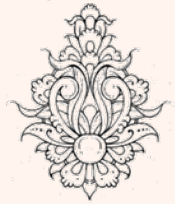
```
void hanoi ( int nDisk, char start, char temp, char finish,int num=0);
int main(){
    hanoi(3, 'A', 'B', 'C');
    cout<<endl;
    hanoi(5, 'A', 'B', 'C');
    return 0;
}
```



مشکلات تابع بازگشتی

- پیاده‌سازی با دستورالعمل‌های کم‌تر همراه با ظرافت و خوانایی بهتر برنامه

- Overhead بالا به دلیل فراخوانی تابع
- نگاه‌داشتن آدرس بازگشتی، ایجاد متغیرها
- مدیریت حافظه
  - در هر بار فراخوانی تابع مقداری حافظه اختصاص داده شده مصرف می‌شود.



آرایه‌ها

آرایه‌ها

- گروهی از محل‌های متوالی حافظه که دارای نام و نوع یکسانی هستند.
- برای مراجعه به محل یا عنصری خاص می‌باید نام آرایه و شماره‌ی عنصر مورد نظر مشخص شود.

شیوه‌ی تعریف

```
type name [elements];
```

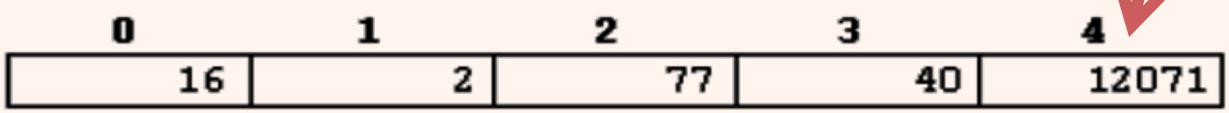
شیوه‌ی دسترسی

```
name[index];
```

```
int A[5] = {16, 2, 77, 40, 12071};
```

```
int A[] = {16, 2, 77, 40, 12071};
```

A[4]



مثال

```
#include <iostream>
using namespace std;
int main (){
    const int SIZE = 7;
    int numbers[SIZE] = {1, 2, 4, 8};
    cout << "Here are the contents of the array:\n";
    for (int index = 0; index < SIZE; index++)
        cout << numbers[index] << " ";
    cout << endl;
    return 0;
}
```

```
Here are the contents of the array:
1 2 4 8 0 0 0
```

مثال

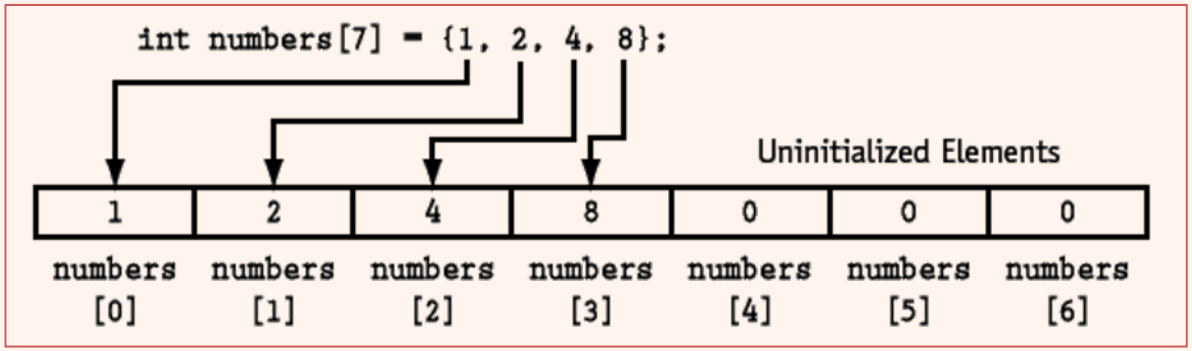
```
int main()
{ const int SIZE=5;                    // defines the size N for 5 elements
double a[SIZE];                        // declares the array's elements as type double
cout << "Enter " << SIZE << " numbers:\t";
for (int i=0; i<SIZE; i++)
cin >> a[i];
cout << "In reverse order: ";
for (int i=SIZE-1; i>=0; i--)
cout << "\t" << a[i];
cout<<endl;
}
```

```
Enter 5 numbers:        11 24 6 7 8
In reverse order:       8       7       6       24      11
```

روش‌های مقداردهی اولیه

- int a[6]={3,2,4,5,6,1};
- int number[7]={1,2,4,8};

باقی صفر می‌شوند.



- int a[6]={0};

تمامی خانه‌ها صفر می‌شوند.

- float a[ ]={22.2,44.4,66.6}

a

0	22.2
1	44.4
2	66.6

Implicit Array Sizing

در این روش کامپایلر با توجه به مقادیر داده شده بعد را در نظر می‌گیرد

- float a[3] = { 22.2, 44.4 ,66.6 , 88.8 };

```
// ERROR: too many values!
```

برای تعریف یک ثابت دو روش کلی وجود دارد:

- استفاده از Const

```
Const int  n=15;
```

- استفاده از پیش‌پردازنده‌ی define

```
#define      n    15
```

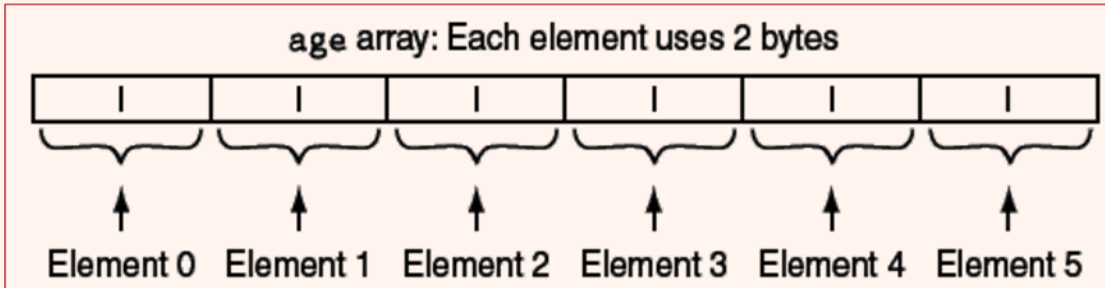
```
#include <iostream>
using namespace std;

int main(){
    const int arraySize = 10;
    int n[arraySize]={1,2,3,4,5,6,7,8,9,10}; // array s has 10 elements
    // initialize elements of array n to 0
    for ( int i = 0; i < arraySize; i++ )
        n[i]=0;
    cout << "Element\t\t\t" << "Value" << endl;
    // output each array element's value
    for ( int j = 0; j < arraySize; j++ )
        cout << j << "\t\t\t" << n[ j ] << endl;
    return 0;
}
```

```
Element                 Value
0                       0
1                       0
2                       0
3                       0
4                       0
5                       0
6                       0
7                       0
8                       0
9                       0
```

مثال

```
short age[6];
```

- اگر short دوبایت فضا نیاز داشته باشد:



ARRAY DECLARATION	NUMBER OF ELEMENTS	SIZE OF EACH ELEMENT	SIZE OF THE ARRAY
`char letter[26];`	26	1 byte	26 bytes
`short ring[100];`	100	2 bytes	200 bytes
`int mile[84];`	84	4 bytes	336 bytes
`float temp[12];`	12	4 bytes	48 bytes
`double distance[1000];`	1000	8 bytes	8,000 bytes

آرایه‌ی مقداردهی نشده

An Uninitialized Array

```
int main()
{ const int SIZE=4; // defines the size N for 4 elements
float a[SIZE]; // declares the array's elements as type float
for (int i=0; i<SIZE; i++)
      cout << "\ta[" << i << "] = " << a[i] << endl;
}
```

```
a[0] = -1.7606
a[1] = 3.98827e-34
a[2] = 1.1934e-38
a[3] = 1.19329e-38
```

اگر آرایه را مقداردهی نکنید ممکن است مقادیر زباله در حافظه وجود داشته باشد

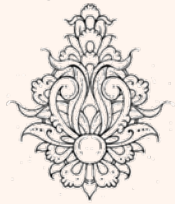
نکات

- نمی‌توان مقدار آن‌ها را به یکدیگر تخصیص داد:

```
float a[7] = { 7.2, 3.4, 4.6 };
float b[7] = { 53.3, 565.5, 7.7 };
b = a;    // ERROR
```

- نمی‌توانیم یک آرایه را به طور مستقیم برای مقداردهی اولیه به آرایهٔ دیگر استفاده کنیم:

```
float a[7] = { 222.2, 344.4, 866.6 };
float b[7] = a;   // ERROR
```



خارج از محدوده

- در زبان c اگر آرایه‌ای را با ۴ عنصر تعریف نماییم، در این صورت هیچ تضمینی وجود ندارد که در برنامه به اندیس‌هایی فراتر از اندازه تعریف شده دست نیابد. (ممکن است به مقادیری دست یابد که بی ارزش است)
- مثال:

```
#include <iostream>
using namespace std;int main()
{  const int SIZE=4; // defines the size N for 4 elements
   double a[SIZE]={ 33.3, 44.4, 55.5, 66.6 };// declares the array's elements as double

   for (int i=0; i<7; i++) //ERROR: index is out of bounds!
      cout << "\ta[" << i << "] = " << a[i] << endl;
}
```

```
a[0]  = 33.3
a[1]  = 44.4
a[2]  = 55.5
a[3]  = 66.6
a[4]  = -9.25596e+061
a[5]  = -9.25596e+061
a[6]  = 1.90727e-307
```