MA144: Problem Solving and Computer Programming

Lecture-21

Arrays

- Why do we need arrays?
- Declaration of Arrays
- How an array is stored in memory?
- Accessing Array Elements
- Initialization of Arrays

Why do we need arrays?

Consider the following scenarios

- Store the marks secured by students of ECE Section C in PSCP course.
  eg. 28, 89, 0, 5, 10, 67 ,.....
- Store the names of I BTech students in NITW
  eg. Lokeshwar, Saketh, Naga Sai, ...
- Store the distances from NITW to all the airports in India
  eg. 157 km, 455 km, 892 km, ...

In all the above situations, we need a structure to store all the similar values under the same name.

Array is a data structure that is a collection of homogeneous (of same data type) data items stored in consecutive memory locations and addressed by a common identifier.

Example - `int marks [55]`

where - `int` is a data type

- `marks` is an identifier name
- 55 is the size of the array (it must be integer constant).

Declaring Arrays

- Like variables, the arrays that are used in a program must be declared before they are used.
- General syntax:

```
type array_name[size];
```

  - type specifies the type of element that will be contained in the array (int, float, char, etc.)
  - size is an integer constant which indicates the maximum number of elements that can be stored inside the array.
- `int marks [55];`
  - marks is an array containing a maximum of 55 integers.

- **Examples**

```
int x[10];
char line[60];
float points[100];
char name[25];
```

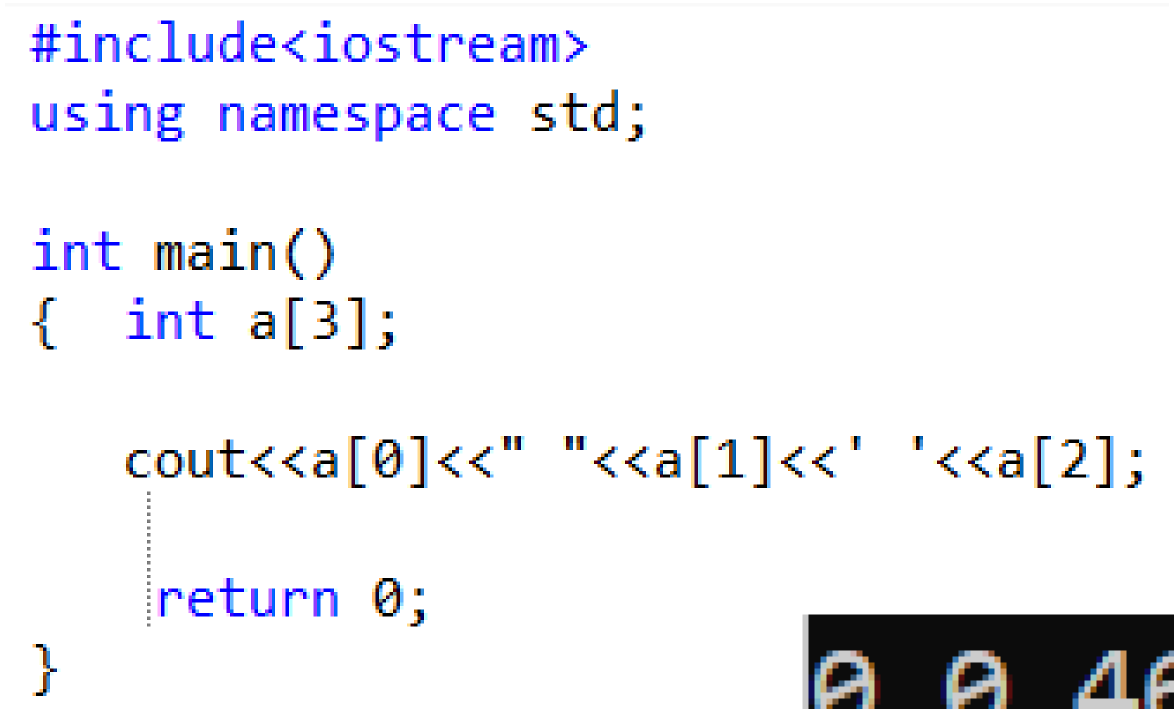
- If we are not sure of the exact size of the array, we can define an array of a large size like

```
  int marks[100];
```

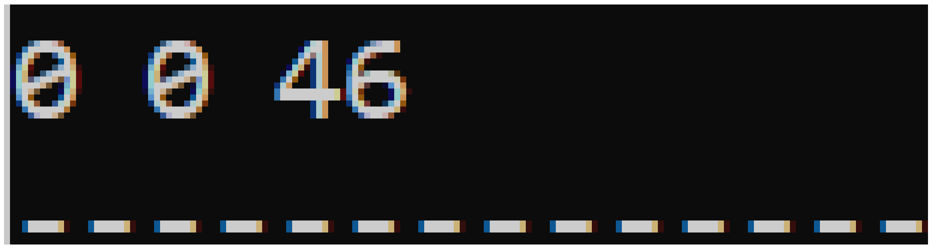
  even though in a particular run, we may only be using, say, 10 elements.

`int marks[10];`

- By the above declaration, `marks` has no values initialized for its elements.
- Such an array contains garbage values initially.



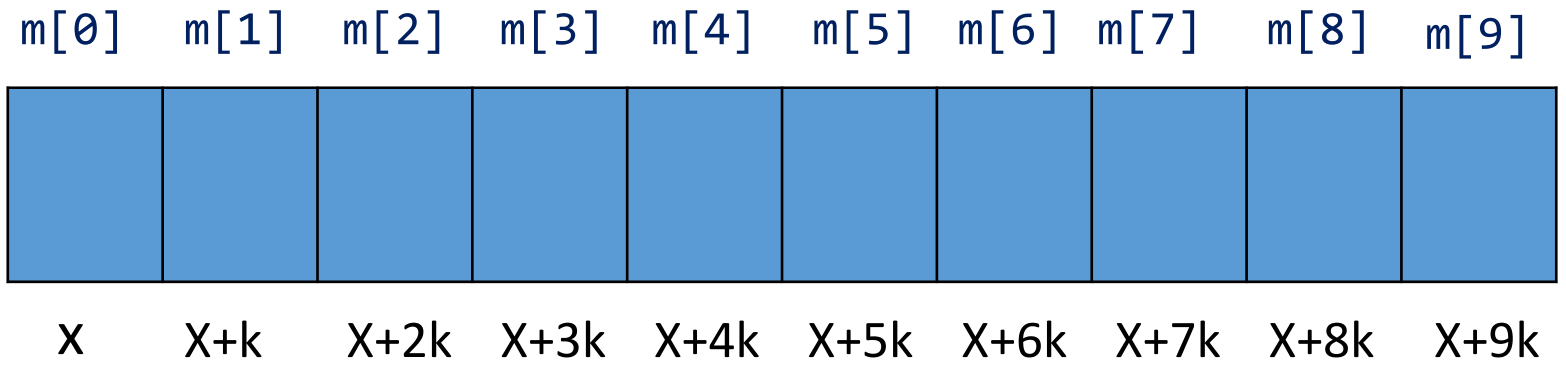
```
#include<iostream>
using namespace std;

int main()
{  int a[3];

   cout<<a[0]<<" "<<a[1]<<' '<<a[2];

    return 0;
}
```



```
0 0 46
```

How an array is stored in memory?

```
int a, b, c; // memory allocation is not consecutive

int m[10]; // always it is consecutive memory allocation
           // Hence, faster access of data
```

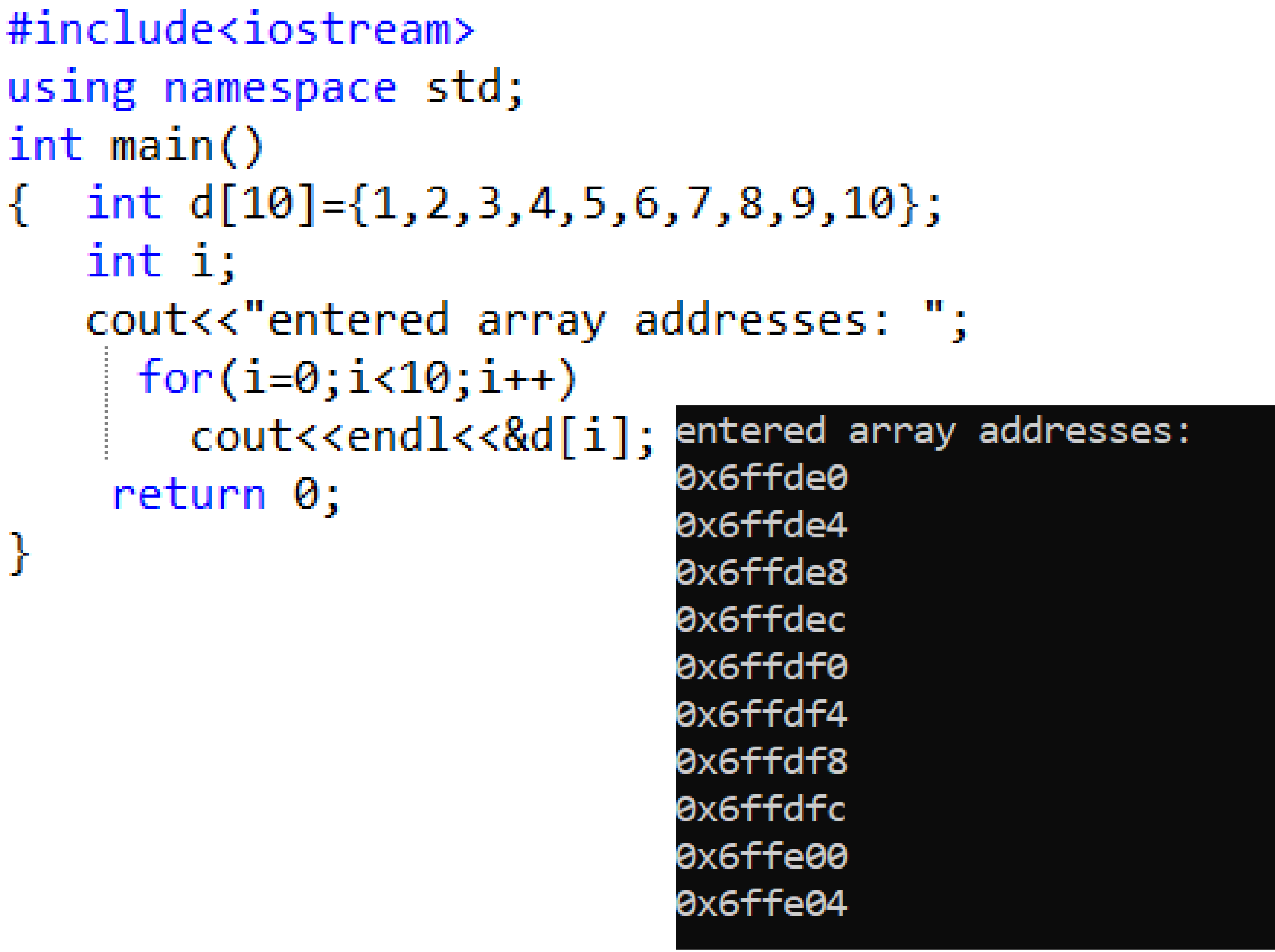


Starting from a given memory location, the successive array elements are allocated space in consecutive memory locations.

X - starting address of the array in memory

k - number of bytes allocated per array element

Element `m[i]` - allocated memory location at address x + i*k



```cpp
#include<iostream>
using namespace std;
int main()
{   int d[10]={1,2,3,4,5,6,7,8,9,10};
    int i;
    cout<<"entered array addresses: ";
      for(i=0;i<10;i++)
        cout<<endl<<&d[i];
     return 0;
}
```

```
entered array addresses:
0x6ffde0
0x6ffde4
0x6ffde8
0x6ffdec
0x6ffdf0
0x6ffdf4
0x6ffdf8
0x6ffdfc
0x6ffe00
0x6ffe04
```

Accessing Array Elements

- A particular element of the array can be accessed by specifying two things:
  - Name of the array.
  - Index (relative position) of the element in the array.
- The index of an array starts from zero.
- The index of the array always goes from 0 to n - 1, where n is the size of the array.
- An array is defined as `int m[10];`
  - The first element of the array `m` can be accessed as `m[0]`
  - second element as `m[1]`
  - third element as `m[2]`

    …….
  - tenth element as `m[9]`

```
int a[10], b[20], i,j,k,x,y;
```

```
a[0] = 1;
a[i] = 5;
a[j] = a[i] + 3;
a[j+1] = a[i] + a[0];
a[a[j]] = 12;
cin >> a[k];
cout<<a[2*j+3];
a[x+2] = 25;
b[3*x-y] = a[10-x] + 5;
```

- The index of the array to be in the range from 0 to n - 1, if n is the size of the array.

Initialization of Arrays

Initialization can be done in several ways.

```
int m[4]; // array declaration
```

(an integer array `m` of size 4)

```
int m[4]={5,10,15,20};
```

// Both declaration and initialization in single step

```
m[0]=5;
m[1]=10;
m[2]=15;
m[3]=20;
```

// by accessing individual locations

```
cin>> m[0];
cin>> m[1];
cin>> m[2];
cin>> m[3];
```

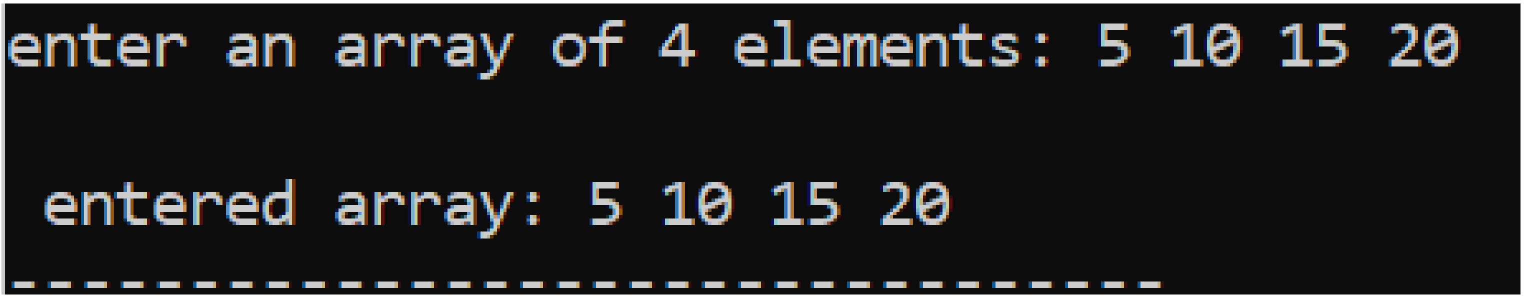
//by user

```
for(i=0;i<size;i++)
    cin>> m[i];
```

Printing array elements

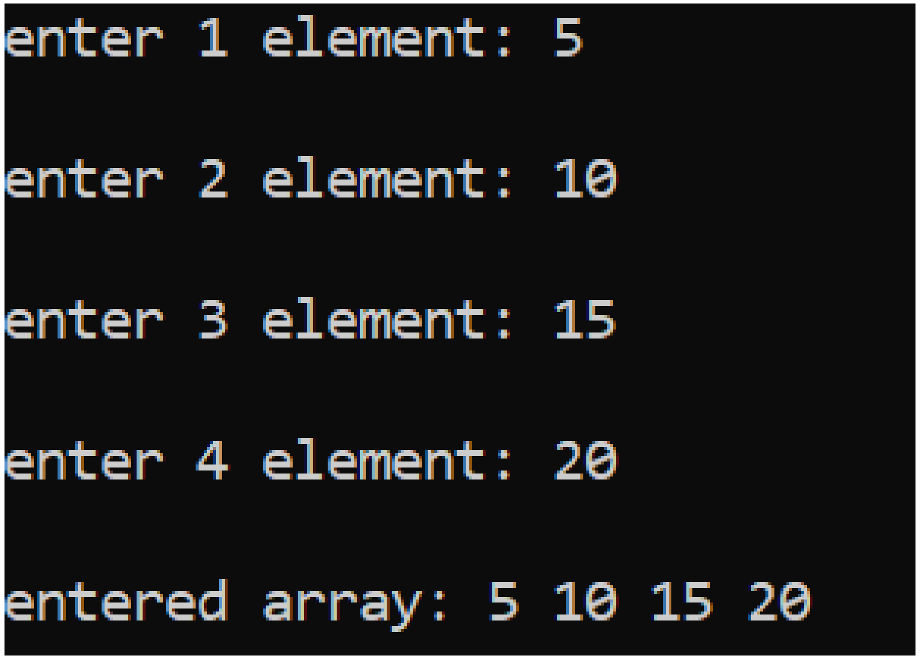
```
for(i=0;i<size;i++)
    cout<< m[i];
```

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

int main()
{  int a[4], i;
  cout<<"enter an array of 4 elements: ";
   for(i=0;i<4;i++)
      cin>>a[i];
   cout<<"\n entered array: ";
   for(i=0;i<4;i++)
     cout<<a[i]<<" ";
    return 0;
}
```



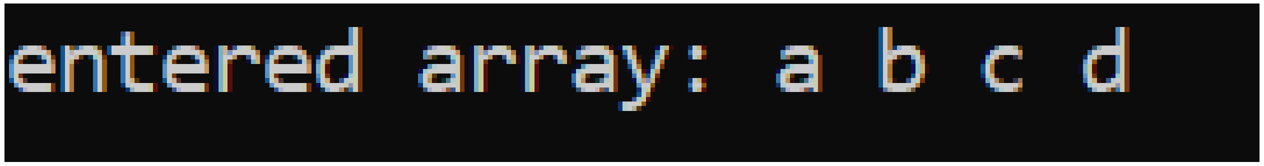
```
enter an array of 4 elements: 5 10 15 20

 entered array: 5 10 15 20
--------------------------------
```

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

int main()
{  int a[4], i;
   for(i=0;i<4;i++)
    { cout<<"enter "<<i+1<<" element: ";
      cin>>a[i];
      cout<<endl;
    }
    cout<<"entered array: ";
     for(i=0;i<4;i++)
     cout<<a[i]<<" ";
    return 0;
}
```



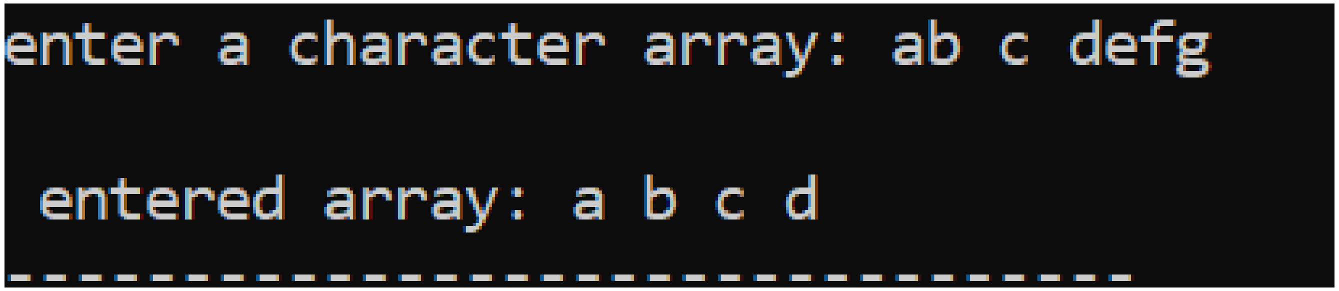
```
enter 1 element: 5

enter 2 element: 10

enter 3 element: 15

enter 4 element: 20

entered array: 5 10 15 20
```

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

int main()
{   char c[4]={'a', 'b', 'c','d'};
    int i;
     cout<<"entered array: ";
      for(i=0;i<4;i++)
        cout<<c[i]<<" ";
     return 0;
}
```

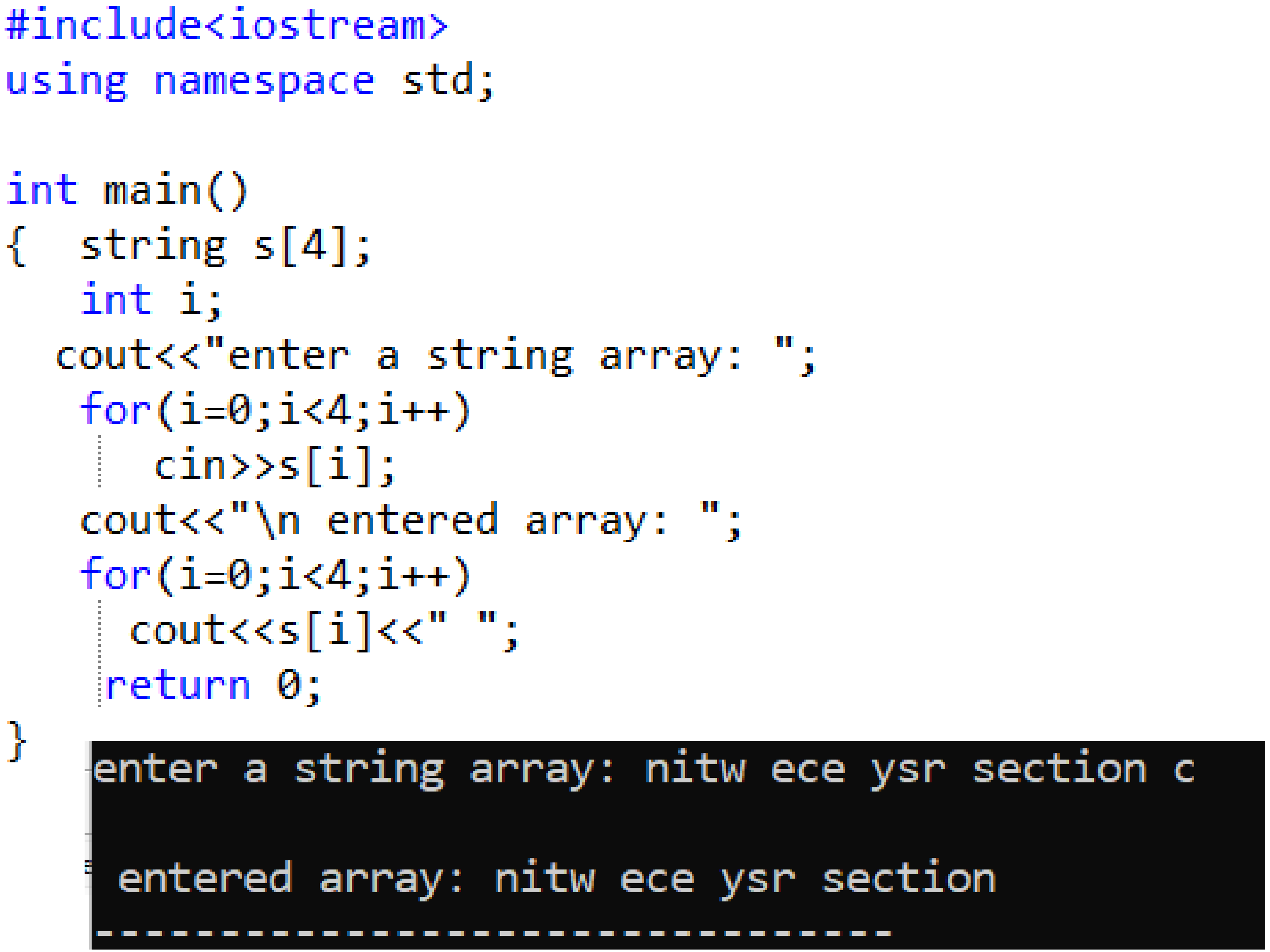


```
entered array: a b c d
```

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

int main()
{  char a[4];
   int i;
  cout<<"enter a character array: ";
   for(i=0;i<4;i++)
      cin>>a[i];
   cout<<"\n entered array: ";
   for(i=0;i<4;i++)
     cout<<a[i]<<" ";
    return 0;
}
```



```
enter a character array: ab c defg

 entered array: a b c d
----------------------------------
```



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

int main()
{   string s[4];
    int i;
   cout<<"enter a string array: ";
    for(i=0;i<4;i++)
        cin>>s[i];
    cout<<"\n entered array: ";
    for(i=0;i<4;i++)
      cout<<s[i]<<" ";
     return 0;
}
```

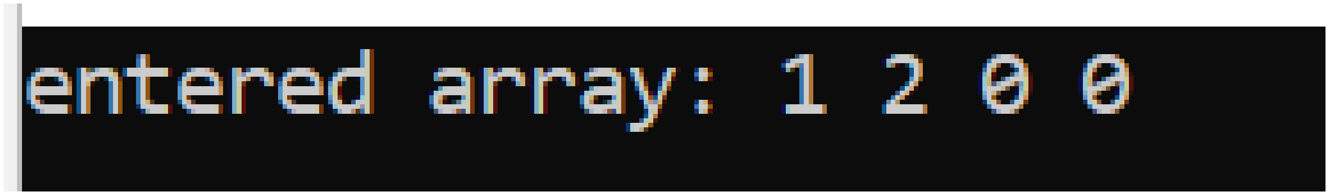
```
enter a string array: nitw ece ysr section c

 entered array: nitw ece ysr section
--------------------------------
```

Special Cases

Special Cases

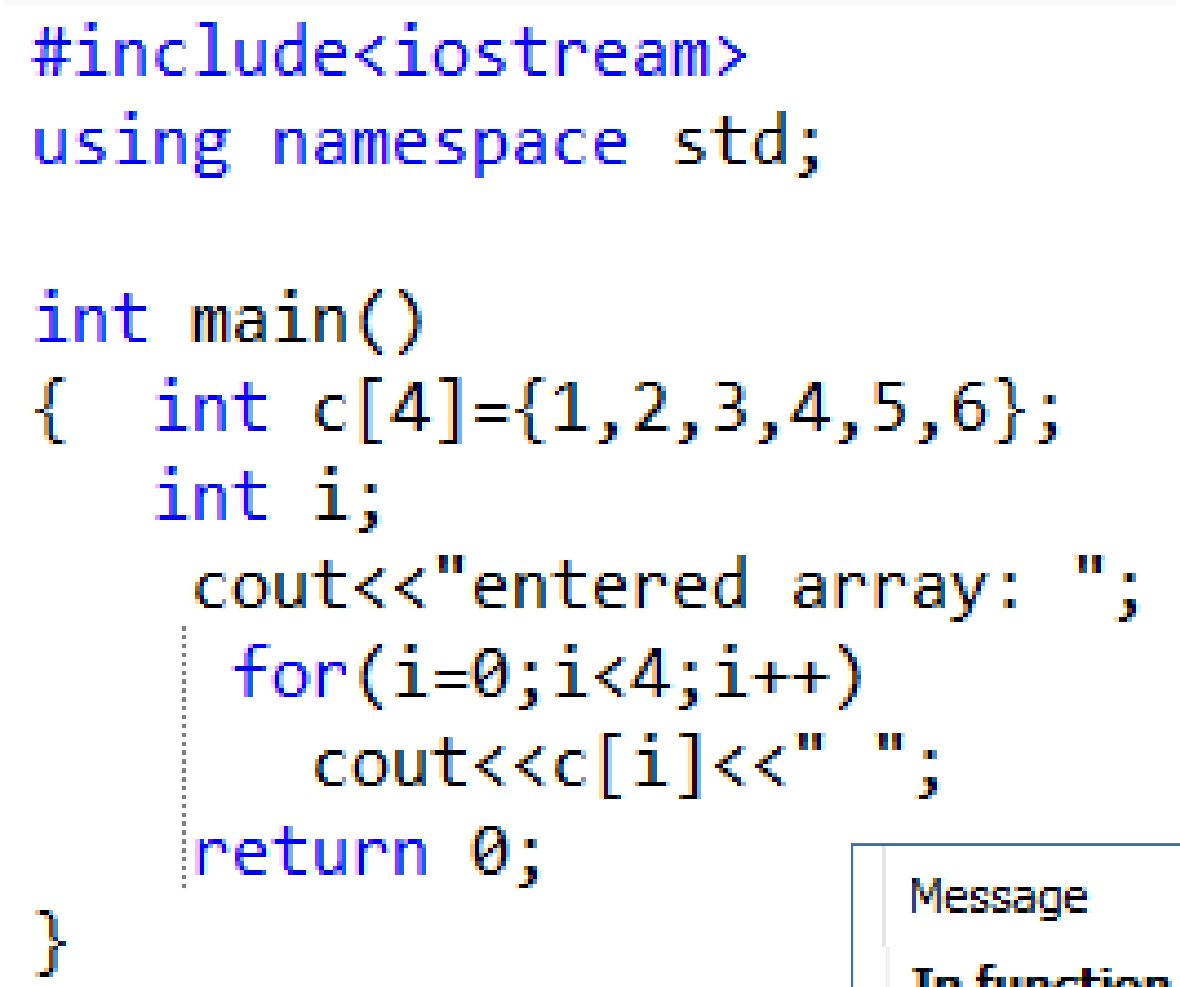
If size > the number of values in the list, then the remaining elements are automatically set to zero.

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

int main()
{   int c[4]={1,2};
    int i;
     cout<<"entered array: ";
      for(i=0;i<4;i++)
         cout<<c[i]<<" ";
     return 0;
}
```

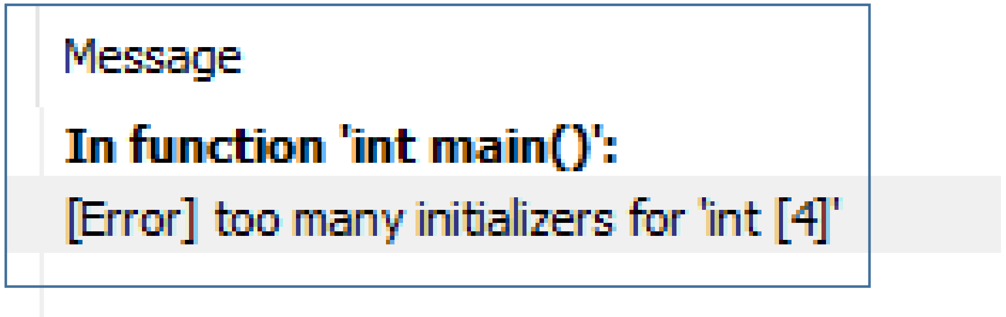


```
entered array: 1 2 0 0
```

If size < the number of values in the list, then compiler reports an error.

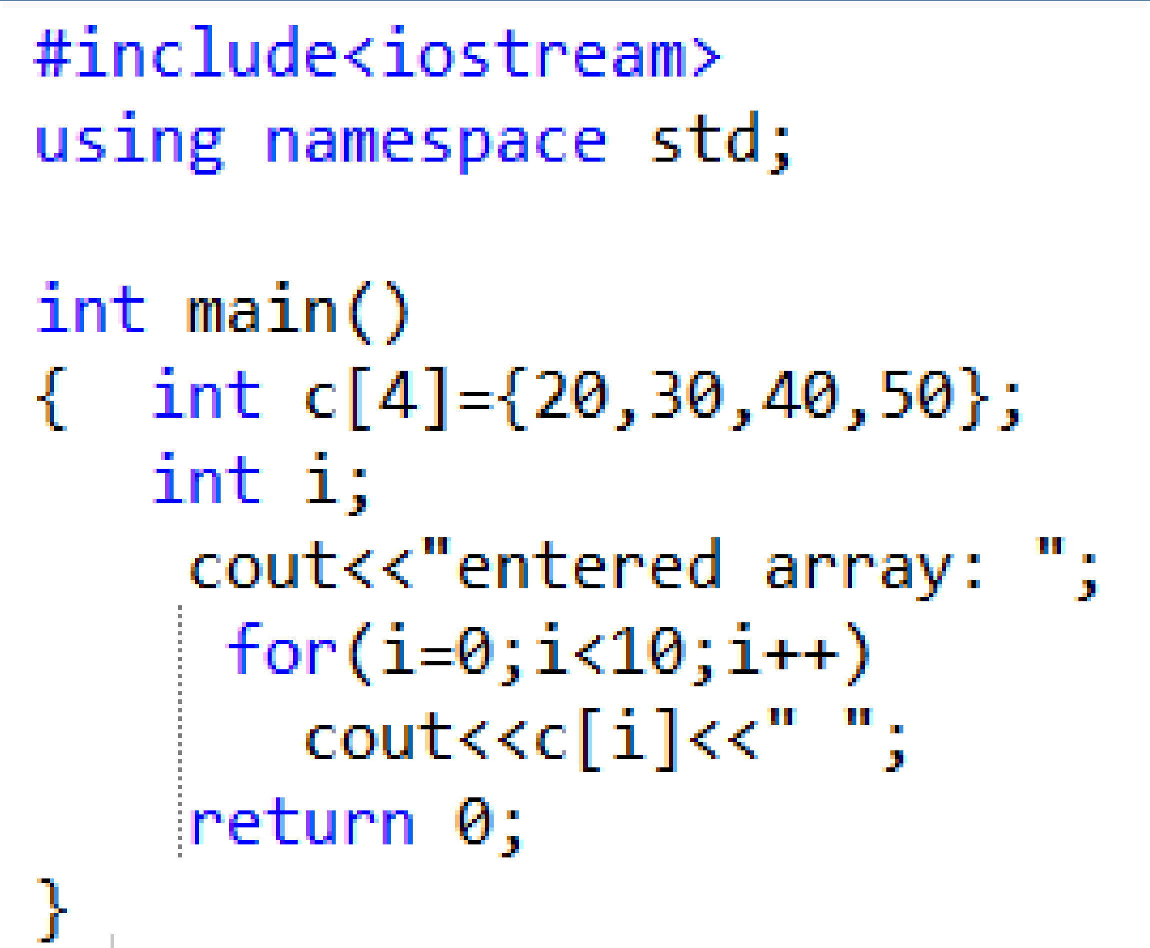


```
#include<iostream>
using namespace std;

int main()
{   int c[4]={1,2,3,4,5,6};
    int i;
     cout<<"entered array: ";
      for(i=0;i<4;i++)
        cout<<c[i]<<" ";
     return 0;
}
```

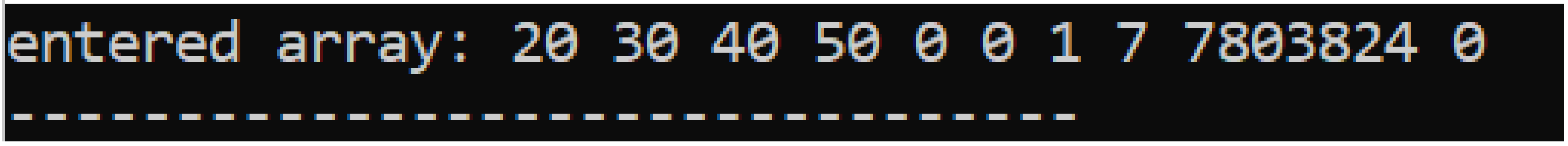


```
Message
In function 'int main()':
[Error] too many initializers for 'int [4]'
```

If size < the number of writing values,
then the remaining values are garbage values.

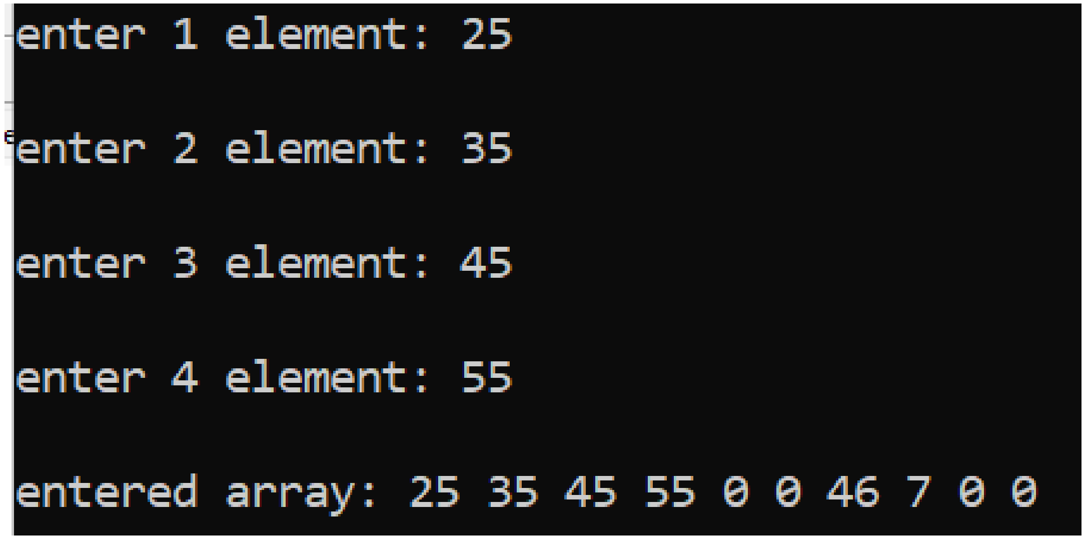


```
#include<iostream>
using namespace std;

int main()
{   int c[4]={20,30,40,50};
    int i;
     cout<<"entered array: ";
      for(i=0;i<10;i++)
         cout<<c[i]<<" ";
     return 0;
}
```



```
entered array: 20 30 40 50 0 0 1 7 7803824 0
--------------------------------
```

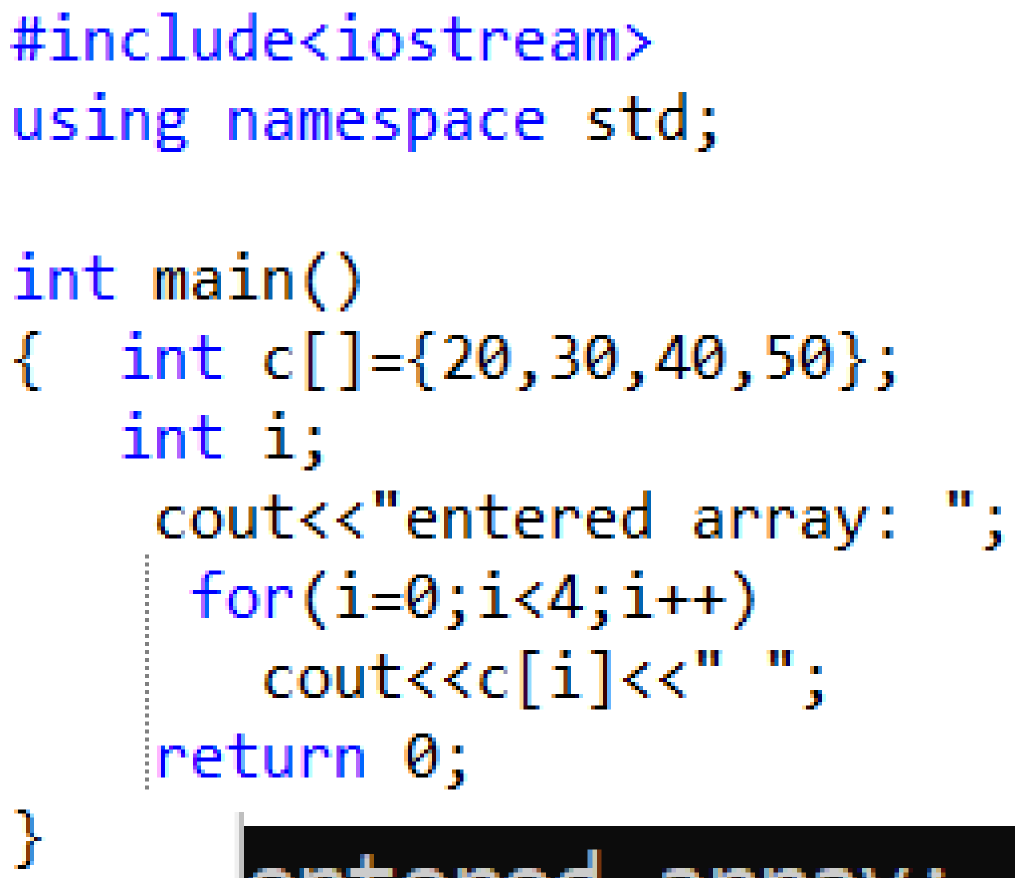
```cpp
#include<iostream>
using namespace std;
int main()
{   int a[4], i;
    for(i=0;i<4;i++)
     { cout<<"enter "<<i+1<<" element: ";
       cin>>a[i];
       cout<<endl;
     }
     cout<<"entered array: ";
      for(i=0;i<10;i++)
      cout<<a[i]<<" ";
     return 0;
}
```



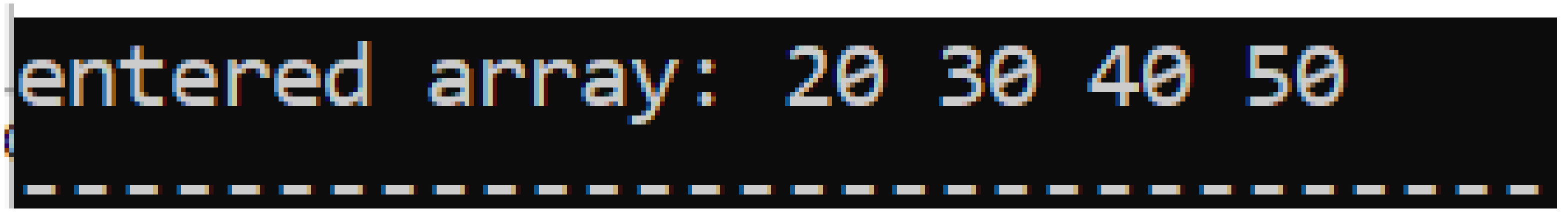
```
enter 1 element: 25

enter 2 element: 35

enter 3 element: 45

enter 4 element: 55

entered array: 25 35 45 55 0 0 46 7 0 0
```

The size may be omitted.

In such cases, the compiler automatically allocates enough space for all initialized elements.

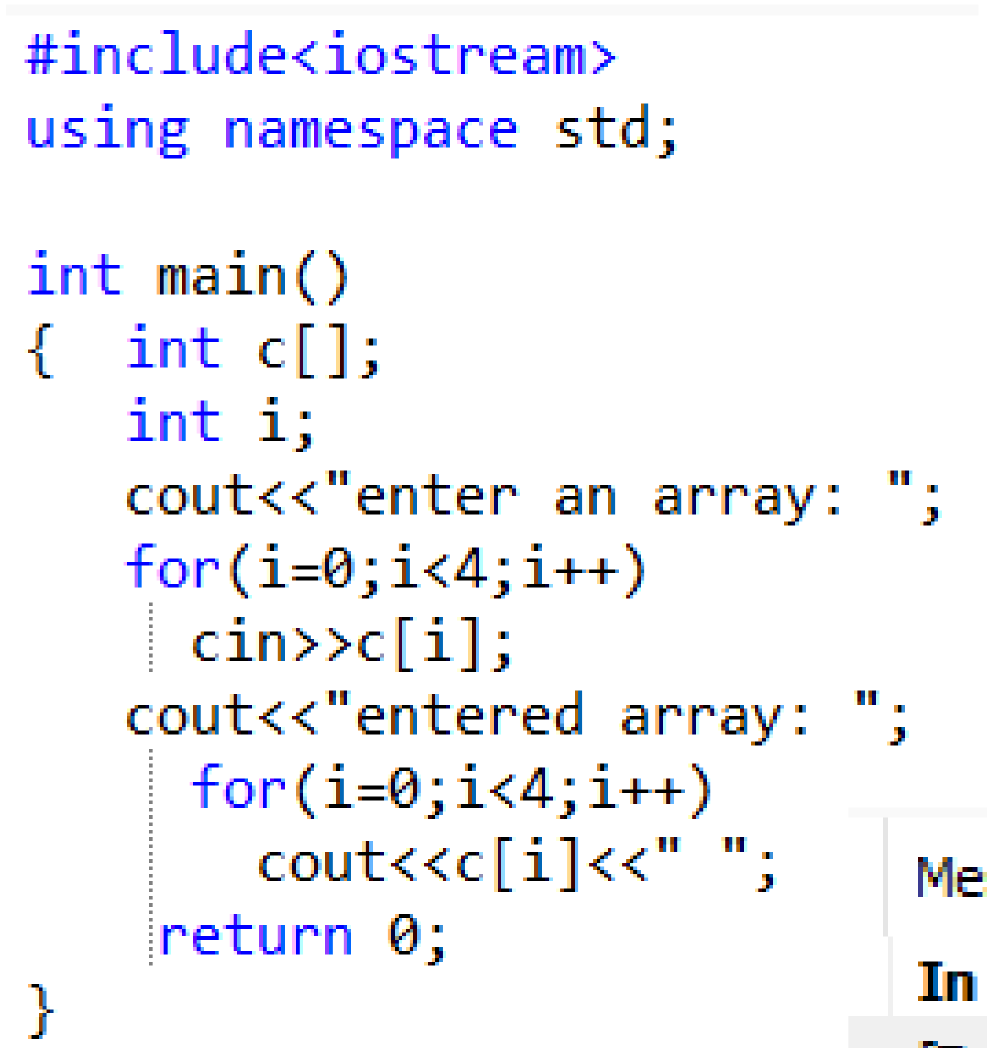


```
#include<iostream>
using namespace std;

int main()
{   int c[]={20,30,40,50};
    int i;
     cout<<"entered array: ";
      for(i=0;i<4;i++)
        cout<<c[i]<<" ";
     return 0;
}
```

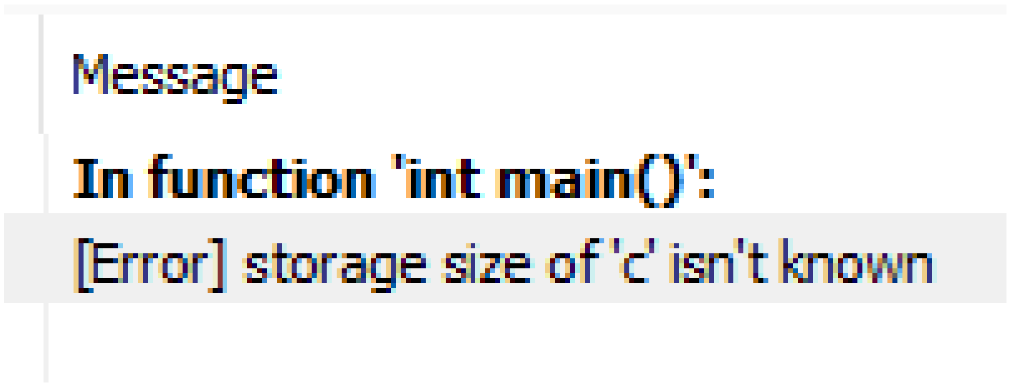


```
entered array: 20 30 40 50
```

The omission of size without initialization reporting an error.

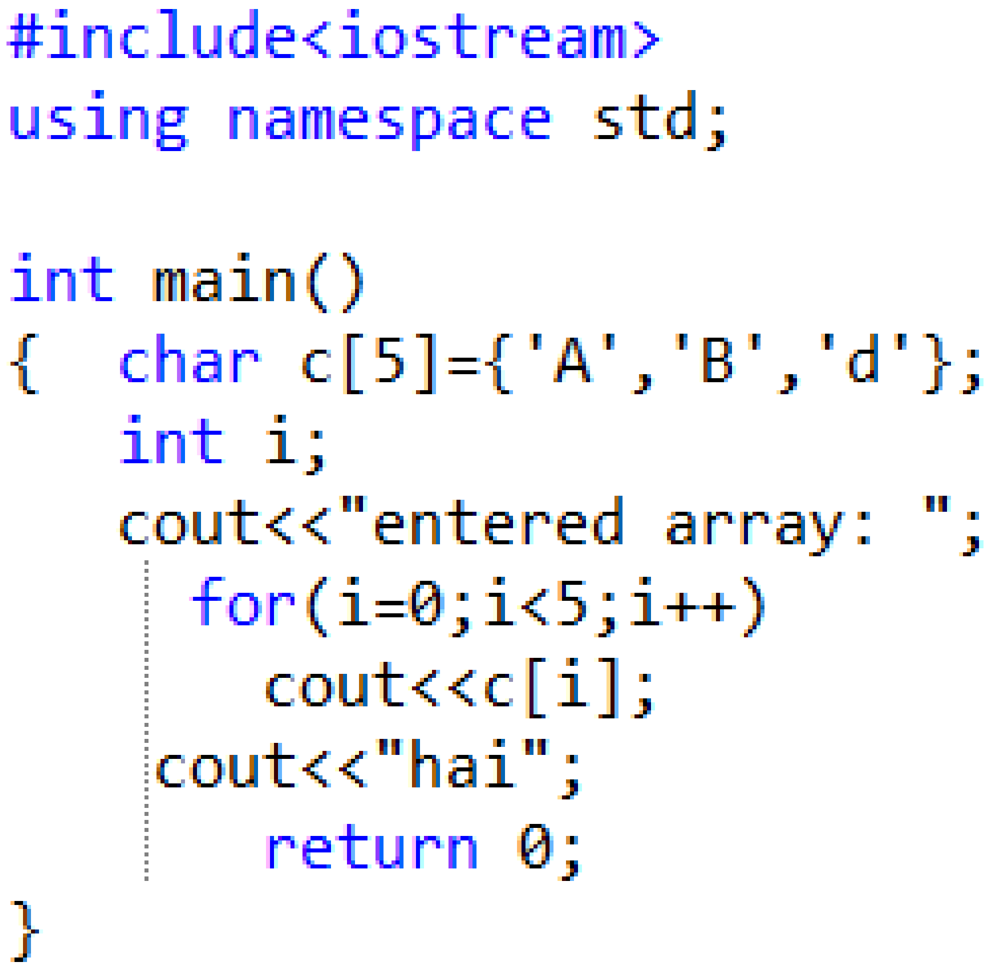


```
#include<iostream>
using namespace std;

int main()
{   int c[];
    int i;
    cout<<"enter an array: ";
    for(i=0;i<4;i++)
      cin>>c[i];
    cout<<"entered array: ";
      for(i=0;i<4;i++)
        cout<<c[i]<<" ";
      return 0;
}
```

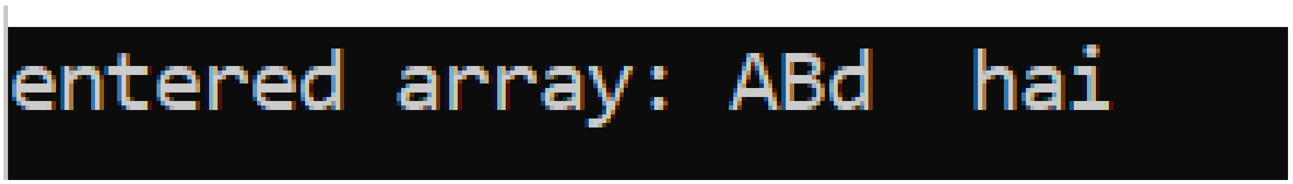


```
Message
In function 'int main()':
[Error] storage size of 'c' isn't known
```

In character array, uninitialized locations are filled with whitespaces.



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

int main()
{   char c[5]={'A','B','d'};
    int i;
    cout<<"entered array: ";
      for(i=0;i<5;i++)
        cout<<c[i];
     cout<<"hai";
        return 0;
}
```

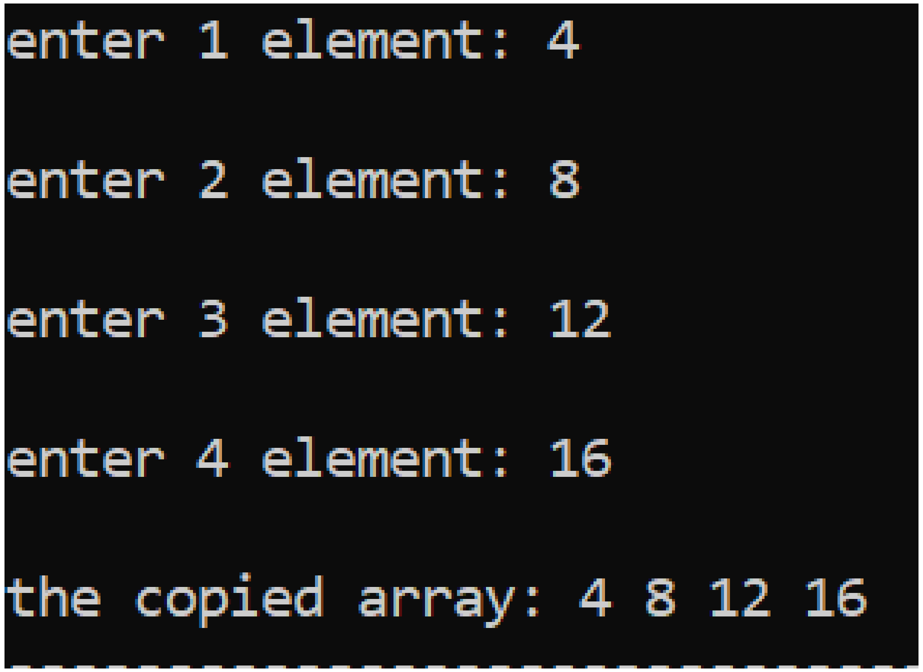


```
entered array: ABd  hai
```

Some programs on arrays

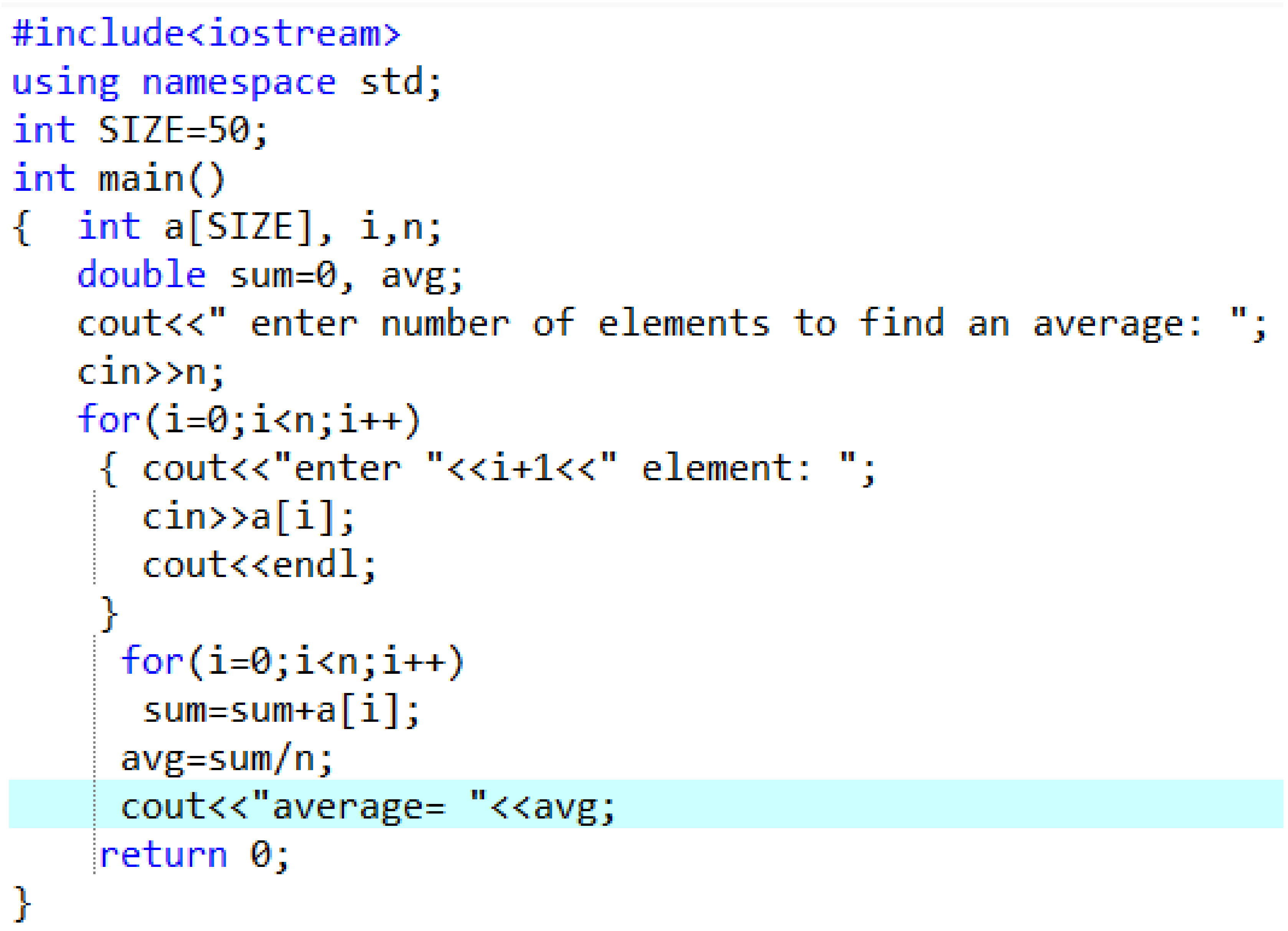
Copying the elements of one array to another

```cpp
#include<iostream>
using namespace std;
int main()
{  int a[4], b[4], i;
   for(i=0;i<4;i++)
    { cout<<"enter "<<i+1<<" element: ";
      cin>>a[i];
      cout<<endl;
    }
    for(i=0;i<4;i++)
      b[i]=a[i];
    cout<<"the copied array: ";
     for(i=0;i<4;i++)
      cout<<b[i]<<" ";
    return 0;
}
```

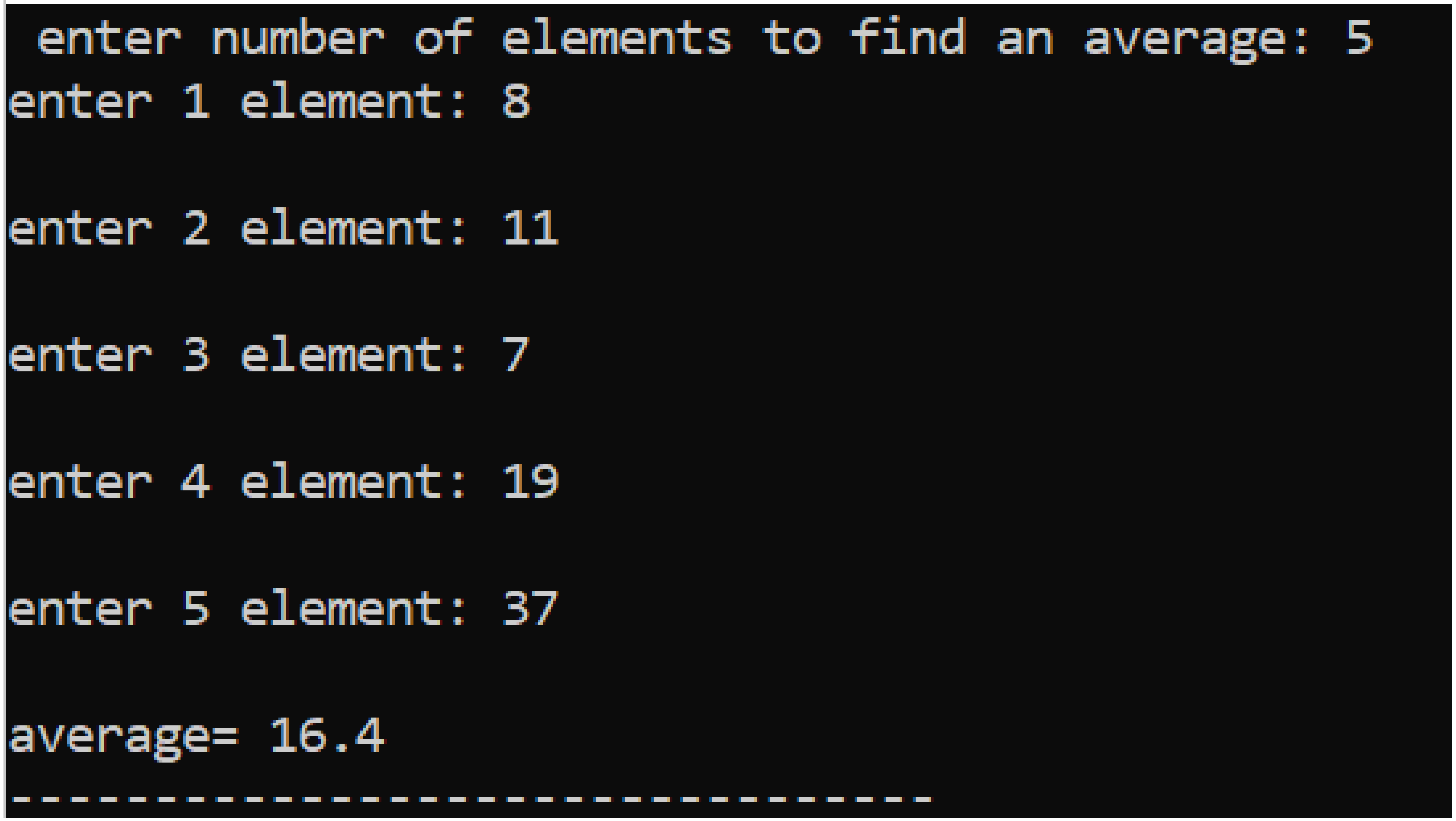


```
enter 1 element: 4

enter 2 element: 8

enter 3 element: 12

enter 4 element: 16

the copied array: 4 8 12 16
```

Find an average of n numbers

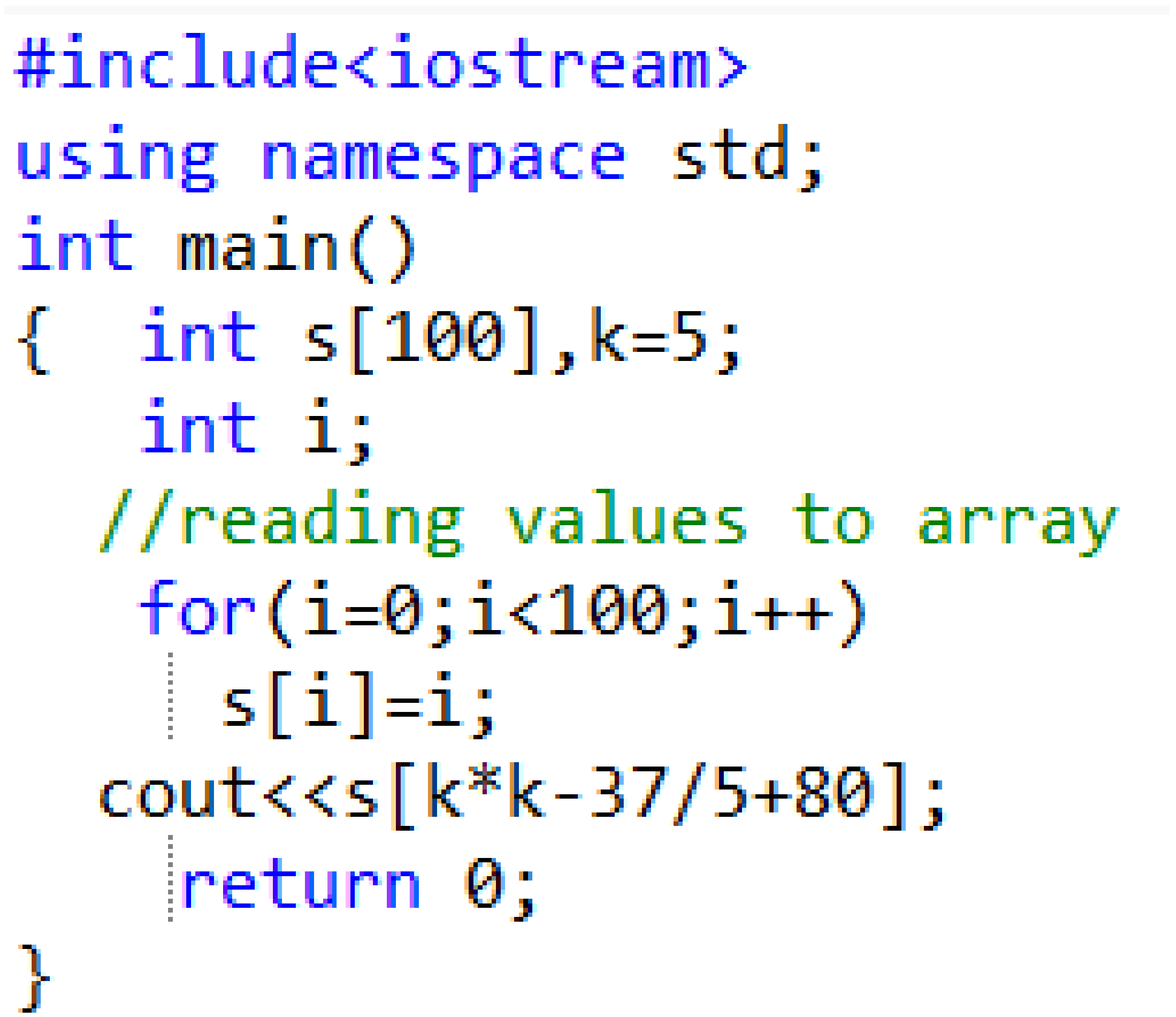


```cpp
#include<iostream>
using namespace std;
int SIZE=50;
int main()
{  int a[SIZE], i,n;
   double sum=0, avg;
   cout<<" enter number of elements to find an average: ";
   cin>>n;
   for(i=0;i<n;i++)
    { cout<<"enter "<<i+1<<" element: ";
      cin>>a[i];
      cout<<endl;
    }
     for(i=0;i<n;i++)
      sum=sum+a[i];
     avg=sum/n;
     cout<<"average= "<<avg;
    return 0;
}
```



```
 enter number of elements to find an average: 5
enter 1 element: 8

enter 2 element: 11

enter 3 element: 7

enter 4 element: 19

enter 5 element: 37

average= 16.4
---------------------------------
```

Find out the output of the following program



```
#include<iostream>
using namespace std;
int main()
{   int s[100],k=5;
    int i;
  //reading values to array
    for(i=0;i<100;i++)
      s[i]=i;
  cout<<s[k*k-37/5+80];
    return 0;
}
```

Next lecture,
Sorting and searching