Computer Programming & Problem Solving

CS100

Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa
May, 2023

ARRAYS

Contents

Arrays

1. **English: An ordered series of a particular type of thing**
2. **In C Language: An array is a data structure which can represent a collection of data items having the same data type (float/int/char/…)**

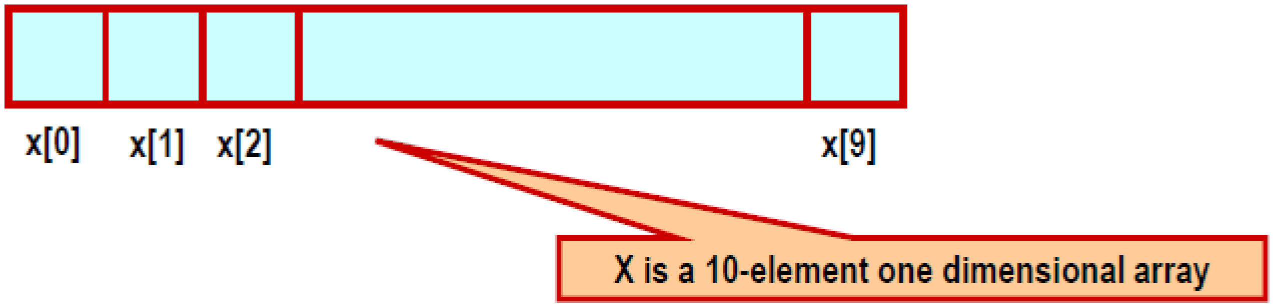
Why Arrays? – Examples

1. **Many applications require multiple data items that have common characteristics**
2. **Example: Finding the minimum of a set of *n* numbers.**
   - **works fine if *n* value is low.**
   - **But what happens if *n* = 100? Or even more?**
   - **Do we use 100 different variables? No.**
   - **We use arrays - one variable capable of storing or holding all the hundred values.**

Using Arrays

1. **In mathematics, we often express such groups of data items in indexed form:** $x_1, x_2, x_3, \ldots, x_n$
2. **All the data items constituting the group share the same name.**
3. **Individual elements are accessed by specifying the index**

```
int x[10];
```



Declaring Arrays

1. **Like variables, the arrays 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

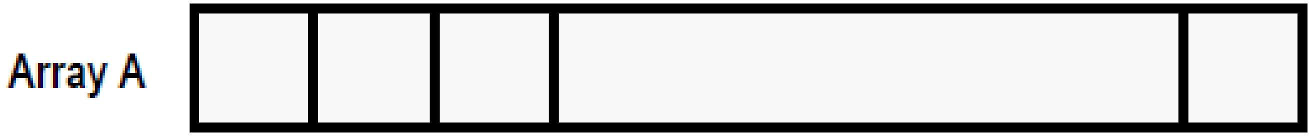
Some more Array Declarations

```
int  x[10];
char  line[80];
float  points[150];
char  name[35];
```

1. **If we are not sure of the exact size of the array, we can define an array of a large enough size.**

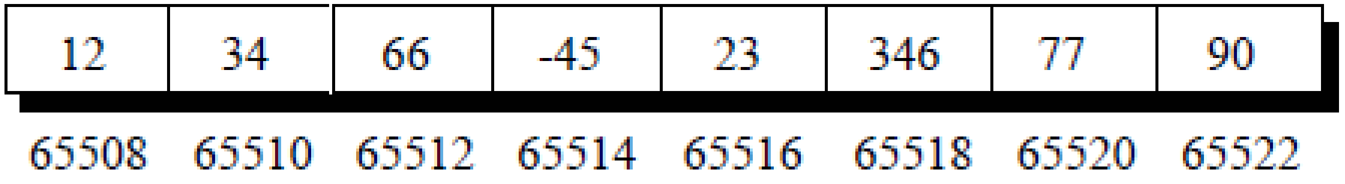
How are Arrays stored in Memory?

1. **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

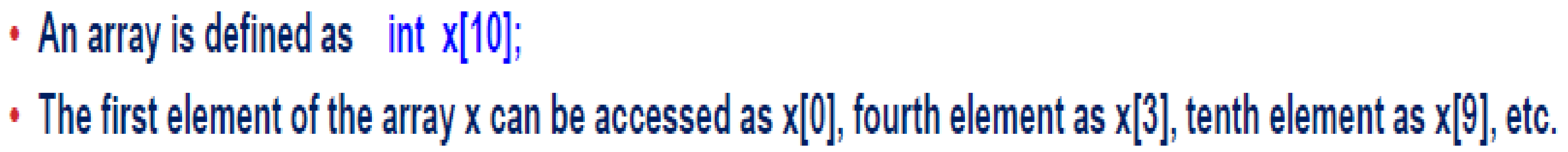
A[i] is allocated memory location at address x + (i*k)



12	34	66	-45	23	346	77	90
65508	65510	65512	65514	65516	65518	65520	65522

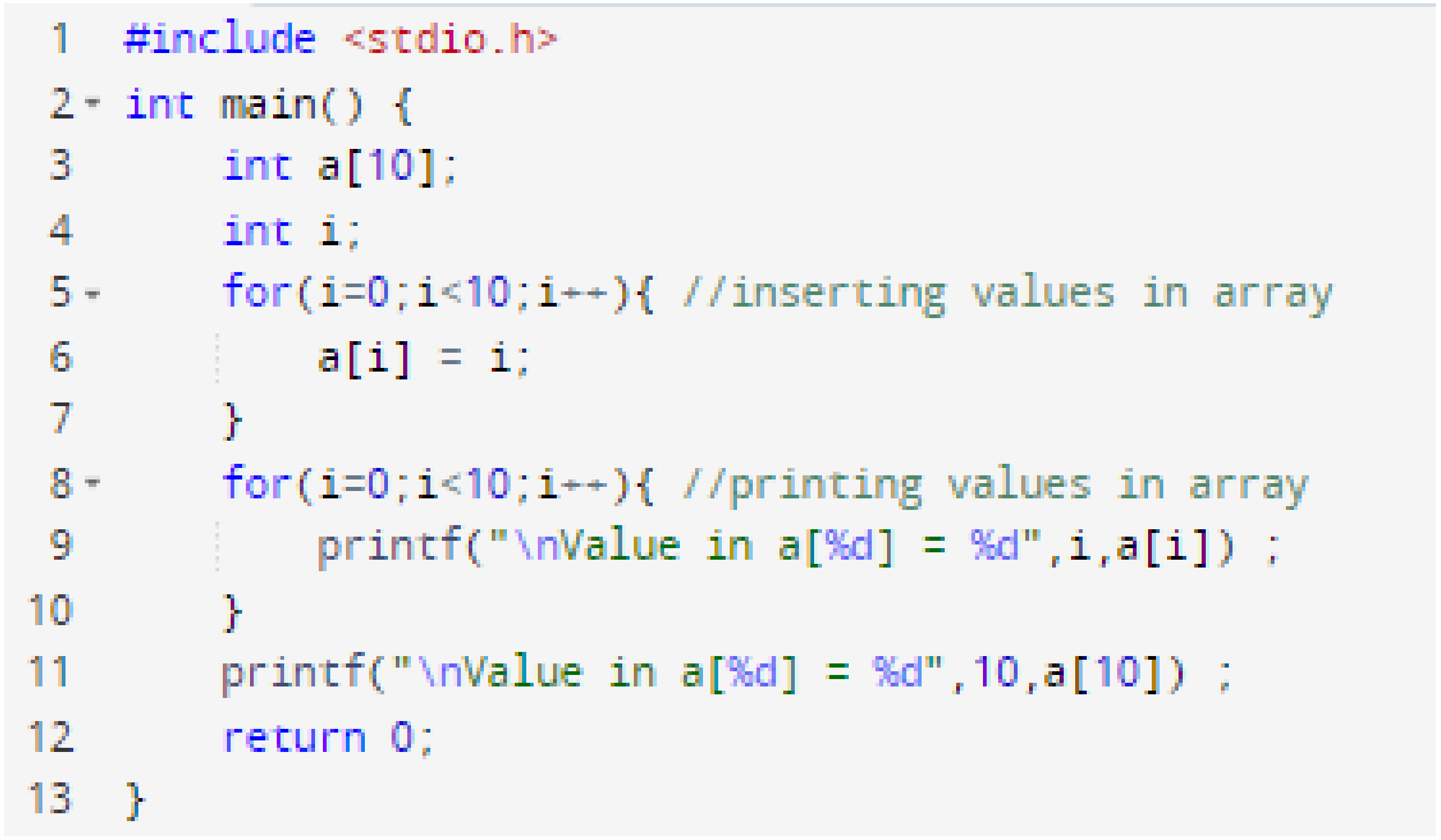
Accessing Array Elements

1. **A particular element of the array can be accessed by specifying two things:**
   a) **Name of the array**
   b) **Index (relative position) of the element in the array**
2. **In C, the index of an array starts from 0, not 1**



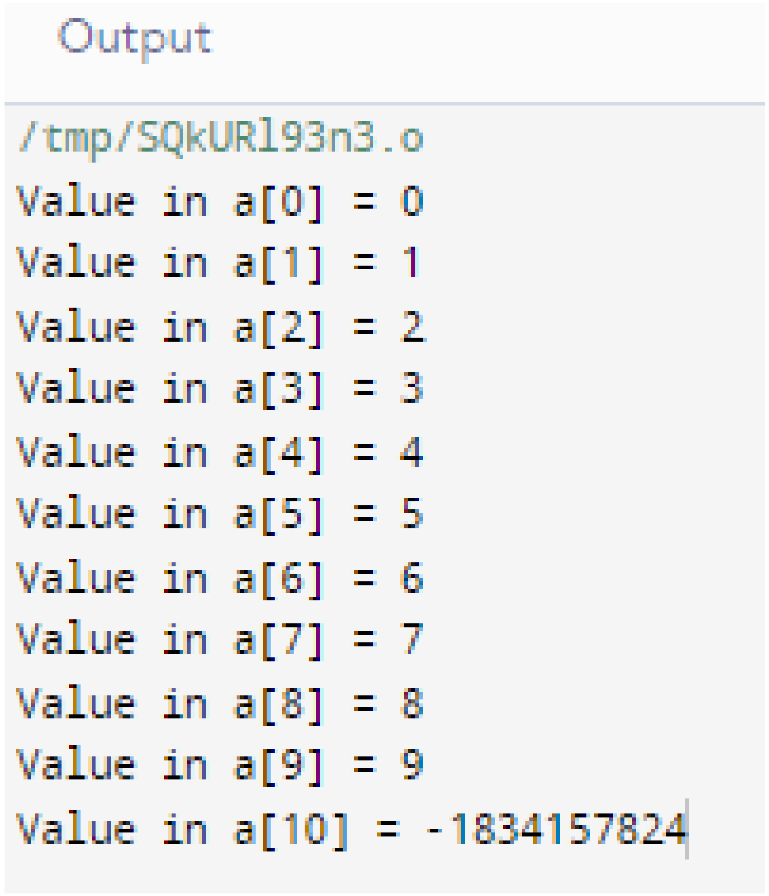
- An array is defined as `int x[10];`
- The first element of the array x can be accessed as x[0], fourth element as x[3], tenth element as x[9], etc.

Example



```c
#include <stdio.h>
int main() {
    int a[10];
    int i;
    for(i=0;i<10;i++){ //inserting values in array
        a[i] = i;
    }
    for(i=0;i<10;i++){ //printing values in array
        printf("\nValue in a[%d] = %d",i,a[i]) ;
    }
    printf("\nValue in a[%d] = %d",10,a[10]) ;
    return 0;
}
```

Example



Output

```
/tmp/SQkUR193n3.o
Value in a[0] = 0
Value in a[1] = 1
Value in a[2] = 2
Value in a[3] = 3
Value in a[4] = 4
Value in a[5] = 5
Value in a[6] = 6
Value in a[7] = 7
Value in a[8] = 8
Value in a[9] = 9
Value in a[10] = -1834157824
```

Example

1. In the above example you can use scanf() also

Initializing Arrays

- **General form:**

```
type array_name[size] = {list of values};
```

- **Examples:**

```
 int  marks[5] = {72, 83, 65, 80, 76};
char name[4] = {'A', 'm', 'i', 't'};
```

The size may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

```
int  flag[] = {1, 1, 1, 0};
char name[] = {'A', 'm', 'i', 't'};
```

Copy the elements of one array to another

Copy individual elements

```
for  ( j = 0; j < 25; j++ )
     a[ j ] = b[ j ];
```

A Warning!

In C, while accessing array elements, array bounds are not checked

Example:

```
int   marks[5];
:
:
marks[8] = 75;
```

- The above assignment would not necessarily cause an error during compilation
- Rather, it may result in unpredictable program results, which are very hard to debug

Things you cannot do

- use = to assign one array variable to another

```
a = b;   /* a and b are arrays */
```

- use == to directly compare array variables

```
if  (a = = b)  ...........
```

- directly scanf or printf arrays

```
printf (“……”, a);
```

What is happening here?

```
main( )
{
        int  avg, sum = 0 ;
        int  i ;
        int  marks[30] ;  /* array declaration */

        for ( i = 0 ; i <= 29 ; i++ )
        {
                printf ( "\nEnter marks " ) ;
                scanf ( "%d", &marks[i] ) ;  /* store data in array */
        }

        for ( i = 0 ; i <= 29 ; i++ )
                sum = sum + marks[i] ;  /* read data from an array*/

        avg = sum / 30 ;
        printf ( "\nAverage marks = %d", avg ) ;
}
```