LAB #07

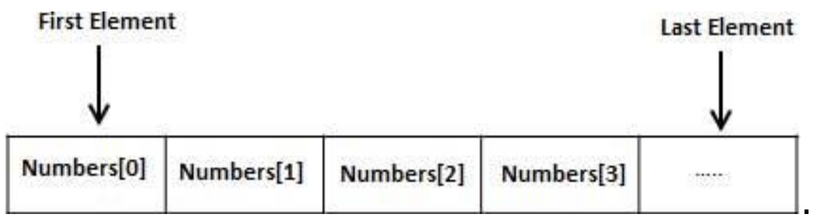
Introduction to single dimensional arrays:

What are arrays?

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element



- ### How to Declare an array?

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement −

```
double balance[10];
```

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

- ### How to initialize an array?

You can initialize an array in C either one by one or using a single statement as follows −

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

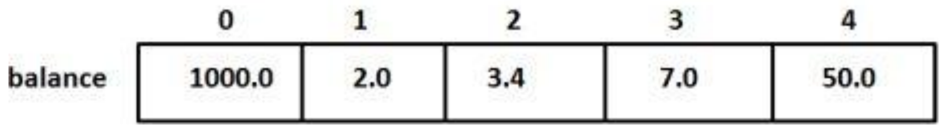
If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

```
balance[4] = 50.0;
```

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above −

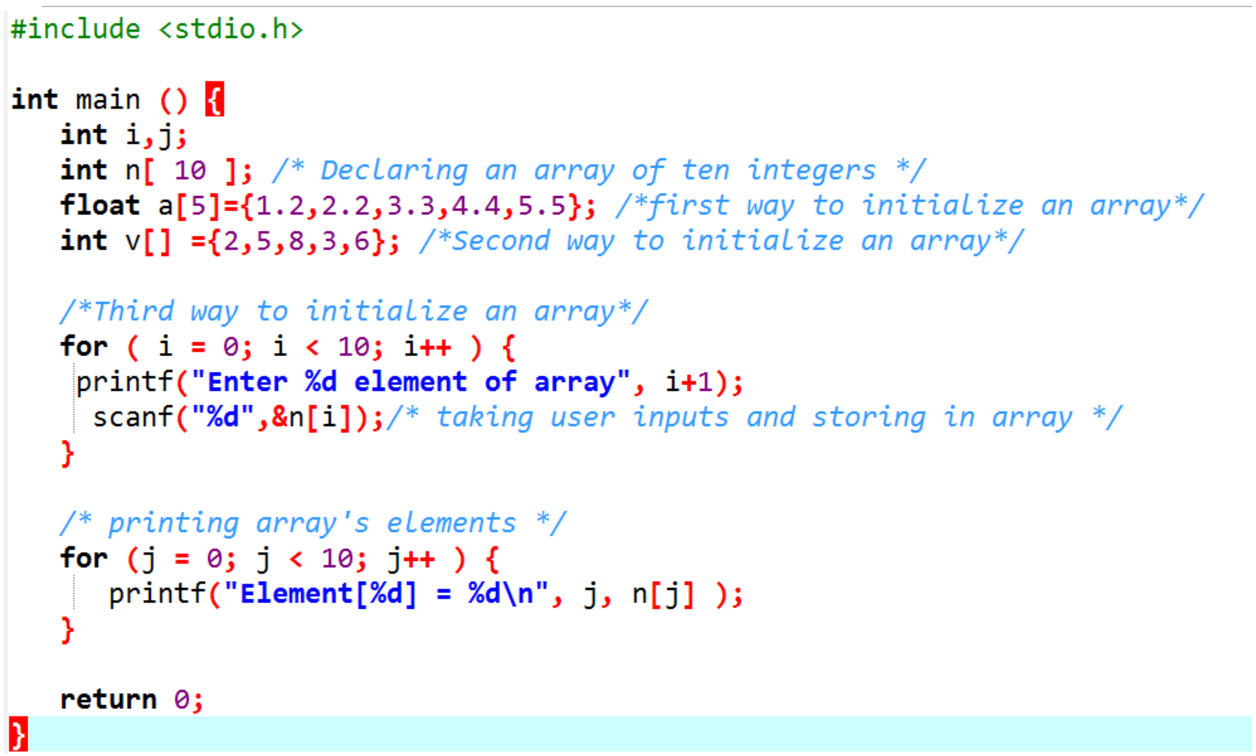


- **<u>How to Access an element</u>?**

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array.

```
double salary = balance[9];
```

Example:



```
#include <stdio.h>

int main () {
   int i,j;
   int n[ 10 ]; /* Declaring an array of ten integers */
   float a[5]={1.2,2.2,3.3,4.4,5.5}; /*first way to initialize an array*/
   int v[] ={2,5,8,3,6}; /*Second way to initialize an array*/

   /*Third way to initialize an array*/
   for ( i = 0; i < 10; i++ ) {
    printf("Enter %d element of array", i+1);
     scanf("%d",&n[i]);/* taking user inputs and storing in array */
   }

   /* printing array's elements */
   for (j = 0; j < 10; j++ ) {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

<u>Passing a complete One-dimensional array to a function:</u>

- **Formal parameter Vs Actual parameter**
- **Example :**

```c
#include <stdio.h>

/* function declaration */
double getAverage(int arr[], int size);

int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass the array as an argument */
   avg = getAverage( balance, 5 ) ;/* function calling*/

   /* output the returned value */
   printf( "Average value is: %f ", avg );

   return 0;
}

/*function Definition*/

/* double getAverage(int arr[5], int size)
 double getAverage(int *arr, int size) */


double getAverage(int arr[], int size) { // one way to pass array in function

   int i;
   double avg;
   double sum = 0;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = sum / size;

   return avg;
}
```

String and Character Array

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A **string** is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

For example: The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.

There are different ways to initialize a character array variable.

```
char name[13] = "StudyTonight";        // valid character array initialization
char name[10] = {'L','e','s','s','o','n','s','\0'};        // valid
```

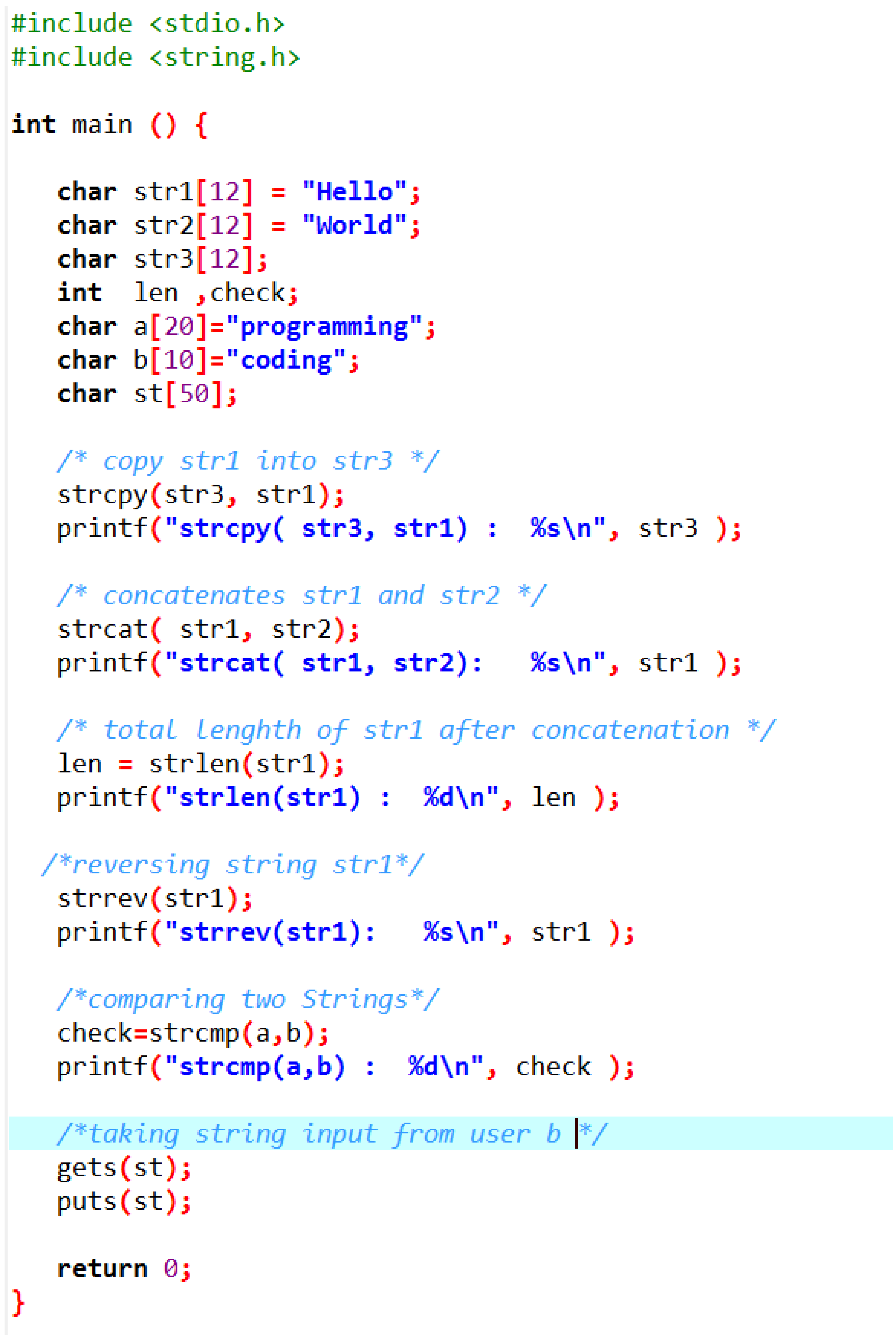
String Handling Functions:

Method	**Description**
strcat()	It is used to concatenate(combine) two strings
strlen()	It is used to show length of a string
strrev()	It is used to show reverse of a string
strcpy()	Copies one string into another
strcmp()	It is used to compare two string

strcmp(s1, s2);

Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.

Example:



```c
#include <stdio.h>
#include <string.h>

int main () {

   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ,check;
   char a[20]="programming";
   char b[10]="coding";
   char st[50];

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

  /*reversing string str1*/
   strrev(str1);
   printf("strrev(str1):   %s\n", str1 );

   /*comparing two Strings*/
   check=strcmp(a,b);
   printf("strcmp(a,b) :  %d\n", check );

   /*taking string input from user b */
   gets(st);
   puts(st);

   return 0;
}
```