Arrays

- Array Declaration and Memory allocation
- Array Initialization
- Constant Arrays and Parallel Arrays
- Searching from Array
- String processing

Array

- An array is a series or list of variables in computer memory, all of which have the same name and data type but are differentiated with special numbers called subscripts.
- A subscript, also called an index, is a number that indicates the position of a particular item within an array.

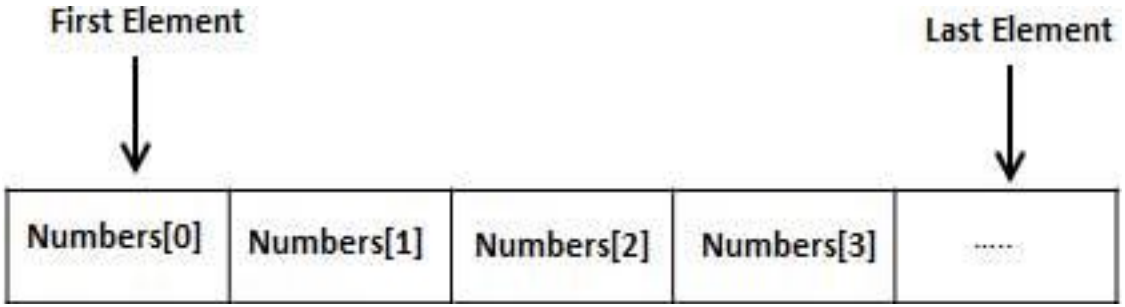
- Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

1D Array Declaration

- Syntax:
- type arrayName [arraySize];
- 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-
- Example:
- double balance[10];
- Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Memory allocation

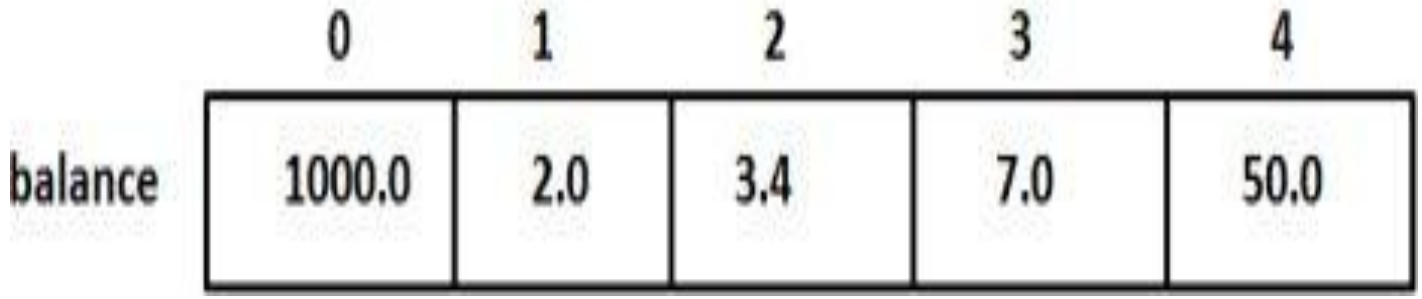
- All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Array Initialization

- 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 [ ].

- 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 –



- 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, 50.0,60.3};
- You will create array with 6 values.

```
int ary[3];
Int val;
ary[0]=3;
ary[1]=5;
ary[2]=7;
val=ary[1];
printf(“%d”, val);        //5
```

```
#include <stdio.h>
int main ()
{
int n[ 10 ];
int i,j;
for ( i = 0; i < 10; i++ )
{
      n[ i ] = i + 100;
}
for (j = 0; j < 10; j++ )
{
      printf("Element[%d] = %d\n", j, n[j] );
}
return 0;
}
```

```
#include<stdio.h>
void main( )
{
    int i, a = 2, b = 3 ;
    int arr[ 2 + 3] ;
    for ( i = 0 ; i < a+b ; i++ )
    {
       printf ( "\nEnter %d", i ) ;
       scanf ( "%d", &arr[i] ) ;
    }
    for ( i = 0 ; i < a+b ; i++ )
    {
         printf ( "\n%d", arr[i] ) ;
    }
}
```

Constant Arrays

```
const int ary[3] = {22, 40, 60};
```

- When an array is constant, it can be assigned its permanent and final values when you write the program code.

Parallel Arrays

- Each element in one array is associated with the same relative position in the other array is called Parallel array.
- int size=5;
- int item[size]={101,102,103,104,105};
- int price[size]={150,240,100,50,90};

Searching from Array

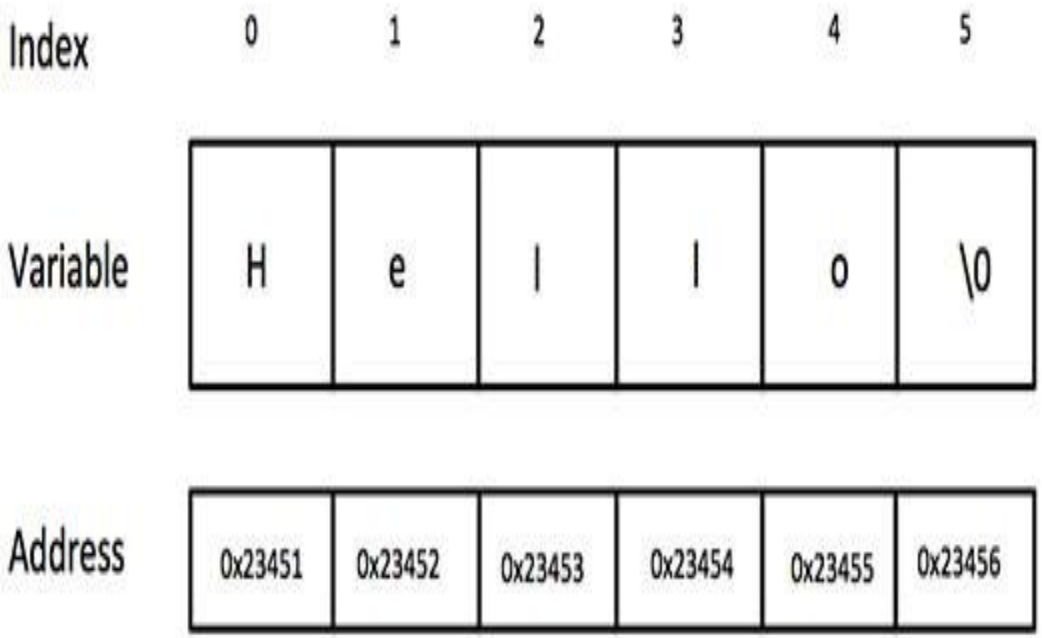
```
#include<stdio.h>

int main() {
int a[30], ele, num, i;

  printf("\nEnter no of elements :");
  scanf("%d", &num);

  printf("\nEnter the values :");
  for (i = 0; i < num; i++) {
    scanf("%d", &a[i]);
  }

  //Read the element to be searched
  printf("\nEnter the elements to be
     searched :");
  scanf("%d", &ele);

//Search starts from the zeroth location
  i = 0;
  while (i < num && ele != a[i]) {
    i++;
  }

  //If i < num then Match found
  if (i < num) {
    printf("Number found at the location
     = %d", i + 1);
  } else {
    printf("Number not found");
  }

  return (0);
}
```

String processing

- Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.

- char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
- The above declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello"

- char greeting[] = "Hello";
- Actually, you do not place the *null* character at the end of a string constant.
- The C compiler automatically places the '\0' at the end of the string when it initializes the array.


Index
0
1
2
3
4
5
Variable
H
e
l
l
o
\0
Address
0x23451
0x23452
0x23453
0x23454
0x23455
0x23456

- scanf()
- gets()
- puts()
- getch()

- **strcpy(s1, s2);**
  - Copies string s2 into string s1.
- **strcat(s1, s2);**
  - Concatenates string s2 onto the end of string s1.
- **strlen(s1);**
  - Returns the length of string s1.
- **strcmp(s1, s2);**
  - Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
- **strchr(s1, ch);**
  - Returns a pointer to the first occurrence of character ch in string s1.
- **strstr(s1, s2);**
  - Returns a pointer to the first occurrence of string s2 in string s1.

```
#include <stdio.h>
#include <string.h>
int main ()
{
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12];
int len ;
strcpy(str3, str1);
printf("strcpy( str3, str1) : %s\n", str3 );
strcat( str1, str2);
printf("strcat( str1, str2): %s\n", str1 );
len = strlen(str1);
printf("strlen(str1) : %d\n", len );
return 0;
}
```

```
#include <stdio.h>
#include <string.h>
int main()
{
char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
int result;
result = strcmp(str1, str2);
printf("strcmp(str1, str2) = %d\n", result);
result = strcmp(str1, str3);
printf("strcmp(str1, str3) = %d\n", result);
return 0;
}
```

- The first unmatched character between string **str1** and **str2** is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67. Hence, when strings **str1** and **str2** are compared, the return value is 32.
- When strings **str1** and **str3** are compared, the result is 0 because both strings are identical.

- strlwr ( )
  - Converts string to lowercase
- strupr ( )
  - Converts string to uppercase
- strrev ( )
  - Reverses the given string

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

int main()
{
   char str[ ] = "Modify This String To Upper";
   printf("%s\n",strupr(str));
   return  0;
}
```