Arrays & Strings

Objective

- Array definition
- One-dimensional array
  - Declaration
  - Initialization
  - Way to access of Elements
  - Representation of array elements in memory
  - Possible operations
- One-dimensional strings
  - Declaration
  - Initialization
  - Manipulation

Introduction

- **Array:** It is a collection of individual data elements that is ordered, fixed in size and homogeneous.
- *A scalar variable is* a single variable whose stored value is an atomic type.
  - For example, each of the variables ch, n, and price declared in the statements
    - char ch;
    - int n;
    - float price;
  - Are of different data types and each variable can only store one value of the declared data type. These types of variables are called *scalar variables.*

Key Words

- **Array identifier:** A name assigned to an array.
- **Array initialization:** The procedure of assigning numerical value or character to each element of an array.
- **Array of strings:** An array that contains strings as its elements.
- **Concatenation of strings:** A kind of string manipulation where one string is appended to another string.

Key words

- **Homogeneous data:** Data of same kind or same data type.
- **Index of an array:** It is an integer constant or variable ranging from 0 to (size – 1).
- **One-dimensional array:** An array that is represented by a name and single index or subscript.
- **Multidimensional array:** An array that is represented by a name and more than one index or subscript.
- **Size of array:** The number of elements in an array.
- **String :** One-dimensional array of characters that contain a NUL at the end.

Characteristics of Array?

- A program that can print its input in reverse order. If there are two or three values, this is easy but what if there are ten or twenty or one hundred values? Then it is not so easy. To avoid above problem we use array .
- An array is a collection of individual data elements that is ordered, fixed in size, and homogeneous.
- An array is considered to be a derived data type.
- Array enables the storing and manipulation of potentially huge quantities of data.

One-Dimensional Array

- There are several forms of an array used in C:
  - One dimensional or single-dimensional and multidimensional array.
  - Since the array is one dimensional, there will be a single subscript or index whose value refers to the individual array element which ranges from 0 to (*n–1), where n is the* total number of elements in the array.
- When defining an array in a program three things need to be specified.
  - The type of data it can hold, i.e., int, char, double, float, etc.
  - The number of values it can hold, i.e., the maximum number of elements it can hold
  - A name

One-Dimensional Array

- The array *size must be a positive integer number or an* expression that evaluates to a positive integer number that must be specified at the time of declaration with the exception that it may be unspecified while initializing the array.
- In C, the array index starts at 0 and ends at (size–1) and provides the means of accessing and modifying the specific values in the array.

Syntax: Array

- The syntax for declaration of a one-dimensional array is data_type array_name [SIZE];
  - All the array elements hold values of type <data type>
  - The size of the array is indicated by <SIZE>, the number of elements in the array. <SIZE> must be an int constant or a constant expression.
- int a[size]; /* memory space for a[0],a[1],…, a[size –1] allocated */
  - lower bound = 0
  - upper bound = size –1
  - size = upper bound – lower bound + 1

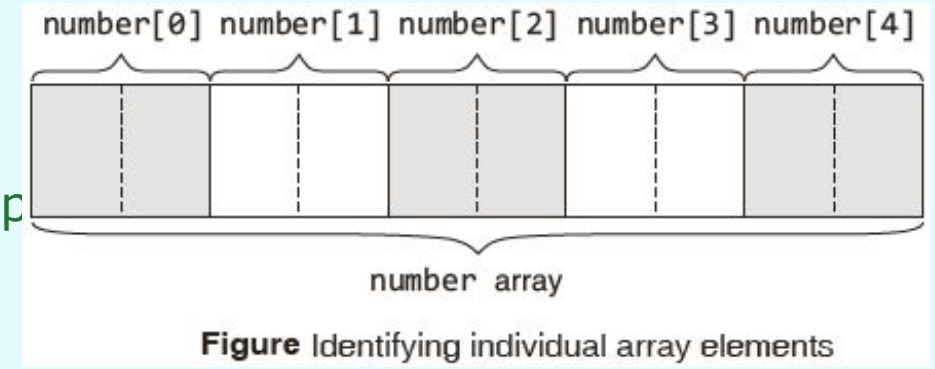

Figure Identifying individual array elements

  - number[0] & number[4] refers to the first & fifth number stored in the 'number' array respectively.

Initializing Integer Arrays

- Variables can be assigned values during declaration like the following example.

  int x = 7;

- Array initialization statements as shown.

  (a) int A[10] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};

  9 8 7 6 5 4 3 2 1 0 ¨ values stored in array elements

  0 1 2 3 4 5 6 7 8 9 ¨index values of array elements

  (b) double a[5] = {3.67, 1.21, 5.87, 7.45, 9.12}

- Automatic sizing While initializing, the size of a one dimensional array can be omitted as shown.

  int arr[] = {3,1,5,7,9};

  - Here, the C compiler will deduce the size of the array from the initialization statement.

Array declaration, initialization and accessing	Example
Array declaration syntax: data_type arr_name [arr_size]; **Array initialization syntax:** data_type arr_name [arr_size]=(value1, value2, value3,….); **Array accessing syntax:** arr_name[index];	**Integer array example:** int age [5]; int age[5]={0, 1, 2, 3, 4}; age[0]; /*0 is accessed*/ age[1]; /*1 is accessed*/ age[2]; /*2 is accessed*/
	Character array example: char str[10]; char str[10]={‘H’,‘a’,‘i’}; (or) char str[0] = ‘H’; char str[1] = ‘a’; char str[2] = ‘i; str[0]; /*H is accessed*/ str[1]; /*a is accessed*/ str[2]; /*i is accessed*/

example

```
#include<stdio.h>

int main()
{
   int i;
   int arr[5] = {10,20,30,40,50};
    for (i=0;i<5;i++)
   {
      // Accessing each variable
      printf("value of arr[%d] is %d \n", i, arr[i]);
   }

}
```

```
value of arr[0] is 10
value of arr[1] is 20
value of arr[2] is 30
value of arr[3] is 40
value of arr[4] is 50
```

example

- double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
- double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
- balance[4] = 50.0;

0	1	2	3	4
1000.0	2.0	3.4	7.0	50.0

pictorial representation of the array we discussed above

```c
#include <stdio.h>
int main ()
{ int n[10];
   /* n is an array of 10 integers */
   int i,j;
   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ )
   {
     n[ i ] = i + 100;
     /* set element at location i to i + 100 */
   }
   /* output each array element's value */
   for (j = 0; j < 10; j++ )
   {
     printf("Element[%d] = %d\n", j, n[j] );
   }
return 0;
}
```

Output

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Accessing Array Elements

- x and y are similar arrays (i.e., of the same data type, dimensionality, and size), then assignment operations, comparison operations, etc., involving these two arrays must be carried out on an element-by-element basis.

Determine total of all elements in a number array

- Examples using the elements of an array named 'numbers' are shown here:

```
numbers[0]=98;
numbers[1]=(numbers[0]-11);
numbers[2]=2*(numbers[0]-6);
numbers[3]=79;
total = numbers[0]+numbers[1]+numbers[2]+numbers[3];
```

Array : Other Allowed Operations

These operations include the following, for an array named 'ar'.

- (a) To **increment the i^{th} element**, the given statements can be used.

```
ar[i]++;
ar[i] += 1;
ar[i] = ar[i] + 1;
```

- (b) To **add n to the i^{th} element**, the following statements may be used,

```
ar[i] += n;
ar[i] = ar[i] + n;
```

Array : Other Allowed Operations

These operations include the following, for an array named 'ar'.

- (c) To **copy the contents** of the i^{th} element to the kth element, the following statement may be written.

```
ar[k] = ar[i];
```

(d) To copy the contents of one array 'ar' to another array 'br', it must again be done one by one.

```
int ar[10],br[10];
ar[10] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
for(i = 0; i < 10; i = i + 1)
{
br[i] = ar[i];
}
```

Array : Other Allowed Operations

These operations include the following, for an array named 'ar'.

- (e) To exchange the values in ar[i] and ar[k], a 'temporary' variable must be declared to hold one value, and it should be the same data type as the array elements being swapped.

```
int temp;
temp = ar[i];
/* save a copy of value in ar[i] */
ar[i] = ar[j];
/* copy value from ar[j] to ar[i] */
ar[j] = temp;
 /* copy saved value of ar[i] to ar[j] */
```

Storing values given by the user in an array

- ***Reading the*** input into an array is done as shown. int a[10]; /* an array with 10 “int” elements */

```
int i;
for(i=0 ; i< 10; i++)
scanf(“%d”, &a[i]);
```

- The idea is that first a value must be read and copied into a[0], then another value read and copied into a[1], and so on, until all the input values have been read.

```c
#include <stdio.h>
int main(void)
{
int numbers[10];
int count = 10;
int sum = 0;
float average = 0.0f;
printf("\nEnter the 10 numbers:\n");

int i;
for(i = 0; i < count; i ++)
{
    printf("%d> ",i+1);
    scanf("%d", &numbers[i]);
    sum += numbers[i];
}
printf("\nsum of the ten numbers entered is: %d\n", sum);
average = (float)sum/count;
printf("\nAverage of the ten numbers entered is: %f\n", average);
return 0; }
```

```
Enter the 10 numbers:
1> 2
2> 3
3> 1
4> 2
5> 1
6> 1
7> 1
8> 2
9> 1
10> 1
Average of the ten
numbers entered is:
1.500000
```

Printing an Array

- The following code segment prints the elements of an array, a[10].

```
for(i=0 ; i< 10; i++)
printf("%d", a[i]);
```

- For printing of numbers entered by the user in the reverse order, the program will be as follows

```
#include <stdio.h>
int main()
{
    int a[5],i;
    printf("Enter 5 Numbers:\n");
    /*For accepting 5-integer numbers from user*/
    for(i=0;i<5;i++)
  a[i]=i+10;
    for(i=4;i>=0;i--)
  printf("Element[%d] = %d\n",i,a[i]);
    return 0;
}
```

- Print odd numbers in an array

```
#include <stdio.h>
int main()
{
    int a[5],i;
    printf("Enter 5 Numbers:\n");
    /*For accepting 5-integer numbers from user*/
    for(i=0;i<5;i++)
        scanf("%d",&a[i]);

    printf("Odd Numbers in Array are:\n");
    for(i=0;i<5;i++)
   {
        if(a[i]%2!=0) //Check number is odd
     {
         printf("%d ",a[i]);
     }
  }
    return 0;
}
```

Write a program to find largest element stored in an array.

```
#include <stdio.h>
int main()
{
int a[50],i,n, large;
printf("\nEnter how many numbers :");
scanf("%d",&n);
printf("\nEnter values for the array: ");
for(i=0; i<n; i++) /* Stores number entered by user. */
      scanf("%d",&a[i]);
/*code to find largest element in an array*/
large=a[0];
for(i=1; i<n; i++)
{
      if(a[i]>large)
      large=a[i];
}
printf("\nThe largest element stored in array a is %d", large);
return 0;
}
```

- How to find second largest element in an array

```
Max 1 = a[0]
Max 2 =0

if a[i] > Max1

{Max2= Max1
Max 1=a[i]}
```

```
#include <stdio.h>
int main()
{
int numbers[10],i;
for(i=0;i<10;i++)
scanf("%d",&numbers[i]);
int max1 = numbers[0];
int max2 = 0;
for(i=0;i<10;i++)
{
if(numbers[i]>max1)
{
max2 = max1;
max1 = numbers[i];
}
}
printf("largest numbers = %d\n",max1);
printf("Second largest numbers = %d\n",max2);
return 0;
}
```

Internal Representation of Arrays in C

- ***References to elements outside of the array bounds It is*** important to realize that there is no array bound checking in C.

- ***A bit of memory allocation It has been seen how arrays*** can be defined and manipulated. It is important to learn how to do this because in more advanced C programs it is necessary to deal with something known as dynamic memory management.
  - It is given a small area of the computer's memory to use. This memory, which is known as the *stack, is used by variables in the* program

```
int a = 10;
float values[100];
```

APPLICATION : One-Dimensional Array

- ***Printing binary equivalent of a decimal number using array***
- ***Here the remainders of the integer division of a*** decimal number by 2 are stored as consecutive array elements.
- ***The division procedure is repeated until the number becomes 0.***

```
#include <stdio.h>
int main()
{
int numbers[20],i,n,r;
printf("Enter a number \n");
scanf("%d", &n);
for(i=0;n>0;i++)
{
r = n%2;
numbers[i] = r;
n = n/2;
}
for(i=i-1;i>=0;i--)
printf("%d",numbers[i]);
return 0;
}
```

Searching an element within an array

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
 int a[30],n,i,key, FOUND=0;

 printf("\n How many numbers");

 scanf("%d",&n);
 if(n>30)
 {
  printf("\n Too many Numbers");

  exit(0);
 }
 printf("\n Enter the array elements \n");

 for(i=0 ; i<n; i++)

 scanf("%d", &a[i]);

 printf("\n Enter the key to be searched \n");

 scanf("%d",&key);

 for(i=0 ; i<n; i++)
```

For loop ➡

```
{
if(a[i] == key)
{
  printf("\n Found at %d",i);
  FOUND=1;
}}
if(FOUND = = 0)
printf("\n NOT FOUND...");
return 0;
}
```

Assignment

- WAP to find out the sum of the numbers stored in an array of integers.
- WAP to find largest element stored in an integer array.
- WAP to copy the elements of one array to another empty array.
- WAP to swap the elements of two arrays. Use third variable.
- WAP to enter the elements of array and print all the even elements.
- WAP to display the array elements in reverse order.
- WAP to search an element in the array.
- WAP to convert a decimal number to a binary number.