Unit-3

14:20 PM
Mon-22-Jan-2024

Arrays and Strings

Array indexing, memory-model, programs with array of integers, two-dimensional arrays, Introduction to strings.

Arrays:-

C supports a derived data type known as array. ~~that can be used as~~

An array is a fixed-size sequenced collection of elements of the same data type. An array can be used to represent a list of numbers, (or) a list of names.

An array is a sequenced collection of related data items that share a common name.

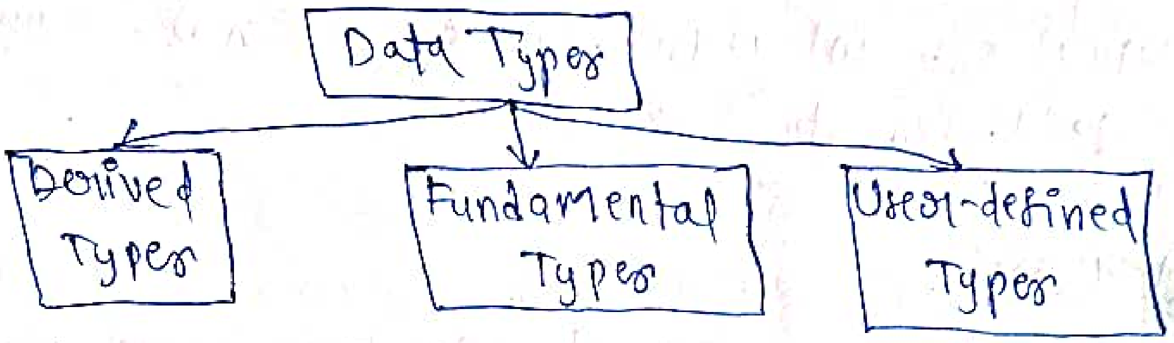
We can refer to the individual elements by writing a number called index (or subscript in brackets after the array name.

While the complete set of ~~arrays~~ values is referred to as an array, individuals values are called elements.

The ability to use a single name to represent a collection of items and to refer to an item by specifying the item number enables us to develop concise and efficient programs.

Types of arrays:

→ One dimensional arrays
→ Two dimensional arrays
→ Multidimensional arrays.



→ Arrays	→ Integral Types	→ Structures
→ Functions	→ Float Types	→ Unions
→ Pointers	→ Character Types	→ Enumerati

Arrays and structures are referred to as structured data types because they can be used to represent data values that have a structure of some sort.

Structured data types provide an organizational scheme that shows the relationships among the individual elements and facilitate efficient data manipulations.

In addition to arrays and structures, C supports creation and manipulation of the following data structures:

→ Linked Lists
→ Stacks
→ Queues
→ Trees

One-dimensional Arrays:

A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or 1-D array.

Like any other variable, arrays must be declared before they are used so that the compiler can allocate space for them in memory.

The general form of array declaration is

```
type variable-name [size];
```

The type specifies the type of element that will be contained in the array, such as int, float & char.

The size indicates the max number of elements that can be stored inside the array.

The C language treats character strings simply as array of characters.

The size in a character string represents the maximum number of characters that the string can hold.

eg: `char name[10];`

Initialization:

After an array is declared, its element must be initialized.

An array can be initialized at either of the following stages:

→ At compile time

→ At run time.

We can initialize the elements of arrays in the same way as the ordinary variables when they are declared.

eg: int number [3] = {1,2,3};

char name [10] = {'J','o','h','n','\0'};

char name [10] = "John";

An array can be explicitly initialized at run time. This approach is usually applied for initializing large arrays.

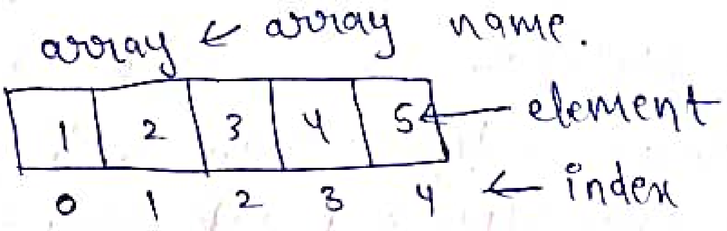
eg:
```
for (i=0; i< 100; i++)
{
    if (i<50)
        sum [i] = 0.0;
    else
        sum [i] = 1.0;
}
```

Access Array Elements:

We can access any element of an array in C using the array subscript operator [] and the index value 'i' of the element.

array-name [index];

Indexing in the array always starts with '0'. i.e, first element is at index '0'.
last element is at 'n-1'
where, 'n' is the number of elements in the array.



array [0] = 1
array [1] = 2
array [2] = 3

eg:
```
#include<stdio.h>
int main()
{
    int array[5] = {1,2,3,4,5};
    printf("Element at array[2] : %d\n",
                                array[2]);
    printf("Element at array[0] : %d\n",
                                array[0]);
    return 0;
}
```

output:
```
Element at array[2] : 3
Element at array[0] : 1
```

eg: Iteration through an array.
```
for (int i=0; i<5; i++)
{
    printf("%d", numbers[i]);
}
```

Array size: The size of an array is fixed during declaration and can not be changed during run time.

Memory Model :-

The memory model refers to how a program's memory is organized and managed during its execution.

The memory model defines how variables, data structures, and program instructions are stored in the computers memory.

Memory Segments:

Text Segment / Code Segment: This is where the compiled programs executable code is stored.

Data Segment: It further divided into.

Initialized Data: Variables that are explicitly initialized by the programmer.
eg. global variables..

Uninitialized Data: Variables that are not initialized by the programmer are stored here.

Heap: Dynamic memory allocation occurs in this segment. It is used for dynamic data structures such as linked lists and trees..

Stack: Function calls and local variables are stored here. It allows the LIFO - Last In First out.

Variables:

Global Variables: Stored in the data segment. They have a life time throughout the program execution.

Local Variables: Stored in the stack. They have a limited scope and lifetime within the function where they are declared.

Pointers: Pointers store memory addresses. They can point to variables, arrays, (or dynamically allocated ~~an~~ memory.

Dynamic Memory Allocation:

The "malloc", calloc, and realloc functions are used to allocate memory in the heap.
The free function is used to deallocate memory.

Programs with array of integers:-

1. sum of elements in an array.

```
#include<stdio.h>
 int main() {
       int numbers[] = {1,2,3,4,5};
       int sum =0;
       for (int i =0; i< sizeof(numbers) /
                              sizeof(numbers[0]); i++)
          {
               sum+= numbers[i];
          }
       printf("Sum of elements : %d \n", sum);
       return 0;
 }
```

2- Finding the maximum element in an array.

```
#include<stdio.h>
int main() {
        int numbers[] = {12,34,56,23,9};
        int max = numbers[0];
        for (int i=1; i< sizeof(numbers)/
                              sizeof(numbers[0]); i++)
        {
            if(numbers[i] > max) {
                max = numbers[i];
            }
        }
        printf(" Maximum element : %d\n", max);
        return 0;
}
```

3. Reversing an array.

```
#include<stdio.h>
int main() {
        int numbers[] = {1,2,3,4,5};
        int length = sizeof (numbers) / sizeof (numbers[0]);
        for (int i=0; i< length / 2; i++)
        {
            int temp = numbers[i];
            numbers[i] = numbers[length - i - 1];
            numbers[length - i - 1] = temp;
        }
        printf ("Reversed array :");
```

```
    for (int i=0; i<length; i++) {
        printf(" %d", numbers[i]);
    }
    return 0;
}
```

Two-dimensional Arrays :-

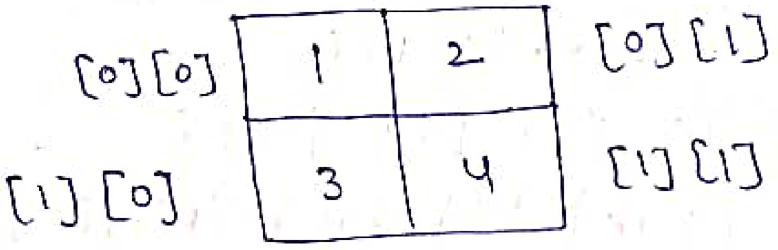
A table is a matrix consisting of [illegible] rows and columns.

We represent a particular value in a matrix by using two subscripts such as V_{ij}.

'V' denotes the entire matrix and V_{ij} refers to the value [illegible] in the ith row and jth column.

The two-dimensional arrays are declared as follows;

type array_name [row-size][column-size];



Representation of 2-D array

Initialization of 2D arrays is

int tab[2][2] = {1,2,3,4};

Two-dimensional arrays provide a way to organize data in a tabular form, where data is arranged in rows and columns.

1. Initializing and Accessing Elements:

```
#include<stdio.h>
int main() {
    int matrix[2][2] = { {1,2},
                         {3,4} };
    printf("Element at matrix[0][1]: %d\n",
                                matrix [0][1]);
    return 0;
}
```

2. Printing a 2D array:

```
#include< stdio.h>
int main() {
    int matrix[3][3] = { {1,2,3},
                         {4,5,6},
                         {7,8,9} };
    for(int i=0; i<2; i++) {
        for(int j=0; j<2; j++) {
            printf("%d", matrix[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Multi-dimensional Arrays:

C allows arrays of 3 or more dimensions. The exact limit is determined by the compiler.

The general form of a multi-dimensional array is

```
type array-name [s1][s2]---[sm];
```

ANSI 'C' does not specify any limit for array dimension. Most compilers permit 7 to 10 dimensions.

Introduction to strings:-

A string is a sequence of characters that is treated as a single data item.
Any group of characters defined b/w double quotation marks is a string constant.

eg:- " My name is Gsbd".

If we want to include double quote in the string.

```
printf("\" Well Done!"\");
```

character strings are often used to build meaningful and readable programs. The common operations performed on character strings include the following.

→ Reading and writing strings.
→ Combining strings together.
→ Copying one string to another.
→ Comparing strings for equality.
→ Extracting a portion of a string.

Declaring and Initializing String variables:

C does not support strings as a data type.
It allows us to represent strings as character array.

A string variable is any valid C variable name and is always declared as an array of characters.
The general form of declaring a string variable is:

```
char string_name [size];
```

The size determines the number of characters in the string name.
eg: char city [10];

When the compiler assigns a character string to a character array, it automatically supplies a null character ('\0') at the end of the string.
C also permits us to initialize a character array without specifying the number of elements. The size of the array will be determined automatically, based on the number of elements initialized.

```
char str[6] = "GSKD";
```

G	S	K	D	\0	\0

Reading String from Terminal:
The input function scanf can be used with %s format specification to read in a string of characters.
eg:

```
char address [10];
scanf("%s", address);
```

The scanf function automatically terminates the string that is read with a null character.

We can also specify the field width using the form %ws in the scanf statement for reading a specified number of characters from the input string.

eg: char name[10];
    scanf("%5s", name);

Reading a line of Text:

C supports a format specification known as the edit set conversion code %[..] that can be used to read a line containing a variety of characters, including white spaces.

eg: char line[80];
    scanf("%[^\n]", line);
    printf("%s", line);

Putting strings together:

The process of combining two strings together is called concatenation.

String Handling Functions:

C library supports a large number of string ~~manipulation~~ -handling functions that can be used to carry out many of the string manipulations.

strcat() ⇒ Concatenates two strings.
strcmp() ⇒ Compare two strings.
strcpy() ⇒ Copies one string over another.
strlen() ⇒ Finds the length of a string.

Strcat()

The strcat() function joins two strings together.

strcat (string 1, string2);

Strcmp()

The strcmp function compares two strings identified by the arguments and has a value '0' if they are equal.

strcmp(string 1, string2);

Strcpy() .

The strcpy function works almost like a string-assignment operator.

strcpy (string 1, string 2);

Strlen()

The function counts and returns the number of characters in a string.

n=strlen(string);

where, n is an integer value.

Strncpy()

The strncpy() function that copies only the left-most 'n' characters of the source string to the target string variable.

strncpy (s1,s2, 5);

Strrev()

The strrev() functions reverses the given string as its argument.

strstr()

It is a two-parameter function that can be used to locate a sub-string in a string.

```
strstr(s1,s2);
strstr(s1,"ABC");
```

1. Concatenation Program:

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

2. Length of a String.

```
#include<string.h> #include<stdio.h>
  int main() {
       char str3[]="Askd".
        int length = strlen (str3);
        printf(" Length of the string:%d\n",
                                   length);

        return 0;
     }
```

19:17 pm = 7:17pm

Mon-22-Jan-2024