Basudev Godabari Degree College,Kesaibahal

Department of Computer Science

"SELF STUDY MODULE"

Module Details :-

- **Class – 1st Semester (2020-21) BATCH**
- **Subject Name : Computer Science**
- **Paper Name** : Programming Using "C"

UNIT-2 : Structure

2.1 Introduction to Loops.
2.2 The WHILE Statement
2.3 The DO Statement
2.4 The DO'WHILE Statement
2.5 The FOR Statement
2.6 Introduction To Array
2.7 Declaration of array
2.8 Simple Example of Array Program
2.9 Types of Array: - (i) One Dimensional Array (ii) Multidimensional Array.
2.10 Limitation of Array.
2.11 Introduction to Pointer.
2.12 Types of Pointer (null Pointer, wild Pointer, dangling pointer, generic Pointer)
2.13 Declaring a Pointer.
2.14 Accessing Variable using Pointer
2.15 Simple Example Of Pointer Program

Learning Objectives :

After Learning this unit you should be able to

- Know the Loops and there types
- Understand how it is used in C Language
- Know the Array Concept.
- Differentiate between One Dimensional Array & Multidimensional Array.
- Write C program using Array.
- Know the Concept of Pointer
- What are the different types of Pointer
- Why pointer is used.
- How a variable can be access using Pointer.

You Can Use the Following Video Link to:-
https://youtu.be/SFo205wqn8w
https://youtu.be/NfrQChcEHWs?list=RDCMUCV7cZwHMX_0vk8DSYrS7GCg
https://youtu.be/BW7PuW1V6kg
https://youtu.be/kdkcOtEU0lE
https://youtu.be/r-M3ESr8WLY
https://youtu.be/QPNRjjIaVLs
https://youtu.be/jpXspGqH4P0
https://youtu.be/Gn7PGIEJejs
https://youtu.be/xHoGGbtFZCo

You Can also use the following link to use the reference book and download it free:

https://www.pdfdrive.com/
https://www.pdfdrive.com/c-programming-language-books.html

➔ In the above site you can get the search option where you can search your book and download.

UNIT _ 2

Introduction to loops in C

This is a short article to introduce you to the concept of loops in C. Once you have an idea of what it is all about, we will go in details for each of them:

- while
- do..while
- for

Definition

A loop is a construction that allows you to execute a block of code multiple times. It consists of two parts - a condition and a body.

Condition and body

Loop condition

Every loop has a condition. This is the part that controls if the loop should continue or stop. All loops in C continue their iterations if the condition is true. In C, the condition is any valid value or expression that could be evaluated to a value. A value of 0 or '\0' (null) is considered as "false" and everything else is "true"

Body

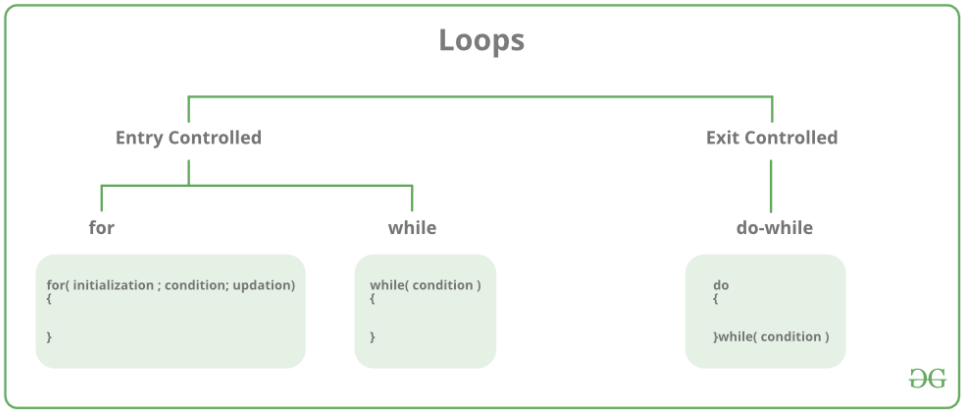
The body is one or more valid C statements. If it consists of more than one statement, the body must be enclosed in curly brackets { } .

How loops in C work

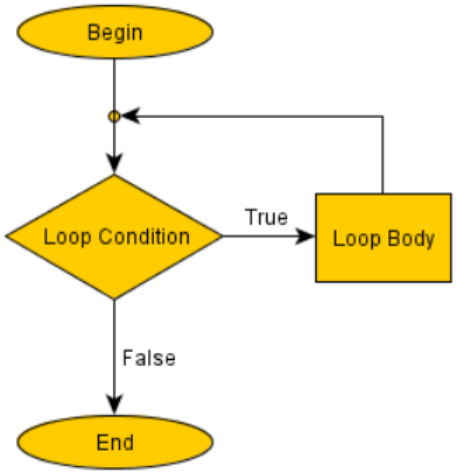
The loop starts by entering the condition. If it is "true" the body will be executed. Then the execution repeats the check of the condition and then the body. It looks like this:

begin -> condition(true)->body-> condition(true)->body ... condition(false)->end.

One check of the condition, followed by one execution of the body is one iteration.



Here is a flowchart to visualize the idea:



While Loop

While studying **for loop** we have seen that the number of iterations is known beforehand, i.e. the number of times the loop body is needed to be executed is known to us. while loops are used in situations where we do not know the exact number of iterations of loop beforehand. The loop execution is terminated on the basis of test condition.

do while loop

In do while loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is in do while loop the condition is tested at the end of loop body, i.e do while loop is exit controlled whereas the other two loops are entry controlled loops.

EXAMPLE:-

```
// C program to illustrate for loop
#include <stdio.h>

int main()
{
   int i=0;

   for (i = 1; i <= 10; i++)
   {
            printf( "Hello World\n");
   }

   return 0;
}
```

ARRAY IN C PROGRAMMING

An array is a group (or collection) of same data types. For example an int array holds the elements of int types while a float array holds the elements of float types.

Why we need Array in C Programming?

Consider a scenario where you need to find out the average of 100 integer numbers entered by user. In C, you have two ways to do this: 1) Define 100 variables with int data type and then perform 100 scanf () operations to store the entered values in the variables and then at last calculate the average of them. 2) Have a single integer array to store all the values, loop the array to store all the entered values in array and later calculate the average.

Input data into the array

Here we are **iterating the array** from 0 to 3 because the size of the array is 4. Inside the loop we are displaying a message to the user to enter the values. All the input values are stored in the corresponding array elements using scanf function.

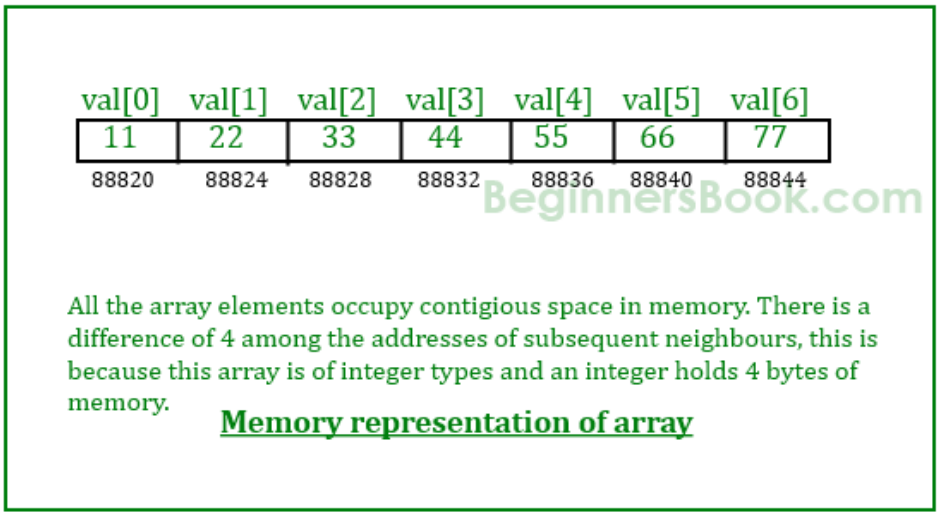
```
for (x=0; x<4;x++)
{
    printf("Enter number %d \n", (x+1));
    scanf("%d", &num[x]);
}
```

Reading out data from an array

Suppose, if we want to display the elements of the array then we can use the for loop in C like this.

```
for (x=0; x<4;x++)
{
    printf("num[%d]\n", num[x]);
}
```

C Array – Memory representation



All the array elements occupy contigious space in memory. There is a difference of 4 among the addresses of subsequent neighbours, this is because this array is of integer types and an integer holds 4 bytes of memory.

Memory representation of array

2D array – We can have multidimensional arrays in C like 2D and 3D array. However the most popular and frequently used array is 2D – two dimensional array. In this post you will learn how to declare, read and write data in 2D array along with various other features of it.

Passing an array to a function– Generally we pass values and variables while calling a function, likewise we can also pass arrays to a function. You can pass array's element as well as whole array (by just specifying the array name, which works as a pointer) to a function.

Pointer to array – Array elements can be accessed and manipulated using pointers in C. Using pointers you can easily handle array. You can have access of all the elements of an array just by assigning the array's base address to pointer variable.

Simple Two dimensional(2D) Array Example

For now don't worry how to initialize a two dimensional array, we will discuss that part later. This program demonstrates how to store the elements entered by user in a 2d array and how to display the elements of a two dimensional array.

```
#include<stdio.h>
int main(){
   /* 2D array declaration*/
   int disp[2][3];
   /*Counter variables for the loop*/
   int i, j;
   for(i=0; i<2; i++) {
      for(j=0;j<3;j++) {
         printf("Enter value for disp[%d][%d]:", i, j);
         scanf("%d", &disp[i][j]);
      }
   }
   //Displaying array elements
   printf("Two Dimensional array elements:\n");
   for(i=0; i<2; i++) {
      for(j=0;j<3;j++) {
         printf("%d ", disp[i][j]);
         if(j==2){
            printf("\n");
         }
      }
   }
   return 0;
}
```

Output:

```
Enter value for disp[0][0]:1
Enter value for disp[0][1]:2
Enter value for disp[0][2]:3
Enter value for disp[1][0]:4
Enter value for disp[1][1]:5
Enter value for disp[1][2]:6
Two Dimensional array elements:
1 2 3
4 5 6
```

Pointer in C Programming

Pointers in C are easy and fun to learn. Some C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers.

What are Pointers?

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address. The general form of a pointer variable declaration is −

```
type *var-name;
```

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk * used to declare a pointer is the same asterisk used for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Take a look at some of the valid pointer declarations −

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

How to Use Pointers?

There are a few important operations, which we will do with the help of pointers very frequently. **(a)** We define a pointer variable, **(b)** assign the address of a variable to a pointer and **(c)** finally access the value at the address available in the pointer variable. This is done by using unary operator ***** that returns the value of the variable located at the address specified by its operand.

NULL Pointers

It is always a good practice to assign a NULL value to a pointer variable in case you do not have an exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned NULL is called a **null** pointer.

The NULL pointer is a constant with a value of zero defined in several standard libraries. Consider the following program −

```
#include <stdio.h>

int main () {

   int  *ptr = NULL;

   printf("The value of ptr is : %x\n", ptr  );

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
The value of ptr is 0
```

In most of the operating systems, programs are not permitted to access memory at address 0 because that memory is reserved by the operating system. However, the memory address 0 has special significance; it signals that the pointer is not intended to point to an accessible memory location. But by convention, if a pointer contains the null (zero) value, it is assumed to point to nothing.

To check for a null pointer, you can use an 'if' statement as follows −

```
if(ptr)     /* succeeds if p is not null */
if(!ptr)    /* succeeds if p is null */
```

Pointers in Detail

Pointers have many but easy concepts and they are very important to C programming. The following important pointer concepts should be clear to any C programmer −

Sr.No.	Concept & Description
1	Pointer arithmetic There are four arithmetic operators that can be used in pointers: ++, --, +, -
2	Array of pointers You can define arrays to hold a number of pointers.
3	Pointer to pointer C allows you to have pointer on a pointer and so on.
4	Passing pointers to functions in C Passing an argument by reference or by address enable the passed argument to be changed in the calling function by the called function.
5	Return pointer from functions in C C allows a function to return a pointer to the local variable, static variable, and dynamically allocated memory as well.

Question Bank

UNIT-2

Long Question

1. What is an array? How to declare and initialize arrays? Explain with examples
2 .Explain how two dimensional arrays can be used to represent matrices.

(or)

Define an array and how the memory is allocated for a 2D array?

2. Write a program to perform matrix multiplication?
3. What are the types of looping statements available in C.

2. MARKS

1. What is an array?
2. What are the main elements of an array declaration?
3. How to initialize an array?
4. Why is it necessary to give the size of an array in an array declaration?
5. What is the difference between an array and pointer?
6. List the characteristics of Arrays.
7. What are the types of Arrays?
8. Compare arrays and structures?

PROGRAMMING WITH C – Practical Question Bank

1. Write a C program to find the sum of first 10 even numbers.
2. Write a C program to check whether a given string is a Palindrome or not.
3. Write a C program to convert temperature from degree Centigrade to Fahrenheit.
4. Write a C program to display the factorial of a given number using for loop.
5. Write a C program to display days of a week using switch-case.
6. Write a C program to print all upper case and lower case alphabets.
7. Write a C program to display Fibonacci series.
8. Write a C program to add two dimensional matrices.
9. Write a C program to search for a given element in an array usingLinear search.
10. Write a C program to find the sum of digits of a given number.

11. Write a C program to find the average marks obtained by a student taking any 3 subject marks as input.
12. Write a C program to find the largest element in a given array of elements.
13. Write a C program to reverse a given number.
14. Write a C program to check whether a given year is a Leap year or not.
15. Write a C program to find the square of a number using a user-defined function.