Contents

[**1.** **Basic Construction of C Programming Language** 3](#_Toc467828562)

[1.1 Documentation Section:- 3](#_Toc467828563)

[1.2 Linking Section:- 4](#_Toc467828564)

[1.3 Symbolic Constant Section:- 5](#_Toc467828565)

[1.4 Global Declaration Section:- 5](#_Toc467828566)

[1.5 Main Function Section:- 6](#_Toc467828567)

[1.6 User Define Function Definition:- 6](#_Toc467828568)

[**Assignment-1** 7](#_Toc467828569)

[**2.** **if / else** 8](#_Toc467828570)

[Assignment-2 10](#_Toc467828571)

[**3.** **LOOP** 12](#_Toc467828572)

[3.1 Entry Controlled Loop 12](#_Toc467828573)

[3.1.1While 13](#_Toc467828574)

[3.1.2For 14](#_Toc467828575)

[3.2 Exit Controlled Loop 16](#_Toc467828576)

[3.2.1Do-while 16](#_Toc467828577)

[3.2.2break 18](#_Toc467828578)

[3.2.3Continue 21](#_Toc467828579)

[**Assignment-3** 22](#_Toc467828580)

[**4.** **Array** 24](#_Toc467828581)

[4.1 1-D Array 25](#_Toc467828582)

[4.2 2-D Array 28](#_Toc467828584)

[**Assignment-4** 31](#_Toc467828585)

[**5** **String** 33](#_Toc467828586)

[**Assignment-5** 35](#_Toc467828587)

[**6** **Function** 37](#_Toc467828588)

[6.1 Components of Function 38](#_Toc467828589)

[6.2 Types Of Function 39](#_Toc467828590)

[6.3 Call by value and call by reference 41](#_Toc467828591)

[6.3.1Call by value 41](#_Toc467828592)

[6.3.2Call by reference 42](#_Toc467828593)

[6.3.2.1Call by reference using reference variable 43](#_Toc467828594)

[6.4 Recursion and recursive function 44](#_Toc467828595)

[**Assignment-6** 47](#_Toc467828596)

[**7** **Pointer** 48](#_Toc467828597)

[Assignment-7 49](#_Toc467828598)

[**8** **Structure** 50](#_Toc467828599)

[Assignment-8 53](#_Toc467828600)

[**9** **Union** 54](#_Toc467828601)

# 

# Basic Construction of C Programming Language

Program construction of C program should include the following segments:-

* 1. **Documentation Section**
  2. **Linking Section**
  3. **Symbolic Constant Section or Macro Section**
  4. **Global declaration Section**
  5. **Main Function Section**

{

* 1. Declaration Section
  2. Executable Section

}

* 1. **User Define Function Definition Section**

# Documentation Section:-

It contains detail description of the problem, the programmer and the logic that means user can write details about the problem, can describe detail information about the programmer. This section can also contain detail algorithm of the problem. Documentation section generally present within comment. Comments are non executable statement present in C program to enhance the understandability and readability of C program. Documentation Section is optional. User may or may not write the documentation section in C program.

**For example**

Write a program to find area & perimeter of a circle.

The documentation of above problem should be: -

Step 1: Start

Step 2: Input radius of the circle

Step 3: Calculate

Area=3.14\*r\*r

Perimeter=2\*3.14\*r

Step 4: Display Area and Perimeter

Step 5: End

# Linking Section:-

After compilation header files are linked with the object program to create object program. Header file contains declaration and definitions of system define function. Function is the collection of statement to perform a particular task which must have a particular name is called Function. According to the time of creation of function, they are divided in two categories-

1. **System Define Function**
2. **User Define Function**

System define function are those functions which are created by the system developer at the time of system creation or, compiler development and store categoriesely under the header files. System Define functions are also called Compiler Define Function or, Library Function. Alternately, we can say that system defined function are those functions which are declared and defined by system developer and used by the user. A function has three place of existence-

1. **Declaration or, Prototype**
2. **Definition**
3. **Calling or, Use**

Header file contains the declaration and definitions of system define function. So, when user used any system define function then the header file which contains the declaration and definition of the function must include with the user program. Linking Section is used to link the user program with system define function for execution. If linking is not present then system define function cannot work properly, a linking error is arise. That means when we use any system define function then the header file of the function must be linked with the user program using the linking statement. Linking is established using

**# include <header filename> or, # include “header filename”.**

User define function are those type of function which are developed, defined and used by the user as the requirement of the user.

# Symbolic Constant Section:-

Constants are those type of entities that’s value cannot be changed during program execution. Value of variable is also called Constant.

Symbolic Constant is that type of constant which are temporary in nature and can be used in program in many positions. Any temporary constant can be declared in C programming language using the concept of symbolic constant as-

**# define Max 100**

The above statement creates a symbolic constant Max that’s value is 10 and can be used in any position in a C program but cannot be changed the value of the constant. Symbolic constant are generally used in those situation where a particular constant is used repeatedly in a C program. Symbolic constants are used to increase the debugging process of a C program.

User can store statements under a particular names, those type of name is called Macro. Macro is an identifier that can store collection of statement to perform some particular task and which is repeatedly used in a C program. When a macro is called then the control is not transferred to the statement present in the macro. The statements are inserted at the calling position of the macro.

**For example**

**# define pf**

**printf(“\n Enter a no.”);**

**Scanf(“%d”,&a)**

# Global Declaration Section:-

Variables are those type of elements that value can be changed during program execution. Alternately, we can say that variable are the name of memory location that’s value can be changed during program execution. According to the scope or visibility, variables are divided in two categories:-

* + - 1. **Global Variable**
      2. **Local Variable**

Global variables are those type of variables that’s value can be accessed in any position of the entire program. Global variables are also called External Variable. If user wants to create any global variable then the variable should be declared in the global declaration section.

Local variable are those types of variables which are visible in a particular function in which they are declared. That means local variables are visible in a particular segment of a C program. Local variables are also called Internal Variable. Local variables are divided in three categories:-

* **Auto Variable**
* **Static Variable**
* **Register Variable**

# Main Function Section:-

It contains a main function which is the execution starting point of a C program. Every C program must contain a main function but only single copy can be present. That means a C program cannot contain more than one main function. The opening brace of main function is the logical starting point of execution and corresponding closing brace is the logical ending point of execution. Main function is divided in two code segment-

1. **Declaration Section**
2. **Executable Section**

Declaration Section contains all the variables which can be used in the main function. This section is generally used to declare all the time variables which can be used in the main function. Since, C language is a typed programming language so, every variable must be declared before their used.

Executable section contains actual segments which can produce output after execution. That means actual executable statement should be present in the executable section.

# User Define Function Definition:-

We know that a function has three place of existence:-

1. **Declaration or, Prototype of function**
2. **Calling or Use of function**
3. **Definition of the function**

User define function are those functions which are declared, defined and used by the user. Function is by default global. So, declaration of the function should be present in the global declaration section. Function cannot be local in any situation. Actually, executable statements are present in the executable section of main function. So, calling of function is performed in the executable section of main function. The definition of user define function are present after the closing brace of main function in any order. That means the definition of user defined functions can be of any order as the choice of the user.

# Assignment-1

1. Write a C program to enter two numbers and find their sum.
2. Write a C program to enter two numbers and perform all arithmetic operations.
3. Write a C program to enter length and breadth of a rectangle and find its perimeter.
4. Write a C program to enter length and breadth of a rectangle and find its area.
5. Write a C program to enter radius of a circle and find its diameter, circumference and area.
6. Write a C program to enter length in centimeter and convert it into meter and kilometer.
7. Write a C program to enter temperature in °Celsius and convert it into °Fahrenheit.
8. Write a C program to enter temperature in Fahrenheit(°F) and convert it into Celsius(°C)
9. Write a C program to convert days into years, weeks and days.
10. Write a C program to find power of any number xy (x^y).
11. Write a C program to enter any number and calculate its square root.
12. Write a C program to enter two angles of a triangle and find the third angle.
13. Write a C program to enter base and height of a triangle and find its area.
14. Write a C program to calculate area of an equilateral triangle.
15. Write a C program to enter marks of five subjects and calculate total, average and percentage.
16. Write a C program to enter P, T, R and calculate Simple Interest.
17. Write a C program to enter P, T, R and calculate Compound Interest.

# if / else

**if :-** “if” is a two-way decision making statement in which according to the condition a particular code block is executed and the other block will be ignore. “if” is a selection based statement in which depending on the condition a particular code lock will be selected and the other block will be deselected. The general syntax of “if” is-

If (Condition)

T

{

F

Code block

}

else :- “Else” is optional with if. It is not necessary to write the else block with if. If “else” is necessary then the user should write the code block. If the code block of “if” contains a single statement then opening brace ({) and closing brace (}) is not necessary to include the code block. Similar concept is applied with else. The general syntax of “else” is-

**If (condition)**

F

T

**{**

**Code block**

**}**

**Else**

**{**

**Code block**

**}**

**Example**

***To test a number which is Odd or, Even.***

**# include <stdio.h>**

**# include <conio.h>**

**void main ()**

**{**

**int n;**

**clrscr ();**

**printf ("\n Enter an Integer Number : ");**

**scanf ("%d",&n);**

**if (n%2==0)**

**{**

**printf ("\n %d is Even Number.",n);**

**}**

**Else**

**{**

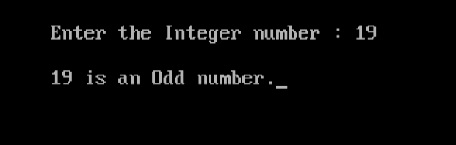
**printf ("\n %d is Odd Number.",n);**

**}**

**getch ();**

**}**

**Output**

****

# 

# Assignment-2

1. Write a C program to find maximum between two numbers.
2. Write a C program to find maximum between three numbers.
3. Write a C program to check whether a number is even or odd.
4. Write a C program to check whether a year is leap year or not.
5. Write a C program to check whether a number is negative, positive or zero.
6. Write a C program to check whether a number is divisible by 5 and 11 or not.
7. Write a C program to count total number of notes in given amount.
8. Write a C program to check whether a character is alphabet or not.
9. Write a C program to input any alphabet and check whether it is vowel or consonant.
10. Write a C program to input any character and check whether it is alphabet, digit or special character.
11. Write a C program to check whether a character is uppercase or lowercase alphabet.
12. Write a C program to input week number and print week day.
13. Write a C program to input month number and print number of days in that month.
14. Write a C program to input angles of a triangle and check whether triangle is valid or not.
15. Write a C program to input all sides of a triangle and check whether triangle is valid or not.
16. Write a C program to check whether the triangle is equilateral, isosceles or scalene triangle.
17. Write a C program to find all roots of a quadratic equation.
18. Write a C program to calculate profit or loss.
19. Write a C program to input marks of five subjects Physics, Chemistry, Biology, Mathematics and Computer. Calculate percentage and grade according to following:

|  |  |
| --- | --- |
| **Percentage** | **Grade** |
| >= 90% | A |
| >= 80 | B |
| >= 70 | C |
| >= 60 | D |
| >= 40 | E |
| <40% | F |

1. Write a C program to input basic salary of an employee and calculate its Gross salary according to following:

|  |  |  |
| --- | --- | --- |
| **Basic Salary** | **HRA** | **DA** |
| >=10000 | 20% | 80% |
| >=20000 | 25% | 90% |
| >=30000 | 30% | 95% |

1. Write a C program to input electricity unit charges and calculate total electricity bill according to the given condition:

|  |  |
| --- | --- |
| **Units** | **Amount (per unit)** |
| First - 50 | 0.50 |
| Next -100 | 0.75 |
| Next – 100 | 1.20 |
| Above - 250 | 1.50 |

An additional surcharge of 20% is added to the bill

# LOOP

Loop is a program construction by which a particular code block will be executed depending on some condition repeatedly. That means Loop is a program construction by which a code block can be repeatedly executed. Generally a loop contains two segments:-

1. **Condition**
2. **Code Block**

According to the position of the condition loops are divided in two categories-

1. **Entry Controlled Loop**
2. **Exit Controlled Loop**

# Entry Controlled Loop

Entry Controlled Loop is those type of looping construction in which before execution of code block condition must be tested. If the condition is true then the code block will be executed and again test the condition. This process will be continued until the condition is false. If the condition is false then the control goes to the immediate next statement after the code block. In C programming language entry controlled loop is divided in two categories:-

* + - 1. **While**
      2. **For**

**Entry Controlled Loop can be designed using the flowchart as-**

**Code Block**

**Condition**

# While

While is the simplest looping construction present in C programming language. While is an Entry Controlled Loop that mans before execution of code block, condition must be tested. If the condition is true then the code block will be executed and again test the condition. This process will continue till the condition is true. If the condition is false then the control goes to the immediate next statement after the code block.

The general syntax of While is-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| **Initialization;**  **while( condition)**  **{**  **Statement/code;**  **Increament/decreament;**  **}** | **i=1;**  **while (i<=10)**  **{**  **Printf (“\n A.K.Singh”);**  **i++;**  **}** |

**Note:-**While loop is appropriate used in those situation where the starting point of the loop and the ending point of the loop is present but the increment and decrement is not present.

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

int i,num;

clrscr();

printf(" Enter the number to print (num): ");

scanf("%d",&num);

i=1;

while(i<=num)

{

printf("%5d",i);

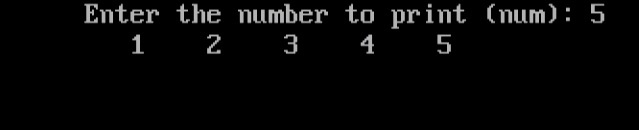
i++;

}

getch();

}

**Output**

****

# For

For is also an Entry Controlled Loop that means before execution of code block condition must be tested. If the condition is true then the code lock will be executed and again test the condition. This process will continue until the condition is false. If the condition is false then the control goes to the immediate next statement after the code block. If the code block contains a single statement then the opening brace and the closing brace is not necessary to inclose the code block. The general syntax of For is:-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| **for(Initialization; Condition; increament/decreament)**  **{**  T  **Statement/code;**  F  **}** | **for (i=1;i<=10;i++)**  **{**  **Printf(“\n A.M.COLLEGE”);**  **}** |

The above code segment will display the name “**A.M.COLLEGE**” 10mtimes.

**Note: -** for loop is appropriately used to handle those type of situation in which initialization, condition and increment/decrement is given. Alternately, we can say that for loop is appropriate to handle Arithmetic Progression.

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

int i,num,t;

clrscr();

printf(" Enter the number to find table (num): ");

scanf("%d",&num);

for(i=1;i<=10;i++)

{

t=i\*num;

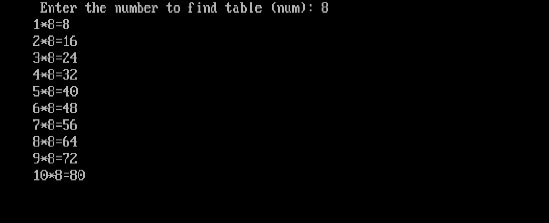
printf("%d\*%d=%d\n",i,num,t);

}

getch();

}

**Output**

****

# Exit Controlled Loop

Exit control loop is those type of looping construction in which after execution of code block condition must be tested if the condition is true then the code block will executed again. This process will continue until the condition is false. When condition is false then the control goes to the immediate next statement after the condition.

The flowchart of Exit Controlled Loop is:-

Code Block

Condition

In C programming language only one loop is present to implement the concept of exit controlled loop. In C language do-while is present to implement the concept of exit controlled loop.

# Do-while

Do-while is an exit controlled loop present in C programming language. That means after execution of code-block condition must be tested. If the condition is true then again execute the code block. This process is continuing till the condition is true. If the condition is false then control goes to immediate next statement after the condition.

The general syntax and example is given in next page:-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| Initialization;  Do  {  Statement/code;  Increament/decreament;  }while(condition); | i=1;  do  {  printf(“\nArif Sir”);  i++;  }while(i<=10); |

The code segment will display the name “**Arif Sir**” ten times.

In do-while initialization and increment/decrement is optional. It is used in those type of situation in which user can interact with the loop in every itaration. It is used to handle that type of situation where the staring point and increment/decrement is known but the ending is unknown. It means Do-while is used to handle infinitely executable loop.

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

char ch;

clrscr();

printf(" \n Alphabate of English-\n ");

ch=65;

do

{

printf("%5c",ch);

ch++;

}while(ch!=91);

getch();

}

**Output**

****

# break

Break is a keyword which is used to terminate loops or, switch. Break statement must be used within a loop or in switch. Break statement cannot be used without loop or switch.

|  |  |
| --- | --- |
| **Using switch method** | **Using do-while method** |
| switch(choice)  {  case 1: Statement1;  ------------;  Break;  case 2: Statement2;  ------------;  Break;  case n: Statement n;  ------------;  Break;  default: Statement;  ------------;  } | Do  {  ---------- ;  ---------- ;  Break;  ---------- ;  ---------- ;  }while (condition); |
| **Using for method** | **Using while method** |
| For (condition)  {  ------------- ;  ------------- ;  Break;  ------------- ;  ------------- ;  } | While(condition)  {  ---------------  ---------------  Break;  ---------------  ---------------  } |

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

int ch;

clrscr();

printf("\n\*\*\*\*\*\*\*\*\*\* Menu \*\*\*\*\*\*\*\*\*\n");

printf("\t1.\t Sunday\n");

printf("\t2.\t Monday\n");

printf("\t3.\t Tuesday\n");

printf("\t4.\t Wednesday\n");

printf("\t5.\t Therseday\n");

printf("\t6.\t Friday\n");

printf("\t7.\t Saterday\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nEnter the choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\tSunday\n");

break;

case 2: printf("\t Monday\n");

break;

case 3: printf("\t Tuesday\n");

break;

case 4: printf("\t Wednesday\n");

break;

case 5: printf("\t Therseday\n");

break;

case 6: printf("\t Friday\n");

break;

case 7: printf("\t Saterday\n");

break;

default:printf("Invalid choice.");

}

getch();

}

**Output**



# Continue

Continue is also a keyword present in C programming language which is used to transfer the control from loop body to the next iteration of the loop. Continue must be used within loop. Without loop continue cannot be used. Example-

**Using while method: -**

while (condition)

{

-----------

continue;

-----------

}

**Using for method: -**

for (initialization; condition; incr/decr)

{

----------

continue;

------------

}

**Using do-while method: -**

Do

{

------------

continue;

------------

}while(condition);

# Assignment-3

1. Write a C program to print all natural numbers from 1 to n. - using while loop
2. Write a C program to print all natural numbers in reverse (from n to 1). - using while loop
3. Write a C program to print all alphabets from a to z. - using while loop
4. Write a C program to print all even numbers between 1 to 100 - using while loop
5. Write a C program to print all odd number between 1 to 100
6. Write a C program to print sum of all even numbers between 1 to n.
7. Write a C program to print sum of all odd numbers between 1 to n.
8. Write a C program to print table of any number.
9. Write a C program to enter any number and calculate sum of all natural numbers between 1 to n.
10. Write a C program to find first and last digit of any number.
11. Write a C program to count number of digits in any number.
12. Write a C program to calculate sum of digits of any number.
13. Write a C program to calculate product of digits of any number.
14. Write a C program to swap first and last digits of any number.
15. Write a C program to find sum of first and last digit of any number.
16. Write a C program to enter any number and print its reverse.
17. Write a C program to enter any number and check whether the number is palindrome or not.
18. Write a C program to find frequency of each digit in a given integer.
19. Write a C program to enter any number and print it in words.
20. Write a C program to print all ASCII character with their values.
21. Write a C program to find power of any number using for loop.
22. Write a C program to enter any number and print all factors of the number.
23. Write a C program to enter any number and calculate its factorial.
24. Write a C program to find HCF (GCD) of two numbers.
25. Write a C program to find LCM of two numbers.
26. Write a C program to check whether a number is Prime number or not.
27. Write a C program to check whether a number is Armstrong number or not.
28. Write a C program to check whether a number is Perfect number or not.
29. Write a C program to check whether a number is Strong number or not.
30. Write a C program to print all Prime numbers between 1 to n.
31. Write a C program to print all Armstrong numbers between 1 to n.
32. Write a C program to print all Perfect numbers between 1 to n.
33. Write a C program to print all Strong numbers between 1 to n.
34. Write a C program to enter any number and print its prime factors.
35. Write a C program to find sum of all prime numbers between 1 to n.
36. Write a C program to print Fibonacci series up to n terms.
37. Write a C program to find one's complement of a binary number.
38. Write a C program to find two's complement of a binary number.
39. Write a C program to convert Binary to Octal number system.
40. Write a C program to convert Binary to Decimal number system.
41. Write a C program to convert Binary to Hexadecimal number system.
42. Write a C program to convert Octal to Binary number system.
43. Write a C program to convert Octal to Decimal number system.
44. Write a C program to convert Octal to Hexadecimal number system.
45. Write a C program to convert Decimal to Binary number system.
46. Write a C program to convert Decimal to Octal number system.
47. Write a C program to convert Decimal to Hexadecimal number system.
48. Write a C program to convert Hexadecimal to Binary number system.
49. Write a C program to convert Hexadecimal to Octal number system.
50. Write a C program to convert Hexadecimal to Decimal number system.
51. Write a C program to print Pascal triangle up to n rows.
52. Star pattern programs - Write a C program to print the given star patterns.
53. Number pattern programs - Write a C program to print the given number patterns.
54. Write a C program to check whether an alphabet is vowel or consonant using switch case.
55. Write a C program to print day of week name using switch case.
56. Write a C program print total number of days in a month using switch case.
57. Write a C program to find maximum between two numbers using switch case.
58. Write a C program to check whether a number is even or odd using switch case.
59. Write a C program to find roots of a quadratic equation using switch case.
60. Write a C program to create Simple Calculator using switch case.

# Array

Collection of similar type of element in a linear form is called Array. Alternately, we can define array is a common name shared by various variable of same type and allocates contiguous memory space in the primary memory RAM. Array is also called vector or subscripted variable or, index variable. The index number is start from o & up to size-1 of the array. Variable present in the array are represented using the name of the array and the subscript of the block. So, it is also called Subscripted variable.

If an array contains a single row then it is called Single Dimensional Array or, Single Subscripted Variable.

If the array contains more then one row then it is called Two-Dimensional Array.

That means collection of single dimensional array is called Two-Dimensional Array. Two-Dimensional Array is also called Double Subscripted Variable. The first subscript indicates the row and the second subscript indicates the column. Two-Dimensional Array is also called Matrix in mathematics, Table in accounting. Collection of TWO-Dimensional Array is called Three-Dimensional Array or, Triple Subscripted Variable. The first subscript indicates the height, second subscript indicates the row and the third subscript indicates the column. In C language maximum 15 dimensional array is allowed. C compiler is not capable to handle more than 15 dimensional array.

**Advantages of Array**

The following **advantages** can be achieved using the concept of an Array

1. Using a single statement huge number of variable can be declared using the concept an Array.

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| Data\_type Array\_name[size]; | int a[100]; |

1. The above statement declares 100 integer type variable.
2. Huge number of variable can be handled very easily and efficiently using the concept of array.
3. Since, array allocates contiguous memory spaces in the primary memory RAM. So, the access time of array variable is minimum. Consiquently, the processing speed is maximum.
4. Internally the concept of array is implemented using the concept of pointer without pointer array cannot be implemented. Since, array internally is a pointer so, the execution speed of array is just four times faster than the speed of normal variable.

**Disadvantages of array**

The following **disadvantages** are present with array-

1. It is not possible to store different types of values in a particular array. Since, all the variable present in array is of same type.
2. We cannot store much number of values than the size of the array. That means, if the size of an array is n then we cannot store more than **n** different values.
3. If the size of an array is maximum(huge) then the chances of memory wastage is maximum.
4. Insertion operation is very time consuming operation due to right shifting of all elements present in the right side of the insertion point.
5. Similarly, deletion is also time consuming process due to left shift operation of all the values present in the right side of the deletion point.

# 1-D Array

General syntax to declare an array is-

|  |
| --- |
| **Syntax** |
| Data\_type Array\_name[size]; |

The data-type indicates the type of values which can be stored in the array. The data-type can be any system defined data-type like int, float, double, long double etc. It can be any derived data-type like pointer. The data can be any user-defined data-type like Structure and Union.

Array name can be any legal identifier name allowed in C-programming. That means array name must be start with an alphabet or underscore (\_) and can be up to 32 character of length.

In array name capital letter, small letter and mixed form characters can be used. Only one special symbol underscore (**\_**) can be used in array name, No other special symbol even space cannot be used in array name. The size of array can be any positive integer number greater than one. Generally, size of array is any positive integer greater than one but positive floating point can also used as array size but the integer part of the number is treated as the size of the array.

**For example:- int a[5];**

The above declaration declares an array of integer types of 10 block and allocates memory in the primary memory RAM as-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 00 | 01 | 02 | 03 | 04 | 05 |

# Example

#include<stdio.h>

#include<conio.h>

void main()

{

int i,array[20],size;

clrscr();

printf("\nEnter the size of array : ");

scanf("%d",&size);

for(i=0;i<=size-1;i++)

{

printf("\tarray[%d] : ",i);

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

}

printf("Element of Array : \n");

for(i=0;i<=size-1;i++)

{

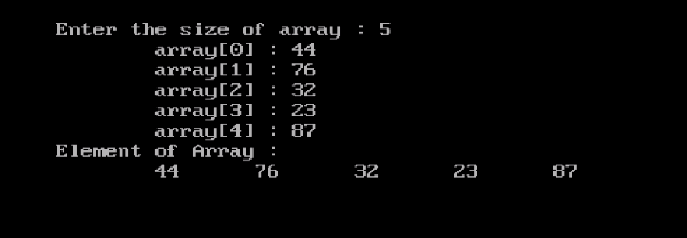
printf("\t%d",array[i]);

}

getch();

}

**Output**



# 2-D Array

Collection of single dimensional array is called Two-Dimensional Array. The first dimension indicates the row and the second indicates the column. Two-Dimensional Array is also called Double-Subscripted variable because every block of a Two-Dimensional array is represented using two subscripts. The first subscript represents the row and the second subscript represents the column. In mathematics Two-Dimensional array is called Matrix and in physics it is called as Two-Dimensional Vector. The main purpose of Two-Dimensional array is to handle table data (the data stored in tubular format).

The general syntax to declare Two-Dimensional Array is-

**Data-type array-name[rows][columns];**

The Data-type indicates the type of values which can be stored in the array. In C programming language can be any standard data-type or, any user defined data-type like Structure and Union.

Array name can be any legal identifier name allowed in C programming language. The rows must be any positive integer number greater than or equal to one. Similarly, columns must be any positive number greater than or equal to one but both the values cannot be one at a time.

**For example:-**

int A[3][4]

The above declaration declares a Two-Dimensional Array of three rows and four columns which depicted (picturised form) as:-

A 0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 01 | 02 | 03 |
| 10 | 11 | 12 | 13 |
| 20 | 21 | 22 | 23 |

0 1

2

The subscript is written in the blocks. It rows and columns are equal then the matrix is called Square Matrix. In the above graphical representation we can see that a row contains that number of elements equal to the size of the column. Similarly, a column contains that number of elements equal to the row of the matrix. It is also noticed that row wise second subscript is incremented and column wise. The first subscript is increased. The memory representation of the above Two-Dimensional array is-

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

int m[3][3],i,j;

clrscr();

printf("\nEnter the element in Metrix : \n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("\tm[%d] : ",i);

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

}

}

printf("Element of Array m[3][3]: \n");

for(i=0;i<3;i++)

{

printf("\t\t");

for(j=0;j<3;j++)

{

printf("\t%d",m[i][j]);

}

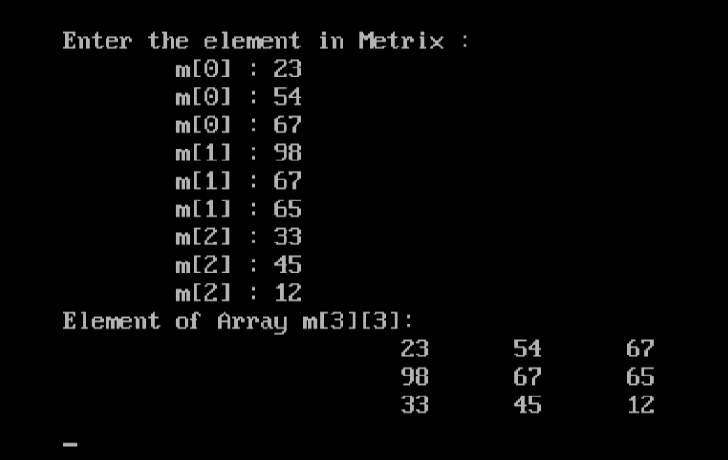
printf("\n");

}

getch();

}

**Output**



# Assignment-4

1. Write a C program to read and print elements of array. - using recursion.
2. Write a C program to find sum of all array elements. - using recursion.
3. Write a C program to find maximum and minimum element in an array. - using recursion.
4. Write a C program to find second largest element in an array.
5. Write a C program to copy all elements from an array to another array.
6. Write a C program to insert an element in an array.
7. Write a C program to delete an element from an array at specified position.
8. Write a C program to print all unique elements in the array.
9. Write a C program to print all negative elements in an array.
10. Write a C program to count total number of even and odd elements in an array.
11. Write a C program to count total number of negative elements in an array.
12. Write a C program to count total number of duplicate elements in an array.
13. Write a C program to delete all duplicate elements from an array.
14. Write a C program to count frequency of each element in an array.
15. Write a C program to merge two array to third array.
16. Write a C program to find reverse of an array.
17. Write a C program to put even and odd elements of array in two separate array
18. Write a C program to search an element in an array.
19. Write a C program to sort array elements in ascending order.
20. Write a C program to sort array elements in descending order.
21. Write a C program to sort even and odd elements of array separately.
22. Write a C program to add two matrices.
23. Write a C program to subtract two matrices.
24. Write a C program to perform Scalar matrix multiplication.
25. Write a C program to multiply two matrices.
26. Write a C program to check whether two matrices are equal or not.
27. Write a C program to find sum of main diagonal elements of a matrix.
28. Write a C program to find sum of minor diagonal elements of a matrix.
29. Write a C program to find sum of each row and column of a matrix.
30. Write a C program to interchange diagonals of a matrix.
31. Write a C program to find upper triangular matrix.
32. Write a C program to find lower triangular matrix.
33. Write a C program to find sum of upper triangular matrix.
34. Write a C program to find transpose of a matrix.
35. Write a C program to find determinant of a matrix.
36. Write a C program to check Identity matrix.
37. Write a C program to check Sparse matrix.
38. Write a C program to check Symmetric matrix.

# String

Array of character is called String. That means linear collection of character in linear form is called String. String value must be enclosed with double quotes and terminated by a null character generally supplied by the compiler. The null character indicates the end of the string. String variable must be a character type. If a string variable contains a single row then it is called Single Dimensional String. If the string contains more than one row then it is called Two-Dimensional String.

The general syntax to declare string variable of single dimension is-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| char variable\_name[size]; | char A[15]; |

The above declaration declares a string variable A of 135 character size. In C language the variable can store 135 character excluding null character but in C++ the variable can store 134 characters excluding null character.

**Note:** - In C &C++ bound checking of string variable is not present. So, string variable can store more number of characters than the size of the variable but it is situation dependent. The general syntax to declare two-dimensional string variable is-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| char varname[size of row][size of column]; | char A[135][8]; |

In the above declaration declares a two dimensional string variable A of **135** row and **8** columns. That means each row can store 8 characters in C language.

**Example**

#include<stdio.h>

#include<conio.h>

void main()

{

char A[15];

char str[7]={'S','h','a','h','i','l','\0'};

clrscr();

printf("Enter the string:");

scanf("%s",&A);

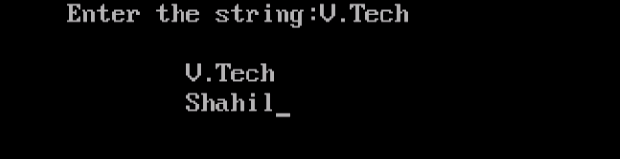
printf("\n\t%s",A);

printf("\n\t%s",str);

getch();

}

**Output**



# Assignment-5

1. Write a C program to find length of a string.
2. Write a C program to copy one string to another string.
3. Write a C program to concatenate two strings.
4. Write a C program to compare two strings.
5. Write a C program to convert lowercase string to uppercase.
6. Write a C program to convert uppercase string to lowercase.
7. Write a C program to toggle case of each character of a string.
8. Write a C program to find total number of alphabets, digits or special character in a string.
9. Write a C program to count total number of vowels and consonants in a string.
10. Write a C program to count total number of words in a string.
11. Write a C program to find reverse of a string.
12. Write a C program to check whether a string is palindrome or not.
13. Write a C program to reverse order of words in a given string.
14. Write a C program to find first occurrence of a character in a given string.
15. Write a C program to find last occurrence of a character in a given string
16. Write a C program to search all occurrences of a character in given string.
17. Write a C program to count occurrences of a character in given string.
18. Write a C program to find highest frequency character in a string.
19. Write a C program to find lowest frequency character in a string.
20. Write a C program to count frequency of each character in a string.
21. Write a C program to remove first occurrence of a character from string.
22. Write a C program to remove last occurrence of a character from string.
23. Write a C program to remove all occurrences of a character from string.
24. Write a C program to remove all repeated characters from a given string.
25. Write a C program to replace first occurrence of a character with another in a string.
26. Write a C program to replace last occurrence of a character with another in a string.
27. Write a C program to replace all occurrences of a character with another in a string.
28. Write a C program to find first occurrence of a word in a given string.
29. Write a C program to find last occurrence of a word in a given string.
30. Write a C program to search all occurrences of a word in given string.
31. Write a C program to count occurrences of a word in a given string.
32. Write a C program to remove first occurrence of a word from string.
33. Write a C program to remove last occurrence of a word in given string.
34. Write a C program to remove all occurrence of a word in given string.
35. Write a C program to trim leading white space characters in a string.
36. Write a C program to trim trailing white space characters in a string.
37. Write a C program to trim both leading and trailing white space characters in a string.
38. Write a C program to remove all extra blank spaces from a given string.

# Function

Named block of statement to perform a particular task is called Function. Collection of statement which must have a particular name to perform a particular task is called Function. The main purpose of function is to re-use the existing code and easily debugging code.

We know that C is a modular programming language i.e. C language has the capability to decompose a large program to smaller segment and at the end all the segment can combined and produces the actual program. In C programming language concept of modular programming is implemented using the concept of function.

**Advantage of Function**

**The following advantages are present in function:-**

1. The concept of code re-use ability can be implemented using function i.e. a function can be used in any program as the requirement of the user without modifying the existing code.
2. Using function existing code can be re-used consequently the productivity of the programmer will be maximized and product cost as well as maintenance cost will be minimized.
3. Product cost and maintenance cost are minimized due to the concept of function. If product cost and maintenance are minimum then the acceptability will be maximum.
4. Using function existing code can be re-used by creating users own header files, which also hides the existing code but can perform particular task as the requirement of the user. Using the concept of header file security of code will be increased.

**Disadvantages of function**

**The following disadvantages are present with function-**

1. Understandability of program is decreased because actual code is present in another file which actually produced the required output which is not visible to the user.
2. Execution speed of program is decreased due to fluctuations of control bellowing Calling Function and Called Function.
3. Space complexity of a program is increased due to reallocation of memory at the time of execution. This also noticed that the size of a program is increased using the concept of function. So, the execution speed will be decreased.

# Components of Function

The general representation is-

**Y= f(x)**

Where, x is the argument or parameter, f is the function name and y is the return value. So, any programming language a function has three components

1. **Function name**
2. **Argument**
3. **Return value or return type**

Function name is any legal identifier allowed in C programming language. The argument is the initial values which must be supplied by the calling function to the called function at the time of activation of the called function. The function which calls another function is called Calling Function and the function which is called by another function is known as Called Function. The arguments supplied by the calling function to the called function at the time of calling. A function can take any number of argument and any type of argument allowed in C programming language. The return type indicates the type of value which should be return by the called function to the calling function after execution of the called function. The return type can be any data type allowed in C programming language.

# Types Of Function

We know that function has three components-Function name, argument and returned type. It is also known to us that function name is fixed for a particular function but argument and return type can be changed. According to the argument and return type function is divided in four categories-

1. **With return with argument**
2. **With return without argument**
3. **Without return with argument**
4. **Without return without argument**

A function has three place of existence that means a function is present in three locations in program-

1. **Declaration or, Prototype**
2. **Calling or Use**
3. **Definition or Body**

The declaration or prototype of a function declares the name of the function and the type of arguments separated by comma and the return type of the returned value. The declaration or prototype is present in the global declaration section of the program because functions are global by default. Functions cannot be local in any situation in C programming language. The prototype can also be present in the declaration section of main function, but in this situation function is also global. The calling or use is the actual invocation point of the called function by the calling function.

In this position actual arguments are supplied by the calling function to the called function. After that, control goes to the function definition to process the actual task. A function may be called many times in any location of another function or, in the same function as the requirement of the user.

The function definition is the actual code block to perform the task specified by the user. Generally the function definition or body present after the closing brace of main function.

If a program contains more than one function then the function definition can be present in any order as the choice of the user. A program can contain any number of functions and can be called in any order and can be declare in any order. A particular function can return a single value after execution to the calling function more than one function cannot be returned by called function to the calling function.

According to the declaration and definition functions are divided in two categories:-

1. **System Defined Function**
2. **User Defined Function**

System Defined Functions are that type of function which are declared and defined by the system developers at the time of system development and stored in compiled form under header files and used by the users as the requirement of the user. The container of system defined function group wise is called Header File.

**For example**

Math.h contains all mathematical functions, graphics.h contains all graphical functions etc. In C programming language total header file ranging between 24 and 29 in previous version (Turboc2) but in **Turboc3** total number of header files are ranging between **100** and **105**. The latest version of **Turbo C** contains **167** header files.

The functions which are declared, defined and used by the user as the requirement of the user are called User-Defined Function. User can also create his own header file by storing all the user defined function under specific name and can be used by user as same as header file as the requirement of the user to implement the code re-use ability.

# Call by value and call by reference

A function which calls another function is called Calling Function and the function which is called by another function is called Function. Arguments are supplied by the calling function to the call function at the time of use of the called function. Return value is returned by the called function to the calling function after execution to the called function. We know that arguments are supplied by the calling function to the called function at the time of calling of the called function. The calling system of called function is divided in two categories-

1. **Call by value**
2. **Call by reference**

# Call by value

***Call by value*** calling system is that type of calling system in which calling function supplies the argument to the called function by supplying values of sum variables as actual argument. The actual arguments are copied into the formal arguments or, dummy arguments of the called function and the dummy arguments must be identical with type but not necessary same named. In call by value call system actual and formal arguments allocate distinct memory and they are not linked internally. So, any changing of dummy variable cannot effect on actual argument. Similarly, any changing of actual argument cannot effect on dummy argument. That means in call by value calling system no internal linking is established between the calling function and the called function.

**For example**

**#include<stdio.h>**

**#include<conio.h>**

**void interchange(int,int);**

**void main ()**

**{**

**int a,b;**

**clrscr ();**

**printf ("\n Enter value of a : ");**

**scanf ("%d",&a);**

**printf ("\n Enter value of b : ");**

**scanf ("%d",&b);**

**printf("\n Before interchange : a=%d b=%d",a,b);**

**interchange (a,b);**

**getch ();**

**}**

**void interchange (int x, int y)**

**{**

**int t;**

**t=x;**

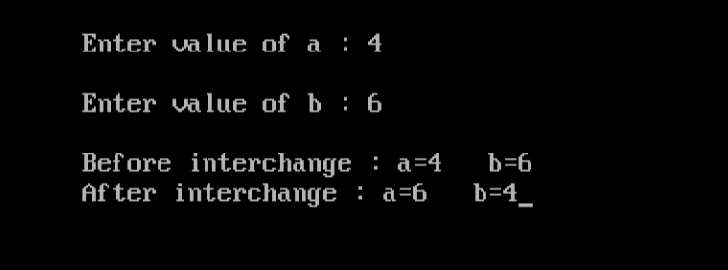
**x=y;**

**y=t;**

**printf("\n After interchange : a=%d b=%d",x,y);**

**}**

**Output**

****

The above function interchange the value of x,y which are the dummy arguments. The value of actual argument a,b are not interchanged. That means in call by value calling system called function cannot effect on actual arguments supplied by the calling function.

# Call by reference

***Call by reference*** calling system is that type of calling system in which calling function supplies arguments address of variables, not values of variables. That means in call by reference calling system actual arguments are address of variables instead of value of variables. Call by reference calling system is suitable when user wants to make an internal linking between the calling function and the called function using actual argument and dummy arguments. In call by reference calling system dummy arguments are internally linked with the actual argument using the concept of pointer and using the concept of reference variable. In call by reference calling system any changing of dummy arguments changes the values of actual arguments. Call by reference calling system is appropriately used to handle those situation where changing of dummy argument must effect on actual argument. Call by reference calling system is implemented using the concept of reference variable and using the concept of pointer.

# Call by reference using reference variable

Reference variables are those types of variable which allocates their memory in the address of another variable that means same memory location different name is called reference variable. Reference variable is also called Alias. Reference variable must be initialized with same variables not values. Reference variable allocates the memory in the same location of the address of the variable by which it initialized.

**For example**

**#include<stdio.h>**

**#include<conio.h>**

**void interchange(int \*,int \*);**

**void main ()**

**{**

**int a,b;**

**clrscr ();**

**printf ("\n Enter value of a : ");**

**scanf ("%d,&a);**

**printf ("\n Enter value of b : ");**

**scanf ("%d",&b);**

**printf("\nValues before adding 10 :\n\ta=%d b=%d",a,b);**

**interchange (&a, &b);**

**getch ();**

**}**

**void interchange (int \*x, int \*y){**

**int t;**

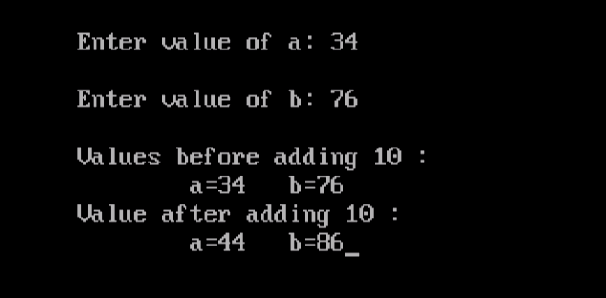
**\*x=\*x+10;**

**\*y=\*y+10;**

**printf("\n Value after adding 10 : a=%d b=%d",\*x,\*y);**

**}**

**Output**



Call by reference calling system can also be implemented using pointer. Pointer is that type of variable which contains address of variable instead of value of variable. That means pointer variable holds address of another variable by which the pointer can access the variable. Pointer variables are internally linked with the variable that the address is stored in the pointer. In C programming language to handle pointer only two operator are used &,\*.

**&- Address of Operator**

**\*- Value of Operator**

# Recursion and recursive function

The function which calls another function is known as Calling Function and the function which is called by another function is known as Called Function. Recursion is those type of calling technique or, program construction in which a function calls itself. That means recursion is those type of program construction in which the calling function and the called function are same. The function which is used to implement the concept of recursion is known as Recursive Function. Recursion is divided in two categories:-

1. **Direct Recursion**
2. **Indirect Recursion**

Direct Recursion is those type of program construction in which the calling function and the called function are same as-

F1()

{

-----------------;

-----------------;

F1();

-----------------;

-----------------;

}

Indirect recursion is those type of program construction in which the cycling is implemented via another function as-

F1(){

-----------------;

-----------------;

F2();

-----------------;

------------------;

}

F2(){

----------------;

----------------;

F1();

-----------------;

-----------------;

}

**Example**

#include<stdio.h>

#include<conio.h>

int fact(int);

void main()

{

int n,f,i;

clrscr ();

printf ("\n Enter the number for factorial: ");

scanf ("%d",&n);

printf("\n %d!=1",n);

for(i=2;i<=n;i++)

{

printf("\*%d",i);

}

f=fact(n);

printf("=%d",f);

getch ();

}

int fact(int num )

{

if(num==0)

{

return 1;

}

else

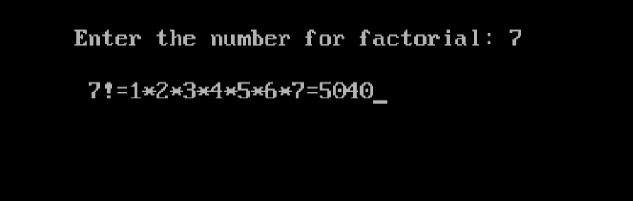
{

return num\*fact(num-1);

}

}

**Output**



# Assignment-6

1. Write a C program to find cube of any number using function.
2. Write a C program to find diameter, circumference and area of circle using functions.
3. Write a C program to find maximum and minimum between two numbers using functions.
4. Write a C program to check whether a number is even or odd using functions.
5. Write a C program to check whether a number is prime, Armstrong or perfect number using functions.
6. Write a C program to find all prime numbers between given interval
7. Write a C program to print all strong numbers between given interval using functions.
8. Write a C program to print all armstrong numbers between given interval using functions.
9. Write a C program to print all perfect numbers between given interval using functions.
10. Write a C program to find power of any number using recursion.
11. Write a C program to print all natural numbers between 1 to n using recursion.
12. Write a C program to print all even or odd numbers in given range using recursion.
13. Write a C program to find sum of all natural numbers between 1 to n using recursion.
14. Write a C program to find sum of all even or odd numbers in given range using recursion.
15. Write a C program to find reverse of any number using recursion.
16. Write a C program to check whether a number is palindrome or not using recursion.
17. Write a C program to find sum of digits of a given number using recursion.
18. Write a C program to find factorial of any number using recursion.
19. Write a C program to generate nth Fibonacci term using recursion.
20. Write a C program to find GCD (HCF) of two numbers using recursion.
21. Write a C program to find LCM of two numbers using recursion.
22. Write a C program to display all array elements using recursion.
23. Write a C program to find sum of elements of array using recursion.
24. Write a C program to find maximum and minimum elements in array using recursion.

# Pointer

Pointer is a type of variable which can store address of another variable. That means pointer is those type of variable which is used to point other variable by storing the address of other variable in it. A pointer can store address of simple variable, address of single pointer, address of double pointer and so on. Pointer variable is declared as same as simple variable but star symbol must be present before the name of the pointer, which indicates the variable can store address of another variable of the type of the pointer.

**Advantages of pointer**

**The following advantages are present with pointer-**

* Using pointer user can access the variable that is defined outside of the function.
* The concept of array and string are purely based on pointer. Without pointer the concept of array and string cannot be implemented.
* Pointer reduces the length of a program, but the complexity of a program is increased.
* Pointer increases the execution speed of a program.
* The concept of data structure cannot be implemented without using pointer.
* The concept of external file is purely based on pointer. A file type pointer is used to interface between RAM and hard disk. Without file pointer the concept of external file cannot be implemented.

**Disadvantages of pointer**

**The following disadvantages are present with pointer-**

* Pointers increase the complexity of a program as well as decrease the understandability of a program.
* In pointer variable arithmetic operation division, multiplication, modular division etc are not allowed, only increment and decrement as well as addition and subtraction.

# Assignment-7

1. C Program to Access Elements of an Array Using Pointer
2. C Program Swap Numbers in Cyclic Order Using Call by Reference
3. C Program to Find Largest Number Using Dynamic Memory Allocation

# Structure

Structure is an user defined data type which is used to create new data type by encapsulating between different variable in a single unit as the choice of the user. That means structure is an abstract data type which is used to create new data type by abstracting different standard variable by other define variable in single unit as the choice of the user.

In C programming language structure is created using the keyword “struct” as-

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| Struct structure\_name  {  Data\_type variable [size];//in array  Data\_type varizble;  }; | Struct student  {  Char name [20];  Int roll;  }; |

In the above declaration student is the name of the structure. It is also called Tangent or, Template. Actually, structured name is new type of name. User can create that type of variable as the requirement of the user. Alternately, we can define Structure is the blue print of logical design of objects. It is also possible to say, common name of a set of object is called Structure. Components of structure not allocate their memory at the time of structure declaration. When structure type variable is created then the internal components of the structure allocates their memory and can store values. At the time of declaration of structure internal component cannot be initialized because they cannot allocate memory at the time of structure declaration. Structure variable allocates distinct memory for the entire internal component present in the structure. The following advantages are present with structure-

* Structure is the encapsulated form of objects. Due to the concept of encapsulation an extra layer is imposed around the component of structure variable so, the internal components are hidden from the outside of the world as well as abstraction is implemented.
* Abstraction is a process by which we can hide internal complexity of structure variable. So, structure variable is also called Abstract Variable.
* Since, structure is encapsulated so, a single unit is created by encapsulation different types of components. So, the single variable can move one location to another at a time.
* The concept of database management system can only be implemented using the concept of structure. Without using structure the concept of record is not possible.
* The concept of data structure cannot be implemented without using structure.
* Using the concept of structure a whole record can be supply as a single unit to an external file.

**Example**

#include<stdio.h>

#include<conio.h>

struct structure

{

int Id,Age;

char Name[40];

}s[20];

int i=0;

void input()

{

int ch;

do

{

printf("\nEnter the Id : ");

scanf("%d",&s[i].Id);

printf("Enter the Name : ");

scanf("%s",&s[i].Name);

printf("Enter the Age : ");

scanf("%d",&s[i].Age);

printf("Do you want next entry (y(1)/n(0)) : ");

scanf("%d",&ch);

i++;

}while(ch!=1);

}

void output()

{

int j;

printf("\tId\t|\tName\t|\tAge\n");

printf("\t--------|--------------------|--------------\n");

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

{

printf("\t%d\t|\t%s\t|\t%d\n",s[j].Id,s[j].Name,s[j].Age);

}

}

void main()

{

clrscr();

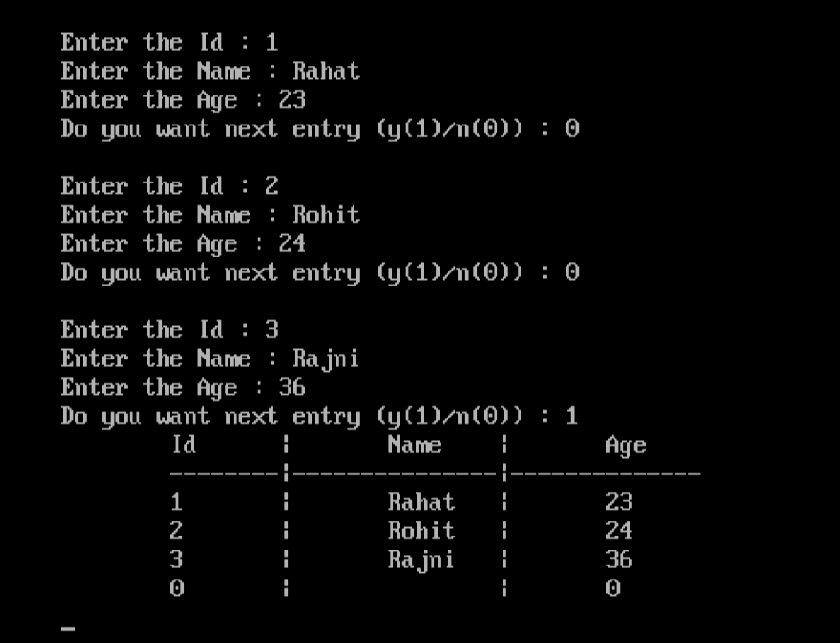
input();

output();

getch();

}

**Output**



# Assignment-8

1. C Program to Store Information(name, roll and marks) of a Student Using Structure
2. C Program to Add Two Distances (in inch-feet) System Using Structures
3. C Program to Add Two Complex Numbers by Passing Structure to a Function

# Union

Union is also a user defined data type which is used to create new data type by encapsulating standard data type and other user defined data type which are already created as the requirement of the user. The only difference between structure and union is that union allocates a common memory space for all the components present in the union which is equal to the size of the largest variable present in the union. The main purpose of union is to decrease space complexity of a program because a common memory can be used for all the variable present in the union and different type of value can be store in the memory location.

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| Union union name  {  Data type variable 1;  Data type variable 2;  }; | Union student  {  Char name [20];  Char class [10];  int year;  }; |