**C Language**

1. **What is Programming ?**

**ANS :**

Programming is the process of creating a set of instructions

that tells a computer how to perform a task.

1. **What is the history of the C language?**

**ANS :**

The C programming language was developed in 1972 by

Dennis Ritchie at Bell laboratories of AT&T (American

Telephone & Telegraph), located in the U.S.A.

* **Dennis Ritchie** is known as the **founder of the C language.**
* It was developed to overcome the problems of previous languages such as **ALGOL, BCPL** & **B**, etc.
* **Dennis Ritchie** is also known as the **Father of Programming.**
* [More information](https://www.javatpoint.com/history-of-c-language#:~:text=C%20programming%20language%20was%20developed,as%20B%2C%20BCPL%2C%20etc.)

1. **What is the importance of the C language?**

**ANS :**

* C is the mother of all programming languages.
* C is the first step to enter the programming field.
* C is POP(Procedural Oriented Programming Language).
* C is one of the most popular programming languages from 1972.

1. **What is Compiler?**

**ANS :**

Compiler is one type of translator which

convert code into machine language. (Code into Binary

language).

Compiler converts high-level language (Human

language,Programming language) to Low-level

language(Machine language,Binary language).

1. **What is an Interpreter?**

**ANS :**

Interpreter is one type of translator which

convert code into machine language. (Code into Binary

language).

Interpreter converts high-level language (Human

language,Programming language) to Low-level

language(Machine language,Binary language).

1. **Explain the difference between Compiler and Interpreter.**

**ANS :**

Compiler :

* Compiler reads the whole code as a single input.
* It’s Faster
* More Memory required
* C, C++

Interpreter:

* Interpreter read code line by line.
* It’s slower than the Compiler.0
* Less Memory required.
* Dart, Java, Python.

1. **Which Header files is used in C language**

**ANS :**

stdio.h : Standard input output Header File, Library.

conio.h : Console input output Header File, Library.

1. **What is Escape Sequence Characters? Explain in detail with example.**

**ANS :**

\n : New Line , or break the line in output.

\t : Tab space , it puts 8 spaces.

**Example** :

#include<stdio.h>

void main()

{

printf("Name\t: My name is Faculty\n");

printf("Age\t: I am 20' year old\n");

printf("School\t: I am study for Red & White Multimedia

Education\n");

}

1. **What is Data Type? Explain with examples.**

**ANS :**

**Data Type :**

* Datatype is simply a type of data.
* Which type of data we have , we can divide into same types according to their nature

Integer :

- Integer data type defined with int keyword.

- This data type considers only natural numbers.

- Ex. -2, -1, 0, 1, 2, 3,...

Float

- Float data type defined with float keyword.

- This data type considers only decimal numbers.

- Ex. -2.3, -1.98, 3.14, 78.98, 37.4,...

Character:

- Character data type defined with char keyword.

- This data type considers only characters.

- Ex. A, B, C, a, b,c,$,%...

- **Note :**  Character stores only 1 value.

1. **What is Variable? Explain with examples.**

**ANS :**

**Variable :**

* Variables are the contener which stores the value.
* Where we can store some value.

**Syntax Declare Variable :**

Datatype variableName;

**Syntax Initialize Variable :**

Datatype variableName = value;

**Example :**

int a; or int a = 10;

float pi; or float pi = 3.14;

char x; or char x = ‘A’;

1. **What is Constant Variable? Explain with examples.**

**ANS :**

**Constant Variable :**

* Constant means to fix value or expression in a variable or any other Word.
* It is used to fix the value of a variable.
* Using the ‘const’ keyword.
* Syntax :const Datatype varName = value;
* Ex. :const float pi = 3.14;

1. **What is the maximum range of each data type as per 16-bit**

**compiler?**

**ANS :**

* char: 8 bits (1 byte)
* int: 16 bits (2 bytes). Range: -32,768 to 32,767
* long: 32 bits (4 bytes). Range: -2,147,483,648 to

2,147,483,647

* float: 32 bits (4 bytes)
* double: 64 bits (8 bytes)

1. **What is Format Specifier? Explain use case of each format**

**specifier.**

**ANS :**

* It Specifies data type when we print data or get data from the user.
* **Ex.**  printf(“%d”,varName);
* %d => format specifier
* When we put the format specifier in “ “, the value will be printed there.
* **%d** or **%i :**  %d is used to specify integer value.
* **%f :** %f is used to specify float value.
* **%c :** %c is used to specify character value.

1. **Explain printf() and scanf() functions in detail.**

**ANS :**

* **printf() :** printf function is used to print messages in output.
  + - **Ex :** printf(“Hello World”);
* **scanf() :** scanf function is used to scan / get value from the user.
  + **Syntax :** scanf(“format specifier”, address of variable);
  + **EX :** scanf(“%d”,&a);

1. **Explain Keywords in C language.**

**ANS :**

Pre - reserved word in language.

- **Ex :** int , float , for , break , for , const , goto , case…

1. **What are the basic rules for creating a Variable?**

**ANS :**

- UPPERCASE

- lowercase

- camelCase => myVariable

- It can contain underscore ( \_ ) => my\_Value

- It cannot contain space.

- It cannot contain digit at start => 1Value => Error

- It can contain a digit at the middle or last => value1

- It cannot contain any symbols => #, @, &, ^, %

1. **What is Operator? Explain with its types.**

**ANS :**

**Operator :**

* Operator is the symbol which helps to perform

mathematical or logical operations between

Operands.

* Operator is used to perform an operation / process on the value.

**Note** : Operands can be either value or variable.

**Types :** 1) Unary Operator :

* Which have / want only 1 Operand.

i) increment : (++)

- Pre Increment => ++a

- Post Increment => a++

ii) decrement : (--)

- Pre decrement => --a

- Post decrement => a–

2) Binary Operator :

* Which have / want minimum 2 Operand.

i) Arithmetic Operator :

+ , - , \* , / ,%

ii) Assignment Operator :

= , += ( a=10; a+=5, Now a=15), -=,

\*=, /=, %=

iii) Conditional Operator :

==, > , >=, < , <=, !=

iv) Logical Operator :

&&, ||, !

v) Bitwise Operator :

&, |, ^, << , >>, ~

3) Ternary Operator :

(condition) ? true statement

: false statement ;

1. **Describe Operator Precedence with example.**

**ANS :**

The precedence of operators dictates the order in which the

operators will be evolved in an expression.

Priority : () = brackets

% = modules

/ , \* = left to right

+ , - = left to right

= = equals to (Right to Left)

1. **Explain Type Casting or Type Conversation.**

**ANS :**

Typecasting is a method in C language of converting one

data type to another data type.

There are two types of typecasting :

1. Implicit Conversation :

* Implicit Conversation means Directly convert.
  1. Convert int data type to char data type.
  2. Convert char data type to int data type.

1. Explicit Conversation :

* Explicit Conversation means Force fully convert.

1. Convert int data type to float data type.
2. Convert float data type to int data type.
3. **Explain types of Control Structure.**

**ANS :**

**Control Structure :**

The control statements used in the C language help a

user to specify a program control's flow.

Three types of Control Structure.

1. Selection Control Structure :

* **‘if’** Statement
* **‘if-else’** Statement
* **‘ladder if-else’** Statement
* **‘nested if-else’** Statement
* **‘switch’** Statement

1. Iteration Control Structure (Loops) :

* **‘while’** Loop
* **‘do-while’** Loop
* **‘for’** Loop

1. Jump Control Structure :

* **‘break’**
* **‘goto’**
* **‘continue’**

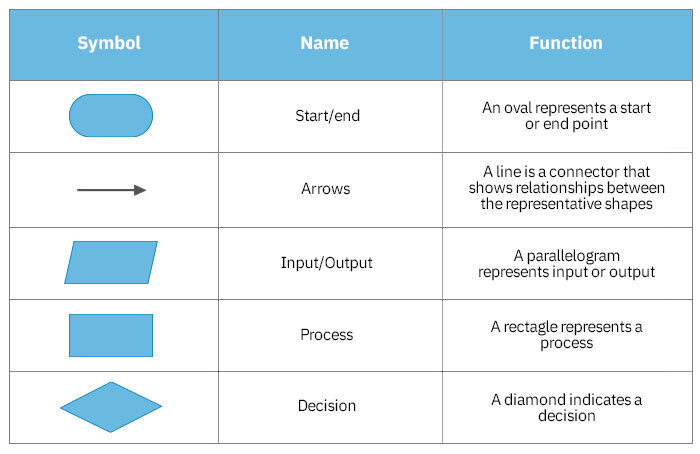
1. **What is Flowchart? Explain all shapes used in Flowchart..**

**ANS :**

* A flowchart is a GUI (Graphical user interface) reposition of

any process

* Flowchart is a blueprint of code.



1. **Explain working of ladder if else with example..**

**ANS :**

**Syntax :**

if(condition)

{

true

// statement

}

else

{

false

// statement

}

* If the given condition is true, then the code inside the if block is executed, otherwise the code inside the else block is executed.

1. **Explain working of ladder if else with example.**

**ANS :**

**Syntax :**

if(condition)

{

true

// statement

}

else if(condition)

{

true

// statement

}

else

{

false

// statement

}

In C ladder if else helps users decide from among multiple options.

1. **Explain working of nested if else with example.**

**ANS :**

Nested if-else statements are just if-else statements inside

other if-else statements to provide better decision making.

if(condition)

{

// true

if(condition)

{

// true

}

else

{

// false

}

}

else

{

//false

if(condition)

{

//true

}

else

{

//false

}

}

1. **Explain structure of ternary operator with example.**

**ANS :**

**Ternary Operator :**

* Ternary Operator is the same if else statement.
* Ternary Operator used when we want to write all conditional statement in a single line.

**Syntax :**

(condition)

? true Statement

: false Statement ;

1. Condition:

Condition representation is evaluated to either true or

false. Like a > b , a < b , a != b ..etc.

1. Expression if True:

If the condition evaluates to true, the second part of the

ternary operator, located after the ?, is executed.

1. Expression if False:

If the condition evaluates to false, the third part of the

ternary operator, located after the :, is executed.

1. **Explain structure of Switch case with example.**

**ANS :**

**Switch Case :**

switch statement is a control flow statement that allows

you to perform multi-way branching based on the value

of an expression.

When we have one input and multiple output / cases

That time we will use a switch case.

Ex: MCQ => Question -1, answer - 4

But we can select only one.

**Syntax :**

switch(variable)

{

case value1 :

// statement

break;

case value2 :

// statement

break;

…

default :

// statement

break; (Optional)

}

1. Variable :

The Variable can be integer data type or a

character data type.

1. Case Statement :

Each case represents a specific constant value

that you want to compare with the expression.

1. Break :

break statement causes the program to exit the

switch statement after executing the

corresponding case block.

1. Default Case :

When users enter value out of created cases.,for

we will print an error message in default.

1. **What is Loop? Explain types of Loops.**

**ANS :**

**Loop :**

When we want to print some line of code multiple

times , then we will use a loop to reduce code.

It means to repeat a specific code until the condition is not satisfied.

**Types of Loop :**

1) Entry Control Loop :

Which loop checks the condition in the entry of

the loop. It's called an entry control loop.

* while loop
* for loop

2) Exit Control Loop :

Which loop checks the condition at the exit of

the loop. It's called an entry control loop.

* do while loop

1. **Explain the while loop with an example.**

**ANS :**

**while loop :**

**Syntax :**

Declaration / Initialization

while(condition)

{

// statement;

updation;

}

**Example :**

int a = 1;

while(a<=10)

{

printf(“%d \t”,a);

a++;

}

1. **Explain the do while loop with an example.**

**ANS :**

**do while loop :**

**Syntax :**

Declaration / Initialization

do

{

// statement;

updation;

}while(condition);

Note : which line ends with (); there compulsory put ;.

**Example :**

int a = 1;

do

{

printf(“%d \t”,a);

a++;

}while(a<=10);

1. **Explain Control Statements with example.**

**ANS :**

1. **break :**

This Keyword terminates execution of a particular block.

1. **Continue:**

This keyword used to skip execution of particular cycle

of loop.

1. **goto :**

This keyword used to jump our execution anywhere to anywhere.

1. **State difference between entry-controlled and**

**exit-controlled loops.**

**ANS :**

**Entry Controller :**

* For every time the first condition checks then going to print output if the condition is true.
* Ex. while loop, for loop.

**Exit Controller :**

* For the first time output will be printed then check condition..
* Ex. do while loop.

1. **What is Array? Explain its types.**

**ANS :**

**Array :**

* Array is Collection/group of elements of the same datatype.

* Array means Collection of integer values. In which all values

​​have their own index.

* The index of the array starts from 0 only.

**Types :**

1. 1D Array (One-Dimensional Arrays) :

These are the most basic type of arrays and

consist of a linear collection of elements.

**Syntax :**

datatype variableName[size] = {integer values};

**Ex :**

int marks[05] = {10,20,30,40,50};

// index 0 1 2 3 4

1. 2D Array (Multidimensional Arrays) :

2D Array is a Collection/group of 1D Array. where

elements are organised in a grid-like structure

with rows and columns.

**Syntax :**

datatype variableName[row][column] = {

{int values},

{int values},

…

{int values},

};

**Ex :**

int a[3][3] = {

{1,2,3},

{4,5,6},

{7,8,9}

};

// index :

{

{(0,0), (0,1), (0,2)},

{(1,0), (1,1), (1,2)},

{(2,0), (2,1), (2,2)}

}

1. **What is String? Explain with example in detail.**

**ANS :**

**String:**

* String is a collection / group of character values.
* The index of the string starts from 0 only.

**Syntax :**

datatype variableName[size] = {character values};

**Ex :**

char name[5] = { ‘h’, ‘e’, ‘l’, ‘l’, ‘o’};

// index : 0 1 2 3 4

Or

char name[5] = “Hello”;

Note : %s will be used as the format specifier in the string.

1. **What is ASCII value? List some important ASCII values.**

**ANS :**

ASCII full form American Standard Code for Information

Interchange.

This ASCII value represents the character variable in

numbers, and each character variable is assigned with some

numbers range from 0 to 127.

List of ASCII values :

A - Z : 65 to 90

a - z : 97 to 122

space : 32

symbol : 33 to 47 AND 58 to 64

digit : 48 to 57

null : 0

1. **What is NULL? Explain with example in detail.**

**ANS :**

**NULL :**

* Unassigned members of string will contain null(empty) value.
* NULL is a single character that compares equal to 0.
* NULL means empty.
* null can be denoted by NULL / '\0'

**Symbol :** ‘\0’ , NULL

**Ex. :**

char name[10] = “Mayank”;

‘M’ , ‘a’ , ‘y’ , ‘a’ , ‘n’ ,’k’ , ‘\0’ , ‘\0’ , ‘\0’ , ‘\0’

0 1 2 3 4 5 6 7 8 9

1. **List some built-in string functions with example.**

**ANS :**

**Built-in string function :**

1. puts() :

The puts function in C is used to output a string.It automatically appends a newline.

Syntax : puts(string);

Ex.: char name[5] = “Hello” ;

puts(name);

Output :

Hello

1. gets() :

The gets function in C is used to input a string.

Syntax : gets(string);

Ex.: char name[5];

printf(“Enter name : ”);

gets(name);

Output :

Enter name : hello

1. strlen() :

The strlen function in C is used to calculate the

length of a string.

Syntax : strlen(string);

Ex. :

char name[10] = “Nayan”;

int len = strlen(name);

printf(“%d”,len);

Output :

5

Note : strlen function is return int value.

1. strupr() :

The strupr function is used to convert all

characters in a given string to uppercase.

Syntax : strupr(string);

Ex. :

char name[10] = “hello”;

printf(“%s”,strupr(name));

Output :

HELLO

1. strlwr() :

The strlwr function is used to convert all

characters in a given string to lowercase.

Syntax : strlwr(string);

Ex. :

char name[10] = “HELLO”;

printf(“%s”,strlwr(name));

Output :

hello

1. strrev() :

The strlrev function is used to convert all

characters in a given string to reverse.

Syntax : strrev(string);

Ex. :

char name[10] = “hello”;

printf(“%s”,strrev(name));

Output :

olleh

1. strcat() :

The strcat function is used to concatenate

(append) one string to the end of another string.

Note : strcat method uses Two strings.

Syntax : strcat(string1 , string2);

string1 = string1 + string2;

Ex. :

char name[10] =“Hello”;

char surName[10]=”World”;

printf(“%s”,strcat(name,surName));

Output :

HelloWorld

1. strcpy() :

The strcat function is used to copy the contents

of one string to another string.

Note : strcpy method uses Two strings.

Syntax : strcpy(string1 , string2);

string1 = string2;

Ex. :

char name[10] =“Hello”;

char sname;

strcpy(sname,name);

printf(“%s”,sname);

Output :

Hello

1. strcmp() :

The strcmp function is used to compare two

strings character by character.

Note : strlen function is return int value.

Syntax : strcmp(string1 , string2);

ans = string1 - string2

=> str1 str2 Ans

A a => -1

a A => 1

A A => 0

Ex. :

char name[10] =“Hello”;

char sname[10]=”Hello”;

int cmp = strcmp(name,sname);

printf(“%d”,cmp);

Output :

0

1. **What is Function? Explain types of it.**

**ANS :**

**Function :**

A re - usable block of code is called a function.

* re - usable block of code which can be accessed any

time just by calling through its name.

* each function has unique signatures(name, return type, arguments) according to its use.
* function can be denoted by '(){}'.
* Life cycle of function

Declaration (Register)

Definition (Put Logic)

Calling (Use)

**Types of Function :**

1. Built - in function :

* built-in function is a function that is provided by the C standard library.
* Functions that are already created are called built in functions.
* printf(), scanf(), clrscr(), getch(), gets(), puts(), ....

1. User Defined Functions (UDF) :

* A user-defined function is a function that you create yourself to perform a specific task within your C program.

1. **What is User Defined Function (UDF)? Explain types of it.**

**ANS :**

**UDF :**

A user-defined function is a function that you create

yourself to perform a specific task within your C

program.

**Types :**

1. TNRN (Take Nothing Return Something)
2. TSRN (Take Something Return Nothing)
3. TNRS (Take Nothing Return Something)
4. TSRS (Take Something Return Something)

* Life cycle of function

Declaration (Register)

Definition (Put Logic)

Calling (Use)

* Syntax :

ReturnDatatype functionName([argument])

{

[statement] / [return value]

}

* [ ] : means Optional.
* ReturnDatatype :
* If returns => int , char , float…
* According to the return value.
* If does not return => void

1. TNRN (Take Nothing Return Nothing) :

Function is not get an argument and does not return any

data type.

Ex :

#include<stdio.h>

void name(); // Declaration

void name() // Definition

{

printf(“\n\t Name\t: MyName”);

}

main()

{

name(); // calling

}

1. TSRN (Take Something Return Nothing) :

Function is get an argument and does not return any

data type.

Ex :

#include<stdio.h>

void sum(int a , int b)

{

printf(“\nSum of : %d ”,a+b);

}

main()

{

sum(10,20);

}

1. TNRS (Take Nothing Return Something)

Function is not get an argument and return any

data type.

Ex :

#include<stdio.h>

int sum()

{

int a , b;

printf(“Enter a : ”);

scanf(“%d”,&a);

printf(“Enter b: ”);

scanf(“%d”,&b);

return a+b;

}

main()

{

int c = sum();

printf(“Addition : %d ”,c);

}

1. TSRS (Take Something Return Something) :

Function is get an argument and return any

data type.

Ex :

#include<stdio.h>

int sum(int a, int b)

{

return a+b;

}

main()

{

int c = sum(10,20);

printf(“Addition : %d ”,c);

}

1. **What is Recursion? Explain working of a Recursion**

**mechanism with example.**

**ANS :**

**Recursion :**

Recursion is a function which calls itself.

Ex. :

#include<stdio.h>

void loop(int start , int end)

{

if(start <= end)

{

printf(“%d”, start++);

loop(start, end);

}

}

void main()

{

loop(1,10);

}

1. **What is Nested Function? Explain with example..**

**ANS :**

**Nested Function :**

Nested function is a function that is defined within

the scope of another function.

Ex.:

#include<stdio.h>

int arraySum(int a[ ], int n)

{

int sum = 0,i;

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

sum += a[i];

return sum;

}

int arrayAverage(int a[ ], int n)

{

int sum = arraySum(a,n);

int avg = sum/n;

return avg;

}

void main()

{

int n,i;

printf(“Enter n”);

scanf(“%d”,&n);

int a[n];

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

{

printf(“Enter value a[%d] = ”);

scanf(“%d”,&a[i]);

}

int avg = arrayAverage(a, n);

printf(“Average : %d”,a);

}

}

1. **What is Pointer? Explain its use case.**

**ANS :**

**Pointer :**

* pointer is a variable which stores address of another

variable.

* pointer can be created with asterisc '\*' operator.
* value of another variable can be stored with address of

operator '&'.

* to print or use the connected variable's value from pointer,

we've to use \* operator.

* to print the address of connected variable, use pointer

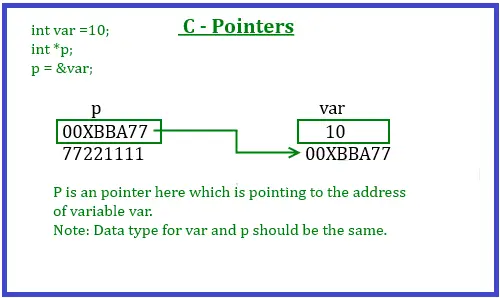
without \* operator.

* pointer of variable must have the same data type.
* To print address of variable :

- %p => Hexadecimal

- %x => Hexadecimal

- %u => Decimal address (Numeric)



Ex. :

#include<stdio.h>

main()

{

int a = 10;

int \*b;

b = &a;

printf("A : %d\n",a);

printf("B : %d\n",\*b);

a = 99;

printf("A : %d\n",a);

printf("B : %d\n",\*b);

\*b = 122;

printf("A : %d\n",a);

printf("B : %d\n",\*b);

}

1. **Explain working of sizeof() operator.**

**ANS :**

* It is a compile-time unary operator which can be used to

compute the size of its operand.

* sizeof() operator is used to determine the size (in bytes) of a data type or a variable.

syntax :

sizeof(expression)

* We use either %lu or %zu format specifiers.

Ex. :

#include<stdio.h>

main()

{

printf("Size of int: %lu bytes\n", sizeof(int));

printf("Size of float: %lu bytes\n", sizeof(float));

printf("Size of char: %lu bytes\n", sizeof(char));

printf("Size of double: %lu bytes\n", sizeof(double));

printf("Size of long int: %lu bytes\n", sizeof(long int));

printf("Size of long long int: %lu bytes\n", sizeof(long

long int));

}

output :

Size of int: 4 bytes

Size of float: 4 bytes

Size of char: 1 bytes

Size of double: 8 bytes

Size of long int: 4 bytes

Size of long long int: 8 bytes

Ex. :

int array[6];

printf("Size of array: %zu bytes\n", sizeof(array));

output :

Size of array: 24 bytes.

Ex. :

char name[6];

printf("Size of string: %zu bytes\n", sizeof(name));

output :

Size of string: 6 bytes

1. **What is the Scale of Pointer? Explain with example.**

**ANS :**

* **Scale of Pointer :**

Ex. :

#include<stdio.h>

main()

{

int a[5] = {11,22,33,44,55},i;

int \*ptr;

ptr = &a;

for(i=0; i<5; i++)

printf("A : %d\n",\*(ptr+i));

}

1. **Describe Array of Pointers with example.**

**ANS :**

An array of pointers in the C programming language is an

array where each element is a pointer.

Ex. :

#include<stdio.h>

main()

{

int a[5] = {10,20,30,40,50};

int i;

int \*ptr[5];

for(i=0; i<5; i++)

{

ptr[i] = &a[i];

}

printf("Pointer of Array\n");

for(i=0; i<5; i++)

printf("%d\n",\*ptr[i]);

}

1. **What is a Chain of Pointer? Describe with example**

**ANS :**

Chain of pointers is when there are multiple levels of

pointers.

Ex. :

#include<stdio.h>

main()

{

int a = 10;

int \*x;

int \*\*y;

int \*\*\*z;

x = &a;

y = &x;

z = &y;

printf("A : %d\n",a);

printf("X : %d\n",\*x);

printf("Y : %d\n",\*\*y);

printf("Z : %d\n",\*\*\*z);

}

1. **Explain Pointer with UDF in detail.**

**ANS :**

#include<stdio.h>

void swap(int \*a, int \*b)

{

int c;

c = \*a;

\*a = \*b;

\*b = c;

}

void main()

{

int a,b;

a = 10;

b = 20;

swap(&a,&b);

printf(“A : %d\n B : %d\n”,a,b);

}

1. **What is Structure? Explain with example.**

**ANS :**

**Structure :**

* Structure is User Defined Data Type.
* It is a combination of multiple data types.
* it can be created using the 'struct' keyword.
* it must be created in a global area.
* it contains the variable declaration only.
* we cannot initialise the variables inside the structure.
* we cannot create UDFs inside the structure.
* the inner variables(attributes) of structure can be accessed using objects of structure.

Syntax :

struct StructureName {

// Declaration of Variables.

};

Ex. :

#include<stdio.h>

// define structure

struct Student {

int rollNo;

char name[20];

float per;

};

void main()

{

struct Student s; // structure object

printf(“Enter Name : ”);

scanf(“%s”,&s.name);

printf(“Enter Rollno : ”);

scanf(“%s”,&s.rollNo);

printf(“Enter Per : ”);

scanf(“%s”,&s.per);

printf(“RollNo\t: %d”,s.rollNo);

printf(“Name\t: %d”,s.name);

printf(“Per\t: %d”,s.per);

}

1. **What is Union? Explain with example.**

**ANS :**

**Union:**

* Union is User Defined Data Type.
* Collection of multiple variables which have multiple data types.
* union can be created using 'union' keyword
* its inner variables can be accessed using the object of union.
* In the case of storing and retrieving multiple attributes, union isn't useful at all because it stores only the last given value properly. The earlier values won't be accessed.

Syntax :

union unionName {

// Declaration of Variables.

};

Ex. :

#include<stdio.h>

union student {

int id;

char name[20];

float per;

};

void main()

{

union student s;

printf("Enter id : ");

scanf("%d",&s.id);

printf("Enter Name : ");

scanf("%s",&s.name);

printf("Enter Per : ");

scanf("%f",&s.per);

printf("\nId : %d\nName : %s\nPer :

%.2f",s.id,s.name,s.per);

}

1. **What is Enumeration? Explain with example.**

**ANS :**

**Enumeration :**

* Union is User Defined Data Type.
* it is used to give the index of attributes
* can be created using 'enum' keyword.
* it doesn't require object to access the inner attributes.
* indexing order of enum can be modified also.
* attributes don't need data types, they are integers by default.

Syntax :

enum varName {

val1, val2,... val N

};

Ex. :

#include<stdio.h>

enum week {

sun , mon , tue , wed , thu , fri , sat

};

void main()

{

printf("Sun = %d\n",sun);

printf("mon = %d\n",mon);

printf("tue = %d\n",tue);

printf("wed = %d\n",wed);

printf("thu = %d\n",thu);

printf("fri = %d\n",fri);

printf("Sat = %d",sat);

}

Output :

Sun = 0

mon = 1

tue = 2

wed = 3

thu = 4

fri = 5

Sat = 6

Second Example :

#include<stdio.h>

enum week{

sun=11,mon,tue,wed=40,thu,fri,sat

};

Void main()

{

printf("Sun = %d\n",sun);

printf("mon = %d\n",mon);

printf("tue = %d\n",tue);

printf("wed = %d\n",wed);

printf("thu = %d\n",thu);

printf("fri = %d\n",fri);

printf("Sat = %d",sat);

}

Output :

Sun = 11

mon = 12

tue = 13

wed = 40

thu = 41

fri = 42

Sat = 43