**Introduction to C**

* The word computer is derived from word compute which means to calculate.
* The process of converting the required input into output form with the help of computer is called as data processing.
* C language is a general-purpose high-level programming language which was introduced by Dennis M Ritchie in 1972 at bell laboratories in ANSI in America.
* C language was invented for finding language for UNIX operating system
* It was named as c because it is the successor of B language.

**Introduction to JavaScript**

* JavaScript is used to create client-side dynamic pages, it is an object-oriented scripting language.
* JavaScript Is not a compiled language but it is a translated language. The JavaScript translator is responsible for translating the JavaScript code for the web browser. It is language that enables dynamic interactivity on websites when applied to the HTML document. Introduced in the year 1995.
* Although JavaScript is not having any connectivity with java but the name was suggested and provided in the times when java was gaining popularity in the market.
* Initially it was named as live script but due to some trademark reasons it was named as JavaScript
* JavaScript code provides 3 places to put in the JavaScript code:
* within the head tag
* within the body tag
* external
* It provides code reusability because single JavaScript can be used in many html pages.
* An external JavaScript file can be saved with .is file extension.

**Disadvantages;**

* The stealer may download the coders code using the URL of the is. file
* If two is files are dependent on one another then a failure in one file may affect the execution of the other file also

**Introduction to java**

* The history of java is very interesting java was originally designed for interactive television.
* Java team members also called as green team initiated the project for develop a language for set up boxes and television etc.
* Firstly, it was called as green talk by James Gosling and file extension for these is .gt
* After it was named as oak and was developed as a project of green project
* why oak? Because it is symbol of strength and chosen a national tree for many countries.
* After oak was renamed as java it was already trademark by oak technologies.
* According to James Gosling java was one of the top choices with silk since java was so unique

most the team members suggested for that.

* And also, java is the island in Indonesia where first coffee was produced java name chosen by James Gosling while he is having a cup of coffee.
* Java is just a name not acronym
* Java syntax is similar to c++, c#.
* Every line of code that run the java inside the class only.
* Java is case sensitive.

Syntax:

Public class main

{

Public static void main(String[] args)

System. Out .println(“hello world”)

}

}

* System is a built in java class that contains useful members such as out which is short for output. The println()

Method short for print line.

* All these system, out, println(), together to print stuff in the screen
* There is also print() method ,which is similar to println but the only difference is that id doesn’t print code by line by line.

**Introduction to python**

* python was developed by Guido van Rossum in the late 1980’s and early 1990’s in Netherlands.
* The inspiration came from the BBC’s Tv Show Monty Pythons Flying circus which is very famous tv show at that time.
* And also, he wanted a short, unique and slightly mysterious so it is named as python.
* Python is high level interpreted, interactive and object-oriented scripting language.

**Features of c:**

* + Simple syntax
  + Easy to learn
  + Easy to code
  + Procedure oriented programming language
  + Memory management
  + Speed

**Features of java:**

* Simple and easy to learn
* Object oriented
* Platform independence
* Security
* Multithreading
* High performance
* Features of JavaScript:
* Dynamic typing
* Light weight scripting language
* Platform independent
* Object oriented programming support

**Features in python**:

* Easy to learn and use
* Rich in library
* Free and open source
* Portable
* Support for GUI
* Interpreted language

**Features In JavaScript:**

* Scripting
* Platform independent
* Scripting Language

**Program Development Cycle**

* Problem definition
* Problem analysis
* Algorithm development
* Coding and documentation
* Testing and debugging
* Maintenance

**Flow chart:**

1. Used for formulas

(a+b)2

1. Used for conditional statements like if, else if, while, for etc.

conditions

3.Used for start and stop

Start or stop

4.Used for i/p and o/p indicators

Print

**Comments** (C, Java, JavaScript)

There are two types of comments that which are available in c language those are:

1. Single line comment (//)
2. Multi line comment (/\*\_\_\_\*/)

**Comments in python**:

There are two types of comment line in python

1. Single Line Comment:

#print (“hello world”)

1. Multiline comments:

#This is comment

#written in

#More than just one line

Or

“”” this is a comment in more than one lines “””

**Types of variables**:

**1.Camel case:**

Here each word except first word starts with capital letter.

myVariableName

**2.Pascal case:**

Each word starts with a capital letter

MyVariableName

**3.Snake case:**

Here each word is separated by an underscore

my\_varaible\_name.

**Variables: [C, python]**

* A variable is a name of memory location. It is used for store the data.
* variables are changeable we can change the value of the variable during the execution of the program.
* Variables must be declared before using them in program.
* If we don’t declare variables then we get error.

**Rules:**

* Variables must start with letters or underscore.
* Noo special symbols are used for declaring variables except underscore.
* Variables names cannot contain special names or spaces.
* Variable name cannot be keyword because keywords have special function in any language.
* 0x or 0X for hexadecimal.
* 0 for octal.
* Nothing for decimal.
* 12,245 invalids.
* 10 20 30 invalids.

**JavaScript variables:**

* Here variables can be declared in 4 ways:
* Automatically
* Using var
* Using Let
* Using const
* The var keyword is used I all JavaScript code from 1995 to 2015.
* The let a const keywords were added to JavaScript in 2015
* The var keyword is used only when the code written in old folders.
* Always use the constants if the value not be changed.
* Only use let if you can’t use const.

**Rules:**

* Name must start with dollar sign or underscore or alphabets
* JavaScript is case sensitive

<!document html>

<html>

<body>

<script>

var x=10;

var y=20;

var z= x+y;

document. Write(z);

</script>

</body>

</html>

**JavaScript local variables:**

<script>

function abc () {

var x=10;//local variable

}

</script>

**JavaScript global variable**:

**<**script**>**

var data=200;//global variable

function a(){

document. Writeln (data);

}

function b(){

document.Writeln (data);

}

a();//calling JavaScript function

b();

</script>

Declaring JavaScript global variable within function

To declare JavaScript variables inside a function you need to use window object

<html>

<body>

<script>

function m(){

window. Value=100;//declaring global variable by window object

}

function n(){

alert(window. Value);//accessing global variable from other function

}

m();

n();

</script>

</body>

</html>

**Variables in java**:

**Rules**

* We can begin them with either dollar sign or underscore or any alphabet.
* Don’t use capital letters as variables.’

Declaration of variables:

Type varibalename=value;

* If you don’t want others to change existing values then in java, we have to use final or constant which means unchangeable or read only.

Final int mynum=15;

System.out.println(mynum);

**Keywords(c)**

* Keywords are reserved words in c language.
* There are 32 keywords which are available in c.
* Each keyword is meant to perform a special work.

Auto double goto signed unsigned

Break do if sizeof void

Case else int static volatile

Char enum long struct while

Const extern register switch

Continue float return typedef

Default for short union

**Keywords (JavaScript)**

Abstract arguments await Boolean

Break byte case catch

Char class const continue

Debugger default delete do

Double else Enum eval

Export extends false final

Finally float for function

Goto if impements import

In instance of int interface

Let long native protected

Public return short static

Super switch synchronized this

Throw throws transient true

Try typeof var void

Volatile while with tield

**Keywords (java)**

Abstract Boolean break byte case

Catch char class continue default

Do double else Enum extends

Final finally float for implements

Import int instance of int interface

Native new null package private

Return short static strictfp long

Super switch synchronized this protected

Throw throws transient try void

volatile while

**Keywords (Python)**

35 keywords available in python.

>>import keyword

>>keyword. Kwlist

>>len(keyword.kwlist)

And def exec if not return

Assert del finally import or try

Break elif for in pass while

Class else from is print with

Continue except global lambda raise yield

**Data types in C**

There are two types of qualifiers in c

1. Size qualifier (short, Long)
2. Sign qualifier (unsigned, signed)

The range of values for signed data type is less than that of unsigned data type. Because in signed data type left most bit is used to represent sign, while in unsigned it is also used for represent value.

In c there are 4 types of data types are there:

1. Primitive or Primary or Built-in data type
2. Non primitive or user defined or secondary
3. Void
4. Enum

**Primitive Data Types**: predefined datatypes

There are:

1. Int
2. float
3. char

**Integer**

* 16 bytes
* -32768 to 32767
* %d or %i used for signed integers or integers

**Unsigned int**:

* 16 bytes
* 0 to 65535

**Short int:**

* 8 bytes
* -128 to 127

**Unsigned short int:**

* 8 bytes
* 0 to 255

**Long int:**

* 32 bytes
* -2147483648 to 2147483647
* %ld

**Unsigned long int**:

* 32 bytes
* 0 to 6,

**Float:**

* 32 bytes
* -3.4E-38 to 3.4E+38
* %f is used for float
* %e or %E is e for 3.6e, for 3.6E.

**Double:**

* 64
* 1.7E-308 to 1.7E+308
* %f or %lf

**Long double:**

* 80
* 3.4E-4932 to 1.1E+4932
* %Lf

**Character Type:**

* 8
* -128 to 127
* %c

**Unsinged char**

* 8
* 0 to 255
* %c

**Other format specifiers**:

1. %x for hexadecimal in lowercase
2. %X for hexadecimal in uppercase
3. %p or %u for address
4. %0x fo

**Non primitive or derived data type**:

These are derived by the primitive data types.

E.g. structures, unions, pointers, arrays, strings

**ENUM:**

**JavaScript data types**:

Here two types of data types are there:

* Primitive data type
* Non primitive data type or reference datatype

There are five types of primitive data types are there:

* String
* Number
* Boolean
* Undefined
* Null

Non primitive data types:

* Object represents instance through which we can access members
* Array represents group of similar values
* Regex represents regular expression

**Data Types in Java**

Data types are divided into 2 types:

* Primitive
* Non primitive

Primitive:

byte, short, int, long, float, double, Boolean and char

Non-primitive:

String, Array, Classes

Primitive:

* Byte: 1 byte: stores whole numbers from -128 to 127, it is used instead of int or other types of integer values to save the memory.
* Short :2bytes: stores whole numbers from -32768 to 32767
* Int:4 bytes: stores whole umbers from -2147483648 to 2147483647
* Long 8bytes stores whole numbers from -9223372036854775808 to 9223372036854775807,, this is used when int is not enough to store.while using these we must end this with L
* Float 4 bytes: stores fractional numbers from 6 to 7 decimal digits, using tehze we must end with f.
* Double:8bytes stores fractional numbers from to 15 decimal digits, using thse we nust end with d.
* Boolean: 1bit stores true or false.
* Char 2bytes stores a single character.

**Data types in python**:

Text Type: str

Numeric Type: int, float, complex

Sequence type: list, tuple, range

Mapping type: dict

Set types: set, frozen set

Boolean type: bool

Binary types: bytes, bytearaary, memory view

None type: nonetype

You can get the data type of any object by using the type () function.

X=str (“hello world”)

x=int (20)

x=float (20.5)

x=complex(1j)

x=list ((“apple”, “banana”, “cherry”`))

x=tuple((“apple” , “banana”, “cherry”))

x=range(6)

x=dict(name=”john” , age=36)

x=set((“apple”, “banana” ,”cherry”))

x=frozenset((“apple” , “banana” , “cherry”))

x=bool(5)

x=bytes(5)

x=bytearray(5)

x=memoryview(bytes(5))

**Operators in c:**

Operators are divided into 3 types:

* Unary
* Binary
* Ternary

operators are nothing but that which acts upon on operands.

In c there are 8 types of operators are there:

* Ari thematic operators (+, -, \*, /, %)
* Relational operators (==, =,>, <, <=,>=)
* Shift operators (left shift and right shift)

1. **Left shift (<<):** The left shift operator moves the bits of the first operand the left by the number of places of the second argument.
2. **Right shift operator (>>):** The right shift operator shifts the bits of the first operand to the right but the number of positions specified by the second operand

* Logical operators: perform logical operations and returns true or false (AND (&&), OR (||), NOT(!))
* Bitwise operators: It only operate son the single bits (&, |, ^, ~)
* Ternary or conditional operator it is same as if else condition in conditional statements.
* Assignment operator assignment operators are used to assign the values to the variables (=, +=, -=, \*=, /=, %=,)
* Miscellaneous operator (The size of and the comma operator comes under the miscellaneous operator)

Comma operator evaluate multiple expressions and gives the last expression.

**Increment operator:**

Whenever more than on format specifier is directly or indirectly linked related to same variable then we need to evaluate the expression from right to left.

#include<stdio.h>

int main()

{

Int i=1;

Printf(“%d%d%d”,i,++i,++i);

}

o/p:3 3 1

**Operators in JavaScript:**

Ther are following types of operators are JavaScript language

* Air thematic operators (+, -, \*, /, %, ++, --)
* Comparison operator (==, ===, =, ==, <,>, <=, <=)
* Bitwise operators (&, |, ^, ~, <<,>>,>>>,)
* Logical operators (&&, ||, !)
* Assignment operator (=, +=, - =, \*=. %= ,/=)
* Special operator

(?:) conditional operator returns values based on the condition it is like if else condition

,

Delete In in operator checks if object has the given property.

Instance of checks if the object is an instance of given type

New create an instance

Type of checks the type of the object

Void it discards the expression return value

Yield checks what is returned in a generator by the generator’s iterator.

**Operators in python**

* Arithmetic
* Comparison or relational
* Assignment
* Logical
* Bitwise
* Membership (in and not in)
* Identity (is and is not)
* In: It evaluates true if it finds a variable in the specified sequence &false otherwise (it means linear search in python)
* Not in: evaluate true if it does not find a variable
* And otherwise, false.
* Is: evaluates true if the variable on the either side of the object points to the same objects and false otherwise
* Is not: evaluates to false if the variables on the either side of the object points to the same object.

**Operators(java)**

* Unary (increment. Decrement)
* Arithmetic
* Shift
* Bitwise
* Logical
* Ternary assignment

**Control statements or conditional statements or selection statements in c**

1. if
2. If else
3. Else if
4. Nested if

**If**

if(expression) {

//code to be executed

}

**Examples:**

#include<stdio.h>

int main () {

int number=0;

printf(Enter a number);

scanf("%d", &number);

if(number%2==0) {

printf(%d is even number ,number);

}

return 0;

}

**If else:**

if(expression) {

//code to be executed if condition is true

} else{

//code to be executed if condition is false

}

**Example:**

#include<stdio.h>

int main () {

int number=0;

printf (enter a number:);

scanf(%d, &number);

if(number%2==0){

printf(%d is even number, number);

}

else{

printf(%d is odd number, number);

}

return 0;

}

**Else if ladder:**

if(condition1) {

//code to be executed if condition1 is true

} else if(condition2) {

//code to be executed if condition2 is true

}

else if(condition3) {

//code to be executed if condition3 is true

}

else{

//code to be executed if all the conditions are false

}

**Examples:**

#include<stdio.h>

int main () {

int number=0;

printf(enter a number:);

scanf(%d, &number);

if(number==10) {

printf(number is equals to 10);

}

else if(number==50) {

printf("number is equal to 50");

}

else if(number==100) {

printf("number is equal to 100");

}

else{

printf("number is not equal to 10, 50 or 100");

}

return 0;

}

**Conditional statements in JavaScrip**t:

* If statement
* If else statement
* If else if statement

**<script>**

var a=20;

if(a==10){

document.write("a is equal to 10");

}

else if(a==15){

document.write("a is equal to 15");

}

else if(a==20){

document.write("a is equal to 20");

}

else{

document.write("a is not equal to 10, 15 or 20");

}

**</script>**

**Conditional operator(java)**

If

If else

If else if

Nested if

If

public class IfExample {

public static void main (String [] args) {

    //defining an 'age' variable

    int age=20;

    //checking the age

**if**(age>18){   than 1

        System. out. print("Age is greater  8");

    }

}

}

If else:

1. **public** **class** IfElseExample {
2. **public** **static** **void** main (String[] args) {
3. //defining a variable
4. **int** number=13;
5. //Check if the number is divisible by 2 or not
6. **if**(number%2==0){
7. System. out. Println("even number");
8. } **else** {
9. System. out. Println("odd number");
10. }
11. }
12. }

If else if

1. **public** **class** IfElseIfExample {
2. **public** **static** **void** main (String[] args) {
3. **int** marks=65;
5. **if**(marks<50) {
6. System.out.println("fail");
7. }
8. **else** **if**(marks>=50 && marks<60){
9. System. out. Println ("D grade");
10. }
11. **else** **if**(marks>=60 && marks<70){
12. System. out. Println ("C grade");
13. }
14. **else** **if**(marks>=70 && marks<80){
15. System. out. Println ("B grade");
16. }
17. **else** **if**(marks>=80 && marks<90){
18. System.out.println("A grade");
19. }**else** **if**(marks>=90 && marks<100){
20. System.out.println("A+ grade");
21. }**else**{
22. System.out.println("Invalid!");
23. }
24. }
25. }

**Conditional statements(python): here it has to follow indentation**.

If

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

Elif

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

If else

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

**Switch:**

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, we can define various statements in the multiple cases for the different values of a single variable.

switch(expression) {

case value1:

 //code to be executed;

break; //optional

case value2:

 //code to be executed;

 break; //optional

......

default:

**Rules**

* The *switch expression* must be of an integer or character type.
* The *case value* must be an integer or character constant.
* The *case value* can be used only inside the switch statement.

The *break statement* in switch case is not must. It is optional. If there is no break statement found in the case, all the cases will be executed present after the matched case. It is known as *fall through* the state of C switch statement. Even if the case constants of the inner outer switch contain common values, no conflicts will arise.

Example:

#include <stdio.h>

**int** main() {

**int** num = 2;

**switch** (num) {

**case** 1:

printf("Value is 1\n");

**break**;

**case** 2:

printf("Value is 2\n");

**break**;

**case** 3:

printf("Value is 3\n");

**break**;

**default**:

printf("Value is not 1, 2, or 3\n");

**break**;

}

**return** 0;

}

Switch statement by JavaScript:

1. **<script>**
2. var grade='B';
3. var result;
4. switch(grade){
5. case 'A':
6. result="A Grade";
7. break;
8. case 'B':
9. result="B Grade";
10. break;
11. case 'C':
12. result="C Grade";
13. break;
14. default:
15. result="No Grade";
16. }
17. document.write(result);
18. **</script>**

**Loops:**

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times.’

1. While loop or sentinel-controlled loop
2. For loop or counter controlled loop

Whenever semicolon is placed at the end of the while loop this means that the body value never gets incremented.

Loops in JavaScript:

**For loop**

<script>

for (i=1; i**<**=5; i++)

{

document.write(i + "<br/>")

}

</script**>**

**While loop**

**<script>**

var i=11;

while (i**<**=15)

{

document.write(i + "**<br/>**");

i++;

}

**</script>**

For in loop

Do while loop

**<script>**

var i=21;

do{

document.write(i + "**<br/>**");

i++;

}while (i**<**=25);

**</script>**

**Loops (java)**

While

Do while

For-Each Loop:

For

For-Each loop:

Mainly used in ararys

for (type variableName : arrayName) {

*// code block to be executed*

}

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (String i : cars) {

System.out.println(i);

}

Loop (Python)

There are two types loops present:

Entry controlled loops (pretest):

Here the condition is tested before the start of the execution of statements.

Exit controlled loops(posttest):

Here the condition is tested at the end of the program)

While

For

Syntax:

For iterator Var in sequence:

Statements(s)  
Here in python language indentation will be followed.\

Jump or loop control or unconditional statements:

* Break
* Continue
* Goto

Break:

By using these we can terminate the further execution of the program.

Syntax: break;

Continue:

Thes e is the fully opposite to break statement.

It is used for continuing the next iteration of loop statements.

When continue statement occurs in the loop it does not terminate but skips the statement after the continue statement.

Goto: It does not require any condition in the loop.

This attement passes control anywhere in the program i.e contro is transfereed to another part of program without testing any condition.

The goto statement requires a label to identify the place to move to exection.a label is valid varable name and must tended with colon.

Goto label;

-------

------- forward branching

Label:

Label:

------ backward branching

-------

Goto label;

Use of goto statement is highly discouraged in any programming language because it makes difficult to trace t eh control flow of a program making the program hard to understand and hard to modify.

Here label can be any plain text except c keyword and it can be set anywhere in the program

A loop becomes infinite if a condition never becomes false.

For(; ;)

You can terminate the infinite loop by using ctrl+c keys.

How does a computer works?

User

Application

Operating system

Hardware

#include<stdio.h> is preprocessor complier directive it is not the statement so is does not end with semicolon.

We know that computer only understands 0’s and 1’s when we give a variable how can it convert to machine language.

Bu using ASCII American Standard code for Information Interchange

Why we are using return 0?

The return 0 value tells us that the program has completed all its task and has terminated successfully and the hardware resources that has been allocated to the program can be now freed and it will be reallocated to other program. Here we can put any number in place of zero either positive or negative it will send the message that the program has been terminated But only zero will send a successful and error free termination message.

c

JavascriptJavaScript function example:

<html>

<body>

<script>

function msg (){

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="call function"/>

</body>

</html>

JavaScript function

<html>

<body>

<script>

function getcube(number){

alert(number\*number\*number);

}

</script>

<form>

<input type="button" value="click" onclick="getcube(4)"/>

</form>

</body>

</html>

Function with return value:

<html>

<body>

<script>

function getInfo(){

return "hello javatpoint! How r u?";

}

</script>

<script>

document.write(getInfo());

</script>

</body>

</html>

Javascript function object

In JavaScript, the purpose of **Function constructor** is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

new Function ([arg1[, arg2[, ....argn]],] functionBody)

**arg1, arg2, .... , argn** - It represents the argument used by function.

**functionBody** - It represents the function definition.

Javacsript function methods:

Apply() it is used to call afunction contains this value and a single array of arguments

Bind() it is used to create a new function

Call() it is used to call a function contains this value and an argument list.

toString() It returns the result in a form of a string

<!DOCTYPE html>

<html>

<body>

<script>

var add=new Function("num1","num2","return num1+num2");

document.writeln(add(2,5));

</script>

</body>

</html>

1. **<script>**
2. var pow=new Function("num1","num2","return Math.pow(num1,num2)");
3. document.writeln(pow(2,3));
4. **</script>**

o/p:8

Type conversion or casting:

Converting one data type to another data type.

There are two types of those are:

1. Implicit
2. Explicit

Implicit:

* Also called as automatic type conversion.
* Here lower data type is assigned to higher data type.

Explicit:

* These is useful when higher data type is converted to lower data type
* Here it doesn’t happen automatically ‘here we have to mention what data type we have to convert
* Because here we are converting the Higer data type to lower data type so higher can take more bits than lower data type.so here wastage happens that’s why we have to mention.

Expressions have also classified that the type that they represent:

1. Boolean expression
2. Integer expression
3. Floating point
4. String expression

Arrays(c):

* Array is a collection of similar data type and stored in continuous memory.
* The lowest address corresponds to the first element and highest element address to the last element.
* Array of character is string
* Each data item in array is called an element. And each element in array is unique and located in separated memory location.
* Each of elements of an array share same variable name but each element is having different subscripts.
* Array can be single or multidimensional number of subscripts determines the type of the dimension.
* It can be used to represent multiple data structures like linked lists, stacks, queues, trees, graphs etc.
* We must know in advance that how many elements we have to store.
* Array is static structure i.e. fixed size the memory which is allocated we are not able to increase or decrease.
* Symbolic also be used to specify the size of the array

#define size 10;

* Index of array starts from 0 and end with -1.
* If during the initialization of number, the initializers is less than size of the array then all the remaining elements assigned to be zero.
* If the number of elements is more than the size given in brackets then the compiler will show an error.

Types:

1. Single dimension
2. Multi dimension

Single- or one-dimensional array

* It is used to represent and store data in a linear form.
* Array having only one subscript is called as one-dimensional value.

Two-dimensional array:

* It is used to represent matrix.
* No of bytes required for 2 dimensional=no of rows \* no of columns \* size of data type.

Structures:

* Structures used to store for the different data types.
* Here we will use struct keyword.

Syntax:

Struct taganme

{

Datatype meber1;

Datatype member2;

Datatype member3;

}

E:g:

Struct student {

Char name [64];

Char course [12];

Int age;

Int year;

};

* Each variable declared in structure is called as member.
* Name given to the structure is called tag.
* Members structures are enclosed within the opening and closing braces.
* Like primary variables structure variables can also be initialized when they are declared.
* Structure templates can be locally or globally. If it is local then it can be used within that function
* If it is global, it can be used by all other functions of program.
* If initializer is less than no of structure variables. Automatically rest of values taken as zero.
* Dot operator is used to access elements.

Structure variable;

S1.anme[];

S1.roll;

S1.age;

* Elements of structurer stored in contiguous memory locations.
* Value of structure variable can be assigned to another variable of same data type using assignment operator.
* Unary, relational, arithmetic, bitwise operators are not allowed within structure variable.
* Size of structure can be found by using size of () of operator

Size of(s1);

Union:

* Unions is almost same as structures. Except one key difference structures allocate enough space to store all their members, whereas unions can only hold one member value at a time. Due to this only one member can store data at the given instance.
* Here we will us e union keyword.
* We use. operator to access members of a union and to access pointer variables we use the -> operator.
* Keep in mind that we have to always end the union declaration with a semicolon.
* The size of the union will always be equal to the size of the largest member of the array. All these sized elements can store the data in the same space without any overflow.

Strings

Strings are nothing but array of characters ended with \0.It indicates end of the string.

Strings can b declared and initialized by using 2 ways:

By using arrays

By using pointers

Datatype \*array\_name;

Char \*variable;

Dataytype arraryname[index];

Char name [10];

String library functions:

Strlen: no of charaters is string excluding the NULL charcter

Strcpy: copies str2 to str1 ,here string2 is source string and string1 is destination.

Strrev

Strlwr

Strupr

Strcmp

Compare two strings

Compiler and interpreter: compiler and interpreter both are used for converting high level language to machine language.

Both compiler and interpreter task are same but the working is different

Compiler read the program at a time and searches the error and lists them. If the program is error free then it is converted to object program.

When program size is large then compiler is preferred.

Whereas interpreter read the code line by line and convert into object code.it check error by statement by statement so it takes so much of time.

Machine language is the only language that understood by computer.

Assembly language overcome some insolvencies of the machine language.

The alphanumeric symbol will be added, START.

If initial condition is false, while loop would not execute its statement on other hand do while loop will be executed at least once.

#include<stdio.h> is preprocessor compiler directive it is not statement so it does not end with semicolon.

Computer works:

User

Application

Operating system

hardware

Round a number:

Ceil():function rounds a number upwards to its nearest integer.

Floor():function rounds a number downwards to its nearest integer. n

Enumerators:

An enum is a keyword it is a user defined data type all properties of integer are applied on enumnertaion data type so size of the enumerator data type is 2byte.it works like a integer

Using enum we can create sequence of integer constant value.

Enum tagname{value1,value2,value3------}

Here enum is user defined data type

Tagname is our own variable

Value1,value2, -----set of enum avlues.

Enum week(sun,mon,tue,wed,thr,fri,sat);

It is start with zero by default and value is incremented by 1 for the sequential identifers in the list.

If constant one value si not initialized thn default sequence will be start from zero

#include<stdio.h>

Enum sbc{x,y,z};

Int main()

{

Int a;

A=x+y+z;

Printf(“sum %d”,a);

Return 0;

}

Sum=3

Type conversion or casting:

Converting one data type to another data type.

There are two types of those are:

1. Implicit
2. Explicit

Implicit:

* Also called as automatic type conversion.
* Here lower data type is assigned to higher data type.

Explicit:

* These is useful when higher data type is converted to lower data type
* Here it doesn’t happen automatically ‘here we have to mention what data type we have to convert
* Because here we are converting the Higer data type to lower data type so higher can take more bits than lower data type.so here wastage happens that’s why we have to mention.

Storage classes in c:

Every variable in c programming has 2 properties

Type& storage class

Type: type reefrs to the what type of dta type it is.

Whereas storage class determines how longit stays in existence.

* These are used to determine the lifetime, visibility, memory location. And initial values of variable.
* Initial value what would be the initial valaue.
* Lifetime it is the time btween creation and distribution of a variable.or how long the variable would exists.
* There ate four types of storage classes available in c language

1. Auto
2. Register
3. Static
4. Extern

Auto:

* The visibility of the automatic is limited the block in which they are defined.
* The scope of the automatic is limited to the block in which they are defined.
* The automatic variables are initialized to garbage by default
* The keyword used for these is auto.
* Every local variable is automatic in c language.
* Default value is garbage value.

Register:

* The variables defined in the register is allocated the memory in the CPU register.
* The initial default value for the register is 0.
* The access time of the register variable is faster than automatic variables.
* The register should only use for variables that require quick access such as counters. It should be noted that defining register does not mean that the variable will be stored in register depending on hardware and implementation restrictions.
* Here the variable is stored in register maximum size of variable=maximum size of the register.
* Register int varaibale\_name;

External or global:

* Declared outside of the function and are accessible to all functions in program
* These are global variables
* Extern varaibles are declared using the extern keyword.
* Stored in RAM
* Scopr of the variable is the global to the function
* Whereas lifetime is till end of the program.
* Default intia value is zero.
* Extern int variable\_name.
* Extern is used whenever we have multile numbers of files while we are trying to define any gloabal avaraiale 0r function. That will be used in various other fies too then we will use the extern in anther file for provideidng the refenrce of the defined function or variable.

Static:

* Used for ssttic storage purpose
* Stired in RAM
* Default value is zero

.

Number

Set

Tuple

Dictionary

List

Indexing is used to obtain the individual characters

Whereas slicing is used to obtained substring

Word[0:2]

Here starting value is always included and stop value is always excluded.

Word=’python’

>>word[-2:]

on

Word[0]=’j’ then we get error

If you need a difernt string you should create a new one.

‘j’+word[1:]

o/p:jython

lists:

lists are mutuable

square=[1,4,9,16,25]

square[0]

1

Square[:]

[1 4 9 16 25]

Square[2]=36

>>square

[1 4 36 16 25]

**Tuple:**

A tuple is acolection objects which ared ordered and immutuable the only difference between the tuple and list is tuple uses brackets whereas list uses square brackets

Tuple cannot be changed unlike lists.

**Dictionary:**

A dictobary dict() object ia an unorederd collection of data that which is represent in key value pairs.

A collection of such pairs are enlosed in curly braces.

Sets:

Sets are unorderd

Duplicate elements are not allowed.

Sets are used to store multiple items in a single variable same as the list and tuple.

If we want printrandom numner in python then we have to import the random variable

Import random

Print(random.randrange(1,10))

Square brackets are used to acces elements of the string

A=”hello world”

Print(a[1])

Check string:

To check certain phrase or charcter pesent in the a string wec na use the keyword in

Txt=”the best things in life are free

Print(“free” in txt)

True

Txt=”the best things in life are free”

If “free” in txt:

Print(“yes”free is present)

o/p: yes, free is present

check if not :

txt=”the best things in life are free”

print(“expensive” not in txt)

true

modify strings:

a=”hello world”

print(a.upper())

HELLO WORLD

A=”hello world”

Print(a.lower())

hello world

a=”hello world”

print(a.replace(“h”,”j”))

jello world

split strings:

a=”hello world”

print(a.split(“,”))

hello,world

string concatenation:

a=”hello”

b=”world’

c=a+b

print(c)

helloworld

a=”hello”

b=”world”

c=a+” “+b

print(c )

hello world

string format:

we cannot combine the string with numbers, so by using string formatting we can combine them.

Most values are true:

Almost any evluvated is true if it has some sort of content.

Any string is true,except empty strings any number is true except 0

Any list,tuple,dictoanry is true except empty ones.

Python sort list:

Sort the lsit alaphabetically.

Thuslist=[“orange”, “mango”,kiwi”,”pineapape”,”banana”]

Thuslist.sort()

Print(thuslist)

o/p:[banana,kiwi,mango,orange,pineappale]

sort the lsit numerically:

thislist=[100,50,65,82,23]

thislsit.sort()

print(thislist)

o/p:[23,50,65,82,100]

sort desecending:

thislist=[“orange”,”mango”,”kiwi”,”pineapple”,”banana”]

thislist.sort(reverse=True)

print(thilsist)

o/p:pineapple,orange,mango,kiwi,banana

sort the list descendically:

thislist=[100,50,65,82,23]

return type:

return

or

return(expression)

return(a);

there are two ways which we can pass the arguments to the function such as

call by value

calla by reference

call by value:

in the call by value copy of the actual argument is passed to the formal arguments and operation is done on formal argument

when the function is called by call by value method it doesn’t affect content of actual arguments

changes made in the formal arguments are local to the block of called functions when control back to calling functions the change will vanish.

Call by reference:

Instead of passing the value to the variable address or reference is passed and the function operate on address of the variable rather than value

Actual parameters can be constants,varaibles, or expressions

C=add(a,b);

C=add(a+5,b);

C=add(10,20);

Formal parameters should be only varaibles, expressions and constnate are not allowed.

Actual parameters seend values to te formal parameters formal parameters receive values from the sctual aorameter

Herein cala by value balue being passed is stored in the stack memory location if yu change the value of function parameter it is chamged for the current function only

In call by refernce function is called with address of actual parameter and the foral arameters receive the addresses of tehe actual parameter now the fromala oarameter don’t contiand the values insed of eteh it contains an address it is calaled as pointer avaraible.

Using pointer varaoble the value of eteh actual parameter get changed

In call by refernce the mmory loctaions are same

Whereas in allaby value the memory loctaions are differtnt.

return(a+b;

there are mainly 5types of errors:

syntax error

run time error

linker error

logical error

semantic error

syntax errors:

these errors generally made by beginners,because they are new to language these errors can be easily corrected or debugged.

int a;

Int a;//incorrect form

Thse errors generally occurred due to mistakes while typing or do not follow the synatax of the specified lanagauge.

Run time errors:

When the program is running and it is not able to perform operation is the main cause for it.

Complilation does not find these errors.

#include<stdio.h>

int main()

{

Int a=2;

Int b=2/0;

Printf(“the value b is %d”,b);

return 0;

}

Linker error:

Linker error are mainly generated when the excetable file of the program is not created.

Logicl error:

The logical error is error that leads to undesired output

These errors produce the incorrect output.

For(i=0;i<=10;i++);

Semantic error:

When the statements are not understanadable by the complier these type of error occur.

#include<stdio.h>

Int amin()

{

Int a,b,c;

a=2;

b=3;

c=1;

a+b=c;

return 0;

}

Input and output functions:

1. Formatted i/o functions
2. Unformatted i/o functions

Formatted i/o fucntions:

It operate son various types of data

Printf()

Scanf()

Unformatted i/o fucntions:

It operates on only on charcter type.

Getchar()

Putchar()

Puts()

Gets()

Getche()

Cgets()

Cputs()

Assoctivity and precedence

Associativity specifies the order in which the operators evaluvated with same precedence iin a complex expression

Asssocity is in 2 types:

Left to right

Right to left

Some default functions in c

Some predefined macros:

* \_\_DATE\_\_ represents date in mm/dd/yyyy
* \_\_TIME\_\_ represents current time in HH:MM:SS format
* \_\_FILE\_\_ represents current file name
* \_\_LINE\_\_ represents current line number
* \_\_STDC\_\_

#include<stdio.h>

Int main()

{

Printtf(“file :%s\n”,\_\_FILE\_\_);

Printf(“date :%s\n”\_\_DATE\_\_);

Printf(“time :%s\n”,\_\_TIME\_\_);

Printf(line%d\n”\_\_LINE\_\_);

Printf(“ANSI %d”,\_\_STDC\_\_);

}

A=3\*\*2;

B=3^2 hare these two are invalid.

Instead of these

#include<stdio.h>

Int main()

{

Int a;

A=pow(3,2);

Printf(“%d”,a);

Return 0;

}

Dynamic memory allocation:

It means that the memory will be allocated at the run time.

Memory management functions are used for allocating and freeing memory.

These fucntions are defined as stdlib.h

Dynamic memory allaoctaion in c lanagauge is possible by 4 fucntions of stdlib.h header file

Malaloc

Calloc

Realaloc

Free

Static:

Memoery alalocted at the compile time

Memeory can’t be execting while exceting the program

Used in arrays.

Dynamic:

Memeory alalocte drun time

Memory can be uncresed whiel execting program

Used ina linke dlist

Methods:

Malloc() allocates single block of requested memory

Calaloc() alaloctes multiple block of requsted memory

Realloc() realaloctes the memory occupied by malaloc or calaloc fucntions

Free() frees the dynamically alalocte dmemory

Global varaibles and tsatatic varaaibles are stored ini permanent storage area whereas local varaaibles get stored in area called stack

The memory space between these two rgeions is known as heap area this region is used for dynamic memory allocation,during execution the size of the heap keep changes.

Malloc():

It stands for memory alaloctaion

The malaloc function aaloctes single block of the requsted memory at runtime .this fuction reserves a block of memory of given size and returns a pointer of type void

This means that he we can assign it to any type of pointer using type casting

It doesn’t initialize memory at exection time so it is having garbage value initially.

If it fails to allocate the enough space then it will returns a NULL pointer.

Syntax;

Ptr=(cast\_type)mallaoc(byte\_size)

Int \*x;

X=(int x)malloc(100\*sizeof(int));

Free(x);

This statement will allocates either 200 or 400bytes according to the size of the 2 or 4 bytes respectively.

And pointers points to the address of the first byte of memory.

Calloc:

Stands for contiguous allocation

Calloc is another memory allocation function that is used for allaocting memory at runtime

Calaloc function is normally used for alalocting memory to derived data types like ararays and structures

The callaoc function allocates multiple block pf requsted memory

It initializes or sets alala values to zero.

If it fails to locate enough space it returns null pointer

The only difference between the caloloc and malloc is malloc single block of memory whereas calloc multiple block of memory

Syntax:

Ptr=(cast\_tyepe\*)calloc(number,elemnt\_size)

Calloc() requires 2 argumenst count and size type

Count wil provides no of eelmnts

Size-type is data type size

EX:

Int \*arr;

arr=(int \*)calloc(10,size of(int));

char \*str;

str=(char\*)calloc(50,size of(char));

realloc():

changes memory size that is already allocated to a variable

if the previously memory is insufficient or more than required you can change the previously allocated memory size using realloc()

by using realloc() we can create the memory dynamically at middle stage.

Realaloc will creates the memory in bytes format and initial value is garbage.

Relaaloc will reqire 2 types of arguments of type void\*

Size\_type()

Free:

When your program comes out operating syatem automatically release alal the memory alalocted by your program but as a good practice when you are not in a ned of memory anymore tehn you should release that memory by calling the fnction free().

Dynamicllcy create dmalaloc() or claloc() doesn’t get freed on its own you must explixtly use free() to release

Syntax:

Free(ptr);