6th Unit

C Programming Language

(Q)(1) What is language ?

→ A language is a set of Instructions. Generaly if you take hindi, english, Odia Language. So generaly Languages we are using to comunicate, if you want to comunicate the another person we are passing instruction using a perticular language. But while using a language we need to follow some of instruction.

(2) what is computer language ?

→ Computer language is also for comunication with a computer. computer language is a set of instruction.

Computer language

↓

Set of programs

↓

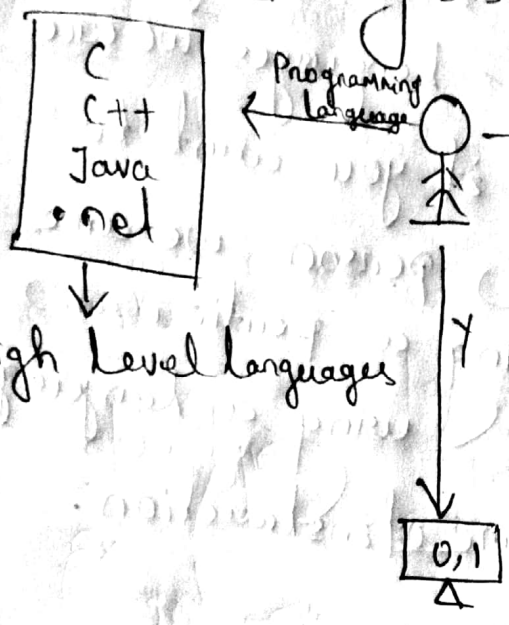
Application / software

→ Computer Lanagauge is a software

→ Computer language is a predefined application

Need of computer language

If one person comunicate with the another person generaly they have to share information (passing instruction).



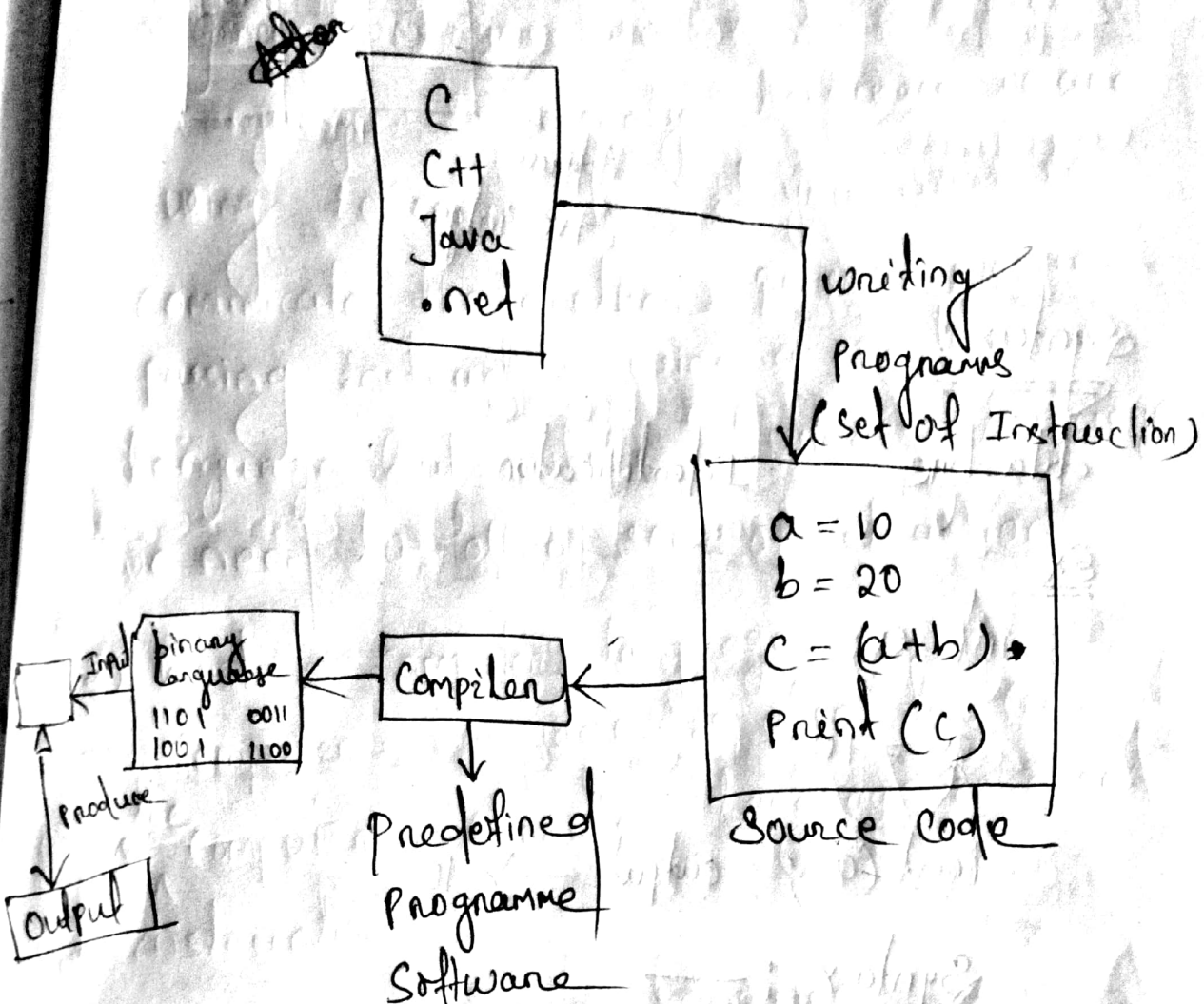
→ A complex operation ~~manipulate~~ manualy it is better to interact with electronic machine.

Ex - Calculator → what even the number give as an input fraction of second she will get the output.

→ If the person want to comunicate with the computer the person need to pass instruction in a machine code only.

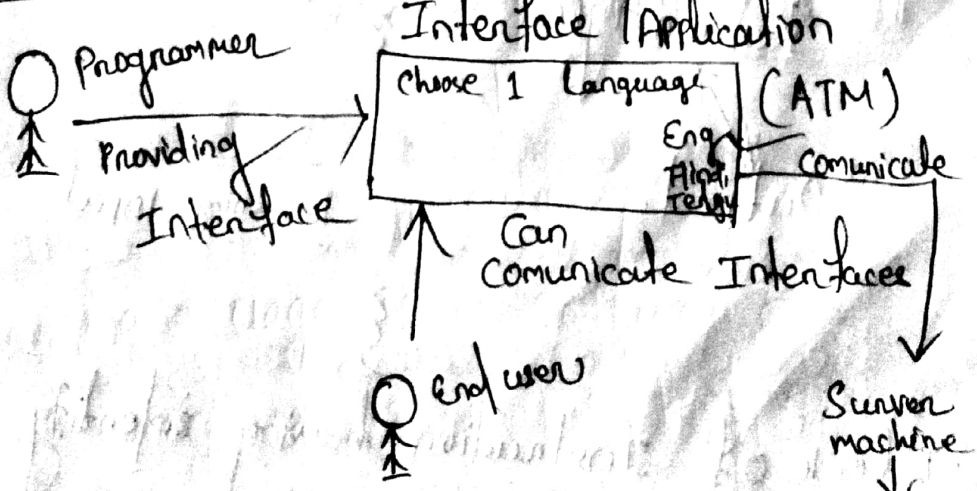
→ Computer understand binary language. (0 and 1).

→ After learning Language we are writing programs.

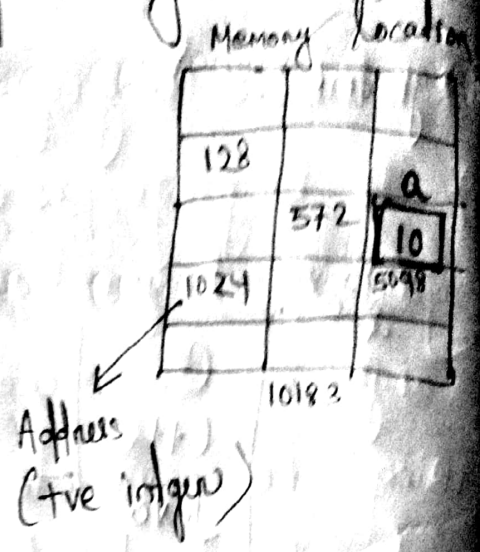


(3) What is an Interface ?

Ex- ATM



Variable : (Named memory location)



Syntax :

data type Identification

Ex

```
int a;
a = 10;
Print (a);  output → 10
```

Syntax :—

```
Identity (arguments list)
{
Return type
Output.
Processing Input
}
```

Input

→ Block of Instruction having identity which is taking Input, Processing the input and producing

the output.

Ex:

```
add (int x, int y)
{
    int z = x+y;
    return z;
}
```

Type Conversions

→ The type conversions are automatic only when the data types involved are built-in types.

```
int m;
float x = 3.14159;
m = x; // convert x to integer before
       its value is assigned //
       to m.
```

→ For user defined data types, the compiler does not support automatic type conversions.

→ We must design the conversion routines by ourselves.

Type Casting

→ some types can be converted (or typecast) to others in C.

→ Typecasting is often unavoidable and essential.

→ Assume :

- x has type type-a
- y has some type that can be cast to type-a.

→ cast y to type-a and assign into x :

- x = (type-a) y;

(11) Decision Control

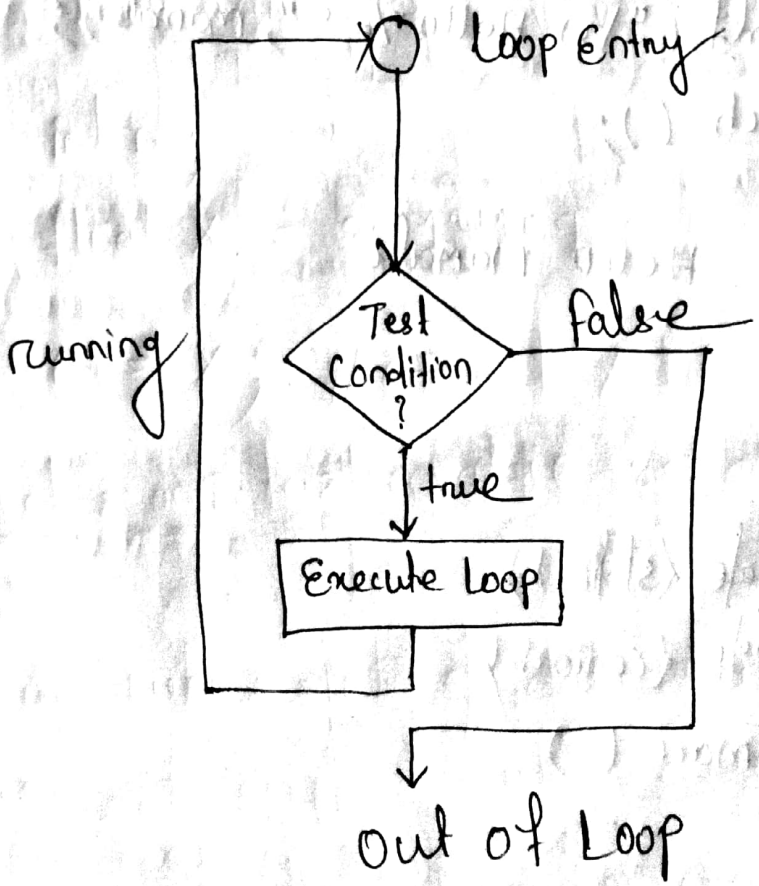
The decision control statement are the decision making statements that decides, the order of execution of statement based on the conditions. In the decision making statement the programmer specify which conditions.

are to be executed on tested with the statements to be executed if the condition is true or false.

Looping statement

These are the statement execute one or more statement repeatedly several number of times. In C programming language there are theree types of Loops;

→ while
→ for
→ do-while



① Write a Program in C to enter your name and display ?

```
# include <conio.h>
# include <stdio.h>
void main ()
{
clrscr ();
char name [20];
Printf ("Enter your name : Manali");
Scanf ("%. s", Manali);
Printf ("\n Hello %. s", Manali);
getch ();
}
```


Out put – Hello Manali

Another way

```
# include <stdio.h>
# include <conio.h>
Void main ()
{
clrscr ();
Print (" manali");
```

```
getch ( );
}
```

Output - Mamali

② // Prog to print sum of two values //

```
# include <stdio.h>
# include <conio.h>
void main ( )
{
int num1 , num2 , num3;
clrscr ( );
Printf (" \n Enter the two numbers : ");
Scanf ("%d %d , & num1, & num2);
num3 = num1 + num2;
Printf ("\ sum of two numbers = %d , num3);
getch ( );
}
```

③ /* WAP to check whether the input integer number is +ve or -ve */

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int num;
  printf("Enter a number: \n");
  scanf("%d", &num);
  if (num>0)
    printf("%d is a positive number
                \n", num);
  else if (num<0)
    printf("%d is a nagative number
                \n", num);
  else
    printf("0 is neither positive nor
                negative");
}
```

output -1

→ Enter a number 0.0 is neither +ve or -ve

output 2 → Enter a number -3 . -3 is -ve

④ /* wap to find greatest of three numbers */

```
# include <stdio.h>
# include <conio.h>
int main( )
{
  int num1, num2, num3;
  printf ("\n Enter value of num1, num2 and num3:");
  scanf (" %d %d %d, &num1, &num2, &num3);
  if ((num1 > num2) && (num1 > num3))
    printf ("\n Number 1 is greatest ");
  else if ((num2 > num3) && (num2 > num1))
    printf ("\n Number2 is greatest");
  else
    printf ("\n Number 3 is greatest ");
  return 0;
}
```

Output

Enter value of num1, num2 & num3

1 2 3

15, 200, 100

Num2 is greater.

7th UNIT

Advanced features of C

In C programming language, there are two methods to pass parameters ~~values are copied to formal parameters and these formal parameters are~~ from calling function to called function and they are as follows.

→ Call by value

→ Call by Reference

Call by value	Call by Reference
→ Calling function sends copies to data	→ calling function send address of data.
→ The formals parameters are ordinary variables.	→ The formal parameters are pointer variables.
→ ATmost only one values can be sent back to the calling function.	→ Several result can be sent back to the calling function.
→ Actual parameters are affected by changes made with in the function.	→ Direct changes are made to the actual parameter.

Call by value Example

```
# include <stdio.h>
# include <conio.h>
 Void main ( )
{
  int num1, num2;
  void swap (int,int );
  clrscr ( );
  num1 = 10;
  num2 = 20;
Printf ("\n before swap:
      num1 = %d, num=2
      = %d, num1 ,num2);
  swap (num1, num2);
Printf ("\n After swap:
      num1 = %d \n num=2=%d"
      num1, num2);
   getch ( );
}
 void swap(inta, intb)
 {
  int temp;
  temp =a;
  a = b;
  b = temp;
}
```

Call by Reference Example

```
# include <stdio.h>
# include <conio.h>
Void main ( )
{
int num1, num2;
void swap (int*, int*);
 clrscr ( );
  num1= 10;
  num2= 20;
Printf("\n Before swap: num1=%
d, num2= %d", num1, num=2);
swap (&num1, &num2);
Printf("\n After swap: num1=%d
num=2 = %d "; num1, num2);
 getch ( );
}
void swap (int *a, int *b)
 {
 int temp;
 temp = *a;
*a = *b;
*b = temp;
}
```

Scope of variable

→ It is a similar concept applies to variables in C. variable scope refers to the accessibility of a variable in a given program of function. For example, a variable may only be available with in a specific function or it may be available to the C program.

```
# include <stdio.h>
# include <conio.h>
int main( )
{
```

Entering into the block →

```
{
    int x = 10;
    {
     Printf ("x = %d", x);
    }
    Printf ("\n x = %d", x);
```

Exit from the block →

```
}
return 0;
}
```

scope of block variable