BASICS OF C++

HELLO WORLD

```
#include<iostream>
int main(){
   std::cout<<"hello world";
   return 0 ;
}
```

OR

```
#include <iostream>
using namespace std;
int main(){
cout<<"hello world";
   return 0;
}
```

PRINTING A VARIABLE

```
#include <iostream>
using namespace std;
int main(){
 //printing a variable
 int a=4;
 cout<<"the variable is "<<a;
 return 0;
}
```

ADDITION OF TWO NUMBERS

```
#include<iostream>
using namespace std;
int main(){
    int num1,num2;
//<< this is called insertion operator
cout<<"enter the value of num1 :\n";
//>> this is called extraction operator
cin>>num1;
cout<<"enter the value of num2 :\n";
cin>>num2;
cout<<"the sum of num1 and num2 is "<<num1+num2;

return 0 ;
}
```

#. [OPERATORS IN C++]

ARTHMETIC OPERATOR

```
#include <iostream>
using namespace std;
int main(){
  int a=3,b=4;
  cout<<"the value of a + b is "<<a+b<<endl;
  cout<<"the value of a - b is "<<a-b<<endl;
  cout<<"the value of a * b is "<<a*b<<endl;
  cout<<"the value of a / b is "<<a/b<<endl;
  cout<<"the value of a % b is "<<a%b<<endl;
  cout<<"the value of a++   is "<<a++<<endl;
  cout<<"the value of a--   is "<<a--<<endl;
  cout<<"the value of ++a   is "<<++a<<endl;
  cout<<"the value of --a   is "<<--a<<endl;

    return 0;
```

```
}
```

ASSIGNMENT OPERATOR

Used to assign values

```
Int a=2 ,  b=5;
Char d='d';
```

COMPARISION OPERATOR

Used to compare two values

```
#include <iostream>
using namespace std;
int main(){
  int a=3,b=4;
  cout<<"the value of a==b is "<<(a==b)<<endl;
  cout<<"the value of a!=b is "<<(a!=b)<<endl;
  cout<<"the value of a>b is "<<(a>b)<<endl;
  cout<<"the value of a<b is "<<(a<b)<<endl;
  cout<<"the value of a>=b is "<<(a>=b)<<endl;
  cout<<"the value of a<=b is "<<(a<=b)<<endl;


    return 0;
}
```

LOGICAL OPERATOR

```
#include <iostream>
using namespace std;
int main(){
  int a=3,b=4;
 cout<<"the value of logical and operator
((a==b)&&(a<b))is:"<<((a==b)&&(a<b))<<endl;
 cout<<"the value of logical or operator ((a==b)||(a<b))is:"<<((a==b)||
(a<b))<<endl;
 cout<<"the value of logical not operator (!(a==b))is:"<<(!(a==b))<<endl;
```

```
    return 0;
}
```

GLOBAL VALUE

```
#include<iostream>
using namespace std;
int c=45;
int main(){
    int a,b,c;
    cout<<"enter the value of a"<<endl;
    cin>>a;
     cout<<"enter the value of b"<<endl;
    cin>>b;
    c=a+b;
    cout<<"the sum is "<<c<<endl;
    cout<<"the value of global c is "<<::c<<endl;
    return 0;
}
```

Reference variable

```
#include<iostream>
using namespace std;
int main(){
    float x=455;
    float &y=x;
    cout<<x<<endl;
        cout<<y<<endl;

    return 0;
}
```

Typecasting

Used to change type of variable into other types of variable
For eg; int to float

```
#include<iostream>
using namespace std;
int main(){
    int a=45;
    float b=45.46;
    cout<<"the value of a is "<<(float)a<<endl;
    cout<<"the value of a is "<<float(a)<<endl;

    cout<<"the value of b is "<<(int)b<<endl;
    cout<<"the value of b is "<<int(b)<<endl;

    //OUIZ
    cout<<"the expression is "<<a+b<<endl;
    cout<<"the expression is "<<a+int(b)<<endl;
    cout<<"the expression is "<<a+(int)b<<endl;


    return 0;
}
```

CONSTANTS

A constant variable can never be changed

```
#include <iostream>
using namespace std;
int main(){
    const int a=45;
    cout<<a<<endl;

    cout<<a<<endl;
    return 0;
}
```

MANIPULATOR

Control code display

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
int a=3;
int b=45;
int c=4736;
cout<<"the value of a without setw"<<a<<endl;
cout<<"the value of b without setw"<<b<<endl;
cout<<"the value of c without setw"<<c<<endl;

cout<<"the value of a with setw"<<setw(6)<<a<<endl;
cout<<"the value of b with setw"<<setw(6)<<b<<endl;
cout<<"the value of c with setw"<<setw(6)<<c<<endl;

   return 0;
}
```

#. [C++ CONTROL STUCTURE]

ELSE IF

```
#include <iostream>
using namespace std;
int main(){
   int age;
   cout<<"tell me your age"<<endl;
   cin>>age;
   if (age<18){
      cout<<"you can not come to my party"<<endl;
   }
   else if(age==18){
      cout<<"you will get a kid pass if you want to come"<<endl;
   }
   else if(age>18){
      cout<<"you can come to the party"<<endl;
   }
   return 0;
}
```

SWITCH CASES STATEMENT

```
#include <iostream>
using namespace std;
int main(){
    int age;
    cout<<"tell me your age"<<endl;
    cin>>age;
   switch(age)
   {
       case 18:
       cout<<"you are 18"<<endl;
        break;
        case 54:
       cout<<"you are 54"<<endl;
        break;
        case 5:
       cout<<"you are 5"<<endl;
        break;
        default:
        cout<<"no special cases"<<endl;

   }
    return 0;
}
```

#. LOOPS IN C++

FOR LOOP

```
For(initialization ; condition ; updation)
{
Loop body (c++ code)
}
```

```
#include<iostream>
using namespace std;
int main(){
    for(int i=0;i<=100;i++){
```

```
        cout<<i<<endl;
    }
    return 0;
}
```

WHILE LOOP

```
While (condition)
{
c++ code
}
```

```
#include<iostream>
using namespace std;
int main(){
    int i;
    while(i<=100){
        cout<<i<<endl;
        i++;
    }
    return 0;
}
```

DO WHILE LOOP

```
Do{
c++ code
}
While (condition);
```

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
   do {
        cout<<i<<endl;
        i++;
```

```
   }while(i<=100);
    return 0;
}
```

{{{{{{{{{{   TABLE OF 6 USING DO WHILE LOOP.   }}}}}}}}

```
#include<iostream>
using namespace std;
int main(){
    int i=6;
    int j=1;
   do {
       cout<<i*j<<endl;
       j++;
   }while(j<=10);
    return 0;
}
```

BREAK AND CONTINUE STATEMENT

BREAK

It will break the code at that particular point

```
#include <iostream>
using namespace std;
int main(){
    for(int i; i<40 ; i++){
        cout<<i<<endl;
        if (i==7){
            break;
        }
    }
    return 0;
}
```

CONTINUE
It will skip the code at that particular point

```
#include <iostream>
using namespace std;
int main(){
    for(int i; i<40 ; i++){

        if (i==7){
            continue;
        }
        cout<<i<<endl;
    }
    return 0;
}
```

POINTERS
Data type which holds the address of other datatypes

```
#include<iostream>
using namespace std;
int main(){
    int a=3;
    int*b=&a;
    //   &--->. (address of) operator
    cout<<"the address of a is "<<&a<<endl;
     cout<<"the address of a is "<<b<<endl;


     //   *--->. (value at) Derefrence operator
     cout<<"the value at address b is "<<*b<<endl;
```

```
    return 0;
}
```

{{{{{{{{{{{   POINTER TO POINTER   }}}}}}}}}}}

```
#include<iostream>
using namespace std;
int main(){
    int a=3;
    int*b=&a;
    //   &--->. (address of) operator
    cout<<"the address of a is "<<&a<<endl;
     cout<<"the address of a is "<<b<<endl;
     int**  c=&b;



     //   *--->. (value at) Derefrence operator
     cout<<"the value at address b is "<<*b<<endl;
     cout<<"the value at address of c is "<<c<<endl;
     cout<<"the value at address value_at(value_at(c)) is "<<**c<<endl;

    return 0;
}
```

ARRAYS

Used to store many variables in one variable.

```
#include <iostream>
using namespace std;
```

```
int main(){
   int marks [4]={35,45,67,87};
   cout<<marks[0]<<endl;
   cout<<marks[1]<<endl;
   cout<<marks[2]<<endl;
   cout<<marks[3]<<endl;

   return 0;
}
```

{{{{{{{{{{{{{{{{ PRINTING ARRAYS USING FOR LOOP }}}}}}}}}}}}}}}}

```
#include <iostream>
using namespace std;
int main(){

   int marks [4]={35,45,67,87};
   for(int i=0 ; i<4 ; i++){
      cout<<"the value of marks "<<i<<" is "<<marks[i]<<endl;
   }

   return 0;
}
```

POINTERS AND ARRAYS

Arrays Mei
Jo naam hota hai waai address phi hota hai.
So (((&naam ——> WRONG)))

```
#include <iostream>
```

```
using namespace std;
int main(){
    int marks [4]={35,45,67,87};

    int* p=marks;

    cout<<"the value of marks [0] is "<<*p++<<endl;
    cout<<"the value of marks [1] is "<<*p++<<endl;
    cout<<"the value of marks [2] is "<<*p++<<endl;
    cout<<"the value of marks [3] is "<<*p++<<endl;
    return 0;
}
```

STRUCTURE

It is a user defined datatype
Used to store different types of variables (eg. int , float , char)

```
#include <iostream>
using namespace std ;
struct employee{
    int ID;
    float salary;
    char favChar ;
};
int main()
{
    struct employee harry;
    harry.ID=1;
    harry.favChar='c';
    harry.salary=43267732564;


    cout<<harry.ID<<endl;
    cout<<harry.favChar<<endl;
    cout<<harry.salary<<endl;
    return 0;
}
```

✯ TYPEDEF ✯ (V.IMP)

Used to give name or short form to datatype.
{ eg. raise niche employee ki jagha ab sirf ep hi likhna padega }

```
#include <iostream>
using namespace std ;
typedef struct employee{
   int ID;
   float salary;
   char favChar ;
}ep;
int main()
{
   ep harry;
   harry.ID=1;
   harry.favChar='c';
   harry.salary=43267732564;


   cout<<harry.ID<<endl;
   cout<<harry.favChar<<endl;
   cout<<harry.salary<<endl;
   return 0;
}
```

UNION

In structures each value is stored
But in union only single value is stored
Because storage is shared between values
If you put two or more values it will give you garbage value as output

```
#include <iostream>
using namespace std ;
union money{
    int money1;
    float money2;
    char money3;
};
int main()
{
    union money vashu;

    vashu.money3='c';

    cout<<vashu.money3;
    return 0;
}
```

{{{{{{{{   EXAMPLE OF GARBAGE OUTPUT   }}}}}}}}

```
#include <iostream>
using namespace std ;
union money{
    int money1;
    float money2;
    char money3;
};
int main()
{
    union money vashu;

    vashu.money3='c';
    cout<<vashu.money1;
    cout<<vashu.money2;
    cout<<vashu.money3;
    return 0;
```

```
}
```

ENUMS

Used to allot integers to given things , starting from 0 to further

```
#include <iostream>
using namespace std ;
int main()
{
    enum HOMIES {vashu , puru , tejas , daksh , himanish};
    HOMIES h3=tejas;
    HOMIES h5=himanish;
    cout<<h3<<endl;
    cout<<h5<<endl;

    return 0;
}
```

FUNCTION

Used to run code in parts

```
#include<iostream>
using namespace std;
int sum(int a, int b){
    int c=a+b;
    return c ;
}
int main(){
```

```
    int num1,num2;
    cout<<"enter the value of num1"<<endl;
    cin>>num1;
    cout<<"enter the value of num2"<<endl;
    cin>>num2;
    cout<<"the sum of num1 and num2 is "<<sum(num1,num2);
    return 0;
}
```

Formal parameter————>
Actual parameter ————>

————> num1 and num 2 are actual parameters.
————>a and b are formal parameters.

FUNCTION PROTOTYPE

If you declare a function after a code it will not work
In that case we use function prototype
It gives surety to your program that the function is ahead

```
#include<iostream>
using namespace std;
int sum(int a, int b);

int main(){
    int num1,num2;
    cout<<"enter the value of num1"<<endl;
    cin>>num1;
    cout<<"enter the value of num2"<<endl;
    cin>>num2;
    cout<<"the sum of num1 and num2 is "<<sum(num1,num2);
    return 0;
}

int sum(int a, int b){
```

```
    int c=a+b;
    return c ;
}
```

VOID FUNCTION

Function which don’t take anything
And neither give any output

```
#include<iostream>
using namespace std;
int sum(int a, int b);
void g(void);


int main(){
    int num1,num2;
    cout<<"enter the value of num1"<<endl;
    cin>>num1;
    cout<<"enter the value of num2"<<endl;
    cin>>num2;
    cout<<"the sum of num1 and num2 is "<<sum(num1,num2);
    g();
    return 0;
}


int sum(int a, int b){
    int c=a+b;
    return c ;
}
void g(){
    cout<<endl<<"hello"<<endl<<"good morning";
}
```

CALL BY REFFERENCE USING POINTER

```
#include<iostream>
using namespace std;
void swap(int* a, int* b){
  int temp=*a;
   *a=*b;
   *b=temp;

}

int main(){
   int x=4, y=6;
   cout <<"the value of x is "<<x<<"the value of y is "<<y<<endl;
   swap (&x,&y);
   cout <<"the value of x after swap is "<<x<<"the value of y after swap is
"<<y<<endl;
   return 0;
}
```

INLINE FUNCTION

```
#include <iostream>
using namespace std;
inline int product(int a , int b){
   return  a*b;
}
int main(){
   int a ,b;
   cout<<"enter the value of a and b"<<endl;
   cin>>a>>b;
   cout<<"the product of a and b is "<<product(a,b)<<endl;
   cout<<"the product of a and b is "<<product(a,b)<<endl;
```

```
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    return 0;
}
```

STATIC VARIABLES

The value of variable is retained

```
#include <iostream>
using namespace std;
inline int product(int a , int b){
   static int  c=0;
   c=c+1;
    return  (a*b)+c;
}
int main(){
    int a ,b;
    cout<<"enter the value of a and b"<<endl;
    cin>>a>>b;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    cout<<"the product of a and b is "<<product(a,b)<<endl;
    return 0;
}
```

RECURSIONS

A function keep call itself until the base condition meets

```
#include <iostream>
using namespace std;
int factorial( int n){
    if (n<=1){
        return n;
    }
    return n* factorial(n-1);
}
int main(){
    int a;
   cout<<" enter a number "<<endl;
   cin>>a;
   cout<<"the factorial of a is "<<factorial(a)<<endl;
    return 0;
}
```

FUNCTION OVERLOADING

```
#include <iostream>
using namespace std;
int sum(int a, int b){
cout<<"using function with 2 arguments"<<endl;
    return a+b;
}
int sum(int a ,int b ,int c){
    cout<<"using function with 3 arguments"<<endl;
    return a+b+c;
```

```
}
int main(){
    cout<<"the sum of 2 and 3 is "<<sum(2,3)<<endl;
    cout<<"the sum of 2 and 3 and 4 is "<<sum(2,3,4)<<endl;

}
```

```
#include <iostream>
using namespace std;

//volume of cube
int volume(int side){
return (side*side*side);
}

//volume of cylinder
int volume (int radius , int height ){
return (3.14*radius*radius*height);
}

//volume of cuboid
int volume(int length ,int breath , int height){
   return (length*breath*height);
}


int main(){
    cout<<"the volume of cube of side 2 is "<<volume(2)<<endl;
    cout<<"the volume of cylinder of radius 2 and height 3 is
"<<volume(2,3)<<endl;
    cout<<"the volume of cuboid of lenght 2 and breath 3 and height 4 is
"<<volume(2,3,4)<<endl;

}
```