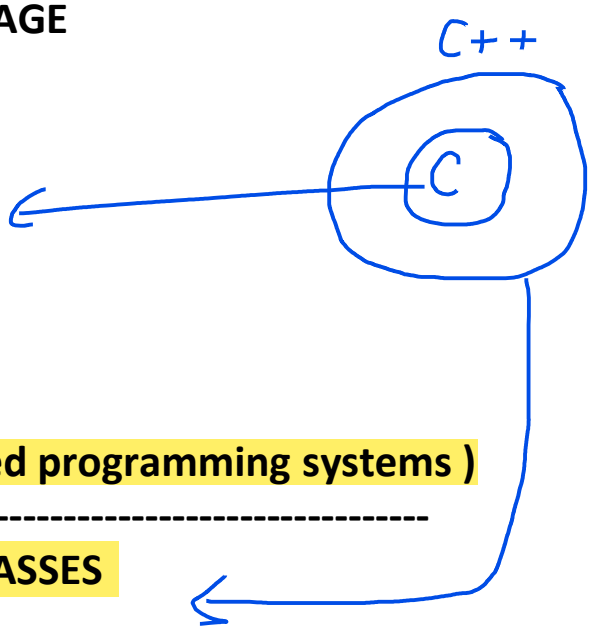
C LANGUAGE



1. control statements
2. array , matrix , string
3. function
4. structrue
5. pointer
6. preprocessor

fops (function oriented programming systems)

C++ OR C WITH CLASSES

1. CLASS AND OBJECT
2. INHERITANCE
3. TEMPLATE
4. EXCEPTION HANDLING
5. POLYMORPHISM

OOPS(OBJECT ORIENTED PROGRAMMING SYSTEMS)

---

object :- noun (place , person , things , ...)

security , user - friendly

example of oops --> windows

C	C++
stdio.h	iostream.h
printf()	cout << (insertion operator)

scanf () **cin** **>> (extraction operator)**

FUNCTION **OBJECT**

\n **endl** **(end line)**

---

1. **printf(" enter two nos \n ");**

   A. cout << " enter two nos " ;
   B. cout << " enter two nos " << endl ;

   C. cout << " enter two nos " << "\n" ;

2. **int a ;**
   scanf("%d", &a);
   cin >> a ;

3. **int a;**
   float b;

   cin >> a >> b ;

4. **printf(" sum = %d \n ", c); // sum = 6**
   cout << " sum = " << c << endl; // sum = 6

// WAP FOR ADDTION OF TWO NOS

SAVE :- Add.cpp [computer]
Mobile C
SAVE :- Add.cc

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

  int    main()
  {
       int   a , b , c ;
              cout << " ENTER TWO NOS " << endl;
              cin >> a >> b;
             c = a + b;
        cout << " SUM = " << c << endl;
   }

//    :: ( scope  resolution  operator )

using namespace std;
#include<iostream>

  int    a = 3 ; // global
  int    main()
  {
        int  a = 5 ;  // local
              cout << a << endl;    // 5
              cout << ::a << endl; //  3
              {
                    int  a = 7;
```

```
            cout << a << endl;    // 7
            cout << ::a << endl; //  3
        }
  }
```

FUNCTION OVERLOADING :- fun. name :- same

argument :- different

sum :- int , float

Adding of two nos using Fun. overloading

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

 void   sum ( int  a , int  b )
 {
        int   c;
        c = a + b;
        cout << " sum = " << c << endl;
  }
  void   sum ( float  a , float  b )
  {
        float  c;
        c = a + b ;
        cout << " sum = " << c << endl;
  }
```

```
int    main()  // user
{
        sum( 2 , 5 );      // integer

        sum( 3.5f, 5.3f ); // float
}
```

// 2. max -> int , float , char

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

int    max ( int  a , int  b)
{
        if( a > b )
            return(a);
        else
            return(b);
}
float  max ( float  a , float  b )
{
       return ( (a > b) ? a : b );
}

char   max ( char   a , char  b )
{
       return( (a > b ) ? a : b ) ;
}
```

```
int    main()  // user
{
        cout << "\t\t\t " << max( 3 , 5 ) << endl ;       // 5
        cout << "\t\t\t " << max( 4.3f , 2.7f ) << endl; // 4.3
        cout << "\t\t\t " << max( 'A' , 'a') << endl ;    // 'a'
 }
```

int t;
t = max(3,5)
cout << t << endl;

REFERENCE VARIABLE (alise variable)

(only in c++)

TYPES OF VARIABLES

1. NORMAL VARIABLE `int a ;`
2. POINTER VARIABLE `int *p ;`
3. REFERENCE VARIABLE `int &b ;`

```
int&  b ; // int &b;
```

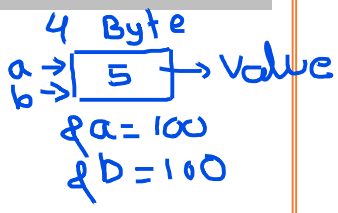
type of b = int&

b = value

&b = self - address

// DECLARE AND INTIALIZATION OF REFERENCE VARIABLE

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



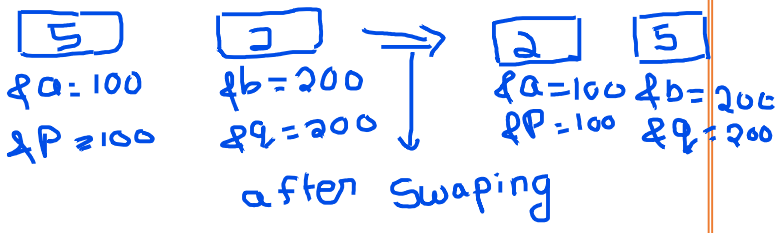

```
int   a = 5 ;
int   &b = a  ;  // declaration and initialization

cout<< " ADDRESS OF a = " << ( unsigned int) &a << endl; // 100
cout<< " ADDRESS of b = " << ( unsigned int) &b << endl; // 100

 b = 3 ;
cout<<  a << endl; // 3
cout<<  b << endl; // 3
 }
```

// call by reference

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

void  swap(int &p, int &q);

int    main()
{
       int   a = 5 , b = 2 ;

       swap(a,b);  // call by reference

       cout<<" a = " << a << endl; // 2
       cout<<" b = " << b << endl; // 5
 }
```

a = 2
b = 5
change

```
void  swap( int  &p , int  &q )
{     int  c;
      c =  p;     // c = 5
      p =  q;   //  p = 2
      q  = c;  //   q = 5
}
```



POINTER VARIABLE	REFERENCE VARIABLE
1. NEW MEMORY (2 byte)	1. NEW MEMORY x
2. NULL	2. NULL x
3. DECLARE int *p;	3. DECLARE AND INITIALIZE int &b = a ; int &b ; x
4. MANY TIME INITIALIZE int a = 2 ,b = 3 , *p ; p = &a ; *p = 2 p = &b ; *p = 3	4. ONE TIME INITIALIZE int &b = a ; int &b = c ; x
5. * , -> (dereference operator)	5. * , -> x int &d = a ;