C++	C
OOP	Procedure programming
1) Pass object oriented programming	
2)	
3)	
4)	

class :-	obj
→ Collection of Related data & function	→ Real world entity/
→ describe the common property & action/ method of object	→ variable of class type
→ blue print for object.	→ which empliment the property describe by class.

class obj
↓ data type ↓ variable.

```
class class name
        ↑ identifier
{
  Body < Private :
         Public :  ] keyword
                   show visibility level
};
```

Data → Private

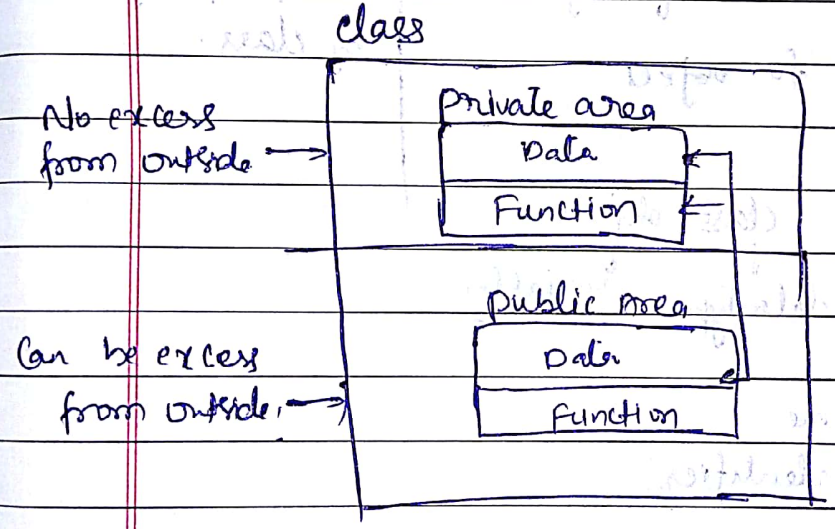
function → Public

Scanned by CamScanner

ex:

```
class Student
{ private:  → excess within the class
    int roll;
    char name[10];   ] Data members

  Public:  → excess outside the class:
    void getdata (void);
    void putdata (void);   ] function member or member function.
};
void main()
{
  Student St1;
  St1. getdata();
```



* Data abstraction

→ Class which contain pure virtual function: called
→ Base class or abstract class.

Operator in C

1) Arithematic +, -, *, /, %

/ → Quotient operator

% → Remainder.

op_1 / op_2 sign is same as op_2

2) '=' Assignment operator

Sum = (a+b)

assigned in Sum.

3) Relational operator

=, >, <, <=, >=

4) logical operator.

(&&) AND (||) OR (!) NOR.

5) unary operator (++, --, (-)

-- → decreament operator

++ → increament operator

pre ++x

post x++

ex:- y = x++

~~y = x+1 → statement (2)~~

~~y = (x) → statement (1)~~

Scanned by CamScanner

Conditional operator

exp_1 ? exp_2 : exp_3

if exp_1 is true print exp_2 and if exp_1 is False, print exp_3

ex:

```
(N%2 == 0) ? printf(even) : printf(odd);
(N%2 == 0) ? cout(even) : cout(odd);
```

Bitwise operator

(|)	(&)	(^)	(>>)
OR	and	ex-or	Right shift
<<	~		
left shift	Compliment		

Scanned by CamScanner

```
include
#<iostream.h>
# include <conio.h>

class sample  -> class defination.
{
 public:      -> Access identifier.
 void sayhello()
 {
  cout<<"hello"
        } };

void main()
{
 Sample s ;  -> object definition
 s. sayhello();
 getch();
 }


# include <iostream.h>
# include <conio.h>

class Sample
{ public:
 int add (int a, int b)
 {
   return (a+b) ;
 }
 int multi (int a, int b)
 { return (a*b);
```

Scanned by CamScanner

```
} };

void main ()
{
  Sample s;

  s.add();
  s.multi();
  getch();
  }.

#
include <iostream.h>
# include <conio.h>
  class Sample
  {public:
  void add (int a, int b)
  {int c=0;
   c= (a+b);
   cout<<"addition is:"<<c;
  }
   void multi (int a, int b)
   {int c=1;
    c= (a*b);
    cout<<"multiplication is:"<<c;
    } };
    void main ()
    {sample s;
```

Scanned by CamScanner

```
clrscr();
int x, y;
cout << "enter the value of x and y";
cin >>x>>y
s.add(x,y);
s.multi(x,y);
getch();
}

#include <iostream.h>
# include <conio.h>

class SI
{
Public:
void SI_I(int p, float t, float r = 5.5)
{
int i; float a;
i= (p*r*t)/100;
a= p+i;
cout << "Total amount is :" <<a;
};
void main()
{ SI a;
clrscr();
int p;
float t, r;
cout<<"please enter value of principal rate time";
cin>>p>>r>>t;
```

Scanned by CamScanner

```
a.si(p,t,r);
getch();
}
```

Q. WAP to make a class employ which contains the data member emp number, empage, emp salary, display this details entered by the user.

Q. WAP to calculate the area of circle, rectangle and square using function overloading.

```
#include <iostream.h>
#include <conio.h>

class sample
{        public:
  void area(int a, int b)
  { area = (a*b);
    cout << "the value is", << area;
  }
  void area (int a).
  { area = (a x a);
    cout << "the value is << area;
  }
  void area (float r)
  { area = 2xπxrxr;
    cout << "the value is << area;
  } };
```

Scanned by CamScanner

```
void main( )
{ clrscr( );
  cin a,b;
  bbat ;
  cout<<'
```

```
sample y ;
y.area(x);
getch( );
}
```

```
cin x, y;
cout<<"enter the
cin<<a>>y
```

* Simple if statement.

```
if ( Condition )
{ Statement-1
  Statement-2.
```

* executed if the condition is true and else part is not present

ex:

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ int salary ; b
  cout<<" enter the salary" ;
  cin >> sal ;
  if ( sal >= 10000 )
  { p = (10000 + (10000 * 10/100)) ;
    cout<<" the bonus is" << b ;
  }
}
```

sal >= 10000
p = 10000 + 10000 × 10/100

'<<' → insertion operator

Scanned by CamScanner

if-else nested

```
if (condition 1)
{ if (condition 2)        ] nested block.
    { statement 1
      ------
      statement n
    }
    else
      { statement
      }
    }
else
    { statement
    }
```

ex (1 - 9) Range 1
(10-18) Range 2
(19 - 50) Range 3

```
#include <iostream.h>
#include <conio.h>
int main ()
{ int age ;
  cout << "enter the age";
  cin >> age ;
  if (age >=10 && age <= 50)
  { if (age >=0 && age <= 18)
    { cout << "age is" << Range 2 ;
    }
```

Scanned by CamScanner

```
else
{ cout << "age lie in" << range 2 and range 3 ;
}.
else
{ if (age <= 9 )
  { cout << " range is 1" ;
  }
  else
  { cout << " Range is 3
  }.
}
```

* if else ladder

```
if (condition 1)
{ statement 1
}
else if (condition 2)
{ statement 2
}
else
{ default statement
}
```

while loop.

* while statement

use in repeated condition.

```
while (condition)
{ statement 1
  ---
  statement 2
}
```

Scanned by CamScanner

ex: print number from 1 to 100.

```
#include<iostream.h>
#include <conio.h>
void main
{ int i, i=0;
while (i<=100)
{ i++;
cout<< i;
}
getch();
}
```

→ variable initilization.
→ while (condition)
→ increasing or decreasing.

Do-while statement

while	do while
→ check the condition at entery	→ check the condition at exist point
→ entery control loop.	→ exist control loop
→ first check the condition then execute.	→ First execute then check the condition

```
do
{ statement 1
statement n
} while (condition);
```

Scanned by CamScanner

Q: print even number from 1 to 100;

```
#include<iostream.h>
#include<conio.h>
void main ()
{ int i=2
do
{ cout << i ;
  i = i+2 ;
}
while ( n<=100);
getch ();
}.
```

condition is true

→ variable initilization.
→ do.
→ while(condition);

#

```
void main ( )
{
int n ;
Cout<<"enter a number";
Cin >> n ;
[for( n=2; n<=100; n=n+2 )]
{
if (n%2==0)
{ cout<<"even" ;
}
else
  Cout<<"not even";
}
}.
```

```
void main
{
int n;
Cout<<"enter the no;
Cin >> n;
n = 2
while(n<=100)
{ n=n+2
Cout<<n=n+2;
}.
```

Scanned by CamScanner

For loop

for (initialization ; condition ; ~~increment~~/decrement)
or
updation.

{ statement 1

- - -

statement n

}.

Function in C++

→ group / Collection of code.

→ Perform some specific task.

Built in	user defined
→ Library fun	→ defined by user.
→ It is already defined in headerfile.	

ex. math.h, conio.h] headerfile.

```
# include <iostream.h>
# include <conio.h>
```

→ void swap (int *, int *)

void swap(&a, &b)

Call by referance

```
#include<iostream.h>
#include<conio.h>
    void swap( int &, int & );

void main ( )
{ int a, b;
    cout<<"ent the value" ;
    cin >>a>>b;
```

```
#include<iostream.h>
#include<conio.h>
{
  void swap( int &a , int &b)
  { int t ;
      t = a;
      a = b;
      b = t;
  cout<<"value is" <<&a <<&b ;
}
  void main ( )
  {.
    int x, y;
    cout <<"enter the value";
    cin >>x>>y;
    swap(x, y)
    cout<<"value is" <<x <<y ;
  }.
```

Scanned by CamScanner

~~Recursion~~ :-

inline function

→ It give request to compiler to put the code of the function where the function is called.

inline function heads
↓
keyword {
function body;
}

→ all the function within a class is inline function.
→ For inline function, function should contain small set of instruction.

→ It minimize the time of jump from programme to function and function to programme at time of execution.

Ex:-

```
# include <iostream.h>
# include < conio.h>
inline int max(int x, int y)
{ return((x>y)? x : y);
}
void main()
{ int a, b
cout << " enter two number";
cin >> a >> b;
cout << " greater value is " << max(a,b);
getch();
}
```

formal parameter.

Scanned by CamScanner

Friend function

↓ access

use the data of two different class. It is a non member function. It can be called by without any object. It is defined as a normal function.

1) (::) x.
2) Call directly as normal function!
3) void disp(BCA b, DCA d)

object of class.

```
{ b.roll;
  d.roll;
}
```

accessed

4) Data of Class is ~~access~~ by object of Class, in friend function!

ex:-

```
# include <iostream.h>
# include <conio.h>
class A ;   <- decleration of class (forward decleration)
class B
{ int b;
  public:
  void getdata(int x)
      b = x ;
  }
  void Putdata (int )
  { cout << " value of b " << b;
```

defenation of function!

Scanned by CamScanner

```
}
friend void add(A , B);
};
class A
{ int a;
void getdata (int x)
{
a = x;
}
void putdata(void)
{ cout << "value of a" << a;
}.
friend void(A.B);
};
void add (A obj1 , B obj2)
{ cout << "addition of A+B" << obj1.a+obj2.B;
}
void main( )
{ A oba;
B obb;
A. getdata (100);
B. get data (200);
A. putdata ()
B. put data ()
add (a.b) ;
getch ();
}.
```

Scanned by CamScanner

Default argument

At the time of defining function value initilize to parameter is called default argument

```
int add (int x, int y, int z=50, int s=0)
{
    Body
}.
```

default argument

If we do not give the value of z and s then at the time of execution compiler use z=50 & s=0 but if we assigned value of z & s then compiler use that assigned value at the time of execution!

eg:

```
#include <iostream.h>

int add(int x=10, int y=20, int z=30)
{ return (x+y+z);
}
void main ()
{ cout<< add ();
}
```

or

```
cout<< add (100);
```

output = 60

out = 150

or `cout<< add (50,50);`

or `cout<< add(100,100);`

output = 130

~~output = 300~~

Scanned by CamScanner

Constant argument :-

int add (int x , int y) ← normal argument / formal parameter.

int add(const int x)

int add(const int &x)

add (const int *n)

Recursion

If the function call itself is called Recursion.

$\lfloor n = n(n-1)$

```
#include <iostream>
int fact(int x)
{ if (x=0)
  { return (1);
  }
  else
  { return( n* fact(n-1));
  }
  void main ()
  {
    int a, f;
    cout<<"print the value of a";
    cin>>a;
    f=fact(a);
    cout<<f;
  }
```

Scanned by CamScanner

Constructor

~~constructor~~ :-

default constructor

```
#include <iostream.h>
#   "    <conio.h>

class test
{ int a, b;
public:
test ()
{ a=0
  b=0
}
void display ( )
{ cout<<"value of"<<a<<b;
}
void main ( )
  test t;
  t. display( );
  getch( );
}
```

without giving any parameters

~~void disp()~~

```
test:: disp ( )
{ cout<<"code:"<<code;
  cout<<"price:"<price
}
void main ( )
  test t1 (101, 506);
  test t2 (t1)
}
```

copy constructor

```
class name (const name & obj)
{
    Body - -
}
```

```
class test
{ int code;
  float price;
  public:
  test (int c , float p);
  test (const test & t);
  void display
};
test:: test (int c float p)
{ code = c;
  price = p;
}
test:: test (const test & t)
{ code = t. code;
  price = t. price;
}
```

Scanned by CamScanner

Constructor overloading

```
#include<iostream>
#   "   <conio.h>
class test
{   int a, b;
    public:
        test ();
        test (int x);
        test (int x, int y);
        void display ();
};
        test:: test ()
        {   a=0, b=0
            }
        test:: test(int x)
        {   a=b=x }
        test: test (int x, int y)
            { x=a, b=y }
        void test:: disp()
        { cout<<"value of a"<<a;
            cout << b ;
            }
void main ()
{ test A;
    test B(100);
    test C(100, 200);
    cout<<"obj A"
    A.disp();
    cout << "object B";
    B. display();
    cout<<"object C";
    C. display();
    getch()
    }
```

Destructor: It is used to destroy object to free memory.

```
#include<iostream.h>
#include<conio.h>
class test
{ static int count;
  public:
  test ()
  { count++;
    cout<<count<<"object created";
  }
  ~ test ()
  { cout<<count<<"object destroyed"
    count--;
  }
};
void main ()
{ cout<<"main Block";
  test t1;
  { cout<<"Block 1";
  test t2, t3
  }  cout<<exist from block!
  cout<<exist from main
```

Static variable always initialize with '0' always

Static member function

```
# include<iostream.h>

class Test
{   int no;
    static int count;   ] static data variable

    public:
      void get no( ) ;
      void dispno( );
      static void dispcount ( );   ]static member function
    };
void test : : get no ( )
  { no = ++ count ;
  }
  void test:: dispno ( )
   { cout<<" object no" <<no ;
   }
   static void test : : dispcount ( )
   { cout <<" counter"<< count ;
   }
        int test : : count ;
   void main ( )
   { Test t1 , t2 ;
     t1. get no ( ) ;
     t2. get no ( ) ;
     ~~t2. dispcount()~~
     test: : dispcount ( ) ;
     t1. dispno ( ) ;
     t2. dispno ( ) ;
   } ;
```

Scanned by CamScanner

array in c++

one dimensional array

```
void main()
{
  int no[5], i, sum=0;
  cout<<"enter five elements";
    for(i=0; i<5; i++)
    {
      cin>> no[i];
    }
For (i=0; i<5; i++)
{ cout<<"no [i];
  sum= sum +no[i];
}
  cout<<" total" <<sum;
  getch();
}
```

Scanned by CamScanner