1. Inline function

```
#include<iostream.h>
#include<conio.h>
inline int cube(int a)
{
return(a*a*a);
}
void main()
{
int x,y;
clrscr();
cout<<"enter a value:";
cin>>x;
y=cube(x);
cout<<"y="<<y;
getch();
}
```

2. Function Overloading

```
#include<iostream.h>
#include<conio.h>
int sum(int a,int b)
{
return(a+b);
}
int sum(int a,int b,int c)
{
return(a+b+c);
}

void main()
{
int a,b,c;
clrscr();
cout<<"enter three numbers:";
cin>>a>>b>>c;
cout<<"sum="<<sum(a,b);
cout<<"sum="<<sum(a,b,c);
getch();
}
```

3. Class program

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

class student
{
private:
      int sno;
      int m1,m2;
public:
    void getdata()
    {
        cin>>sno;
        cin>>m1;
        cin>>m2;
    }
void putdata()
{
        cout<<sno<<endl;
        cout<<m1<<endl;
        cout<<m2;
}
};
void main()
{
student s;
s.getdata();
s.putdata();
}
```

4. Nesting of member Function

```
#include<iostream.h>
#include<conio.h>
class small
{
private:
        int a,b;
public:
void get()
{
cout<<"enter the first number:";
cin>>a;
cout<<"enter the second number:";
cin>>b;
}
int smallest()
{
if(a<b)
return a;
else
return b;
}
void display()
{
int s=smallest();
cout<<"smallest of the two number is "<<s;
}
};

void main()
{
small obj;
obj.get();
obj.display();
getch();
}
```

5. Constructor

```
#include<iostream.h>
#include<conio.h>
class student
{
private:
      int sno;
      int m1,m2;
public:
student(int x,int y,int z)
{
sno=x;
m1=y;
m2=z;
}
void putdata()
{
cout<<sno<<endl;
cout<<m1<<endl;
cout<<m2;
}
};
void main()
{
int x,y,z;
clrscr();
cout<<"enter three values:";
cin>>x>>y>>z;
student s(x,y,z);
s.putdata();
getch();
}
```

6. Friend function

```
#include<iostream.h>
#include<conio.h>
class box
{
private:
int length;
public:
box()
{
length=0;
}
friend int printlength(box);
};
int printlength(box b)
{
b.length+=10;
return b.length;
}
void main()
{
box b;
clrscr();
cout<<"length of box:"<<printlength(b)<<endl;
getch();
}
```

7. Unary Operator Overloading

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

class count
{
private:
    int value;
public:
```

```
count()
{
value=5;
}
void operator ++()
{
value=value+1;
}
void display()
{
cout<<"count!"<<value<<endl;
}
};
void main()
{
count count1;
clrscr();
++count1;
count1.display();
getch();
}
```

8. Binary Operator Overloading

```
#include<iostream.h>
#include<conio.h>
class complex
{
private:

    float real;
    float imag;
public:
void input()
{
cout<<"enter real and imaginary parts respectively:";
cin>>real;
cin>>imag;
}
complex operator+(complex &obj)
{
complex temp;
temp.real=real+obj.real;
temp.imag=imag+obj.imag;
return temp;
}
void output()
{
cout<<"output complex number:"<<real<<"+"<<imag<<"i";
}
};
void main()
{
complex C1,C2,C3;
clrscr();
cout<<"enter first complex number:\n";
C1.input();
cout<<"enter second complex number:\n";
C2.input();
C3=C1+C2;
C3.output();
getch();
}
```

9. Single Inheritance

```
#include<iostream.h>
#include<conio.h>
class student
{
private:
        int rno;
public:
void get()
{
cout<<"enter the roll number:";
cin>>rno;
}
void put()
{
cout<<"rno="<<rno<<endl;
}
};
class mark:public student
{
private:
        int m1,m2;
public:
void getmark()
{
cout<<"enter the two mark:";
cin>>m1>>m2;
}
void putmark()
{
cout<<"m1="<<m1<<endl;
cout<<"m2="<<m2<<endl;
}
};
void main()
{
mark U;
clrscr();
U.get();
U.getmark();
U.put();
U.putmark();
getch();
}
```

10. Multiple Inheritance

```
#include<iostream.h>
#include<conio.h>
class student
{
public:
    int rno,m1,m2;
void get()
{
cout<<"enter the value:";
cin>>rno>>m1>>m2;
}
void put()
{
cout<<"rno="<<rno<<endl;
cout<<"m1="<<m1<<endl;
cout<<"m2="<<m2<<endl;
}
};
class sports
{
public:
int score;
void getscore()
{
cout<<"enter the score:";
cin>>score;
}
void putscore()
{
cout<<"score="<<score<<endl;
}
};
class result:public sports,public student
{
private:
       int tot;
public:
void getresult()
{
tot=m1+m2+score;
}
void putresult()
{
cout<<"tot="<<tot;
}
```

```
};
void main()
{
result U;
clrscr();
U.get();
U.getscore();
U.getresult();
U.put();
U.putscore();
U.putresult();
}
```

11.Multilevel inheritance

```
#include<iostream.h>
#include<conio.h>
class student
{
protected:
      int rno;
public:
void get()
{
cout<<"enter the roll no:";
cin>>rno;
}
};
class mark:public student
{
protected:
int m1,m2;
public:
void getmark()
{
cout<<"enter the two marks:";
cin>>m1>>m2;
}
void display()
{
```

```
cout<<"\n roll no:"<<rno<<"\n";
cout<<"\n mark1:"<<m1<<"\n"<<"mark2:"<<m2;
}
};
class result:public mark
{
int tot;
public:
void total()
{
tot=m1+m2;
}
void output()
{
cout<<"\n"<<"total="<<tot;
}
};
void main()
{
result r;
clrscr();
r.get();
r.getmark();
r.total();
r.display();
getch();
}
```

12. Hierarchical Inheritance

```
#include<iostream.h>
#include<conio.h>
class a
{
public:
    int x,y;
void getdata()
{
cout<<"\n enter value of x and y:\n";cin>>x>>y;
}
};
class b:public a
{
public:
void product()
{
cout<<"\n product="<<x*y;
}
};
class c:public a
{
public:
void sum()
{
cout<<"\nsum="<<x+y;
}
};
void main()
{
b obj1;
c obj2;
clrscr();
obj1.getdata();
obj1.product();
obj2.getdata();
obj2.sum();
getch();
}
```

13. Nested class

```
#include<iostream.h>
#include<conio.h>
class a
{
public:
    class b
     {
```

```
private:
        int num;
public:
void getdata(int n)
{
num=n;
}
void putdata()
{
cout<<"the number is "<<num;
}
};
};
void main()
{
cout<<"nested classes"<<endl;
clrscr();
a::b obj;
obj.getdata(9);
obj.putdata();
getch();
}
```

14. Virtual Function

```
class base
  {
     public:

    virtual void print()
    {
        cout << "print base class\n";
    }

    void show()
    {
        cout << "show base class\n";
    }
};
```

```
class derived : public base
{
public:
    void print()
    {
        cout << "print derived class\n";
    }

    void show()
    {
        cout << "show derived class\n";
    }
};

void main()
{
    base *bptr;
    derived d;
    bptr = &d;
    bptr->print();
    bptr->show();
   getch();
}
```

15. Files

```
#include <iostream>
#include <fstream>

int main()
{
   cout<<"Write";
   ofstream f1("demo.txt");
   f1<<"welcome";
   f1.close();
char c[100];
cout<<"read";
ifstream   f2("filename.txt");
while (getline (f2, c))
{
   cout<<c;
}
 f2.close();
 return 0;
}
```