1.comand line argument

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

void main(int argc, char *argv[])
{
    clrscr();
    cout << "Number of arguments: " << argc
<< endl;
    for(int i = 0; i < argc; i++)
        cout << "Arg " << i << " = " << argv[i] <<
endl;
    getch();
}
```

2.inline function

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

```
inline int square(int x)
{
    return x * x;
}

void main()
{
    clrscr();
    cout << "Square of 5 = " << square(5);
    getch();
}
```

3.Default Argument

```
#include<iostream.h>
#include<conio.h>
Int sum(int a,int b,int c=0,int d=0)
{
    RETURN (a+b+c+d);
}
Void main()
```

```
{
    Cout<<sum(10,20,30,40)<<endl;
}
```

4.Static Member Function

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

class Counter
{
    static int count;
public:
    Counter() { count++; }
    static void showCount()
    {
        cout << "Count = " << count << endl;
    }
};

int Counter::count = 0;
```

```
void main()
{
    clrscr();
    Counter a, b;
    Counter::showCount();
    Counter c;
    Counter::showCount();
    getch();
}
```

5.Function Overloading

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

class Over
{
public:
    void sum(int a, int b)
    {
        cout << "Sum (int) = " << a + b << endl;
```

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

void main()
{
    clrscr();
    Over o;
    o.sum(5, 7);
    o.sum(2.5, 3.6);
    getch();
}
```

6.object as arguments

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

```
class Sample
{
    int num;
public:
    void getData(int n)
    {
        num = n;
    }
    void display()
    {
        cout << "Number = " << num << endl;
    }
    void add(Sample s1, Sample s2)
    {
        num = s1.num + s2.num;
    }
};

void main()
{
```

```
    clrscr();
    Sample s1, s2, s3;
    s1.getData(10);
    s2.getData(20);
    s3.add(s1, s2);
    s3.display();
    getch();
}
```

7.constructor destructor

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

class Demo
{
public:
    Demo()
    {
        cout << "Constructor Called" << endl;
    }
```

```
    ~Demo()
    {
        cout << "Destructor Called" << endl;
    }
};

void main()
{
    clrscr();
    Demo d;
    cout << "Inside main function" << endl;
    getch();
}
```

8.Single Inheritance

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

class A
{
```

```
public:
    void showA()
    {
        cout << "This is Base Class" << endl;
    }
};

class B : public A
{
public:
    void showB()
    {
        cout << "This is Derived Class" << endl;
    }
};

void main()
{
    clrscr();
    B obj;
    obj.showA();
```

```
    obj.showB();
    getch();
}
```

9.Multiple Inheritance

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

class A
{
public:
    void showA()
    {
        cout << "Class A" << endl;
    }
};

class B
{
public:
```

```
    void showB()
    {
        cout << "Class B" << endl;
    }
};

class C : public A, public B
{
public:
    void showC()
    {
        cout << "Class C" << endl;
    }
};

void main()
{
    clrscr();
    C obj;
    obj.showA();
    obj.showB();
```

```
    obj.showC();
    getch();
}
```

10.Hierarchical Inheritance

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

class A
{
public:
    void showA()
    {
        cout << "Base Class" << endl;
    }
};

class B : public A
{
public:
```

```
    void showB()
    {
        cout << "Derived Class 1" << endl;
    }
};

class C : public A
{
public:
    void showC()
    {
        cout << "Derived Class 2" << endl;
    }
};

void main()
{
    clrscr();
    B obj1;
    C obj2;
    obj1.showA();
```

```
    obj1.showB();
    obj2.showA();
    obj2.showC();
    getch();
}
```

11.Virtual Base Class

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

class A
{
public:
    int x;
    A() { x = 10; }
};

class B : virtual public A {};
class C : virtual public A {};
```

```
class D : public B, public C
{
public:
    void show()
    {
        cout << "Value of x = " << x << endl;
    }
};

void main()
{
    clrscr();
    D obj;
    obj.show();
    getch();
}
```

12.Abstract Class

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

```
class Shape
{
public:
    virtual void draw() = 0; // Pure virtual
function
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Drawing Circle..." << endl;
    }
};

void main()
{
    clrscr();
    Shape *s;
```

```
    Circle c;
    s = &c;
    s->draw();
    getch();
}
```