INDEX :

Constructor :

```cpp
#include<iostream>

using namespace std;


class Complex{

    int a, b;

    public:

    // Creating constructor

    // Constructor is a special member function with the same name as of the class.

    // It is used to initialize the objects of its class.

    // It is automatically invoked whenever an object is created.

    Complex();  // Constructor declaration


    void printNumber(){

        cout<<"Your number is "<<a<<" + "<<b<<"i"<<endl;

    }

};


Complex :: Complex(){ // This is a default constructor as it accept no parameter.

    a = 10;

    b = 20;

    cout<<"Hello World : "<<endl;

}


int main(){

    Complex c;

    c.printNumber();


return 0;

}


//************************   Characteristics of Constructors   ************************

/*

1. It should be declared in the public section of the class.

2. They are automatically invoked whenever the object is created.
```

```
3. They cannot return values and do not have return types.

4. It can have default arguments.

5. We cannot refer to their address.

*/
```

Output :-

```
PS D:\9. Tutorial of C++> cd
Hello World :
Your number is 10 + 20i
PS D:\9. Tutorial of C++>
```

Parameterized Constructor :

```
#include<iostream>

using namespace std;


class Complex{

    int a, b;

    public:

    Complex(int, int);


    void printNumber(){

        cout<<"Your number is "<<a<<" + "<<b<<"i"<<endl;

    }

};


Complex :: Complex(int x, int y){ // -->This is a parameterized constructor as it takes 2 parameters.

    a = x;

    b = y;

}


class Point{

    int x, y;

    public:

    Point(int a, int b){

        x = a;

        y = b;

    }


    void displayPoint(){

        cout<<"The point is ("<<x<<", "<<y<<")"<<endl;

    }

};


int main(){

    // Implicit call

    Complex a(4, 6);
```

```
    a.printNumber();


    // Explicit call
    Complex b = Complex(5, 7);
    b.printNumber();


    Point p(1, 2);
    p.displayPoint();


    Point q = Point(3, 5);
    q.displayPoint();


return 0;
}
```

Output :-

```
PS D:\9. Tutorial of C++> cd
}
Your number is 4 + 6i
Your number is 5 + 7i
The point is (1, 2)
The point is (3, 5)
PS D:\9. Tutorial of C++>
```

Constructor Overloding :

```
#include<iostream>

using namespace std;


class Complex{

    int a, b;

    public:

    Complex(int x, int y){

        a = x;

        b = y;

    }


    Complex(int x){

        a = x;

        b = 0;

    }


    Complex(){

        a = 0;

        b = 0;

    }


    void printNumber(){

        cout<<"Your number is "<<a<<" + "<<b<<"i"<<endl;

    }

};


int main(){

    Complex c1(4, 6);

    c1.printNumber();


    Complex c2(5);

    c2.printNumber();


    Complex c3;
```

```
    c3.printNumber();


return 0;

}
```

Output :-

```
PS D:\9. Tutorial of C++> cd
Your number is 4 + 6i
Your number is 5 + 0i
Your number is 0 + 0i
PS D:\9. Tutorial of C++>
```

Default Argument Constructor :

```cpp
#include<iostream>

using namespace std;


class Simple{

    int data1, data2, data3;

    public:

    Simple(int a, int b = 9, int c = 11){

        data1 = a;

        data2 = b;

        data3 = c;

    }


    void printData();


};


void Simple :: printData(){

    cout<<"The value of data1, data2 and data3 is "<<data1<<", "<<data2<<" and "<<data3<<endl;

}


int main(){

    Simple s(1, 4);

    s.printData();


return 0;

}
```

Output :-

```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ;
The value of data1, data2 and data3 is 1, 4 and 11
PS D:\9. Tutorial of C++>
```

Dynamic Initialization Of Objects :

```cpp
#include<iostream>

using namespace std;


class BankDeposit{

    int principal, year;

    float interestRate, returnvalue;


    public:

    BankDeposit(){}

    BankDeposit(int p, int y, float r); // 'r' can be a value like 0.04

    BankDeposit(int p, int y, int r);   // 'r' can be a value like 4 %

    void display();

};


BankDeposit :: BankDeposit(int p, int y, float r){

    principal = p;

    year = y;

    interestRate = r;

    returnvalue = principal;

    for(int i=0; i<y; i++){

        returnvalue = returnvalue * (1+interestRate);

    }

}


BankDeposit :: BankDeposit(int p, int y, int r){

    principal = p;

    year = y;

    interestRate = float(r)/100;

    returnvalue = principal;

    for(int i=0; i<y; i++){

        returnvalue = returnvalue * (1+interestRate);

    }

}
```

```cpp
void BankDeposit :: display(){

    cout<<"Principal amount was "<<principal<<endl;

    cout<<"Return value after "<<year<<" year is "<<returnvalue<<endl;

}


int main(){

    BankDeposit bd1;

    int p, y, R;

    float r;


    cout<<"Enter the value of p, y and r here : "<<endl;

    cin>>p>>y>>r;

    bd1 = BankDeposit(p, y, r);

    bd1.display();


    cout<<"Enter the value of p, y and r here : "<<endl;

    cin>>p>>y>>R;

    BankDeposit bd2(p, y, R);

    bd2.display();


return 0;

}
```

Output :-

```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ;
lizationOfObjects }
Enter the value of p, y and r here :
1000
2
0.04
Principal amount was 1000
Return value after 2 year is 1081.6
Enter the value of p, y and r here :
1000
2
4
Principal amount was 1000
Return value after 2 year is 1081.6
PS D:\9. Tutorial of C++>
```

Copy Constructor :

```cpp
#include<iostream>

using namespace std;


class Number{

    int a;

    public:

    Number(){

        a = 0;

    }

    Number(int num){

        a = num;

    }


    // When no copy constructor is found, compiler supplies its own copy constructor.

    Number(Number &obj){

        cout<<"Copy constructor called "<<endl;

        a = obj.a;

    }


    void display(){

        cout<<"The number for this object is "<<a<<endl;

    }

};


int main(){

    Number x, y, z(45), z2;

    x.display();

    y.display();

    z.display();


    // z1 should exactly resemble z or x or y

    Number z1(z);  // Here copy constructor is invoked.

    z1.display();
```

```
    z2 = y; // Here copy constructor not called.
    z2.display();

    Number z3 = x; // Here copy constructor is invoked.
    z3.display();

return 0;
}
```

Output :-

```
PS D:\9. Tutorial of C++> cd "d:\9.
The number for this object is 0
The number for this object is 0
The number for this object is 45
Copy constructor called
The number for this object is 45
The number for this object is 0
Copy constructor called
The number for this object is 0
PS D:\9. Tutorial of C++>
```

Destructor :

```
#include<iostream>

using namespace std;


// Destructor never takes an argument nor does it return any value.

int count = 0;

class num{

    public:

    num(){

        count++;

        cout<<"This is the time when constructor is called for object no. "<<count<<endl;

    }


    ~num(){

        cout<<"This is the time when Destructor is called for object no. "<<count<<endl;

        count--;

    }

};


int main(){

    cout<<"We are in the main function : "<<endl;

    cout<<"Creating first object n1"<<endl;


    num n1;{

        cout<<"Entering this block "<<endl;

        cout<<"Creating two more objects "<<endl;

        num n2, n3;

        cout<<"Exiting this block"<<endl;

    }


    cout<<"Back to main "<<endl;


return 0;

}
```

Output :-

```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\" ; if ($?)
We are in the main function :
Creating first object n1
This is the time when constructor is called for object no. 1
Entering this block
Creating two more objects
This is the time when constructor is called for object no. 2
This is the time when constructor is called for object no. 3
Exiting this block
This is the time when Destructor is called for object no. 3
This is the time when Destructor is called for object no. 2
Back to main
This is the time when Destructor is called for object no. 1
PS D:\9. Tutorial of C++>
```