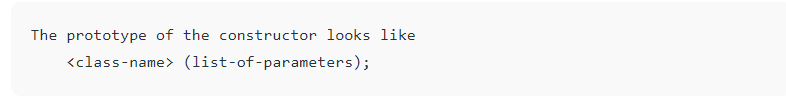
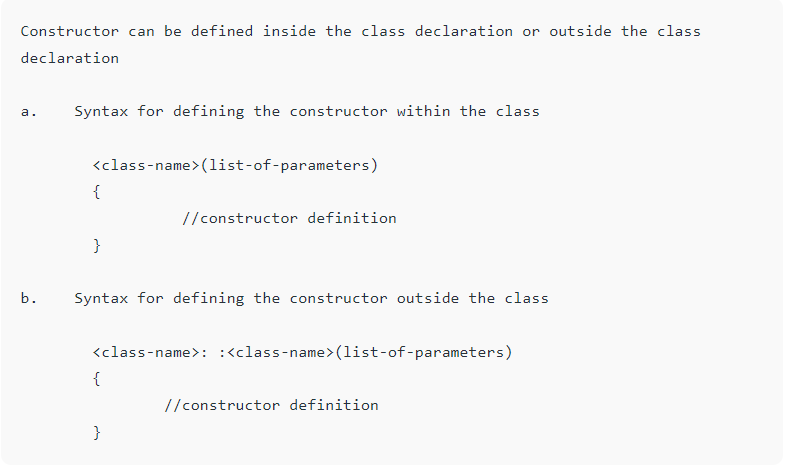
**Constructors in C++**

**Constructor in C++** is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally. The constructor in C++ has the same name as the class or structure. Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object which is why it is known as constructors.

• Constructor is a member function of a class, whose name is same as the class name.  
• Constructor is a special type of member function that is used to initialize the data members for an object of a class automatically, when an object of the same class is created.  
• Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.  
• Constructor do not return value, hence they do not have a return type.





// Example: defining the constructor within the class

#include<iostream>

using namespace std;

class student

{

int rno;

char name[50];

double fee;

public:

student()

{

cout<<"Enter the RollNo:";

cin>>rno;

cout<<"Enter the Name:";

cin>>name;

cout<<"Enter the Fee:";

cin>>fee;

}

void display()

{

cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

}

};

int main()

{

student s; //constructor gets called automatically when we create the object of the class

s.display();

return 0;

}

// Example: defining the constructor outside the class

#include<iostream>

using namespace std;

class student

{

int rno;

char name[50];

double fee;

public:

student();

void display();

};

student::student()

{

cout<<"Enter the RollNo:";

cin>>rno;

cout<<"Enter the Name:";

cin>>name;

cout<<"Enter the Fee:";

cin>>fee;

}

void student::display()

{

cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

}

int main()

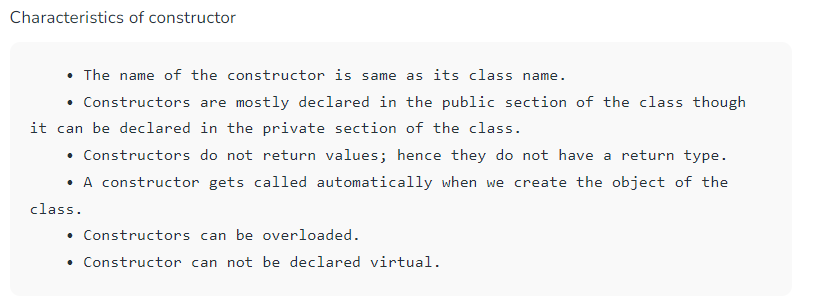
{

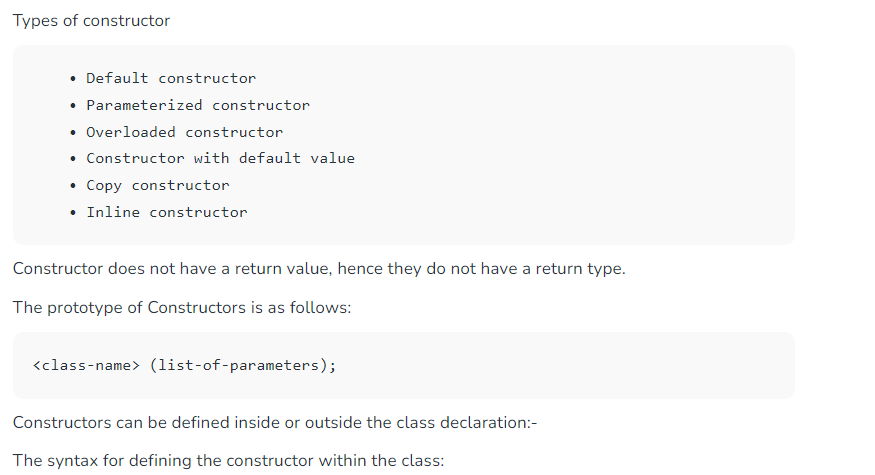
student s;

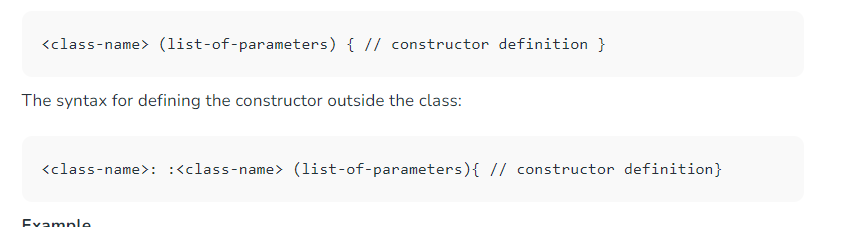
s.display();

return 0;

}

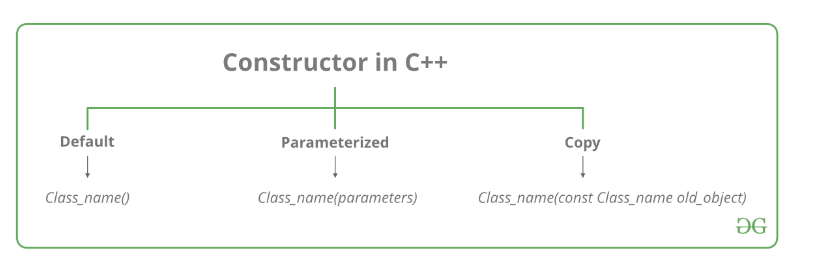






A constructor is different from normal functions in following ways:

* Constructor has same name as the class itself
* Default Constructors don’t have input argument however, Copy and Parameterized Constructors have input arguments
* Constructors don’t have return type
* A constructor is automatically called when an object is created.
* It must be placed in public section of class.
* If we do not specify a constructor, C++ compiler generates a default constructor for object (expects no parameters and has an empty body).



### **Types of Constructors**

[**1. Default Constructors:**](https://www.geeksforgeeks.org/c-internals-default-constructors-set-1/) Default constructor is the constructor which doesn’t take any argument. It has no parameters. It is also called a zero-argument constructor.

// Cpp program to illustrate the

// concept of Constructors

#include <iostream>

using namespace std;

class construct {

public:

int a, b;

// Default Constructor

construct()

{

a = 10;

b = 20;

}

};

int main()

{

// Default constructor called automatically

// when the object is created

construct c;

cout << "a: " << c.a << endl << "b: " << c.b;

return 1;

}

**Output**

a: 10

b: 20

***Note:****Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly.*

// Example

#include<iostream>

using namespace std;

class student

{

int rno;

char name[50];

double fee;

public:

student() // Explicit Default constructor

{

cout<<"Enter the RollNo:";

cin>>rno;

cout<<"Enter the Name:";

cin>>name;

cout<<"Enter the Fee:";

cin>>fee;

}

void display()

{

cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

}

};

int main()

{

student s;

s.display();

return 0;

}

***Operator overloading is a compile-time polymorphism***. It is an idea of giving special meaning to an existing operator in C++ without changing its original meaning.

In C++, we can make operators work for user-defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading. For example, we can overload an operator ‘+’ in a class like String so that we can concatenate two strings by just using +. Other example classes where arithmetic operators may be overloaded are Complex Numbers, Fractional Numbers, Big integers, etc.

**Example:**

int a;

float b,sum;

sum = a + b;

// C++ Program to Demonstrate

// Operator Overloading

#include <iostream>

using namespace std;

class Complex {

private:

int real, imag;

public:

Complex(int r = 0, int i = 0)

{

real = r;

imag = i;

}

// This is automatically called when '+' is used with

// between two Complex objects

Complex operator+(Complex const& obj)

{

Complex res;

res.real = real + obj.real;

res.imag = imag + obj.imag;

return res;

}

void print() { cout << real << " + i" << imag << '\n'; }

};

int main()

{

Complex c1(10, 5), c2(2, 4);

Complex c3 = c1 + c2;

c3.print();

}