**Constructors In C++ | C++ Tutorials for Beginners #29**

In this tutorial, we will discuss constructors in C++

**Constructors in C++**

A constructor is a special member function with the same name as the class. The constructor doesn’t have a return type. Constructors are used to initialize the objects of its class. Constructors are automatically invoked whenever an object is created.

**Important Characteristics of Constructors in C++**

* A constructor should be declared in the public section of the class
* They are automatically invoked whenever the object is created
* They cannot return values and do not have return types
* It can have default arguments
* We cannot refer to their address

An example program to demonstrate the concept of the constructor is shown below.

#include <iostream>

using namespace std;

class Complex

{

int a, b;

public:

// Creating a 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(void); // Constructor declaration

void printNumber()

{

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

}

};

Complex ::Complex(void) // ----> This is a default constructor as it takes no parameters

{

a = 10;

b = 0;

// cout<<"Hello world";

}

Copy

**Code Snippet 1: Constructor Example Program**

As shown in a code snippet 1,

* 1st “complex” class is defined which consists of private data members “a” and “b”.
* 2nd default constructor of the “complex” class is declared.
* 3rd function “printNumber” is defined which will print the values of the data members “a” and “b”.
* 4th default constructor is defined which will assign the values to the data members “a” and “b”. The main things to note here are that whenever a new object will be created this constructor will run and if the parameters are not passed to the constructor it is called a default constructor.

The main program is shown in code snippet 2.

int main()

{

Complex c1, c2, c3;

c1.printNumber();

c2.printNumber();

c3.printNumber();

return 0;

}

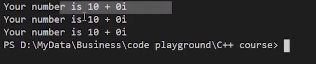
Copy

**Code Snippet 2: Main Program**

As shown in Code Snippet 2,

* 1st objects “c1”, “c2”, and “c3” of the complex data type are created. The main thing to note here is that when we are creating objects the constructor will run for each object and will assign the values.
* 2nd function “printNumber” is called by the objects “c1”, “c2”, and “c3”.

The output for the following program is shown in figure 1.



**Figure 1:** Program Output

As shown in figure 1, whenever a “printNumber” function is called it prints the values which are being assigned through the constructor.

**Parameterized and Default Constructors In C++ | C++ Tutorials for Beginners #30**

In this tutorial, we will discuss parameterized and default constructors in C++

**Parameterized and Default Constructors in C++**

Parameterized constructors are those constructors that take one or more parameters. Default constructors are those constructors that take no parameters. The main things to note here are that constructors are written in the public section of the class and the constructors don’t have a return type. An example program to demonstrate the concept of the constructor is shown below.

**Parameterized Constructors Example Program 1**

#include<iostream>

using namespace std;

class Complex

{

int a, b;

public:

Complex(int, int); // Constructor declaration

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;

// cout<<"Hello world";

}

Copy

**Code Snippet 1:  Parameterized Constructor Example Program 1**

As shown in a code snippet 1,

* 1st “complex” class is defined which consists of private data members “a” and “b”.
* 2nd parameterized constructor of the “complex” class is declared which takes two parameters.
* 3rd function “printNumber” is defined which will print the values of the data members “a” and “b”.
* 4th parameterized constructor is defined which takes two parameters and assigns the values to the data members “a” and “b”. The main things to note here are that whenever a new object will be created this constructor will run.

The main program is shown in code snippet 2.

int main(){

// Implicit call

Complex a(4, 6);

a.printNumber();

// Explicit call

Complex b = Complex(5, 7);

b.printNumber();

return 0;

}

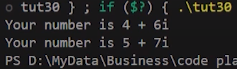
Copy

**Code Snippet 2: Main Program**

As shown in Code Snippet 2,

* 1st parameterized constructor is called implicitly with the object “a” and the values “4” and “6” are passed
* 2nd function “printNumber” is called which will print the values of data members
* 3rd parameterized constructor is called explicitly with the object “b” and the values “5” and “7” are passed
* 4th function “printNumber” is called again which will print the values of data members

The output for the following program is shown in figure 1.



**Figure 1:** Program Output 1

**Parameterized Constructors Example Program 2**

#include<iostream>

using namespace std;

class Point{

int x, y;

public:

Point(int a, int b){

x = a;

y = b;

}

void displayPoint(){

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

}

};

Copy

**Code Snippet 3: Parameterized Constructor Example Program 2**

As shown in Code Snippet 3,

* 1st “point” class is defined which consists of private data members “x” and “y”.
* 2nd parameterized constructor of the “point” class is defined which takes two parameters and assigns the values to the private data members of the class.
* 3rd function “displayPoint” is defined which will print the values of the data members “x” and “y”.

The main program is shown in code snippet 4.

int main(){

Point p(1, 1);

p.displayPoint();

Point q(4, 6);

q.displayPoint();

return 0;

}

Copy

**Code Snippet 4: Main Program**

As shown in Code Snippet 4,

* 1st parameterized constructor is called implicitly with the object “p” and the values “1” and “1” are passed
* 2nd function “displayPoint” is called which will print the values of data members
* 3rd parameterized constructor is called implicitly with the object “q” and the values “4” and “6” are passed
* 4th function “displayPoint” is called which will print the values of data members

The output for the following program is shown in figure 2.

https://cwh-full-next-space.fra1.digitaloceanspaces.com/videos/cpp-tutorials-in-hindi-30/Program_Output_2.webp

**Figure 2:** Program Output 2

**Constructor Overloading In C++ | C++ Tutorials for Beginners #31**

In this tutorial, we will discuss constructor overloading in C++

**Constructor Overloading in C++**

Constructor overloading is a concept in which one class can have multiple constructors with different parameters. The main thing to note here is that the constructors will run according to the arguments for example if a program consists of 3 constructors with 0, 1, and 2 arguments, so if we pass 1 argument to the constructor the compiler will automatically run the constructor which is taking 1 argument. An example program to demonstrate the concept of Constructor overloading in C++ is shown below.

#include <iostream>

using namespace std;

class Complex

{

int a, b;

public:

Complex(){

a = 0;

b =0;

}

Complex(int x, int y)

{

a = x;

b = y;

}

Complex(int x){

a = x;

b = 0;

}

void printNumber()

{

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

}

};

Copy

**Code Snippet 1: Constructor Overloading Program Example**

As shown in Code Snippet 1,

* 1st we created a “complex” class which consists of private data members “a” and “b”.
* 2nd default constructor of the “complex” class is declared which has no parameters and assigns “0” to the data members “a” and “b”.
* 3rd parameterized constructor of the “complex” class is declared which takes two parameters and assigns values to the data members “a” and “b”.
* 4th parameterized constructor of the “complex” class is declared which takes one parameter and assigns values to the data members “a” and “b”.
* 5th function “printNumber” is defined which will print the values of the data members “a” and “b”.

The main program is shown in code snippet 2.

int main()

{

Complex c1(4, 6);

c1.printNumber();

Complex c2(5);

c2.printNumber();

Complex c3;

c3.printNumber();

return 0;

}

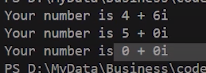
Copy

**Code Snippet 2: Main Program**

As shown in Code Snippet 2,

* 1st parameterized constructor is called with the object “c1” and the values “4” and “6” are passed. The main thing to note here is that this will run the constructor with two parameters.
* 2nd function “printNumber” is called which will print the values of data members
* 3rd parameterized constructor is called with the object “c2” and the value “5” is passed. The main thing to note here is that this will run the constructor with one parameter.
* 4th function “printNumber” is called which will print the values of data members
* 5th default constructor is called with the object “c3”. The main thing to note here is that this will run the constructor with no parameters.
* 6th function “printNumber” is called which will print the values of data members

The output for the following program is shown in figure 1.



**Figure 1:** Program Output

As shown in figure 1, all the values which were passed and assigned through parameterized constructors and the values which were assigned through the default constructor are printed.

**Constructors With Default Arguments In C++ | C++ Tutorials for Beginners #32**

In this tutorial, we will discuss constructors with default arguments in C++

**Constructors with Default Arguments in C++**

Default arguments of the constructor are those which are provided in the constructor declaration. If the values are not provided when calling the constructor the constructor uses the default arguments automatically. An example program to demonstrate the concept default arguments in C++ is shown below.

#include<iostream>

using namespace std;

class Simple{

int data1;

int data2;

int data3;

public:

Simple(int a, int b=9, int c=8){

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;

}

Copy

**Code Snippet 1: Constructor with Default Arguments Program Example**

As shown in a code snippet 1,

* 1st we created a “simple” class which consists of private data members “data1”, “data2” and “data3”.
* 2nd parameterized constructor of the “simple” class is defined which takes three parameters and assigns values to the data members “a” and “b”. The main thing to note here is that the value “9” and “8” are the default values for the variables “b” and “c”.
* 3rd function “printData” is defined which prints the values of the data members “data1”, “data2”, and “data3”.

The main program is shown in code snippet 2.

int main(){

Simple s(12, 13);

s.printData();

return 0;

}

Copy

**Code Snippet 2: Main Program**

As shown in code snippet 2,

* 1st parameterized constructor is called with the object “s” of the data type “simple” and the values “12” and “13” are passed. The main thing to note here is that the value of the parameter “c” will be automatically set by the default value.
* 2nd function “printData” is called which will print the values of data members.

The output for the following program is shown in figure 1.

https://cwh-full-next-space.fra1.digitaloceanspaces.com/videos/cpp-tutorials-in-hindi-32/Program_Output.webp

**Figure 1:**Program Output

As shown in figure 1, the value “12”, “13”, and “8” are printed. The constructor assigned the values “12” and “13” to the variables “a” and “b” but the value for the variable “c” was not passed that’s why constructors set the value “8” which was the default value for the variable “c”.

**Dynamic Initialization of Objects Using Constructors | C++ Tutorials for Beginners #33**

In this tutorial, we will discuss the dynamic initialization of objects using constructors in C++

**Dynamic Initialization of Objects Using Constructors**

The dynamic initialization of the object means that the object is initialized at the runtime.  Dynamic initialization of the object using a constructor is beneficial when the data is of different formats. An example program is shown below to demonstrate the concept of dynamic initialization of objects using constructors.

#include<iostream>

using namespace std;

class BankDeposit{

int principal;

int years;

float interestRate;

float 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 14

void show();

};

Copy

**Code Snippet 1: Dynamic Initialization of Objects using Constructor Example**

As shown in Code Snippet 1,

* 1st we created a “BankDeposit” class which consists of private data members “principal”, “years”, “interestRate”, and “returnValue”.
* 2nd default constructor of the “BankDeposit” class is declared.
* 3rd parameterized constructor of the “BankDeposit” class is declared which takes three parameters “p”, “y”, and “r”. The main thing to note here is that the parameter “r” is of a float data type.
* 4th parameterized constructor of the “BankDeposit” class is declared which takes three parameters “p”, “y”, and “r”. The main thing to note here is that the parameter “r” is of an integer data type.
* 5th function “show” is declared.

The definition of constructors and function is shown below.

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

{

principal = p;

years = 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;

years = y;

interestRate = float(r)/100;

returnValue = principal;

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

{

returnValue = returnValue \* (1+interestRate);

}

}

void BankDeposit :: show(){

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

<< ". Return value after "<<years

<< " years is "<<returnValue<<endl;

}

Copy

**Code Snippet 2: Definition of Constructors and Function**

As shown in Code snippet 2,

* 1st the constructor “BankDeposit” is defined in which the value of the parameter “p” is assigned to the data member “principal”; the value of the parameter “y” is assigned to the data member “year”; the value of the parameter “r” is assigned to the data member “interestRate”. At the end “for” loop is defined which will run till the length of the variable “y” and add “1” in the “interestRate”; then multiply the value with the “returnValue”. The main thing to note here is that in this constructor the data type of the parameter “r” is float.
* 2nd another constructor “BankDeposit” is defined in which the value of the parameter “p” is assigned to the data member “principal”; the value of the parameter “y” is assigned to the data member “year”; the value of the parameter “r” is converted to “float” and divided by “100” then assigned to the data member “interestRate”. At the end “for” loop is defined which will run till the length of the variable “y” and add “1” in the “interestRate”; then multiply the value with the “returnValue”. The main thing to note here is that in this constructor the data type of the parameter “r” is float.
* 3rd the function “show” is defined which will print the values of the data members “principal”, “year”, and “returnValue”.

The main program is shown in code snippet 3.

int main(){

BankDeposit bd1, bd2, bd3;

int p, y;

float r;

int R;

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

cin>>p>>y>>r;

bd1 = BankDeposit(p, y, r);

bd1.show();

cout<<"Enter the value of p y and R"<<endl;

cin>>p>>y>>R;

bd2 = BankDeposit(p, y, R);

bd2.show();

return 0;

}

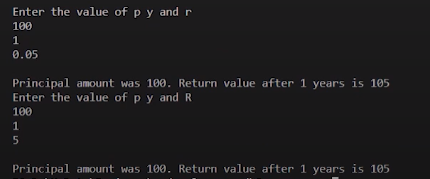
Copy

**Code Snippet 3: Main Program**

As shown in a code snippet 3,

* 1st the object “bd1”, “bd2”, and “bd3” of the data type “BankDeposit” are created.
* 2nd the integer variables “p” and “y” are declared; the float variable “r” is declared, and the integer variable “R” is declared.
* 3rd the values for the variables “p”, “y”, and”r” are taken from the user on the runtime.
* 4th parameterized constructor “BankDeposit” is called with the object “bd1” and the variables “p”, “y”, and “r” are passed. The main thing to note here is that this will run the constructor with float parameters “r”.
* 5th function “show” is called which will print the values of data members
* 6th the values for the variables “p”, “y”, and ”R” are taken from the user on the runtime.
* 7th parameterized constructor “BankDeposit” is called with the object “bd2” and the variables “p”, “y”, and “R” are passed. The main thing to note here is that this will run the constructor with integer parameters “R”.
* 8th function “show” is called which will print the values of data members.

The output for the following program is shown in figure 1.



**Figure 1:** Program Output

As shown in figure 1, the first time the values “100”, “1”, and “0.05” are entered and it gives us the return value of “105”. The second time the values “100”, “1”, and “5” are entered and it gives us the return value of “105”. So the main thing to note here is that the compiler figures out the run time by seeing the data type and runs the relevant constructor.

**Copy Constructor in C++ | C++ Tutorials for Beginners #34**

In this tutorial, we will discuss copy constructor in C++

**Copy Constructor in C++**

A copy constructor is a type of constructor that creates a copy of another object. If we want one object to resemble another object we can use a copy constructor. If no copy constructor is written in the program compiler will supply its own copy constructor. An example program to demonstrate the concept of a Copy constructor in C++ is shown below.

#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;

}

};

Copy

**Code Snippet 1: Copy Constructor Example Program**

As shown in Code Snippet 1,

* 1st we created a “number” class which consists of private data member “a”.
* 2nd default constructor of the “number” class is defined which has no parameters and assign “0” to the data members “a”.
* 3rd parameterized constructor of the “number” class is defined which takes one parameter and assigns values to the data members “a”.
* 4th copy constructor of the “number” class is defined which takes its own reference object as a parameter and assigns values to the data members “a”.
* 5th function “display” is defined which will print the values of the data members “a”.

The main program is shown in code snippet 2.

int main(){

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

x.display();

y.display();

z.display();

Number z1(z); // Copy constructor invoked

z1.display();

z2 = z; // Copy constructor not called

z2.display();

Number z3 = z; // Copy constructor invoked

z3.display();

// z1 should exactly resemble z or x or y

return 0;

}

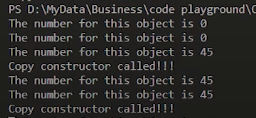
Copy

**Code Snippet 2: Main Program**

As shown in Code Snippet 2,

* 1st objects “x”, “y”, “z”, and “z1” are created of the “number” data type. The main thing to note here is that the object “z” has a value “45”.
* 2nd function “display” is called by the objects “x”, “y”, and “z”.
* 3rd copy constructor is invoked and the object “z” is passed to “z1”
* 4th function “display” is called by the object “z1”
* 5th the value of “z” is assigned to “z1”. The main thing to note here is that it will not invoke a copy constructor because the object “z” is already created.
* 6th function “display” is called by the object “z2”
* 7th the value of “z” is assigned to “z3”. The main thing to note here is that it will invoke a copy constructor because the object “z3” is being created.
* 8th function “display” is called by the object “z3”

The output for the following program is shown in figure 1.



**Figure 1:**Program Output

As shown in figure 1, all the values which were passed and assigned through copy constructors are printed.

**Destructor in C++ in Hindi | C++ Tutorials for Beginners #35**

In this tutorial, we will discuss Destructor in C++

**Destructor in C++**

A destructor is a type of function which is called when the object is destroyed. Destructor never takes an argument nor does it return any value. An example program to demonstrate the concept of destructors in C++ is shown below.

#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 number"<<count<<endl;

}

~num(){

cout<<"This is the time when my destructor is called for object number"<<count<<endl;

count--;

}

};

Copy

**Code Snippet 1: Destructor Example Program**

As shown in Code Snippet 1,

* 1st global variable “count” is initialized.
* 2nd we created a “num” class.
* 3rd default constructor of the “num” class is defined which has no parameters and does increment in the variable “count” and prints its value. The main thing to note here is that every time the new object will be created this constructor will run.
* 4th destructor of the “num” class is defined. The destructor prints the value of the variable “count” and decrement in the value of “count”. The main thing to note here is that every time the object has been destroyed this destructor will run.

The main program is shown in code snippet 2.

int main(){

cout<<"We are inside our 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;

}

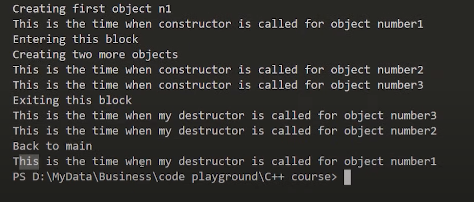
Copy

**Code Snippet 2: Main Program**

As shown in Code Snippet 2,

* 1st object “n1” is created of the “num” data type. The main thing to note here is that when the object “n1” is created the constructor will run.
* 2nd inside the block two objects “n2” and “n3” are created of the “num” data type. The main things to note here are that when the objects “n2” and “n3” are created the constructor will run for both objects and when the block ends the destructor will run for both objects “n2” and “n3”.
* 3rd when the program ends the destructor for the object “n1” will run.

The output for the following program is shown in figure 1.



**Figure 1:**Program Output

As shown in figure 1, first the constructor for the object “n1” was called; second the constructor for the objects “n2” and “n3” was called; third the destructor was called for the objects “n2” and “n3”; at the end destructor for the object “n1” was called.