
FACE

Constructor

- Constructor in C++ has the same name as class.
- Constructors never have return type.
- Invoked automatically at the time of object creation
- Used to initialize the data members
- Constructors can be defined either inside the class or outside the class

FACE

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket()
7        {
8              customer_Id=12345;
9              product_purchased=10;
10       }
11 };
12 int main()
13 {
14       Supermarket customer_1;
15       std::cout<<customer_1.customer_Id<<"\n";
16       std::cout<<customer_1.product_purchased;
17 }
18
```

Output

```
12345
10
```

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket();
7  };
8  Supermarket::Supermarket()
9  {
10       customer_Id=132407;
11       product_purchased=1;
12 }
13 int main()
14 {
15       Supermarket customer_1;
16       std::cout<<customer_1.customer_Id<<"\n";
17       std::cout<<customer_1.product_purchased;
18 }
```

Output

```
132407
1
```

Types of Constructor

- Default Constructor
- Parameterized Constructor
- Copy Constructor

FACE

Default Constructor

- It has no parameter.
- If we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.
- Default constructor provided by the compiler - initialize the data members to default value (i.e., 0)

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket()
7        {
8              customer_Id=132407;
9              product_purchased=12;
10       }
11 };
12 int main()
13 {
14       Supermarket customer_1;
15       std::cout<<customer_1.customer_Id<<"\n";
16       std::cout<<customer_1.product_purchased;
17 }
```

Output

```
132407
12
```

Parameterized Constructor

- Constructors with parameter
- Used to provide different values, by passing the values as argument during the object creation.

FACE

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket(int x,int y)
7        {
8              customer_Id=x;
9              product_purchased=y;
10       }
11 };
12 int main()
13 {
14       Supermarket customer_1(1963,7);
15       std::cout<<customer_1.customer_Id<<"\n";
16       std::cout<<customer_1.product_purchased;
17 }
```

Output

```
1963
7
```

Copy Constructor

Copy constructor creates a **new object,** which is exact copy of the existing objects.

Syntax of Copy Constructor:

```
Classname(const classname &objectname)
{
    ....
}
```

FACE

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket(int x,int y)
7        {
8              customer_Id=x;product_purchased=y;
9        }
10       Supermarket(const Supermarket &cust)
11       {
12             customer_Id=cust.customer_Id;
13             product_purchased=cust.product_purchased;
14       }
15 };
16 int main()
17 {
18       Supermarket customer_1(1963,7);
19       Supermarket customer_2(customer_1);
20       std::cout<<customer_2.customer_Id<<"\n";
21       std::cout<<customer_2.product_purchased;
22 }
```

OUTPUT

```
1963
7
```

```
1  #include<iostream>
2  using namespace std;
3
4  class Point
5  {
6  private:
7        int x, y;
8  public:
9        Point(int x1, int y1) { x = x1; y = y1; }
10       Point(const Point &p2) {x = p2.x; y = p2.y; }
11       int getX()           { return x; }
12       int getY()           { return y; }
13 };
14
15 int main()
16 {
17       Point p1(10, 15);
18       Point p2 = p1;
19       cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
20       cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();
21       return 0;
22 }
```

OUTPUT

```
p1.x = 10, p1.y = 15
p2.x = 10, p2.y = 15
```

Copy Constructor Vs Assignment Operator

Which of the following two statements call copy constructor and which one calls assignment operator?

```
Mycon t1, t2;
Mycon t3 = t1;        // -------→1
t2 = t1;              // -------→2
```

FACE

Why copy constructor argument should be const in C++?

- So that objects are not modified accidentally

```
1  #include<iostream>
2  using namespace std;
3
4  class Point
5  {
6  private:
7        int x, y;
8  public:
9        Point(int x1, int y1) { x = x1; y = y1; }
10       Point(const Point &p2) {x = p2.x; y = p2.y; }
11       int getX()           { return x; }
12       int getY()           { return y; }
13 };
14
15 int main()
16 {
17       Point p1(10, 15);
18       Point p2 = p1;
19       cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
20       cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();
21       return 0;
22 }
```

Destructor

Syntax of Destructor:

```
~Classname()
{
        //code
}
```

FACE

Destructor

A destructor is automatically called when

- The program finished execution.
- When a scope({ }) contains a local variable ends.
- When you call the delete operator.

FACE

```
1  #include<iostream>
2  class Supermarket
3  {
4        public:
5        int customer_Id,product_purchased;
6        Supermarket()
7        {
8              std::cout<<"Customer 1 purchased\n";
9        }
10       ~Supermarket()
11       {
12             std::cout<<"Customer 1 bill paid";
13       }
14 };
15 int main()
16 {
17       Supermarket customer_1;
18
19 }
```

Output

```
Customer 1 purchased
Customer 1 paid the bill
```

Destructor

- Can there be more than one destructor in a class?
- Can we write our own destructor(user-defined)?
- When do we need to write a user-defined destructor?

FACE

Private Destructor

```
1  #include <iostream>
2  using namespace std;
3  class Test
4  {
5        private:
6        ~Test()
7        {
8        }
9  };
10 int main()
11 {
12 }
13
14
15
16
17
18
19
20
21
22
```

```
1  #include <iostream>
2  using namespace std;
3  class Test
4  {
5        private:
6        ~Test()
7        {
8        }
9  };
10 int main()
11 {
12       Test t;
13 }
14
15
16
17
18
19
20
21
22
```

```
1  #include <iostream>
2  using namespace std;
3  class Test
4  {
5        private:
6        ~Test()
7        {
8        }
9  };
10 int main()
11 {
12       Test* t;
13 }
14
15
16
17
18
19
20
21
22
```

```
1  #include <iostream>
2  class Test {
3  private:
4      ~Test() {}
5      friend void destructTest(Test*);
6  };
7  void destructTest(Test* ptr)
8  {
9      delete ptr;
10 }
11
12 int main()
13 {
14     Test* ptr = new Test;
15     destructTest(ptr);
16     return 0;
17 }
18
19
20
21
22
```

Question 1

C++ provides a special ..... called the constructor, which enables an object to initialize itself when it is created.

A) friend function

B) member function

C) public function

D) private function

Question 2

An ..... with a constructor or destructor cannot be used as a member or a union.

A) class

B) object

C) function

D) variable

Question 3

A destructor is used to destroy the objects that have been created by a .....

A) object

B) class

C) function

D) constructor

Question 4

Which of the following statements are not true about destructor?

1. It is invoked when object goes out of the scope
2. Like constructor, it can also have parameters
3. It can be virtual
4. It can be declared in private section
5. It bears same name as that of the class and precedes Lambda sign.

Question 4

A) Only 2, 3, 5

B) Only 2, 3, 4

C) Only 2, 4, 5

D) Only 3, 4, 5

Question 5

Which of the following gets called when an object is being created?

A) Constuctor

B) Virtual Function

C) Destructors

D) Main

Question 6

Which of the following is true about constructors.
i) They cannot be virtual
ii) They cannot be private.
iii) They are automatically called by new operator.

A) All i,ii,iii

B) i & iii

C) ii & iii

D) i & ii

Question 7

Destructors _____ for automatic objects if the program terminates with a call to function exit or function abort

A) Are called

B) Are not called

C) Are inherited

D) Are created

Question 8

Which of the following represents the correct explicit call to a constructor of class A?

```
class A
{
int a;
  public:
       A(int i)
       {
       a = i;
       }
}
```

Question 8

A) A a(5);

B) A a;

C) A a = A(5);

D) A a = A();

FACE

Question 9

Constructors are normally used to ..... and to allocate memory.

A) define variables

B) allocate variables

C) initialize variables

D) initialize object

Question 10

If default constructor is not defined, then how the objects of the class will be created?

A) The compiler will generate error

B) Error will occur at run-time.

C) Compiler provides its default constructor to build the object.

D) None of these

Question 11

Which is the correct form of default constructor for following class?

```
#include <iostream>
using namespace std;

class sample
{
    private:
        int x,y;
};
```

Question 11

A) public: void sample(){}

B) public: void sample(){ x=0; y=0;}

C) public: void sample(int a,int b){ x=a; y=b;}

D) Both 1 and 2

Question 12

For automatic objects, constructors and destructors are called each time the objects

A) enter and leave scope

B) inherit parent class

C) are constructed

D) are destroyed

Question 13

Which of the following statement is correct about the program given below?

```
#include <iostream>
using namespace std;
class Face
{
    public:
    Face()
    {
        cout<< "Face";
    }
```

```
 ~Face()
    {
        cout<< "Academy";
    }
};
int main()
{
    Face objBix;
    return 0;
}
```

FACE

Question 13

A) The program will print the output Face.

B) The program will print the output Academy.

C) The program will print the output FaceAcademy.

D) The program will report compile time error.

Question 14

What happens when a class with parameterized constructors and having no default constructor is used in a program and we create an object that needs a zero-argument constructor?

A) Compile-time error.

B) Preprocessing error.

C) Runtime error.

D) Runtime exception.

Question 15

What is the difference between constructors and destructors?

A) They have a different function name

B) Constructors does not have return type whereas destructors do have

C) Constructors allow function parameters whereas destructors do not

D) Constructors does not function parameters

Question 16

If you created a parameterized and a copy constructor in the class and you create an object with no arguments (0 arguments), what will be the output?

A) Program will execute successfully

B) A compile-time will occur

C) A run-time error will occur

D) A syntax error will occur

Question 17

What will be the output of the following code?

```
#include<iostream>
using namespace std;
//class definition
class Example {
    public:
        void ~Example()
        {
            cout<<"Destroying
the object";
```

```
}
};

//main() code
int main()
{
    Example Ex;
    return 0;
}
```

FACE

Question 17

A) Run-time error

B) Compile-time error

C) Destroying the object

D) Executes successfully but no output

Question 18

What will be the output of the following code?

```
#include<iostream>
using namespace std;
class Example
{
    Example()
    {
        cout << "Constructor
called";
    }
};
```

```
//main() code
int main()
{
   Example Ex;
   return 0;
}
```

Question 18

A) Constructor called

B) Program successfully executed – no output

C) Compile time error

D) Run time error

Question 19

What will be the output of the following code?

```
#include<iostream>
using namespace std;

//class definition
class Example {
    public:
    Example() {
        cout << "Constructor
called ";
    }
```

```
//main() code
int main()
{
   Example Ex1, Ex2;
   return 0;
}
```

Question 19

A) Constructor called

B) Constructor called Constructor called

C) Compile time error

D) Run time error

Question 20

What will be the output of the following code?

```
#include<iostream>
using namespace std;

//class definition
class Example {
    public:
        int a;
        int b;
};
```

```
//main() code
int main()
{
   Example Ex1  = {10, 20};
   cout << "a = " << Ex1.a <<",
b = " << Ex1.b;
   return 0;
}
```

Question 20

A) Compile time error

B) Run time error

C) a = 10, b = 20

D) a = 10, b = 10

THANK YOU

FACE