GeeksforGeeks

Related Articles

Private Destructor

Difficulty Level : Medium ● Last Updated : 31 May, 2021

Also read : Can a constructor be private in C++ ?

Predict the output of following programs.

CPP

```
// CPP program to illustrate
// Private Destructor
#include <iostream>
using namespace std;

class Test {
private:
    ~Test() {}
};
int main()
{
}
```

The above program compiles and runs fine. Hence, we can say that : It is **not** compiler error to create private destructors.
Now, What do you say about below program.

CPP

```
// CPP program to illustrate
// Private Destructor
#include <iostream>
using namespace std;

class Test {
private:
    ~Test() {}
```

```
};
int main()
{
    Test t;
}
```

The above program fails in compilation. The compiler notices that the local variable 't' cannot be destructed because the destructor is private.

Now, What about the below program?

CPP

```
// CPP program to illustrate
// Private Destructor
#include <iostream>
using namespace std;

class Test {
private:
    ~Test() {}
};
int main()
{
    Test* t;
}
```

The above program works fine. There is no object being constructed, the program just creates a pointer of type "Test *", so nothing is destructed.

Next, What about the below program?

CPP

```
// CPP program to illustrate
// Private Destructor

#include <iostream>
using namespace std;

class Test {
private:
    ~Test() {}
};
int main()
{
```

```
    Test* t = new Test;
}
```

The above program also works fine. When something is created using dynamic memory allocation, it is programmer's responsibility to delete it. So compiler doesn't bother.

In the case where the destructor is declared private, an instance of the class can also be created using malloc() function. Same is implemented in below program.

CPP

```
// CPP program to illustrate
// Private Destructor

#include <bits/stdc++.h>
using namespace std;

class Test {
public:
    Test() // Constructor
    {
        cout << "Constructor called\n";
    }

private:
    ~Test() // Private Destructor
    {
        cout << "Destructor called\n";
    }
};

int main()
{
    Test* t = (Test*)malloc(sizeof(Test));
    return 0;
}
```

Output:

However, The below program fails in compilation. When we call delete, destructor is called.

CPP

```
// CPP program to illustrate
// Private Destructor
#include <iostream>
using namespace std;

class Test {
private:
    ~Test() {}
};
int main()
{
    Test* t = new Test;
    delete t;
}
```

We noticed in the above programs, when a class has private destructur, only dynamic objects of that class can be created. Following is a way to **create classes with private destructors and have a function as friend of the class.** The function can only delete the objects.

CPP

```
// CPP program to illustrate
// Private Destructor
#include <iostream>

// A class with private destuctor
class Test {
private:
    ~Test() {}
public:
    friend void destructTest(Test*);
};

// Only this function can destruct objects of Test
void destructTest(Test* ptr)
{
    delete ptr;
}

int main()
```

```
{
    // create an object
    Test* ptr = new Test;

    // destruct the object
    destructTest(ptr);

    return 0;
}
```

What is the use of private destructor?

Whenever we want to control destruction of objects of a class, we make the destructor private. For dynamically created objects, it may happen that you pass a pointer to the object to a function and the function deletes the object. If the object is referred after the function call, the reference will become dangling. See this for more details.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Want to learn from the best curated videos and practice problems, check out the C++ Foundation Course for Basic to Advanced C++ and C++ STL Course for foundation plus STL. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course.**

Previous Next