Practice Compete Jobs Rank Leaderboard

Dashboard > C++ > Inheritance > Accessing Inherited Functions

Points: 475 Rank: 6579

Your Accessing Inherited Functions submission got 30.00 points. Share Tweet

Try the Next Challenge | Try a Random Challenge

Accessing Inherited Functions

by vatsalchanana

Problem Submissions Leaderboard Discussions

You are given three classes *A, B* and *C*. All three classes implement their own version of *func*.

In class *A*, *func* multiplies the value passed as a parameter by 2:

```
class A
{
    public:
        A(){
            callA = 0;
        }
    private:
        int callA;
        void inc(){
            callA++;
        }

    protected:
        void func(int & a)
        {
            a = a * 2;
            inc();
        }
    public:
        int getA(){
            return callA;
        }
};
```

In class *B*, *func* multiplies the value passed as a parameter by 3:

```
class B
{
    public:
        B(){
            callB = 0;
        }
    private:
        int callB;
        void inc(){
            callB++;
        }
    protected:
        void func(int & a)
        {
            a = a * 3;
            inc();
        }
    public:
        int getB(){
            return callB;
        }
};
```

In class *C*, *func* multiplies the value passed as a parameter by 5:

```
class C
{
    public:
        C(){
            callC = 0;
        }
    private:
        int callC;
        void inc(){
            callC++;
        }
    protected:
        void func(int & a)
        {
            a = a * 5;
            inc();
        }
    public:
        int getC(){
            return callC;
        }
};
```

You are given a class *D*:

```
class D
{

        int val;
        public:
                //Initially val is 1
                 D()
                 {
                        val = 1;
                 }


                 //Implement this function
                 void update_val(int new_val)
                 {


                 }
                 //For Checking Purpose
                 void check(int); //Do not delete this line.
};
```

You need to modify the class *D* and implement the function `update_val` which sets *D*'s *val* to *new_val* by manipulating the value by only calling the *func* defined in classes *A*, *B* and *C*.

It is guaranteed that *new_val* has only $2, 3$ and 5 as its prime factors.

Input Format

Implement class *D*'s function *update_val*. This function should update *D*'s *val* only by calling *A*, *B* and *C*'s *func*.

Constraints

$1 \le new\_val \le 10000$

Note: The *new_val* only has $2, 3$ and 5 as its prime factors.

Sample Input

$new\_val = 30$

Sample Output

A's *func* will be called once.

B's *func* will be called once.

C's *func* will be called once.

Explanation

Initially, $val = 1$.

A's *func* is called once:

```
val = val*2
val = 2
```

B's *func* is called once:

```
val = val*3
val = 6
```

C's *func* is called once:

```
val = val*5
val = 30
```

Medium Submitted 12228 times
Max Score **30**

Need Help?

View Discussions

View Top Submissions

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

Download problem statement

Download sample test cases

Suggest Edits

Current Buffer (saved locally, editable)

C++14

```
#include<iostream>

using namespace std;

class A
{
    public:
        A(){
            callA = 0;
        }
    private:
        int callA;
        void inc(){
            callA++;
        }

    protected:
        void func(int & a)
        {
            a = a * 2;
            inc();
        }
    public:
        int getA(){
            return callA;
        }
};

class B
{
    public:
        B(){
            callB = 0;
        }
    private:
        int callB;
        void inc(){
            callB++;
        }
    protected:
        void func(int & a)
        {
            a = a * 3;
            inc();
        }
    public:
        int getB(){
            return callB;
        }
};

class C
{
    public:
        C(){
            callC = 0;
        }
    private:
        int callC;
        void inc(){
            callC++;
        }
    protected:
        void func(int & a)
        {
            a = a * 5;
            inc();
        }
    public:
        int getC(){
            return callC;
        }
};
```

```
class D: private A,B,C
{
private:
    int val;
public:
    D(): val(1){}//Initially val is 1

    void update_val(int new_val)
    {
        while(new_val % 2 == 0)
        {
            new_val /= 2;
            A::func(val);
        }
        while(new_val % 3 == 0)
        {
            new_val /= 3;
            B::func(val);
        }
        while(new_val % 5 == 0)
        {
            new_val /= 5;
            C::func(val);
        }
    }
    void check(int);
};


void D::check(int new_val)
{
    update_val(new_val);
    cout << "Value = " << val << endl << "A's func called " << getA() << " times " << endl << "B's func called " << getB() << " times" << endl << "C's func called " << getC() << " times" << endl;
}


int main()
{
    D d;
    int new_val;
    cin >> new_val;
    d.check(new_val);

}
```

Line: 100 Col: 3

□ Test against custom input

Run Code Submit Code

Upload Code as File

```
1080
```

Congrats, you solved this challenge!

Challenge your friends:

✓ Test Case #0
✓ Test Case #1
✓ Test Case #2
✓ Test Case #3
✓ Test Case #4
✓ Test Case #5
✓ Test Case #6
✓ Test Case #7
✓ Test Case #8
✓ Test Case #9

You've earned 30.00 points.

Next Challenge

Contest Calendar|Blog|Scoring|Environment|FAQ|About Us|Support|Careers|Terms Of Service|Privacy Policy|Request a Feature