

由于类模板的特殊性

把类的声明和成员函数体放在同一个文件中"head.h"

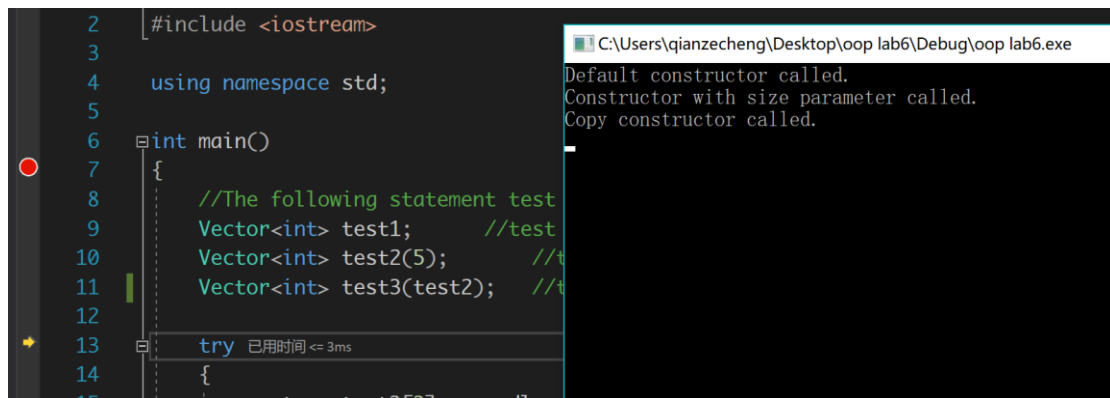
该文件中实现了：默认构造函数，带一个参数的构造函数，一个拷贝构造函数，一个默认的析构函数，重载了索引操作符[]并在里面实现了抛出异常（访问越界的时候）以及另外两个成员函数 size 和 inflate。

首先声明，我没用显示调用析构函数，因此当 main 函数运行完的时候析构函数才会被调用，要下了断点才能看到。另外在实现的时候使用了类模板，而测试的时候仅就 int 情况做了测试，因为其他的数据类型道理是相同的。

测试方法是单步调试，在每个成员函数中输出提示语句表示该成员函数被调用，以下是我的测试代码：

```
1. #include "head.h"
2. #include <iostream>
3.
4. using namespace std;
5.
6. int main()
7. {
8.     //The following statement test both constructors and "size" member function
9.     Vector<int> test1;           //test default constructor
10.    Vector<int> test2(5);        //test constructor with size specified
11.    Vector<int> test3(test2);    //test copy constructor
12.
13.    try
14.    {
15.        cout << test3[3] << endl;    //in range, so output 0(initialization value)
16.        cout << test3[5] << endl;    //index out of range, exception caught
17.    }
18.    catch (IndexOutOfBounds)
19.    {
20.        cerr << "Index Out Of Bounds!!!" << endl;
21.        //to avoid trouble when testing, this line is noted
22.        //exit(1);
23.    }
24.
25.    int size = test3.size();        //test "size" function
26.    cout << "Old size: " << size << endl;
27.    int inflate_size = test3.inflate(5);
28.    cout << "New size: " << inflate_size << endl;
29.    return 0;
30. }
```

测试过程：



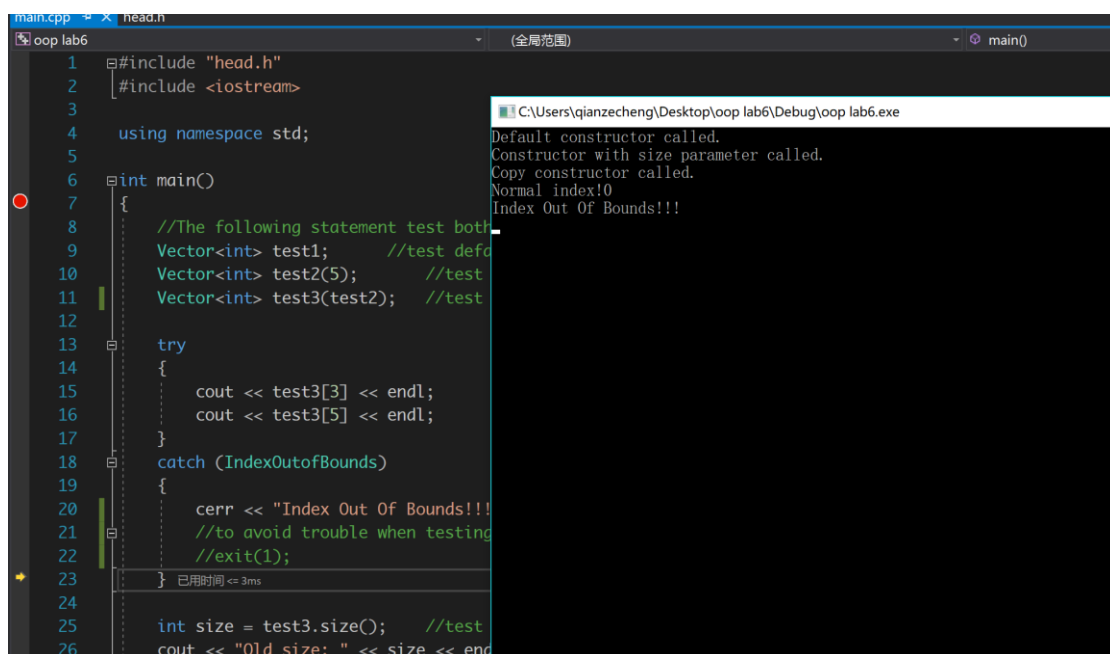
The screenshot shows a C++ IDE with a code editor on the left and a debug console on the right. The code in the editor is as follows:

```
2  #include <iostream>
3
4  using namespace std;
5
6  int main()
7  {
8      //The following statement test
9      Vector<int> test1;    //test
10     Vector<int> test2(5); //test
11     Vector<int> test3(test2); //test
12
13     try 已用时间 <= 3ms
14     {
15         cout << test2[3] << endl;
16     }
```

The debug console on the right shows the following output:

```
C:\Users\qianzecheng\Desktop\oop lab6\Debug\oop lab6.exe
Default constructor called.
Constructor with size parameter called.
Copy constructor called.
```

先是测试构造函数，如上图所示，可以看到三个构造函数都被成功调用了



The screenshot shows a C++ IDE with a code editor on the left and a debug console on the right. The code in the editor is as follows:

```
1  #include "head.h"
2  #include <iostream>
3
4  using namespace std;
5
6  int main()
7  {
8      //The following statement test both
9      Vector<int> test1;    //test defa
10     Vector<int> test2(5);  //test
11     Vector<int> test3(test2); //test
12
13     try
14     {
15         cout << test3[3] << endl;
16         cout << test3[5] << endl;
17     }
18     catch (IndexOutOfBounds)
19     {
20         cerr << "Index Out Of Bounds!!!
21         //to avoid trouble when testing
22         //exit(1);
23     } 已用时间 <= 3ms
24
25     int size = test3.size(); //test
26     cout << "Old size: " << size << endl;
```

The debug console on the right shows the following output:

```
C:\Users\qianzecheng\Desktop\oop lab6\Debug\oop lab6.exe
Default constructor called.
Constructor with size parameter called.
Copy constructor called.
Normal index!0
Index Out Of Bounds!!!
```

上面测试操作符[]以及测试异常，可以看到第一行 test3[3]的时候，未触发异常，正常输出（提示为 Normal index! 0，其中 0 是哪个索引位置对应的值）

当执行 test3[5]的时候，发生了越界访问，异常被触发，执行 catch 语句块，输出错误提示语句。正常来说此时应该要处理这个异常，终止程序（或其他处理异常的方法），但是为了继续进行测试，这边没有处理这个异常。

```
11 Vector<int> test3(test2); //test copy constructor
12
13 try
14 {
15     cout << test3[3] << endl;
16     cout << test3[5] << endl;
17 }
18 catch (IndexOutOfBounds)
19 {
20     cerr << "Index Out Of Bounds!!!"
21     //to avoid trouble when testing
22     //exit(1);
23 }
24
25 int size = test3.size(); //test
26 cout << "Old size: " << size << endl;
27 int inflate_size = test3.inflate(5)
28 cout << "New size: " << inflate_size
29 return 0;
30 }
```

C:\Users\qianzecheng\Desktop\oop lab6\Debug\oop lab6.exe
Default constructor called.
Constructor with size parameter called.
Copy constructor called.
Normal index!0
Index Out Of Bounds!!!
Size function called.
Old size: 5

上图测试了 size 成员函数，由于之前定义的时候 test3 的 size 为 5，这边输出也是 5，说明 size 成员函数正确

```
oop lab6 (全局范围)
10 Vector<int> test2(5); //test construcot with size specified
11 Vector<int> test3(test2); //test copy constructor
12
13 try
14 {
15     cout << test3[3] << endl;
16     cout << test3[5] << endl;
17 }
18 catch (IndexOutOfBounds)
19 {
20     cerr << "Index Out Of Bounds!!!"
21     //to avoid trouble when testing
22     //exit(1);
23 }
24
25 int size = test3.size(); //test
26 cout << "Old size: " << size << endl;
27 int inflate_size = test3.inflate(5)
28 cout << "New size: " << inflate_size
29 return 0; 已用时间 <= 2ms
30 }
```

C:\Users\qianzecheng\Desktop\oop lab6\Debug\oop lab6.exe
Default constructor called.
Constructor with size parameter called.
Copy constructor called.
Normal index!0
Index Out Of Bounds!!!
Size function called.
Old size: 5
Inflate function called.
New size: 10

上图测试了 inflate 成员函数，由于调用 inflate 的时候传入参数为 5，加上原来的 size，新的 size 变成了 10，正确。

```
main.cpp x head.h
oop lab6 (全局范围)
10 Vector<int> test2(5); //test construcot with size specified
11 Vector<int> test3(test2); //test copy constructor
12
13 try
14 {
15     cout << test3[3] << endl;
16     cout << test3[5] << endl;
17 }
18 catch (IndexOutOfBounds)
19 {
20     cerr << "Index Out Of Bounds!!!"
21     //to avoid trouble when testing
22     //exit(1);
23 }
24
25 int size = test3.size(); //test
26 cout << "Old size: " << size << endl;
27 int inflate_size = test3.inflate(5)
28 cout << "New size: " << inflate_size << endl;
29 return 0;
30 } 已用时间 <= 5ms
```

C:\Users\qianzecheng\Desktop\oop lab6\Debug\oop lab6.

Default constructor called.
Constructor with size parameter called.
Copy constructor called.
Normal index!0
Index Out Of Bounds!!!
Size functon called.
Old size: 5
Inflate function called.
New size: 10
Destructor called.
Destructor called.
Destructor called.

上图测试了析构函数，可以观察到，main 函数返回时，三个对象均被析构了。