

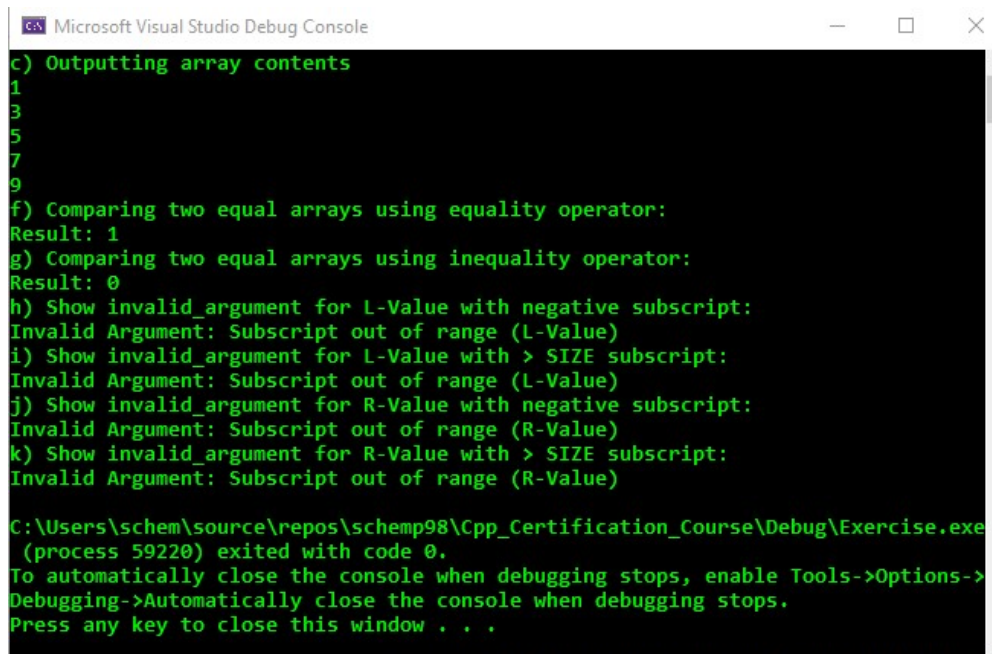
```
1 //
2 // Shaun Chemplavil U08713628
3 // shaun.chemplavil@gmail.com
4 // C/C++ Programming III : Intermediate Programming with Objects
5 // 151116 Raymond L. Mitchell III
6 // Array.h
7 // Win10
8 // Visual C++ 19.0
9 //
10 // File contains the class definition for Array template class
11 //
12
13 #ifndef SHAUNCHEMPLAVIL_ARRAY_H
14 #define SHAUNCHEMPLAVIL_ARRAY_H
15
16 #include<iostream>
17 #include <stdexcept>
18
19 using std::invalid_argument;
20 using std::cerr;
21
22 namespace ShaunChemplavil
23 {
24     template <typename ElemType = int, int SIZE = 3>
25     class Array
26     {
27     public:
28         // Default Constructor
29         Array() {}
30
31         // Copy Constructor
32         Array(const Array &source)
33         {
34             for (int idx = 0; idx < SIZE; idx++)
35                 elements[idx] = source.elements[idx];
36         }
37
38         // Copy Assignment Operator
39         const Array &operator=(const Array &source)
40         {
41             // Prevent Self-Assignment
42             if (this == &source)
43                 return *this;
44
45             for (int idx = 0; idx < SIZE; idx++)
46                 elements[idx] = source.elements[idx];
47
48             /*this = source;
49
50             return *this;
51         }
52
```

```
53     bool operator==(const Array &other) const
54     {
55         for (int idx = 0; idx < SIZE; idx++)
56             if (elements[idx] != other.elements[idx])
57                 return false;
58
59         return true;
60     }
61
62     bool operator!=(const Array &other) const
63     {
64         return (!(*this == other));
65     }
66
67     // Subscript operator (L-value version)
68     ElemType& operator[](int index)
69     {
70         try
71         {
72             // check if valid index
73             if ((index < 0) || (index >= SIZE))
74                 throw invalid_argument{"Subscript out of range (L-Value)"};
75
76             return elements[index];
77         }
78         catch (invalid_argument &ex)
79         {
80             cerr << "Invalid Argument: " << ex.what() << "\n";
81             return elements[0];
82         }
83     }
84
85     // Subscript operator (R-value version)
86     ElemType operator[](int index) const
87     {
88         try
89         {
90             // check if valid index
91             if ((index < 0) || (index >= SIZE))
92                 throw invalid_argument{"Subscript out of range (R-Value)"};
93
94             return elements[index];
95         }
96         catch (invalid_argument &ex)
97         {
98             cerr << "Invalid Argument: " << ex.what() << "\n";
99             return elements[0];
100         }
101     }
102
103 private:
104     ElemType elements[SIZE];
```

```
105     };  
106 }  
107  
108 #endif  
109
```

```
1  //
2  // Shaun Chemplavil U08713628
3  // shaun.chemplavil@gmail.com
4  // C/C++ Programming III : Intermediate Programming with Objects
5  // 151116 Raymond L. Mitchell III
6  // hw6.cpp
7  // Win10
8  // Visual C++ 19.0
9  //
10 // Test Program for the Array Template class
11 //
12
13 #include "Array.h"
14
15 #include <iostream>
16
17 using std::cout;
18 using ShaunChemplavil::Array;
19
20 const int ARRAY_SIZE = 5;
21
22 int main()
23 {
24     // a ) Creating default array with " << ARRAY_SIZE << " ints
25     Array<int, ARRAY_SIZE> arrayOfFiveInts;
26
27     // b) Modifying array with arbitrary values
28     for (int idx = 0; idx < ARRAY_SIZE; idx++)
29         arrayOfFiveInts[idx] = 2 * idx + 1;
30
31     cout << "c) Outputting array contents\n";
32     for (int idx = 0; idx < ARRAY_SIZE; idx++)
33         cout << arrayOfFiveInts[idx] << "\n";
34
35     // d) creating a const Array using the copy constructor
36     const Array<int, ARRAY_SIZE> constArrayOfFiveInts(arrayOfFiveInts);
37
38     // e) Assigning Array to existing Array using copy assignment
39     Array<int, ARRAY_SIZE> copyAssignArrayOfFiveInts;
40     copyAssignArrayOfFiveInts = arrayOfFiveInts;
41
42     cout << "f) Comparing two equal arrays using equality operator:\n"
43         << "Result: " << (arrayOfFiveInts == constArrayOfFiveInts) << "\n";
44
45     cout << "g) Comparing two equal arrays using inequality operator:\n"
46         << "Result: " << (arrayOfFiveInts != copyAssignArrayOfFiveInts) << "\n";
47
48     cout << "h) Show invalid_argument for L-Value with negative subscript:\n";
49     copyAssignArrayOfFiveInts[-1] = 5;
50
51     cout << "i) Show invalid_argument for L-Value with > SIZE subscript:\n";
52     copyAssignArrayOfFiveInts[ARRAY_SIZE + 1] = 5;
```

```
53
54     cout << "j) Show invalid_argument for R-Value with negative subscript:\n";
55     int temp = constArrayOfFiveInts[-1];
56
57     cout << "k) Show invalid_argument for R-Value with > SIZE subscript:\n";
58     temp = constArrayOfFiveInts[ARRAY_SIZE + 1];
59 }
60
```



The screenshot shows the Microsoft Visual Studio Debug Console window. The output text is as follows:

```
c) Outputting array contents
1
3
5
7
9
f) Comparing two equal arrays using equality operator:
Result: 1
g) Comparing two equal arrays using inequality operator:
Result: 0
h) Show invalid_argument for L-Value with negative subscript:
Invalid Argument: Subscript out of range (L-Value)
i) Show invalid_argument for L-Value with > SIZE subscript:
Invalid Argument: Subscript out of range (L-Value)
j) Show invalid_argument for R-Value with negative subscript:
Invalid Argument: Subscript out of range (R-Value)
k) Show invalid_argument for R-Value with > SIZE subscript:
Invalid Argument: Subscript out of range (R-Value)

C:\Users\schem\source\repos\schemp98\Cpp_Certification_Course\Debug\Exercise.exe
(process 59220) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->
Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```