Lab 8: Templates

1.1 Information

Topics: Templates and functions, templates and classes

Turn in: All source files (.cpp and .hpp).

Starter files: Download on GitHub or D2L.

```
Lab 08 - Templates/
├── lab8.cpp ...  Contains main()
├── lab8_sorting.hpp ...  File for 1st part of
│                         lab
├── lab8_arraywrapper.cpp ...  File for 2nd part of
│                              lab
├── CodeBlocks Project/ ...  Code::Blocks project here
├── Makefile/ ...  Makefile here
└── VS2015 Project/ ...  Visual Studio 2015 project here
```

The `lab8.cpp` file contains `main()` as well as the program code that tests your code. However, these aren't unit tests - it is up to you to check the program output and make sure it is being displayed correctly.

Contents

1.1 Information 1
1.2 Templates and functions: A sort function 3
1.2.1 DisplayArray 3
1.2.2 Swap 3
1.2.3 SelectionSort 3
1.2.4 Testing 4
1.3 Templates and classes: An array wrapper 5
1.3.1 ArrayWrapper constructor 6
1.3.2 Push 6
1.3.3 Display 6
1.3.4 Testing 7
1.4 Appendix A: Starter code 9

1.2 Templates and functions: A sort function

In the `Part1()` test in the `lab8.cpp` file, several arrays are created and then sorted via the `SelectionSort` function that is declared in the `lab8_sorting.hpp` file. Using templates will allow us to write one version of this function that will work for any data-type that has the relational operators implemented for that type (i.e., for your own class, you would implement `operator<`, `operator>`, etc.)

There are three functions that you will have to implement in `lab8_sorting.hpp`: SelectionSort, Swap, and DisplayArray.

1.2.1 DisplayArray

For this function, use a for loop to iterate through all the elements of the array `a[]` in a horizontal line. The output should look like this:

```
'a' 'b' 'c' 'd'
```

1.2.2 Swap

The Swap function is used by the SelectionSort function. This function takes two parameters: `a` and `b`, two elements of some array. Swap should change it so that `a` has `b`'s value, and vice versa.

The following is an **incorrect** way to implement Swap:

```
// This won't work
a = b;
b = a;
```

You will need to declare a local variable within the function to help you implement the Swap functionality correctly.

1.2.3 SelectionSort

Within this function, you will implement the simple SelectionSort algorithm. You can find the algorithm's implementation on Wikipedia:

https://en.wikipedia.org/wiki/Selection_sort#Implementation

Make sure to add a comment in your code "citing" this algorithm within your code - do not pass it off as your own work.

1.2.4 Testing

When you run the program, the output should look as follows. (Note that this example shows text wrapping, which happens if the window is too small, but your DisplayArray function should display all the elements on the same line.)

```
------------------------------
PART 1: TEMPLATES & FUNCTIONS
------------------------------

TEST 1
Before sort:
'5' '3' '6' '1' '2'
After sort:
'1' '2' '3' '5' '6'

TEST 2
Before sort:
'h' 'e' 'q' 'a' 'd' 'z' 'c'
After sort:
'a' 'c' 'd' 'e' 'h' 'q' 'z'

TEST 3
Before sort:
'Andhra Pradesh' 'Goa' 'Haryana' 'Bihar' '
   Uttarakhand' 'Telangana' 'Rajasthan' 'Tamil Nadu'
    'Punjab'
After sort:
'Andhra Pradesh' 'Bihar' 'Goa' 'Haryana' 'Punjab' '
   Rajasthan' 'Tamil Nadu' 'Telangana' 'Uttarakhand'
```

1.3 Templates and classes: An array wrapper

For this part of the lab, you will implement a barebones static array wrapper, but using a template. The starter code contains the following in `lab8_arraywrapper`:

```
1 #ifndef _ARRAY_WRAPPER
2 #define _ARRAY_WRAPPER
3
4 const int MAX_SIZE = 10;
5
6 #endif
```

Within this file, you will declare a class and use templates.

NOTE: Since we're using templates, the class/function declarations AND the function definitions must all go in this file.

Declare a class named `ArrayWrapper`. It will have the following **private member variables:**

- `m_data`, an array whose data type is `T`, the placeholder for the template. Set this array's size to the `MAX_SIZE` named constant.
- `m_itemCount`, an integer.

Within the class, you will declare and define the functions. As an example, here's a different class as an example:

```
class Bob
{
    Bob() // defined WITHIN the class
    {
        name = "Bob";
    }

    void Display() // defined WITHIN the class
    {
        cout << name << endl;
    }

    private:
    string name;
};
```

Implement the following functions:

1.3.1 ArrayWrapper constructor

Parameters: None

This constructor should initialize `m_itemCount` to 0.

1.3.2 Push

Parameters: `data`, type is the placeholder `T`.

Return type: void

If the `m_itemCount` is less than `MAX_SIZE`, then set `m_data` at the next available index (at `m_itemCount`) to the value passed in as `data` and increment `m_itemCount`.

1.3.3 Display

Parameters: None

Return type: void

Use a for loop to iterate through each element of the array, displaying the value of the element to the screen via `cout`. Have every element outputted on the same line.

1.3.4 Testing

Before running the program, make sure to comment out the test code:

```
1  void Part2()
2  {
3      cout << "--------------------------" << endl;
4      cout << "PART 2: TEMPLATES & CLASSES" << endl;
5      cout << "--------------------------" << endl;
6
7      cout << "Uncomment out this code once the
       ArrayWrapper class is implemented" << endl;
8
9  //     // Test 1
10 //     cout << endl << "TEST 1" << endl;
11 //     ArrayWrapper<int> numArray;
12 //     numArray.Push( 2 );
13 //     numArray.Push( 4 );
14 //     numArray.Push( 6 );
15 //     numArray.Push( 8 );
16 //     numArray.Push( 10 );
17 //     numArray.Display();
18 //
19 //     // Test 2
20 //     cout << endl << "TEST 2" << endl;
21 //     ArrayWrapper<char> charArray;
22 //     charArray.Push( 'c' );
23 //     charArray.Push( 'a' );
24 //     charArray.Push( 't' );
25 //     charArray.Push( 's' );
26 //     charArray.Push( '\0' );
27 //     charArray.Display();
28 //
29 //     // Test 3
30 //     cout << endl << "TEST 3" << endl;
31 //     ArrayWrapper<string> strArray;
32 //     strArray.Push( "Ontario" );
33 //     strArray.Push( "Quebec" );
34 //     strArray.Push( "Nova Scotia" );
35 //     strArray.Push( "Alberta" );
36 //     strArray.Push( "British Columbia" );
37 //     strArray.Display();
38 }
```

When you run the program, the output should look as follows. (Note that this example shows text wrapping, which happens if the window is too small, but your DisplayArray function should display all the elements on the same line.)

```
---------------------------
PART 2: TEMPLATES & CLASSES
---------------------------

TEST 1
'2' '4' '6' '8' '10'

TEST 2
'c' 'a' 't' 's' ''

TEST 3
'Ontario' 'Quebec' 'Nova Scotia' 'Alberta' 'British
   Columbia'
```

1.4 Appendix A: Starter code

lab8.cpp

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

#include "lab8_sorting.hpp"
#include "lab8_arraywrapper.hpp"

void Part1();
void Part2();

int main()
{
    Part1();
    Part2();

    return 0;
}

void Part1()
{
    cout << "-----------------------------" << endl;
    cout << "PART 1: TEMPLATES & FUNCTIONS" << endl;
    cout << "-----------------------------" << endl;

    // Test 1
    cout << endl << "TEST 1" << endl;
    int numArray[] = { 5, 3, 6, 1, 2 };
    int numArraySize = 5;

    cout << "Before sort: " << endl;
    DisplayArray( numArray, numArraySize );

    SelectionSort( numArray, numArraySize );

    cout << "After sort: " << endl;
    DisplayArray( numArray, numArraySize );

    // Test 2
    cout << endl << "TEST 2" << endl;
```

```
40      char charArray[] = { 'h', 'e', 'q', 'a', 'd', 'z', '
    c' };
41      int charArraySize = 7;
42
43      cout << "Before sort: " << endl;
44      DisplayArray( charArray, charArraySize );
45
46      SelectionSort( charArray, charArraySize );
47
48      cout << "After sort: " << endl;
49      DisplayArray( charArray, charArraySize );
50
51      // Test 3
52      cout << endl << "TEST 3" << endl;
53      string strArray[] = { "Andhra Pradesh", "Goa", "
    Haryana", "Bihar", "Uttarakhand", "Telangana", "
    Rajasthan", "Tamil Nadu", "Punjab" };
54      int strArraySize = 9;
55
56      cout << "Before sort: " << endl;
57      DisplayArray( strArray, strArraySize );
58
59      SelectionSort( strArray, strArraySize );
60
61      cout << "After sort: " << endl;
62      DisplayArray( strArray, strArraySize );
63  }
64
65  void Part2()
66  {
67      cout << "--------------------------" << endl;
68      cout << "PART 2: TEMPLATES & CLASSES" << endl;
69      cout << "--------------------------" << endl;
70
71      cout << "Uncomment out this code once the
    ArrayWrapper class is implemented" << endl;
72
73  //      // Test 1
74  //      cout << endl << "TEST 1" << endl;
75  //      ArrayWrapper<int> numArray;
76  //      numArray.Push( 2 );
77  //      numArray.Push( 4 );
78  //      numArray.Push( 6 );
```

```
79 //     numArray.Push( 8 );
80 //     numArray.Push( 10 );
81 //     numArray.Display();
82 //
83 //     // Test 2
84 //     cout << endl << "TEST 2" << endl;
85 //     ArrayWrapper<char> charArray;
86 //     charArray.Push( 'c' );
87 //     charArray.Push( 'a' );
88 //     charArray.Push( 't' );
89 //     charArray.Push( 's' );
90 //     charArray.Push( '\0' );
91 //     charArray.Display();
92 //
93 //     // Test 3
94 //     cout << endl << "TEST 3" << endl;
95 //     ArrayWrapper<string> strArray;
96 //     strArray.Push( "Ontario" );
97 //     strArray.Push( "Quebec" );
98 //     strArray.Push( "Nova Scotia" );
99 //     strArray.Push( "Alberta" );
100 //     strArray.Push( "British Columbia" );
101 //     strArray.Display();
102 }
103
```

lab8_sorting.hpp

```
1  #ifndef _SORTING
2  #define _SORTING
3
4  /**
5      Sets the value of a to the original value of b,
6      and sets the value of b to the original value of a.
7
8      @param <T&> a   The reference to an element of an
       array
9      @param <T&> b   The reference to another element of
       an array
10     @return <void>  No return - values are changed via
       pass-by-reference parameters
11 */
```

```
12 template <typename T>
13 void Swap( T& a, T& b )
14 {
15 }
16
17 /**
18     Implementation of Selection Sort.
19     See for reference: https://en.wikipedia.org/wiki/
   Selection_sort#Implementation
20
21     @param <T[]> a  An array to sort
22     @param <int> n  The size of the array a[]
23     @return <void>  No return - array is passed-by-
   reference and modified directly
24 */
25 template <typename T>
26 void SelectionSort( T a[], int n )
27 {
28 }
29
30 /**
31     Display each element of the array, from 0 to n-1
32
33     @param <T[]> a  An array to sort
34     @param <int> n  The size of the array a[]
35     @return <void>  No return
36 */
37 template <typename T>
38 void DisplayArray( const T a[], int n )
39 {
40 }
41
42 #endif
43
```

lab8_arraywrapper.hpp

```
1 #ifndef _ARRAY_WRAPPER
2 #define _ARRAY_WRAPPER
3
4 const int MAX_SIZE = 10;
5
```

```
6 #endif
7
```