COMP 2012 Object-Oriented Programming and Data Structures

Lab 5 Template + Operator Overloading

Image by Freepik

Overview

In this lab, you are going to implement a sorted array that can support different data types to demonstrate your understanding of template and operator overloading in C++. In particular, data type of `int` and a self-defined type of `Course` will be used for testing of the program.

You can download the skeleton code in a zip package HERE.

SortedArray

```
template <typename T, int MAX_SIZE = 10, bool ASCENDING = true>
class SortedArray {
  public:
    /* Constructors */
    // Default constructor
    SortedArray();
    // Copy constructor
    SortedArray(const SortedArray& other) = default;
    // Conversion constructor
    SortedArray(int x);
    // Another constructor
    SortedArray(const T arr[], int n);

    /* Accessors */
    // Checks whether the array is full or not
    bool full() const;
    // Checks whether the array is empty or not
    bool empty() const;
    // Returns the number of elements
    int size() const;
    // Returns the maximum possible number of elements
    int max_size() const;

    /* Mutators */
    // Inserts an element in the appropriate position
    void insert(const T& value);

    /* Operators */
    // subscription operator: to return an element at the given index
    T operator[](int index) const;
    // equal-to operator: to compare two SortedArray objects
    bool operator==(const SortedArray& other) const;

  private:
    T data[MAX_SIZE];
    int num_elements;
};
```

The definition of this class can be found in **sortedArray.h**. It is a **class template** with a type parameter, `T`, and two non-type parameters, namely, `MAX_SIZE` and `ASCENDING`. It maintains a sorted array of items/objects.

You are required to add the whole function definition (i.e. header + body) outside the class definition for the 3 operators (`[]`, `==`, `+`) in **sortedArray.h** (TODO #1, TODO #2, TODO #3). The output operator `<<` and other member functions have already been made available to you.

Given two sorted arrays **A** and **B** which are instantiated with the same template arguments, i.e. they are for storing the values with the same type (T), with the same capacity (MAX_SIZE) and the same order (ASCENDING).

- The result of **A** `+` **B** can be obtained by creating a sorted array **C** and copying all the elements of **A** to it
- Then, inserting elements from **B** to **C** one by one
- The process would end when all the elements of **B** have been inserted to **C** or when the number of elements in **C** reaches the pre-defined maximum size of `MAX_SIZE`.

Notice that **A** `+` **B** and **B** `+` **A** might give different results. For example, when we have:

```
A = {3, 5, 8, 24, 36, 82, 91}
B = {7, 12, 44, 52}
MAX_SIZE = 10
```

The results would be:

```
A + B = {3, 5, 7, 8, 12, 24, 36, 44, 82, 91}
B + A = {3, 5, 7, 8, 12, 24, 36, 44, 52, 82}
```

Course

```
class Course {
  public:

    /* Constructors */
    // Default constructor
    Course();

    // Conversion constructor
    Course(const string& s);

    /* Accessor */
    // Gets the course code
    string get_code() const;

    /* Mutator */
    // Sets the course code if it is valid
    void set_code(const string& s);

    /* Operator overloading */
    // smaller-than operator:
    // Checks whether the course code is smaller than the other course code
    bool operator<(const Course& other) const;

    // larger-than operator:
    // Checks whether the course code is larger than the other course code
    bool operator>(const Course& other) const;

    // equal-to operator:
    // Checks whether the two courses are with the same course code
    bool operator==(const Course& other) const;

    // not-equal-to operator:
    // Checks whether the two courses are with different course code
    bool operator!=(const Course& other) const;

  private:
```

Lab Tasks

Your task is to implement all missing function definition for the specific operators so that the program can produce the correct results as expected. You are required to add appropriate lines of code in 8 different places in **sortedArray.h** and **course.cpp**.

TODO #1
Implement the subscription operator `[]` in **sortedArray.h**

TODO #2
Implement the equal-to operator `==` in **sortedArray.h**

TODO #3
Implement the global addition operator `+` in **sortedArray.h**

TODO #4
Implement the smaller-than operator `<` in **course.cpp**

TODO #5
Implement the larger-than operator `>` in **course.cpp**

TODO #6
Implement the equal-to operator `==` in **course.cpp**

TODO #7

Implement the not-equal-to operator `!=` in **course.cpp**

TODO #8
Implement the global input operator `>>` in **course.cpp**

Sample Sessions

Here are three sample sessions of using the program. User inputs are shown in bold and red, like this: **3 36 8**

Sample Session #1 (user inputs for this session can be found in **input/input1.txt** in the zip package)

```
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 1
For integer sequence 1
Enter the number of elements: 3
3 36 8
For integer sequence 2
Enter the number of elements: 3
36 3 8
Is {3, 8, 36} equal to {3, 8, 36}? Yes
{3, 8, 36} + {3, 8, 36} = {3, 3, 8, 8, 36, 36}
{3, 8, 36} + {3, 8, 36} = {3, 3, 8, 8, 36, 36}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 1
For integer sequence 1
Enter the number of elements: 7
3 5 8 24 36 82 91
For integer sequence 2
Enter the number of elements: 4
7 12 44 52
Is {3, 5, 8, 24, 36, 82, 91} equal to {7, 12, 44, 52}? No
{3, 5, 8, 24, 36, 82, 91} + {7, 12, 44, 52} = {3, 5, 7, 8, 12, 24, 36, 44, 82, 91}
{7, 12, 44, 52} + {3, 5, 8, 24, 36, 82, 91} = {3, 5, 7, 8, 12, 24, 36, 44, 52, 82}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): -1
```

Sample Session #2 (user inputs for this session can be found in **input/input2.txt** in the zip package)

```
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 2
For course sequence 1
Enter the number of elements: 2
COMP2012 COMP2611
For course sequence 2
Enter the number of elements: 2
comp2611 cOMP2012
Is {COMP2611, COMP2012} equal to {COMP2611, COMP2012}? Yes
{COMP2611, COMP2012} + {COMP2611, COMP2012} = {COMP2611, COMP2611, COMP2012, COMP2012}
{COMP2611, COMP2012} + {COMP2611, COMP2012} = {COMP2611, COMP2611, COMP2012, COMP2012}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 2
For course sequence 1
Enter the number of elements: 3
COMP2012 LANG4030 ENGG2010
For course sequence 2
Enter the number of elements: 2
comp2011 COMP2611
Is {LANG4030, ENGG2010, COMP2012} equal to {COMP2611, COMP2011}? No
{LANG4030, ENGG2010, COMP2012} + {COMP2611, COMP2011} = {LANG4030, ENGG2010, COMP2611, COMP2012,
COMP2011}
{COMP2611, COMP2011} + {LANG4030, ENGG2010, COMP2012} = {LANG4030, ENGG2010, COMP2611, COMP2012,
COMP2011}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): -1
```

Sample Session #3 (user inputs for this session can be found in **input/input3.txt** in the zip package)

```
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 1
For integer sequence 1
Enter the number of elements: 4
3 5 -8 24
For integer sequence 2
Enter the number of elements: 5
12 4 -27 -5 52
Is {-8, 3, 5, 24} equal to {-27, -5, 4, 12, 52}? No
{-8, 3, 5, 24} + {-27, -5, 4, 12, 52} = {-27, -8, -5, 3, 4, 5, 12, 24, 52}
{-27, -5, 4, 12, 52} + {-8, 3, 5, 24} = {-27, -8, -5, 3, 4, 5, 12, 24, 52}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): 2
For course sequence 1
Enter the number of elements: 4
COMP2012 LANG4030 ENGG2010 MATH2411
For course sequence 2
Enter the number of elements: 2
comp2011 COMP2611
Is {MATH2411, LANG4030, ENGG2010, COMP2012} equal to {COMP2611, COMP2011}? No
{MATH2411, LANG4030, ENGG2010, COMP2012} + {COMP2611, COMP2011} = {MATH2411, LANG4030, ENGG2010,
COMP2611, COMP2012, COMP2011}
{COMP2611, COMP2011} + {MATH2411, LANG4030, ENGG2010, COMP2012} = {MATH2411, LANG4030, ENGG2010,
COMP2611, COMP2012, COMP2011}
Option 1: Integer sequences
Option 2: Course sequences
Select an option (enter -1 to exit): -1
```

Compile and Test

Submission & Deadline

The deadline of submission for all lab sections is **20:00:00 on 14 April 2023 (Fri)**.

Please submit your completed work to ZINC by zipping the following 2 files.

```
sortedArray.h
course.cpp
```

ZINC usage instructions can be found here. You can make multiple submissions to ZINC before the deadline. Only the last submission received before the deadline will be graded.

Menu

Page maintained by

Namkiu Chan
Last Modified: 03/29/2023 23:24:15

Homepage

Course Homepage