Programming Languages: Project #1 - Extendable Array

SWE3006 Programming Languages - Fall 2019

Due date: September 27 (Fri) 11:59pm

1 Extendable Array

Write `ExtendableArray`, an array of integers class, which supports the following functionalities:

- 1. Conceptually, an ExtendableArray is an infinite length array initialized to all zeros.
- 2. A default ExtendableArray constructor should allocate memory space to hold 2 integers.
- 3. It should be expandable. Assigning to an element of the array that hasn't been allocated yet should transparently reallocate the memory space. I.e. operator[] may need more memory space than currently allocated, which results in memory reallocation. Note that your object should not allocate memory space more than necessary.
- 4. Write operator[] so that it returns an ElementRef (see ExtendableArray.h provided).
- 5. If the index to operator[] used on the right side of an assignment operation is outside the currently allocated space, the operation returns 0 without reallocating more memory space (e.g., having int x = a[10000000000]; shouldn't blow out your memory).

2 Header Files

You are not allowed to make any change to the provided ExtendableArray.h. Your class should be implemented in ExtendableArray.cpp. Provided source files are located in /home/swe3021/proj1.

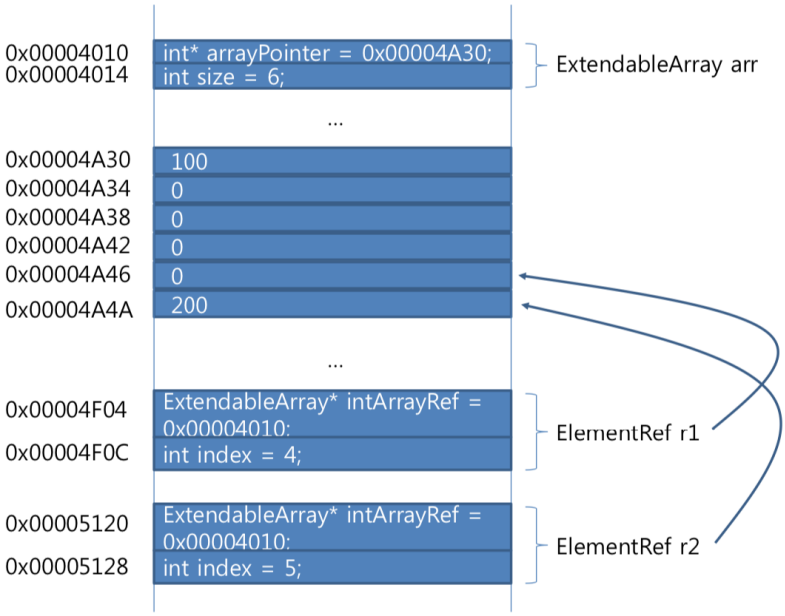
2.1 ExtendableArray.h

```
//
// Expandable integer array class //
#ifndef EARRAY_H
#define EARRAY_H

#include <ostream>
using namespace std;

class ExtendableArray;

class ElementRef
{
    private:
        ExtendableArray *intArrayRef; //pointer to the array
        int index; // index of this element
    public:
        ElementRef( ExtendableArray& theArray, int i );
        ElementRef( const ElementRef& other ); // copy constructor
        ~ElementRef();

        ElementRef& operator=( const ElementRef& rhs );
        ElementRef& operator=( int val );
```

```
        operator int() const;
};

class ExtendableArray {
    private:
        int *arrayPointer; // integer array pointer
        int size; // the size of array
    public:
        ExtendableArray(); //allocates memory space for 2 integers
        ExtendableArray( const ExtendableArray& other ); // copy constructor
        ~ExtendableArray(); // destructor

        ExtendableArray& operator=( const ExtendableArray& rhs );
        ElementRef operator[]( int i );

        friend class ElementRef; // ElementRef class can access my private members
        friend ostream& operator<< (ostream& o, const ExtendableArray& t){
            for(int i=0;i<t.size;i++){
                o << t.arrayPointer[i] << " ";
            }
            return o;
        }
};
#endif // EARRAY_H
```

3 ElementRef Class

The following figure shows how ElementRef is used to access array element of ExtendableArray object.



Remember that "reference" is a variable that a compiler uses to refer to a memory space. When we use a reference (variable) on the left hand side of `=`, for example, `myRef = 1;`, 1 will be stored in the memory referenced by `myRef`. We call this memory location as "L-value" of a variable (or reference).

If you use the reference on the right hand side of `=`, for example, `int var = myRef;`, the value stored in the memory space referenced by `myRef` will be used, i.e., its value will copied to the memory space referenced by `var`. We call the value stored in a reference variable as "R-value".

Consider the following example code.

```
ExtendableArray arr;
arr[10000] = 1;
cout << arr[10000] << endl;
```

Since `arr[10000]` can be used as either L-value or R-value, `ExtendableArray::operator[]` must return a reference type object. However as your `arrayPointer` is pointing to a sequence of integers that do not have "references", we have to use `ElementRef` class object that pretends to be a reference of an integer in the array.

When ElementRef object is returned from `operator[]`, the object will call the overloaded `operator=` if the object is used as L-value. Otherwise, it automatically calls type conversion operator so that it can be used as R-value if necessary.

When the ElementRef object is used as L-value, it must check if the index is greater than the array size. If so, it has to allocate more memory space as the ExtendableArray is extendable.

4 Test Driver

The file driver1.cpp contains the function main(). The purpose of this file is to make various calls to the functions in your classes to test them as thoroughly as possible. How you choose to use this file is entirely up to you. Keep in mind that any functions included in this file are for your own basic testing purposes only. You should continually make changes to this file so that it tests your code while you are making chnages to your class.

```
#include "ExtendableArray.h"
#include <iostream>
using namespace std;

void stuff_20(ExtendableArray arr) {
    for (int i=0; i < 20; i++) {
        arr[ i ] = i;
    }
    cout << arr << endl; // 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
}

int main() {
    ExtendableArray a1;
    for (int i=0;i<20;i++)
        a1[i] = i;
    cout << a1 << endl; //0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

    if(1){
        ExtendableArray a2;

        for (int i=0;i<10;i++)
            a2[i+5] = a1[i];

        cout << a2 << endl; //0 0 0 0 0 0 1 2 3 4 5 6 7 8 9

        a1 = a2;
        for (int i=0;i<10;i++)
            a2[i] = i;

        cout << a1 << endl; //0 0 0 0 0 0 1 2 3 4 5 6 7 8 9
        cout<< a2 <<endl; //012345678956789
    }

    cout << a1 << endl; //0 0 0 0 0 0 1 2 3 4 5 6 7 8 9
```

```
    ExtendableArray a3;
    a3[0] = 1;
    cout << a3 << endl; // 1 0
    stuff_20(a3);
    cout << a3 << endl; // 1 0
    cout << a3[2147483647] << endl; // 0

    return 0;
}
```

5 How to Submit

The only file you submit is ExtendableArray.cpp. You must not submit the header nor the test driver code. To submit ExtendableArray.cpp file, you must execute the following command.

```
$ cd /home/your_working_directory
$ pl_submit proj1 ./ExtendableArray.cpp
```

Note that you can submit multiple times. But only the last submission will be graded. Using the following command, you can check whether your file has been correctly submitted.

```
$ pl_check_submission proj1
```

Note that these commands only work in swin, swui, swye, swji machines. If you implemented your codes in your desktop, you must upload the file to these machines before you submit. Otherwise, pl_submit command will fail to read your implementation.

For any questions, please post them in Piazza so that we can share your questions and answers with other students and TAs. Please feel free to raise any issues and post any questions. Also, if you can answer other students' questions, you are welcome to do so. You will get some credits for posting questions and answering other students' questions.