

CS 251: Intermediate Software Design

Programming Assignment 1

The first assignment is intended to reacquaint you with C++ and your programming environment. If you are familiar with C++ this assignment will be straightforward. If this assignment isn't straightforward, you may not have enough C++ background to take this class.

Array

A array is an Abstract Data Type (ADT) with operations such as `get`, `set`, and `size`. The first homework assignment focuses upon building and using an array. Your task is to implement an array of characters in C++. This array will be different from C++ built-in arrays in the following ways:

- *The initial size of the array can be a run-time parameter* – to implement this you'll need to use the C++ `new` and `delete` operators.
- *Operations on the array will be range-checked* – thus, if you try to get or set an array element that is out of range the operation will return a “failure” status.

Your task is to write the following C++ methods that operate upon objects of class `Array`. Here's the class declaration:

```
// File Array.h

// This will be an array of chars.
typedef char T;

class Array
{
public:
    // = Initialization and termination methods.

    // Define a "trait"
    typedef T element_type;

    // Dynamically create an uninitialized array. Throws
    // <std::bad_alloc> if allocation fails.
    Array (size_t size);

    // Dynamically initialize an array. Throws
    // <std::bad_alloc> if allocation fails.
    Array (size_t size, const T &default_value);

    // The copy constructor (performs initialization). Throws
    // <std::bad_alloc> if allocation fails.
    Array (const Array &s);

    // Assignment operator performs an assignment by making a copy of
    // the contents of parameter <s>, i.e., *this == s will return true.
    // Throws <std::bad_alloc> if allocation fails.
    void operator= (const Array &s);

    // Clean up the array (e.g., delete dynamically allocated memory).
    ~Array (void);

    // = Set/get methods.

    // Set an item in the array at location index. Returns -1 if
```

```

// index is larger than the size() of the array, else 0.
int set (const T &new_item, size_t index);

// Get an item in the array at location index. Returns -1 if
// index is larger than the size() of the array, else 0.
int get (T &item, size_t index) const;

// Returns a reference to the <index> element in the <Array> without
// checking for range errors.
const T &operator[] (size_t index) const;

// Set an item in the array at location index without
// checking for range errors.
T &operator[] (size_t index);

// Returns the current size of the array.
size_t size (void) const;

// Compare this array with <s> for equality. Returns true if the
// size()'s of the two arrays are equal and all the elements from 0
// .. size() are equal, else false.
bool operator== (const Array<T> &s) const;

// Compare this array with <s> for inequality such that <*this> !=
// <s> is always the complement of the boolean return value of
// <*this> == <s>.
bool operator!= (const Array<T> &s) const;

private:
// Returns true if <index> is within range, i.e., 0 <= <index> <
// <cur_size_>, else returns false.
bool in_range (size_t index) const;

// Maximum size of the array, i.e., the total number of <T> elements
// in <array_>.
size_t max_size_;

// Current size of the array. This starts out being == to
// <max_size_>. However, if we are assigned a smaller array, then
// <cur_size_> will become less than <max_size_>. The purpose of
// keeping track of both sizes is to avoid reallocating memory if we
// don't have to.
size_t cur_size_;

// Pointer to the array's storage buffer.
T *array_;
};

```

Note that `get()` and `set()` explicitly check whether the `index` is within the bounds of the array, whereas the `operator[]` methods do not.

You can get the “shells” for the program from www.cs.wustl.edu/~schmidt/cs251/assignment1. The `Makefile`, `main.cpp`, and `Array.h` files are written for you. All you need to do is edit the `Array.cpp` and `Array.i` files to add the methods that implement the Array ADT.

If you are an undergraduate student please use the shells that are in the `ugrad` directory at the URL above. If you are a graduate student please use the shells that are in the `grad` directory at the URL above. Graduates taking the class need to use exception handling to propagate range errors back from the `get()` and `set()` methods. Undergraduates taking the class don't need to use exception handling, though you can if you'd like.