Overview  For this project, we will implement a templated class called Vector that is essentially a variable length array. The Vector can grow as necessary as new elements are added to the array. Further we will implement an iterator called VectorIterator that exists to “refer to”, or “point to” an existing element in the Vector. The required API’s are specified in the provided Vector.h file.

Copying the Project Skeletons

1. Log into deepthought19.cc.gatech.edu using ssh and your prism log-in name.
2. Copy the files from the ECE8893 user account using the following command:

/usr/bin/rsync -avu /nethome/ECE6122/Vector .

Be sure to notice the period at the end of the above command.
3. Change your working directory to Vector

   cd Vector

4. Copy the provided Vector-skeleton.cc to Vector.cc as follows:

   cp Vector-skeleton.cc Vector.cc

5. If you are an undergraduate, comment out the definition of GRAD_STUDENT in Vector.h.
6. Then edit Vector.cc to implement your Vector and VectorIterator
7. Compile your code using make as follows:

   make

8. This builds a program called testVec that has eight different test cases to test your implementation.
9. Once you have gotten the program compiled and ready to test, you can begin running the tests. Running testVec with no arguments runs all tests; running with a single argument is assumed to be the test number you want to run. Grad students must run all eight tests, undergrads only need tests 1-6. You should consider debugging by running one test at a time.

Resources

1. Vector-skeleton.cc is a starting point for your program.
2. Vector.h describes the classes and required API.
3. testVec.cc is the test program to test your implementation. Do not change this program.
4. String.h and String.cc are used by the testing program. You should not need to edit either of these.
5. Makefile is the makefile for the testVec program.
6. expected-alltests.txt is the expected output from running all tests.
Checking for Memory Leaks  An important requirement for this assignment is to properly manage memory and to have NO MEMORY LEAKS. On the deepthought system, you can run the valgrind program, which give detailed memory usage statistics and leak statistics. Run valgrind as follows:

valgrind --tool=memcheck ./testVec (your test number here)

At the end of the execution, valgrind prints a summary of the memory leaks. You are hoping for the message All heap blocks were freed -- no leaks are possible. If you have memory leaks, you will only get half credit for the particular test (see the section below on grading).

In–Place new and delete  Recall that we said in class that the new operator, by default, does two separate and independent things. First is to find some memory to store the new object, and second is to call the constructor for the object. In this assignment, you will often want to call a constructor, but to use existing memory. This is called an in–place new. The syntax is:

    new (address) constructor

The address is the address of an existing memory region where the new object will be stored. The constructor is the constructor (default, copy, etc) to use.

Similarly, the delete operator does two independent things. First is to call the destructor for the object, and the second is to free the memory associated with the object. Again, in this assignment you will frequently want to call the destructor for an object, but do not want the memory freed. Given an object of type T, the syntax for in–place delete is:

    object.~T(); or objPointer->~T();

Here, object is a variable of type T and objPointer is a pointer to an object of type T.

Grading  For undergrads, there are six individual tests, worth 20 points each. For graduates, there are eight test worth 15 points each. In both cases, the total possible is 120 points (this is a very difficult assignment!). If you get the right answer but have memory leaks, you will get half credit for the particular test.

Using the GNU Debugger  The GNU debugger gdb is invaluable in debugging this assignment. We will discuss gdb and other issues for this assignment in the next class lecture, Wed Oct 19.

Turning in your Project.  The system administrator for the deepthought cluster has created a script that you are to use to turn in your project. The script is called turnin-ece6122, and is found in /usr/local/bin, which should be in the search path for everyone. From your home directory (not the Vector subdirectory), enter:

    turnin-ece6122 Vector.

This automatically copies everything in your Vector directory to a place that I can access (and grade) it.