

Next Week's recitations: TBA

Only 20 pts. on this assignment, for typing up a short technical C++ program on `timberlake` and also trying it out on your home IDE. This is mainly to encourage practice with a UNIX text-editor of choice, so you have the option of using it during “hands-on” parts of recitations in upcoming weeks. Please submit answers to the questions at the end *using the separate answer sheet*—i.e. in hardcopy. (There is an item to submit your file online also, to test the online submission system, but we are not asking for the question answers to be submitted online.) The answer sheet includes affirming that you've either completed the UNIX set-up/tutorial from Week 1 recitations or office-hours, or you have specific UNIX command-line experience by other means.

Reading For This Month

As I said in class, the text assumes only one previous main-sequence CS course, and includes a C++ reference good enough that previous instructors and I do not feel a separate C++ book is needed. The effect of this, however, is that much of the first half is conceptual review (for those who have had 2 previous main-sequence courses like 115–116 at UB), and much is C++ language reference that we don't need to learn all-at-once now. The following attempts to chart a reasonable path through the maze.

For next week's recitations, please read these sections of the text's “C++ Primer”: Read P.1, and read P.2 but **ignore** “macros” apart from `NULL`. Skim P.3 and P.4—there are some differences esp. in Table P.5 on p20, but we'll take them as they come. Read P.5 intently—this parallels my Week 1 lectures. Read P.6 as a “forward reference”—some things I said this week were preparatory for call-by-(constant)-reference, but full details are on tap for my lectures later this month. Read P.7, *but then forget you did*—we will treat “raw” C strings and arrays as anathema. Instead read P.8 as replacing the former. Skim Table P.6 especially all the “find” methods—the operations in my `HelloString.cpp` and similar files are enough for now. Read Section P.9, which along with P.1 and P.2 will be the other main focus of recitations—but not getting into the gory details in the tables, which are really for reference later.

For next week's lectures: Read Chapter 1, sections 1.1–1.2. You may treat this as a review of ideas from CSE 115–116 or your equivalent “CS I” and/or/if “CS II” courses elsewhere, but note some little C++ details. Then read the long section 1.3 intently—it will be a main focus of my upcoming slides— and also sections 3.1 and 3.2 in Chapter 3. The short section 1.4 is 115–116 conceptual review again. Only skim the “Case Study” in sections 1.5–1.7.

For Week 3 and beyond: Skim Chapter 2, sections 2.1–2.4 as FYI—again there is far more detail than I want to cover ahead-of-time, like in Table 2.2 on p142. The main fact to know about C++ exceptions is they're not classes like in Java—and my exercises will not highlight them. But I will highlight sections 2.5 and 2.6 in Week 3 lectures, and as done by previous instructors, I will give out some supplementary hardcopy notes on “Little-Oh” and “Theta” in addition to “Big Oh” notation next week—for now they are defined in full in my slides on the course webpage.

Focus on Chapter 3: sections 3.1 and 3.2 for now, the rest for later. Ordinarily I would say to skip sections 3.3–3.7 for now, but you have a long weekend and then a week with no hardcopy homework, so let me urge you to get the material “in your head” sooner. Chapters 4–6, however, I've decided it's fine to postpone—except just read these pages of section 4.1 for now: 231–235, and the boxes on p237 and p241. Please, however, read ALL of Chapter 7, now or at least by Week 3. It reviews some routines such as binary search which you may have seen in earlier courses, but I wish to make them examples of the messages about code-correctness and “PRE/POST/INV” (and

“requires/ensures”) which I am building on the text’s chapter 2. jumping-around; from chapter 8 onward coverage will be sequential.

Assignment Part

While logged in to `timberlake`, please type manually the following short C++ program—which is a subset of `HelloString.cpp` to be demo’ed next week. Type it in whatever UNIX text editor (e.g. `vi/vim` or `nano` or `pico`) you prefer to try to learn. Name it according to the top-of-file comment, *which you should change appropriately*. Compile and run it on `timberlake`. Submit it online by entering at the UNIX prompt,

```
submit_cse250 <file>
```

Then, if you intend to use a home system such as the Eclipse IDE (which can also be used on Macs) or a commercial-or-freeware C++ compiler, please set up your environment and transfer this program to it (can use FTP via e.g. FileZilla, or just mouse-copy). Show and compare the results between `timberlake` and your system. (Or, if you intend to use only `g++` on `timberlake`, please affirm so,)

```
/** File "memMapNNN.cpp", by KWR for CSE250, Fall 2010. Substitute NNN by
    your initials to name your file. *To be typed up on "timberlake" first.*
    May illustrate differences in memory-map between timberlake and your IDE.
 */

#include<iostream>
using namespace std;

int main() {
    //Pointers to Named (Stack) Objects
    string xsphost = "Jello"; //xsphost and ysphost are concrete names
    string ysphost = "Bello"; //in the "symbol table" for the program.
    string* xsp = &xsphost;
    string* ysp = &ysphost;
    ysp = xsp; //copies the address value, "5004" in slides
    xsp->at(0) = 'H';
    cout << "\nAs stack pointers: xsp = " << xsp << " and ysp = " << ysp << endl;
    cout << "Oops!---forgot to de-reference the pointers..." << endl;
    cout << "As stack pointers: *xsp = " << *xsp << " and *ysp = " << *ysp
        << endl;

    //Pointers to Anonymous (Heap) Objects
    string* xp = new string("Jello");
    string* yp = new string("Bello");
    yp = xp;
    xp->at(0) = 'H';
    cout << "\nAs heap pointers: xp = " << xp << " and yp = " << yp << endl;
    cout << "Oops!---forgot to de-reference the pointers..." << endl;
    cout << "As heap pointers: *xp = " << *xp << " and *yp = " << *yp << endl;

    return (0);
}
```

Assignment 1 Questions—due in lecture Fri. 9/10/10.

Name and Student ID Number:

1. Did you access `timberlake` directly on-campus, from home via Putty, or from home via some other means?

2. Which text editor did you use on `timberlake`?

3. What values for `xsp` and the ordinary (heap) pointer `xp` were printed? (If `yyp` is different from `xsp`, ditto `yp` and `xp`, scream...) If they print in hexadecimal, you may copy that—i.e. you need not convert to base 10.

4. What values are printed on your home system? (If you have not set up a home system, please come in for setup help, or say you'll just use `timberlake`.)

Please circle *and sign your name to affirm* one of the following:

(a) I have carried out the UNIX tutorial parts 1–2 and have set up a working course directory on `timberlake` as provided in Week 1 recitations/office-hours; or

(b) I have the following specific experience with command-line UNIX (or Linux or equivalent):

-----..