CSCI251 Advanced Programming

Spring 2024

Lab 5 for Week 10 & 11

General Note

In these series of laboratory exercises you will be learning and practicing various programming concepts enabled by C++ programming language. Mostly, these concepts will be introduced through simple and sometimes complex examples. You are encouraged to think about how and why things work the way they do. The exercises are not set up to trip you, but you need to think carefully.

In order to focus on the task at hand, you can use an integrated development environment (IDE) to write, compile and test your code. In one of the notes provided on Moodle, you can find the steps required to install Code::Blocks on several OS (e.g. Windows, Linux, MacOS). You are encouraged to follow the instructions and install it for your operating system. The operating system on which you develop your code should not matter. All that needs to happen is that your code is C++17-compliant. The system on which your code will be tested has g++ 7.5 installed and your code must work on that system (i.e. **capa.its.uow.edu.au**).

In order to test that your code is compliant and will work on **capa.its.uow.edu.au**, you will need to do a secure login to capa and test your code. The appropriate option required to ensure that you are compiling against C++17 is the -std=c++17 added to your command line instruction. For example:

g++ file1.cpp file2.cpp file3.cpp -std=c++17 -o output_exec

Get familiar with your IDE and ensure that the build option is set to use c++17.

Use sensible names for your variables and insert comments to let the reader know what your code is doing. If your code contains no comments you will lose marks.

Code Submission

For each exercise, name your code as instructed. Each Lab will be due for submission at midnight of Weeks 3, 5, 7, 9 and 11. You will also be reminded accordingly as the session progresses. Submission points will be set up on Moodle. For each task you might save the code file(s) like:

t1.cpp: only one file.

t1-1.cpp: if multiple subtasks

t1.zip: if there are multiple files

At the end, please submit a (big) zip _le that includes all task for the lab, and name it like: “familyname_studentId_lab#.zip”. The symbol # represents the lab number. For example, in Lab 1 (week 2&3), the submission can be named as “Jordan_12345_lab1.zip”.

1 Task One: Function Template (20 points)

1. (10 points) Debug file *Debug-A.cpp.*
2. (10 points) In *findMax.cpp*, we have a function template *findMax* that uses passing by values. Change the code to have pass by reference to this function template. In *main()* function, declare the appropriate data type to call the function template passing by reference.

2 Task Two: More Templates (20 points)

Write code *Symbolic.cpp* that contains a function template to display a value preceded and succeeded by n elements of a symbol x on a line, with a space on each side of the value. The arguments to the function template should be the value, and n, and x. Write a *main()* function that tests the function with char, int, double and string arguments. The output could be, for example,

*** 47 ***
000 39.25 000
aaaa Bob aaaa

3 Task Three: Debug (20 points)

1. (10 points) *Debug-B.cpp* only has one problem, a missing constructor for CSL. Add an appropriate one!

2. (10 points) Consider the provide file *fibT.cpp*. What is it trying to do? When will it work on a type? Demonstrate it "working" for at least one type other than unsigned int or int.

3. (0 point) Container, iterator, function. Look at the source in *One.cpp* and in *Two.cpp*. This is partially to see how some generic functionality fits together.
(a) How do the two files differ?
(b) Compile and run each of the programs.
(c) What do the programs do?

4 Task Four: Some template exercises (20 points)

1. (0 point) Write *doubled.cpp* to include a function template *doubled* that can take a type and return the addition of the element to itself, using +.
(a) Show it working for int, oat, char, and string. Show means you need to demonstrate how the value is in some way doubled.
(b) What functionality does a type need in order to work with your function doubled and with your demonstration?
(c) Show it working for one ADT that you write.

2. (20 points) This activity involves implementing a template class in a file *doubledClass.cpp*.
(a) Write a class template *hold* that holds a single data type of to-be-specified type T.

(b) Copy your function template *doubled* from the previous question into this class template.
(c) Demonstrate the use of doubled through an instance of the template class for a range of types.

5 Task Five: More generically...(20 points)

Consider the function template and definitions below.

```
template <typename T>
T funcExp(T list[], int size){
int j;
T x = list[0];
T y = list[size-1];
for(j=1; j<(size-1)/2;j++){
if (x < list[j]) x = list[j];
if (y > list[size-1-j]) y = list[size-1-j];}
return (x+y); }

int list[8]={1,2,9,3,5,8,13,10};
string strlist[]={"one","fish","two","fish","red","fish","blue","fish"};

cout << funcExp(list,8) << " :: " << funcExp(strlist,8) << endl;
```

What is the functionality of funcExp (i.e., what it wants to achieve) and what is the meaning of the output? (Upload you answers in a text file.)