CSCI 262 Lecture 18 – Templates

Outline

- Generic programming – allows us to write functions or classes in terms of unspecified types (type parameters)
  - Allows the creation of data structures such as container types which can be instantiated to contain specific types, without having to explicitly rewrite the data structure for every type we wish to contain,
  - Implemented using templates in C++
- Templates define a pattern for code which will be generated by the compiler at compile time as needed
  - Templates are not compiled, but their instantiations for particular type parameters are compiled
  - All template code belongs in header files, therefore, not in source (.cpp) files

Readings

Read chapter 16 for next week (section 16.5 is optional).

Before templates, particularly in the old C language, if you wanted to do something like generic programming, you wrote functions (or made data structures) that worked with void pointers (i.e., variables of type void *). This type is compatible with any pointer type in C – you can assign the address of any object to a void pointer. The trick, though, is that somewhere along the line you need to actually turn the value being pointed to back into its real type so you could work with it. This is done by casting. (I’m not going to cover that topic, because it is endlessly complex in C++, and generally we can avoid it thanks to generic programming!)

You can see an example of this applied to binary search and sorting algorithms if you look at the bsearch() and qsort() functions in the <cstdlib> header.

Self Check

Examine the following code and explain the bug in it – where does it go wrong, and why?

```cpp
template<class T>
void print_it(T val) {
    cout << to_string(val) << endl;
}

int main() {
    int x = 42;
    vector<string> list = {"apple", "orange", "pear");
    print_it(x);
    print_it(list);
}
```

For Further Practice

If you’d like to experience the “good old days” before generic programming, see if you can create an array of some arbitrary type and sort it properly using the qsort().