C++ Programming

Exercise sheet 6
Software Engineering Group EIM-I
Philipp Schubert
June 05, 2020

Solutions to this sheet are due on 12.06.2020 til 16:00. Please hand in a digital version of your answers via e-mail. The e-mail’s subject has to contain cppp20. Do zip-compress your solutions.

Note: If you copy text elements/code elements from other sources, clearly mark those elements and state the source. Copying solutions from other students is prohibited. All of your files that belong to your solution have to be contained in a single .zip file that is named according to the following naming scheme: <name>_<surname>_solution_<X>.zip. Replace <name> and <surname> with your actual name and replace <X> with the number of the exercise sheet. You can look up your results using this link [https://docs.google.com/spreadsheets/d/1LtRFGuJ2kkXqU1G4UjXR73a0nW3Ldp10hxa9s638EdC3o/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1LtRFGuJ2kkXqU1G4UjXR73a0nW3Ldp10hxa9s638EdC3o/edit?usp=sharing)

This exercise sheet will help to familiarize yourself with C/C++’s preprocessor (CPP). You will also learn about programming using C++’s templates. Using templates you can write more abstract and generic code that can be used to solve a whole bunch of tasks rather than just one specific task. You can achieve 16 points in total.

Exercise 1.
In this exercise, you have to use C/C++’s preprocessor.

a) Define a preprocessor macro `MY_ASSERT(BOOL_EXPR, MESSAGE)` that checks if `BOOL_EXPR` evaluates to true or false. If `BOOL_EXPR` evaluates to false print the message `MESSAGE` as well as the file and the line number the failure has been detected to the command line and call `abort()`, which is declared in the `<cstdlib>` header. The call to `abort()` will exit the program abnormally.
(2 P.)

b) Define another preprocessor macro `POWER(RESULT, BASE, EXPONENT)` that computes `BASE`\(^{EXPONENT}\) (the mathematical power function) and stores the result in `RESULT`.
(2 P.)

c) Explain why it is not a bright idea to define a macro like `#define FAC(N) (N > 1) ? N * FAC(N-1) : 1`—using recursion—in order to compute the factorial function? Does this even work? If it goes wrong, explain why.
(2 P.)
Exercise 2.
Defining a data type for triples.

a) Define a `struct` by the name of `triple` that is capable of storing three variables of arbitrary types!
That is, each of the variables may hold a value of an arbitrary type, say A, B, and C.
(2 P.)

b) Provide the following two members for `triple`:

- `triple(A a, B b, C c);` // a simple constructor that initializes the data members
- `friend ostream& operator<< (ostream& os, const triple& t);` // an operator to print the data
members to the command line in a convenient manner

(2 P.)

Exercise 3.
Consider the bubble sort algorithm from exercise 04.2.a. A possible implementation that you can use for
this exercise is shown here:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>  // contains the for_each algorithm and swap
#include <functional>  // needed for part b)
using namespace std;

void bubble_sort(vector<int>& v) {
    bool has_swapped;
    size_t n = v.size();
    do {
        has_swapped = false;
        for (size_t i = 0; i < n - 1; ++i) {
            if (v[i] > v[i+1]) {
                swap(v[i], v[i+1]);
                has_swapped = true;
            }
        }
        // After each iteration the biggest element has swapped to the end.
        // Therefore, we can shorten our loop after each iteration.
        --n;
        // If no swap has taken place, we are done.
    } while (has_swapped);
}

int main() {
    vector<int> v = {10, 9, 8, 7, 6, 5, 4, 1, 3, 2};
    for_each(v.begin(), v.end(), [](int i) { cout << i << " "; }); cout << '\n';
    // Do the sorting!
    bubble_sort(v);
    for_each(v.begin(), v.end(), [](int i) { cout << i << " "; }); cout << '\n';
    return 0;
}
```

The `bubble_sort()` implementation in the above is able to sort vectors of integers. Sorting, however, is a
more general task. Given a certain predicate, one can basically sort everything (that can be ordered).
In this task, you will craft a more abstract bubble sort implementation that is able to sort a vector of
"anything".
a) First, change the `bubble_sort()` implementation such that it can operate on anything that implements the `operator<`. In order to do so, make `bubble_sort()` a function template such that it can sort a vector of an arbitrary type `T`! Test the function by instantiating a template function that sorts `double` values and check if your function still works correctly. (3 P.)

b) In order to sort a vector of an arbitrary type `T` that does not implement `operator<`, adjust the signature of function `bubble_sort()` to receive a second parameter of type function that serves as a predicate. The signature looks like: `void bubble_sort(vector<T> &v, function<bool(T, T)> predicate);`

Then, rather than performing a check for `<` in the `if` condition, apply the predicate function to compare two values. An example call of the adjusted `bubble_sort()` is shown in the following: (3 P.)

```cpp
// ... your adjusted implementation

bool cmp_string_size(const string& a, const string& b) { return a.size() < b.size(); }

int main() {
    vector<string> v = {"A", "BBB", "CC", "DDDDD", "EE", "FFFFFF", "G", "HHHH"};
    // sort v according to the predicate cmp_string_size
    bubble_sort<string>(v, cmp_string_size);
    // or you can just pass a cool lambda function and sort in reverse order
    bubble_sort<string>(v, [](const string& a, const string& b) { return a.size() > b.size(); });
    return 0;
}
```

Exercise 4.
This is an optional exercise:

```cpp
template<typename T, typename... Args>
T add(T t, Args... args) {
    // TODO
}
```

Implement the above function template such that it adds arbitrary many values of an arbitrary type. The keyword here is variadic template arguments (use google or a book). Hint: use recursion, you will need a helper function template that represents the trivial case. Having implemented the above and the helper function template, you can use the `add()` function as follows:

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    int isum = add(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    cout << isum << \n';
    string s1 = "Hello";
    string s2 = ", ";
    string s3 = "World";
    string s4 = ",!";
    string ssum = add(s1, s2, s3, s4);
    cout << ssum << \n';
    return 0;
}
```

(0 P.)