C++ Programming

Exercise Sheet 06
Software Engineering, EIM-I
Philipp Schubert

November 24, 2017

Solutions to this sheet are due on 01.12.2016 til 14:00. Please hand in a digital version of your answers via e-mail. The e-mails subject has to contain cpp. Do zip-compress your solutions. For questions please send mail or speak to me during the exercises.

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>_solution<XX>.zip. Replace <name> and <surname> with your actual name and replace <XX> with the sheet number the solutions belong to. You can look up your results using this link, https://docs.google.com/spreadsheets/d/1V8rKtimsQS6thKGkTh6CCh1v-LwulBIA3RvKA1ZSH2M/edit?usp=sharing

During this exercise sheet you will learn about the C/C++ preprocessor (CPP) and how it can be used in a rational way. You will also learn about programming using templates. Using templates one can write more abstract and generic code that can be used to solve a whole bunch of tasks rather than just one specific one. You can achieve 16 points in total.

Exercise 1.
In this exercise you have to use the C/C++ preprocessor.

a) Define a preprocessor macro MY_ASSERT(COND, MSG) that checks if the condition COND is true or false. If COND is evaluated to false print the message MSG as well as the file and the line number this condition failed in to the command line and call exit(-1), which is defined in the <cstdlib> header, in order to 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) Why is it 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, why is that?
(2 P.)
Exercise 2.
Defining a triple data type.

a) Define a struct by the name of triple that is capable of storing three variables of an arbitrary type. Each of the variables can have a different type, say A, B and C!
(2 P.)

b) Provide the two following member functions 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 out the contents 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
#include <functional>  // needed for part b)
using namespace std;

void bubble_sort(vector<int>& v) {
    bool has_swapped;
    size_t n = v.size();
    int tmp;
    do {
        has_swapped = false;
        for (size_t i = 0; i < n - 1; ++i) {
            if (v[i] > v[i+1]) {
                tmp = v[i];
                v[i] = v[i+1];
                v[i+1] = tmp;
                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()` function in the above code is capable of sorting vectors of integers. But sorting is a more general task. Given a certain predicate, one can basically sort everything (that can be ordered). In this task you will implement a more abstract bubble sort that is able to sort a vector of "everything".

a) The first change we would like to make to our `bubble_sort()` function is that is can sort everything that implements an `operator<`. In order to do so, make `bubble_sort()` a function template such that is 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 need to 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` statement, apply the predicate function to the values to be compared. An example call of the adjusted `bubble_sort()` is shown in the following:

(3 P.)

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

bool my_predicate(int a, int b) { return a > b; }

int main() {
    // ... declare a vector of int 'v' for example and sort it according to the predicate 'my_predicate'
    bubble_sort<int>(v, my_predicate);
    // or you can just pass a cool lambda function and sort in reverse order
    bubble_sort<int>(v, [](int a, int b) { return a < b; });
    return 0;
}
```

Exercise 4.

This is an optional exercise:

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

Implement the above function template to add arbitrary many values of the same arbitrary type. The keyword here is variadic template arguments (use google or a book). You will need a helper function template that represents the trivial case. (Hint: use recursion!) Having implemented the above and the helper function template, you can use the `add()` function like 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 ", s2 = "World", s3 = "!";
    string ssum = add(s1, s2, s3);
    cout << ssum << "\n";
    return 0;
}
```

(0 P.)