LINKÖPING UNIVERSITY
Department of Computer and Information Science
Software and Systems
Christoffer Holm 2019-06-03

Computer examination in
TDDD38 Advanced Programming in C++

Date 2019-06-03

Time 14-19

Department IDA

Course code TDDD38

Exam code DAT1

Examiner

Klas Arvidsson (klas.arvidsson@liu.se)

Administrator

Anna Grabska Eklund, 28 2362

Teacher on call

Christoffer Holm (christoffer.holm@liu.se)
Will primarily answer exam questions using the student client.
Will only visit the exam rooms for system-related problems.

Allowed Aids (tillåtna hjälpmedel)

An English-* dictionary may be brought to the exam.
No other printed or electronic material are allowed.
The cppreference.com reference is available in the exam system, except for the language section.

Grading

The exam has a total of 25 points.
0-10 for grade U/FX
11-14 for grade 3/C
15-18 for grade 4/B
19-25 for grade 5/A

Special instructions

- All communication with staff during the exam can be done in both English and Swedish.
- Don't log out at any time during the exam, only when you have finished.
- Given files are found in subdirectory `given_files` (write protected). The exam will be available as a pdf in this directory at the start of the exam.
- Files for examination must be submitted via the Student Client.
- When using standard library components, such as algorithms and containers, try to chose ”best fit“ regarding the problem to solve. Avoid unrelated/unnecessary computations and unnecessary data structures.
- C style coding may cause point reduction where C++ alternatives are available.
- Your code should compile. Commented out regions of non-compiling code may still give some points. Resource leaks and undefined behavior is important to fix.

Available commands

`e++17` is used to compile with “all” warnings as *errors*.
`w++17` is used to compile with “all” warnings. **Recommended.**
`g++17` is used to compile *without* warnings.
`valgrind --tool=memcheck` is used to find memory leaks.

C++ reference

During the exam you will have *partial* access to `http://www.cppreference.com/` with the chromium browser. You can start the broweser by either running `chromium-browser` in the terminal or choose an appropriate option in the start menu. Do note that everything except cppreference will be unavailable. If you are unable to access a page that should be available (it might have been blocked by mistake) then you can send a message through the exam client. Since it is an experimental feature there might be some quirks.

Theory questions

1. Answers may be given in either Swedish or English. Write all your answers in one text file and submit as assignment 1.

   (a) Write a function template that is equivalent to this function: [1p]

   ```
   auto& fun(auto& a, auto b)
   {
     return a += b;
   }
   ```

   (b) Which are the six special member functions? [1p]

   (c) What does it mean to mark a constructor as `explicit`? [1p]

   (d) What is the difference between the `sizeof`-operator and the `sizeof...`-operator? [1p]

   (e) Give an example of when it is reasonable to call a destructor explicitly. [1p]

Practical questions

2. When dealing with networks, binary files or similar situations where we require objects to be sent as binary data we usually make sure that we only send objects that are *Serializable*. [5p]

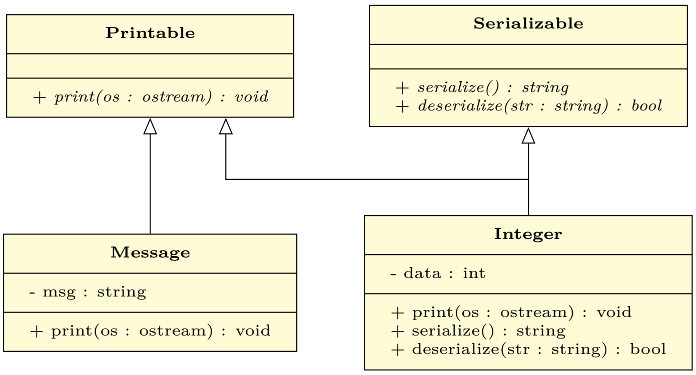
Serializable means that an object can be converted to and from strings (or a sequence of bytes). Usually when we want to send an object over a network, we first serialize it and then send the resulting string over the network. The reciever must then deserialize the string to get a copy of the object.

In this assignment we have two types of objects; those that are `Printable` and those that are `Serializable`.

Printable objects are those that can be written to an arbitrary `ostream`.

Serializable objects are those that can be serialized (converted to a string) and *deserialized* (constructed from a string).

You are to implement the following UML diagram:


Printable
+ print(os : ostream) : void
Serializable
+ serialize() : string
+ deserialize(str : string) : bool
Message
- msg : string
+ print(os : ostream) : void
Integer
- data : int
+ print(os : ostream) : void
+ serialize() : string
+ deserialize(str : string) : bool

Here we can see that both `Message` and `Integer` are both `Printable`, however; `Integer` is also `Serializable`. This means that `Integer` has *two* base classes (multiple inheritance).

Message Must have a constructor that initializes a data member of type `string` called `msg`. It must also override the `print` function.

`print` takes an `ostream` and prints `msg` to it (followed by a newline).

Integer Must have a constructor that initializes a data member of type `int` called `data`. It must override `print`, `serialize` and `deserialize`.

`print` should simply print `data` followed by a newline to the supplied `ostream`

`serialize` should convert `data` to a `string` and return the result.

`deserialize` should take a `string` and convert it to `int`, and then assign the result to `data`.

There are testcases in `given_files/program2.cc`. There are also two incomplete functions, `serialize` and `print` that you are supposed to complete (more details in the file).

3. The way the ternary conditional operator works is as follows: [5p]

Given `(condition ? a : b)` it works such that if `condition` is `true` then `a` will be returned from the expression, otherwise `b` will be returned.

Note that `a` and `b` doesn't have to be variables, but can be full expressions.

As you might see, this can easily be replaced with an if-statement or a function (`std::max`) and *should* be replaced with those things in all but a select few cases. However; one peculiar property of the ternary conditional operator arises whenever `a` and `b` are different types.

Since this counts as an expression it must return a singular type, but which one? The type of `a` or the type of `b`? The compiler will choose the type that can most accurately represent both types.

Example: The type of `(true ? 1.5 : 2)` will be `double` since *both* `1.5` and `2` can be represented accurately as `double`. `int` is not a valid alternative since `1.5` cannot be represented accurately as `int`.

With this knowledge in mind, you are to implement a class-template called `common_type` which takes two template-parameters `T1` and `T2`. There should be a type alias called `type` defined inside `common_type` which is an alias for the type which can most accurately represent both `T1` and `T2`.

You should also specialize `common_type` such that if any or all of `T1` and `T2` are `void`, then `common_type::type` is also `void`.

Note: `common_type::type` should *not* be a reference-type; `std::remove_reference` (defined in `<type_traits>`) might be useful to make sure that it is not a reference-type.

There are some testcases given in `given_files/program3.cc`. Using the already defined `std::common_type` for this assignment is not acceptable.

Partial credits will still be given if the `void` specializations of `common_type` are missing.

4. The binomial coefficient [5p]

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\cdot \ldots \cdot (n-k+1)}{1 \cdot 2 \cdot \ldots \cdot (n-k)}$$

is prevalent in many fields of computer science. It appears when working with combinatorics, polynomials, probabilities etc. However it is not obvious how to best calculate it since one have to multiply many numbers together and then divide two (probably) very large numbers to get the result. Most attempts will lead to integer overflow very quickly since we multiply a lot of integers.

There is a way to calculate this with minimized overflow risk by dividing each factor in the numerator with the factors in the denominator *before* we multiply everything together. There is a program that does this and some extra optimizations (for precision) in `given_files/program4.cc`.

In this assignment you are to refactor the given code so that it uses standard algorithms instead hand-written loops. You should use STL as much as possible. A full solution shouldn't require any standard iteration statements. Your solution should mimic the the steps taken by the program as closely as possible.

5. In functional programming one of the fundamental operations are filter which allows us to iterate over a container and extract those elements that fulfill some kind of predicate (condition). [5p]

In this assignment you are to implement a class-template filter that stores a reference to a container and a *UnaryPredicate* function (a function that takes one element from the container and returns false if this element should be filtered out and true if it should be kept). filter will then provide an iterator range with iterators filter_iterator through the member functions begin() and end(). Dereferencing filter_iterator will return the current element in the container which fullfills the given predicate function.

filter_iterator must store an iterator range (2 iterators) from the container. This iterator range will be from the current element to the end iterator of the container. The increment operator must search for the next element that fulfills the predicate in the stored range. Note that the constructor must search for the *first* element that fulfills the predicate (if no such element is found, it should be the end-iterator) and let that be the current iterator.

filter_iterator is the *end-iterator* if the two iterators stored are the same (end-iterator of the container).

filter is a class-template with two template parameters; the container type and the predicate type. The constructor must take a constant reference to the container and an instance of the predicate function-object. filter should also have two member functions begin() and end() which returns filter_iterator corresponding to the begin and end of the container.

filter_iterator must be an *InputIterator* (https://en.cppreference.com/w/cpp/named_req/InputIterator). The comparison members should only examine the iterators stored in filter_iterator, it is not necessary to compare the predicate functions. Note that operator-> **must** be implemented.

Example: The following program requires you to compile with C++17 (compile with w++17). This example should output 2 4 6 10.

```
vector<int> v { 1, 2, 3, 4, 5, 6, 9, 10 };
auto predicate = [](int n) { return n % 2 == 0; };
for (int n : filter(v, predicate))
{
  cout << n << " ";
}
```

A test program is given in given_files/program5.cc.