LINKÖPING UNIVERSITY
Department of Computer and Information Science
Software and Systems
Eric Elfving 2017-01-12

Computer examination in

TDDD38 Advanced Programming in C++

Date 2017-01-12

Time 08-13

Department IDA

Course code TDDD38

Exam code DAT1

Examiner

Klas Arvidsson (klas.arvidsson@liu.se)

Administrator

Anna Grabska Eklund, 28 2362

Teacher on call

Eric Elfving (eric.elfving@liu.se, 013-28 2419)
Will primarialy 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.

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, see separate instructions (`given_files/student_client.pdf`)!
- 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. Memory leaks and undefined behavior is important to fix.

Theory questions

Answers may be given in either Swedish or English. Submit your answers to all theory questions in one text file called `THEORY.TXT` and submit it as `Assignment #1`.

1. The standard library has a function `std::addressof` that is used in generic code to find the address of an object. Usually, the operator `&` is used for finding addresses of objects, why do you think they added a function as well? [1p]

2. A forwarding reference and an r-value reference are declared in the same way (`Type &&`), but are semantically very different. Why? [1p]

3. C++14 added generic lambdas. Give an example of a generic lambda and give a definition of a function object class that is equivalent to the compiler-generated closure object created by your lambda. [2p]

4. Why shouldn't you use `operator<` when comparing iterators in the code below (`container` is some sort of STL container): [1p]

```
for ( auto it = begin(container); it < end(container); ++it )
    // do something with it
```

Programming exercises

5. You are to create a class hierarchy to model a group of games. The types of games to model are board games and card-based games. The goal of this program is to help the user choose a game from a list of predefined games by asking some questions for each game and if the answers match the given game, a description of the game will be printed. [5p]

   See the file `given_files/assignment5.cc` for examples on the questions that should be asked as well as usage of the member functions below.

   Create an abstract base class `Game`. A Game object has to be initialized with a name (std::string) and the number of players needed (as two integers, minimum and maximum). `Game` has the following member functions:

   name returns the name of the game.

   choose a filtering function that asks the user questions depending on the current game and returns true if the game matches the answers, false otherwise.

   print prints information about the current game to a given stream.

   Create a concrete subclass of `Game`, `Board_Game`. This class also needs the board size (expressed in cm as two integers), as well as the data needed by `Game`. `Board_Game` should have a member function `size` that returns a string representation of the board size.

   Some board games are cooperative. In these games you play in teams against mechanisms in the game. Sometimes there is a traitor (a player who works against the team). Create a subclass to `Board_Game` called `Cooperative`. The extra data needed by this class is whether there is a possibility of having a traitor or not. By default, a cooperative game should not have a traitor.

   Finally create a concrete class `Card_Game` as a subclass of `Game`. Many card games uses a regular deck of 52 cards, but many have their own deck instead. Whether a standard deck or not must be provided when creating an object of type `Card_Game`.

   None of the classes should have any copy or move operations.

   Encapulation is important and unnecessary code duplication will incur point deductions.

6. The programming language python has a function called `range` that generates a range of values. A range is a set of values from a starting point up to (but not including) en end value, possbly with a step size. This could be done quite easily just by using a simple loop or the `std::iota` algorithm, but these solutions require us to store all values in the range. In this exercise, you will create a variant of the python function range by creating an input iterator that calculates and stores the current value when you increment it and let you access the value with the standard dereference operator. [5p]

   See the file `given_files/assignment6.cc` for examples on how the parts below should work.

   Create a class `Range_Iterator` that fulfils the `InputIterator` concept templated on the type generated. To do this, your class needs the following:

   - A constructor taking two arguments, a starting value and a step size (defaulted to 1) of the template type.
   - The following type declarations:

     `iterator_category` Type of iterator (`std::input_iterator_tag`)

     `value_type` The type returned by `operator*` (the template type).

     `difference_type` What you get when subtracting two iterators. Since we're not supporting subtraction, just use **`long`**.

     `reference` Reference to the type iterated over.

     `pointer` Pointer to the type iterated over.

     Please note that it should not be possible to change the current value with any operations other than incrementing. This should be represented by your selected types.
   - `operator++` (prefix and postfix) to increment the current value with a given step size.
   - `reference operator*()` returns the current value.
   - operators for equality and inequality. Two iterators are unequal as long as one has not passed the other (depending on the sign of the step size). In this assignment, you can assume that the left hand side object is the one being incremented.

   Create another class `Range` that has one template type argument `T`. This class should have the following:

   - Data members to store starting value, end value and step size, all having type `T`.
   - Constructor(s) to initialize the data members according to the given examples.
   - A inner type alias `iterator` of type `Range_Iterator<T>`.
   - `iterator begin()` that creates an iterator that starts at the starting value and has the given step size
   - `iterator end()` gives an iterator that stores the end value.

   Lastly create a free template function `range` that has one simple task, take the number of arguments needed to initialize a `Range` object and pass these to the correct constructor and return the created object. This is possible to do with several overloads of this function, but to get full marks on this assignment you are only allowed to have one (hint: variadic templates is needed).

   Your templates will only be tested with arithmetic types, you do not have to check this (even if it is rather simple to do with concepts and `std::is_arithmatic_v`).

7. Suppose that you have a system that generates a list of student results for a certain exam. The format of each student result in this list is seen below (where `<PNR>` is the Swedish *personnummer*): [5p]

```
<NAME>:<SURNAME>:<PNR>:<POINTS>:<GRADE>
```

The file `given_files/STUDENT_RESULTS.TXT` is a sample set of data from this system (all names are randomized).

Your task is to create a program that reads a result list from STDIN and prints the list in sorted order to STDOUT (sort by grade first - 5,4,3,U, then surname and lastly by name) according to the format below (`...` symbolizes several discarded lines to save space, your program should print all lines). Note that the name part is 30 characters in total.

```
Name                           Personnummer  Grade (Points)
Ahl, Jonatan                   920828-XXXX       5 (20)
Almkvist, Therese              870225-XXXX       5 (24)
...
Ruggiu, Jonathan               891030-XXXX       4 (15)
Vilkancis, Adam                961102-XXXX       4 (17)
Akhtar, Antymos                960815-XXXX       3 (14)
Balzereit, Rikard              890617-XXXX       3 (13)
...
Wasing, Lena                   950724-XXXX       U ( 2)
Zornic, Joanna                 880731-XXXX       U ( 2)
```

Requirements

1. Each line in the input should be saved as an object of your own class type `Student`.
2. It is (of course) important to use encapsulation and const correctly in `Student`.
3. All students should be saved in a `vector`.
4. For full marks, use the STL for everything in main. The problem has three parts; input, sorting and output and you should be able to solve it with three expressions in main (possibly some declarations as well).

Hint: To save some typing, you can redirect the input stream to read all the contents from the given file:

```
a.out < given_files/STUDENT_RESULTS.TXT
```

8. This assignment tests your ability to use the standard library. It is important that you use the algorithm best suitable task, `for_each` is not a valid choice for all parts. You should not use any built-in iteration statements. Don't store more data than necessary unless you have a good reason (please comment). [5p]

   You have two files that together represent a set of time intervals. Each line in the file `given_files/START_TIME` is the start of the interval that ends on the corresponding line in `given_files/END_TIME`.

   Each timestamp is in the format HHMM.

   Your task is to create a program that does the following:

   1. Check command-line arguments. The program should be started with the both file names as arguments and these should be readable files. If anything goes wrong, some informative message is printed and the program ends.
   2. Calculate and store the length of each time interval in a vector called `intervals`. If the end time is before than the start time, the end time is assumed to be the next day (24h later).
   3. Print the largest and smallest time interval.
   4. Print all intervals separated with newline.

   The following example shows a valid output for the given files:

```
Largest interval : 1444
Smallest interval: 0001
All intervals
0001
0058
0017
1444
```