Programming Expertise
University of Potsdam SS 2021
Detlef Groth, Christian Kappel
Exam July 22nd, 2021

You have 80 minutes time for the implementation of the programming tasks. Make after the steps 1-3 intermediate versions: GooboParser1.cpp, GooboParser2.cpp and GooboParser3.cpp . Submit at the end of the exam those single versions by USB stick or by E-Mail preferentially as zip archive (surname.zip) to me (E-Mail: dgroth@uni-potsdam.de). 75% of the tasks will be on execute correctness and 25% of the tasks will be weighted by usefulness and clearness of the implementation.
The theory questions in 4-6. are to be answered first without any aids and the sheet with the answers is to be handed in after about 15 minutes. For the computer tasks 1-3 all aids are allowed during the exam. This does not include personal support from fellow students or other persons.

Clarification

With this I state, that I will not take and give any not allowed support during the exam.

Name, Matrikel-Number.: Signature:

1. 2 points (layout console application)
2. 4 points (implementation of console application)
3. 4 points (advanced terminal task 3A or GUI task 3B)
4. 2 points (theory – C)
5. 2 points (theory – C)
6. 2 points (theory - C++)

Sum: 16 points

Good luck !!

1. Layout of console application and utilizing command line arguments (2 points):
Create the basic outline of a console application with main function, help function and usage of command line arguments. Save the possible command line arguments in two variables . If not at least two argument were given call the help function and exit the application. The two arguments should be a goobo input filename and a taskname like *summary* or *children*
The database file was downloaded from: http://purl.obolibrary.org/obo/go/go-basic.obo
You find the obofiles on Moodle. Further check if the given filename points to a valid filename. If not tell the user, that the file does not exists.
C++ filename first task: ______________________________

2. Opening and searching in the Goobo file (4 points): Our program should work with any goobofile file. Please don't hardcode the filename in your application, if you do, you get a minus point. Implement the *summary* function for a specific Gene Ontology namespace. If the user gives the for example the three arguments: *filename.obo summary molecular_function* on the command line, then go-obo file is parsed and the number of valid and obsolete entries should be shown on the terminal Hint: You can print the text inside of the function or you return the text to the function caller, for instance as a vector of strings, print it then outside of the function. This approach better fits with task 3. Hint: If your code is very slow use limit your search to the first 100 entries first to save your programming time during development. You should uncomment this in your final program. BTW: The other namespace are: *cellular_component* and *biological_process*
C++ filename(s) second task: ____

3A. Terminal - ECSummary (4 points, Alternatively you can do task 3B - GUI):

1 (1 point): Extend your application so that it can handle more than 1 input file at the same time, your output should then have as well the filename as the first column entry.

Hint: handle the multiple files in main, not within the function/method! If you skip this you can do taks 3A2 as well with one file, but the filename should be shown there as well.

2) (3 points): Add an additional command ECSummary which summarizes the first two digits of EC-entries and prints out the number of how often such an EC category appears in an ordered manner, so that the most often terms appears first, here an Example output:

```
$ goutils gene_ontology_edit.obo.2021-01-01  gene_ontology_edit.obo.2016-
01-01  ECsummary
gene_ontology_edit.obo.2021-01-01       EC:1.1  474
gene_ontology_edit.obo.2021-01-01       EC:2.7  423
gene_ontology_edit.obo.2021-01-01       EC:1.14 383
gene_ontology_edit.obo.2021-01-01       EC:3.1  375
…
gene_ontology_edit.obo.2016-01-01       EC:1.1  434
gene_ontology_edit.obo.2016-01-01       EC:2.7  413
gene_ontology_edit.obo.2016-01-01       EC:1.14 378
```

. . .

3B. Building a small GUI for biologists (4 points):
If the user starts the application with the string GUI as the second argument a small application with a graphical user interface should be starting.

$ ./GooboParser3 /srv/groth/data/gene_ontology-2019-01-01.obo GUI

You might create a class GooboGui or just a simple gui function. If you build a class you should have a look at: https://www.fltk.org/articles.php?L379+I0+TFAQ+P1+Q. Your graphical interface should consist of three Fl_Button(s) and one Fl_Input control on top. The Fl_Input widget will be used for entering the ID or name search and the three buttons, “open”„”search“ and „exit“. Below on the left is a Fl_Tree widget which should be filled with the three GO-ID’s covering the three major namespaces and search entries of the Goid search based on the Fl_Input widget. If the user enters an incomplete GO-ID all GO-ID should be shown which match this GO-ID. For example if the users enters: GO:00012 all GO-ID’s which start with those numbers should be displayed in the tree. Please be sure to update the widget after adding new entries. If a new search is started the tree widget should be cleared. Each time the three mjor namespaces should be added on top. If no Id is found, the text „id not found“ should be shown in the tree widget. On the right you should display a Fl_Multline_Output or a similar widget for displaying the information found in the GO entry clicked on the left. If the user clicks the main namespace names the summary info should be displayed on the right widget (see task 2), otherwise for normal GO-Ids the info belonging to the Go-ID up to the next term is displayed. The “open” button should allow the user to select an other Obofile, “exit” should exit the application.

A) Create the layout of the GUI app (2 points)
B) Implement the described functionality (2 points)
C++ filename(s) for third task: ____

Name, Matrikel-Number.: **Signature:**

4-6. Theoretical questions:

4. C – Question 1 (see other page)

5. C – Question 2 (see other page)

6. Explain in 4-5 sentences the difference of static variables and static member functions in comparison to ordinary member variables and functions in C++ classes. Explain also how are static variables in classes different to static variables in non-class functions?