Preliminary description

24 декабря 2019 г.

Description of the test structure

The *programming part* of the exam consists of three related tasks. The 1st and 2nd tasks are considered independently.
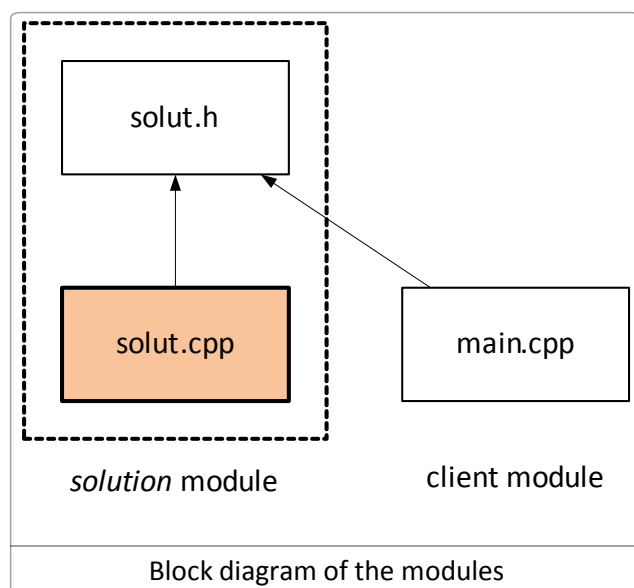
The first task gives up to 0.35 of the grade for the complete solution, the second — up to 0.65. The third task is optional and does not affect the main grade given for the programming part. The third task gives you up to 20 bonus points that can be used to improve *OA* grade.

The third task is an optional task that can be considered only if two mandatory tasks are completely solved.

Each of the parts in the contest is a separate task, submission is carried out sequentially from the first one to the second and then the third, by adding necessary code. You are asked to submit only requested fuctions at each step. See the task notes in Yandex Contest.

The task notes for each of the parts are represented in Yandex Contest and also given in the form of comments on the corresponding function skeletons that need to be implemented.

To simplify the solution of the problem, the **structure of a CMake project has been prepared**, which can be cloned from the repository (the link is indicated before the test) or downloaded from there in the form of a zip archive. In case of problems with access to the Internet, the project is also issued in the form of a password-protected archive in advance.

Block diagram of the modules

The source code modules given in the project contain three files: `solut.cpp`, `solut.h`, `main.h`. The dependencies between them are presented in the diagram above.

The main working file is `solut.cpp`, it solves the problems of the test by writing code of methods, whose skeletons are presented in it.
To perform the task decomposition, the methods highlighted by students are also added to this file.
After developing the functions, this file is submitted to Yandex Contest.

The `solut.h` file contains aliases for types that will help (and will be needed) for developing interface functions (that is, those that are exported to a test environment, for example, to `main.cpp`), as well as their prototypes. This file should not be modified by students.

The `main.cpp` file contains sample code for using the developed functions and can be used for debugging and testing. It is recommended to refer to test functions 1, 2, 3 in the course of the implementation of the relevant parts of the tasks. Prior to this, they should be commented out to avoid compilation and runtime errors.

The `/data/` directory contains a file with test data, named `input.txt`, which can also be used for debugging. It is presented in the universal form according to the POSIX-standard (each line is ended up with the only LF character). This file is apropriate for all of the following OS: Windows, Linux, Mac.

While mastering a program, you have to put the test file in the directory with the target execution file. Alternative way is to put it somewhere in a known location (e.g., in the root directory of disc `C:`) and specify the path to the file in test functions in `main.cpp` module.
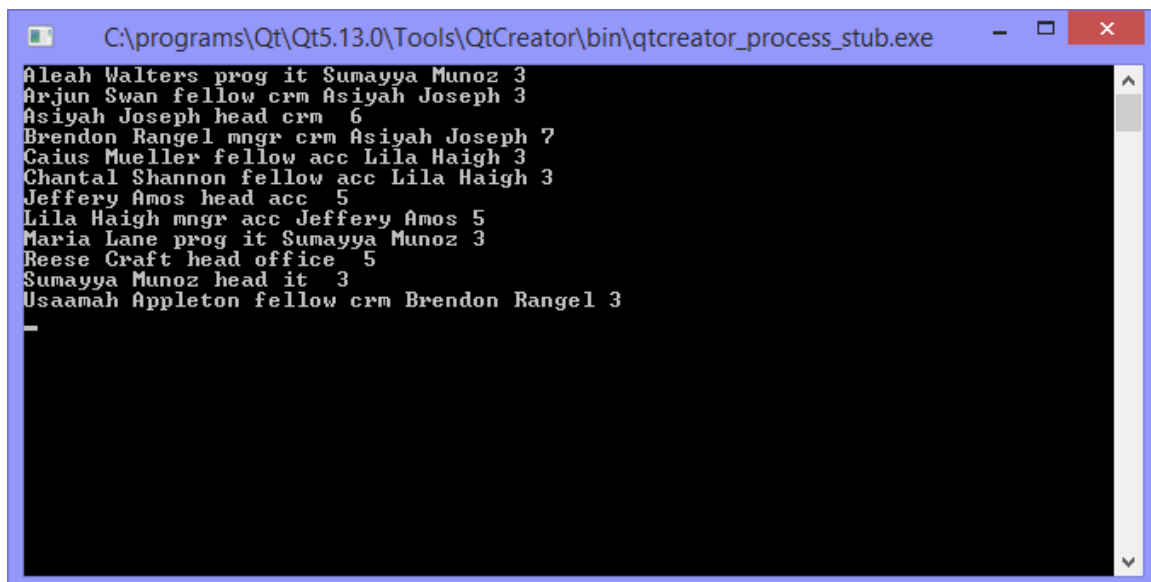
Task 3 uses modified version of test file named `input-task3.txt`.

Submitting results

Only the `solut.cpp` file with specified functions is to be uploaded to Yandex Contest, therefore no other files are to be modified. Any decomposition (i.e. splitting into smaller methods) should be done within the `solut.cpp` file without changing the other files.

Program output examples

Output for Task 1

Output for Task 2

```
acc
        Caius Mueller, fellow; 3 w/d
        Chantal Shannon, fellow; 3 w/d
        Jeffery Amos, head; 5 w/d
        Lila Haigh, mngr; 5 w/d
crm
        Arjun Swan, fellow; 3 w/d
        Asiyah Joseph, head; 6 w/d
        Brendon Rangel, mngr; 7 w/d
        Usaamah Appleton, fellow; 3 w/d
it
        Aleah Walters, prog; 3 w/d
        Maria Lane, prog; 3 w/d
        Sumayya Munoz, head; 3 w/d
office
        Reese Craft, head; 5 w/d
```

Output for Task 3

```
Caius Mueller
Chantal Shannon
Lila Haigh
```