Развој софтвера – пример испита 2020/2021

Испит траје 3 сата.

Задатак	1	2	3	4	5	6	7	8	9	10	11	12	Укупно
Макс.	5	5	5	5	5	5	5	5	5	5	30	30	110

Чување рада:

Директоријум који сте отпаковали и у којем ћете смештати решења назовите Вашим налогом на серверу Алас (на пример, `mi10050`). **Ако сте мењали индекс, ставите нови број индекса у Ваш налог на серверу Алас**. Само ће садржај овог директоријума бити сакупљен и оцењен. Ако оставите више решења за неки задатак, или више директоријума или директоријум назовете у другом формату, Ваш рад неће бити прегледан и аутоматски добијате 0 поена.

Документација:

Да бисте приступили документацији за програмски језик `C++`, потребно је да испратите упутство у датотеци `cppreference.pdf`. У случају да је документација већ била учитана у алату `Qt Creator`, онда је довољно да је отворите као што је приказано на последњој слици у датотеци `cppreference.pdf`.

ТЕОРИЈСКИ ДЕО — РАДИТИ НА ПАПИРУ
ПИСАТИ ЧИТКО — НЕЧИТКИ ЗАДАЦИ НЕЋЕ БИТИ ПРЕГЛЕДАНИ

1. ...
2. ...
3. ...
4. ...
5. ...
6. ...
7. ...
8. ...
9. ...
10. ...

ПРАКТИЧНИ ДЕО

11. Коришћењем технике *развоја вођеног тестирањем* (*TDD*), имплементирати класу `Encoder` која шифрује произвољне ниске. Алгоритам шифровања зависи од "корака шифровања" Π, целог броја. Алгоритам сваки карактер K који представља цифру, велико или мало слово улазне ниске замењује карактером који се налази на растојању Π од карактера K у ASCII кодној шеми, уз циклично померање (циклус се понавља на нивоу потклase цифара, потклase великих и потклase малих слова засебно). На пример, за $\Pi = 3$, алгоритам шифрује карактер `0` у `3`, карактер `n` у `q`, а карактер `Z` у `C`[1], док за $\Pi = -55$, алгоритам шифрује карактер `d` у `a`[2], и слично, карактер `2` у `7`. Сваки карактер који није цифра, велико или мало слово се кодира карактером `#`.

    Обезбедити наредни јавни интерфејс класе `Encoder`:

    - Конструктор класе прихвата цели број који представља "корак шифровања".
    - Написати *оператор позива функцијског објекта* (енг. *call operator*) који прихвата ниску и враћа њену шифровану вредност.

    Напомене:

    - Неопходно је написати макар 6 јединичних тестова.
    - Водити рачуна о именовању и парадигми писања тестова.
    - Имплементација која је урађена без *TDD*-а носи највише 40% поена.

[1] Циклични померај $Z \mapsto A \mapsto B \mapsto C$.
[2] Циклични померај $d \mapsto c \mapsto \ldots \mapsto d \mapsto c \mapsto \ldots \mapsto d \mapsto c \mapsto b \mapsto a$.

12. Написати Qt апликацију која конкурентно симулира једну трку паса на 100 метара на основу података из датотеке. Слика 1 илуструје како програм треба да изгледа приликом покретања. У наставку следи опис рада програма.

(а) Графички кориснички интерфејс имплементирати тако да изгледа као интерфејс приказан на слици 1. Прозор направити тако да се приликом промене његове величине, компоненте аутоматски померају да задрже дати распоред.

(б) Креирати класу `Racer` која садржи податке о једном тркачу. Један тркач је описан својим именом (`QString`), брзином (`unsigned`) и процентом пређеног пута трке (`double`). Обезбедити наредни јавни интерфејс класе `Racer`:

- Креирати метод `QString toQString() const` који израчунава ниску која илуструје пређени пут. Ниска се састоји од имена тркача, праћена карактером двотачке, а затим следи онолико понављања карактера `#` колико има десетица у запису процента пређеног пута тог тркача. На пример, уколико је у неком тренутку тркач имена `Blitz` прешао $55,3\%$ пута, онда овај метод враћа ниску `"Blitz:#####"`.
- Креирати метод `void fromVariant(const QVariant &variant)` којим се врши десеријализација објекта класе `Racer`.

(в) Кликом на дугме *Региструј учеснике трке*:

- Отворити дијалог за одабир датотеке као на слици 2. Корисник треба да одабере XML датотеку која садржи податке о позитивном броју тркача. Претпоставити да је датотека добро форматирана (пример XML датотеке `racers.xml` је дат у поставкама).
- Из одабране датотеке десеријализовати податке о тркачима као објекте класе `Racer`. Након учитавања података, приказати их у вишелинијском текстуалном пољу као на слици 3.

(г) Кликом на дугме *Започни трку*, покренути за сваког тркача по једну нит. Након што је покренута, нит понавља наредне кораке све док одговарајући тркач не заврши трку (тј. док проценат пређеног пута не премаши 100%):

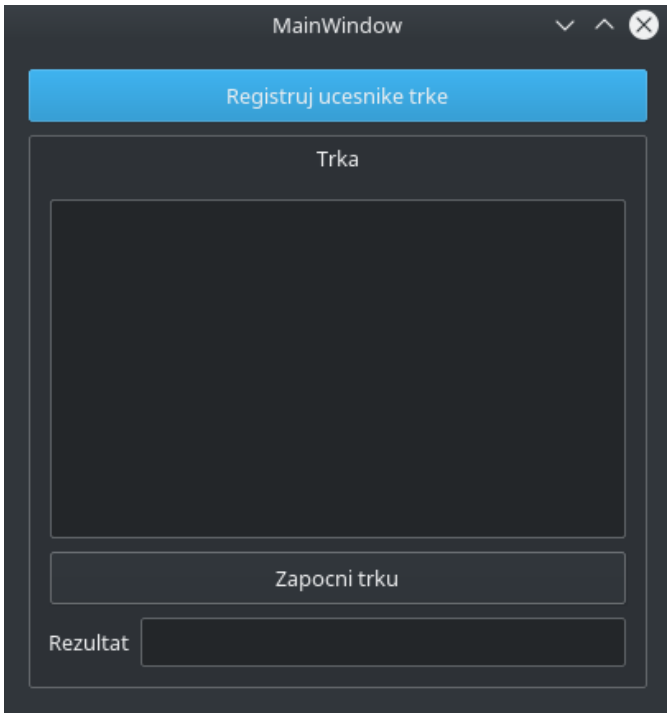
- Успављује се на 1 секунду.
- Ажурира проценат пређеног пута тркача са датим редним бројем за вредност брзине.

Додатно, омогућити да, када се нити покрену, све контроле буду онемогућене као на слици 12.

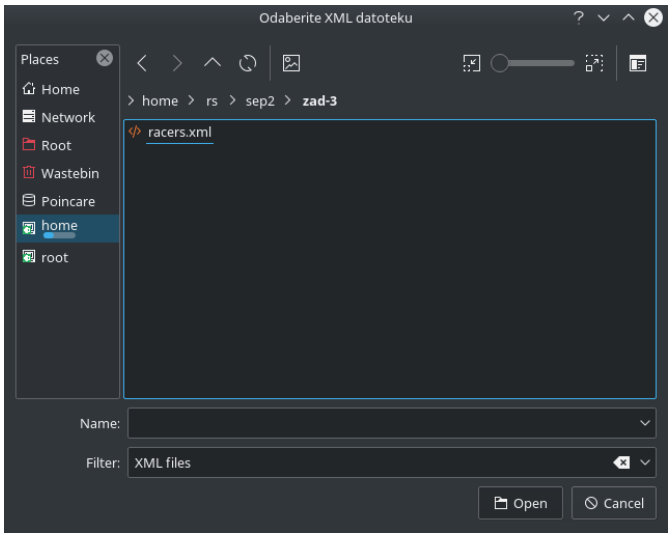
(д) Када једна нит ажурира проценат пређеног пута, потребно је обавестити главну нит да је дошло до измене, на основу чега је потребно ажурирати приказ у вишелинијском текстуалном пољу. За сваког тркача исписати у новом реду ниску која се добија позивом метода `toQString()` из класе `Racer` (део под б). Пример приказа овог понашања је дат на сликама 5 и 6.

(ђ) Када први тркач заврши трку, његова нит се обуставља и његово име се уписује у једнолинијско поље са ознаком *Резултат*. Када сви тркачи заврше трку, омогућити све контроле, као на слици 6.

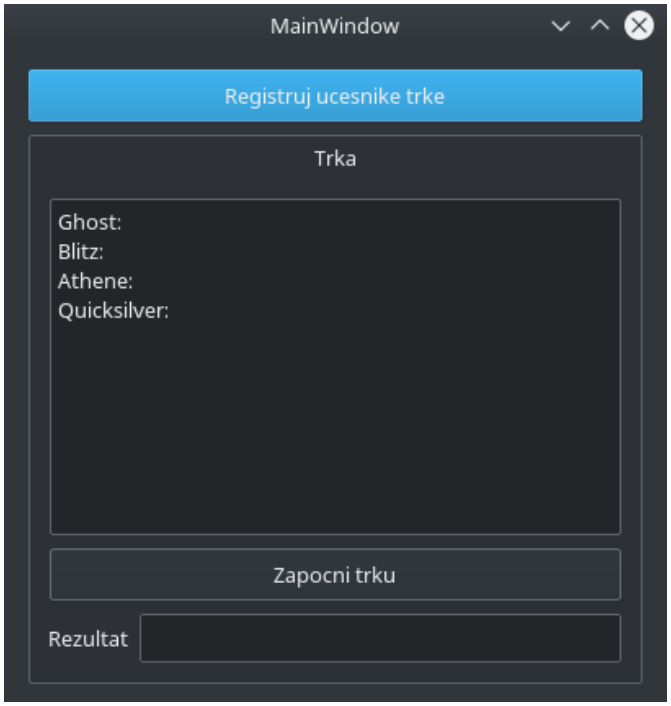
(е) Осигурати се да не долази до проблема у конкурентном окружењу и водити рачуна о раду са динамичким ресурсима.



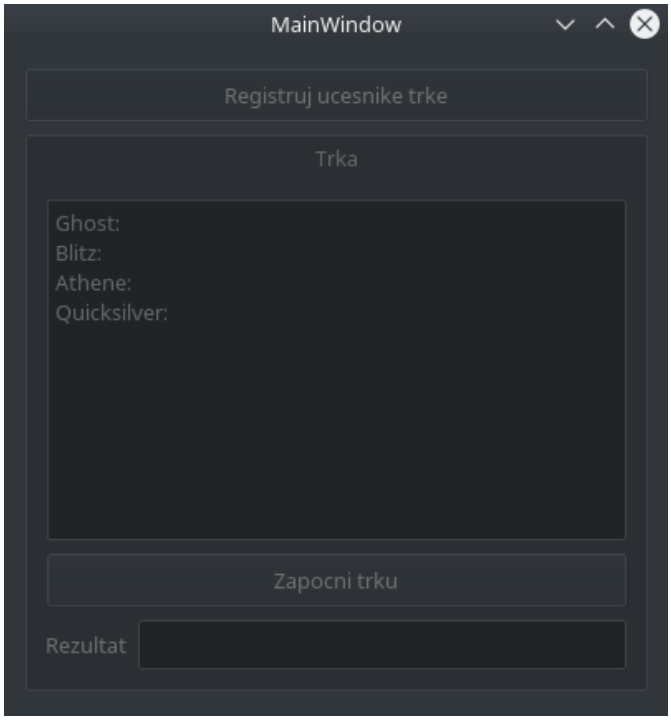
Слика 1: Иницијални приказ графичког интерфејса.



Слика 2: Пример покретања дијалога за одабир датотеке након клика на дугме *Региструј учеснике трке*.

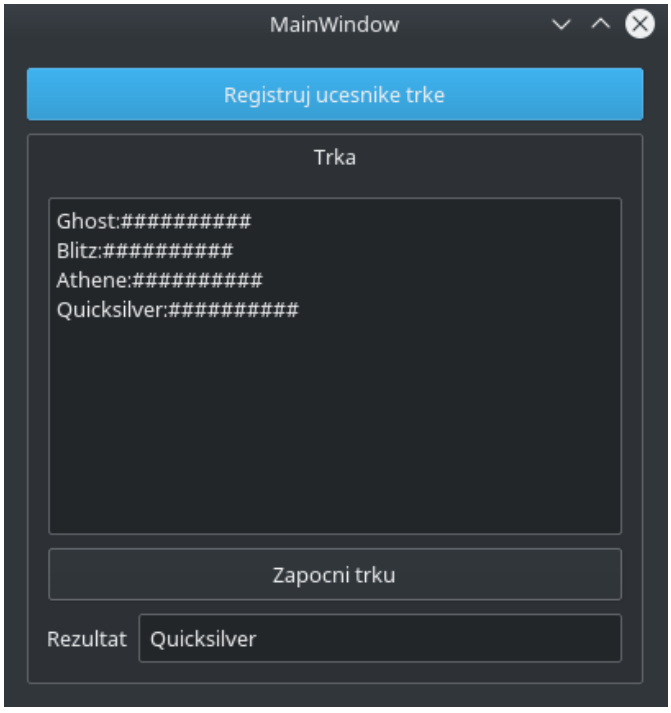


Слика 3: Пример приказивања података о тркачима након учитавања података из одабране датотеке.



Слика 4: Приказ након клика на дугме *Започни трку*.

Слика 5: Пример приказа након што је један тркач завршио трку.



Слика 6: Пример приказа након што су сви тркачи завршили трку.