Индивидуални пројекат

Последњи рок за предају пројекта: 08.01.2019. 23:59

Пројектни захтеви

Случај коришћења

Стицајем непредвиђених околности на које нисте могли да утичете касните на лет за Лондон. На том аеродрому сте први пут и баш се не сналазите најбоље а време вам је битан фактор. Наилазите на таблу са информацијама о летовима али не можете да се снађете, тј. не можете да пронађете ваш лет јер летови нису сортирани. Можда летови јесу сортирани али не по критеријуму који вама одговара?

FLIGHT NUMBER	DESTINATION	DEPARTURE TIME	GATE NUMBER
AA223	LAS VEGAS	21:15	A3
BA036	DALLAS	21:00	A3
AA220	LONDON	20:30	B4
VI303	MEXICO	19:00	B4
BA087	LONDON	17:45	B4
AA342	PARIS	16:00	A7
VI309	PRAGUE	13:20	F2
QU607	TORONTO	8:30	F2
AA224	SYDNEY	8:20	A7
AF342	WASHINGTON	7:45	A3

Претпоставимо да је горе приказана листа летова сада сортирана на другачији начин. Како би било ако би ви сами могли да бирате критеријум сортирања тако што ће вам на располагању бити и назив дестинације и број лета а не само време поласка?

FLIGHT NUMBER	DESTINATION	DEPARTURE TIME	GATE NUMBER
AF342	WASHINGTON	7:45	A3
AA224	SYDNEY	8:20	A7
QU607	TORONTO	8:30	F2
VI309	PRAGUE	13:20	F2
AA342	PARIS	16:00	A7
BA087	LONDON	17:45	B4
VI303	MEXICO	19:00	B4
AA220	LONDON	20:30	B4
BA036	DALLAS	21:00	A3
AA223	LAS VEGAS	21:15	A3

Функционални захтеви

Написати Це++ програм који омогућује следеће:

- Учитава списак полазака у задатом формату.
- Сортира списак полазака по одабраном критеријуму.
- Омогућује прикупљање и чување података у току сортирања.
- Употребом FLTK библиотеке приказати (визуализовати) померање елемената у свакој итерацији сортирања.
- Подржати прослеђивање неопходних улазних параметара програма преко аргумената команде линије.

Имплемнтациони кораци

1. Из улазне датотеке учитати податке о авионским поласцима. Податке о поласцима учитати као објекте класе *Flight.* Класа треба да има атрибуте за сваки учитани податак који описује конкретни полазак: *lightNo*, *destination*, *departure* и *gateNo*. Користити знаковни низ (*string*).
2. Из улазне датотеке учитати тачно 10 полазака. *Напомена: У сврху тестирања препоручљиво је учитати више од 10 полазака.*
3. За сортирање користити SELECTION SORT и један алгоритам по избору. Изборни алгоритми су: MERGE-SORT, QUICK-SORT и HEAP-SORT.
4. Поласци треба да буду сортирани у лексикографском редоследу у односу на одабрани критеријум сортирања који се дефинише од стране корисника било кроз аргумент команде линије или кроз графичку корисничку спрегу.
5. Омогућити чување података у току сортирања: број итерације, тренутни распоред елемената, број поређења и број померања елемената.
6. Обезбедити подршку за испис сортираних елемената и прикупљених података у току сортирања. Подржати упис у излазну текстуалну датотеку. Дефинисати формат и структуру излазне датотеке.
7. Употребом FLTK библиотеке омогућити: унос путање до улазне и излазне датотеке, одабир или дефинисање критеријума сортирања и одабир алгоритма за визуелизацију.

8. Употребом FLTK библиотеке приказати (визуелизовати) померање елемената у зависности од итерације сортирања – пример излаза је дат у наставку. [1]
9. Обезбедити посебно дугме за приказ наредне итерације сортирања.

НАПОМЕНЕ:

- По потреби додати нове и/или прилагодити већ постојеће делове кода.
- Обратити пажњу на употребу адекватних структура података.
- Водити рачуна о динамичким структурама података.

[1] У свакој итерацији сортирања елемент потенцијално промени постојећу позицију што може бити искоришћено као механизам за праћење елемената.

Пример сортирања употребом *Selection sort* алгоритма

У *Selection sort* алгоритму након *i*-те итерације (*i*-1)-ти елемент је замењен са најмањим елементом низа из интервала од *i*-тог до *n*-тог елемента.

Пример сортирања низа бројева:

(низ је подељен са "|", сортирани елементи се налазе са леве стране)

- Почетни низ:

| 5 -2 4 7 3 2 -4 0 2 1 1

5 је замењено са -4 (најмањи елемент)

- Низ након итерације 1:

-4 | -2 4 7 3 2 5 0 2 1 1

-2 је на правом месту па према томе нема замене.

- Низ након итерације 2:

-4 -2 | 4 7 3 2 5 0 2 1 1

4 је замењено са 0 (најмањи елемент из низа који није сортиран)

- After iteration 3:

-4 -2 0 | 7 3 2 5 4 2 1 1

7 је замењено са 1 (најмањи елемент из низа који није сортиран)

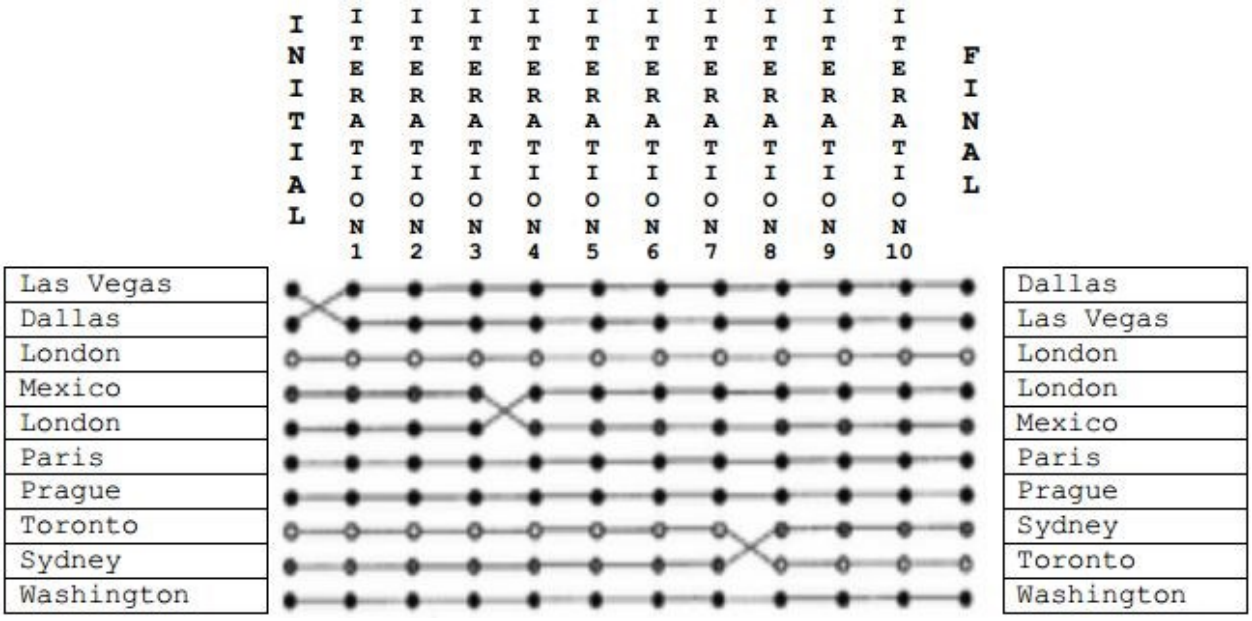
итд.

Пример визуелизације сортирања – *Selection sort* алгоритам

Пример сортирања градова:

Завршни корак у пројекту је приказ промене позиција елемената односно повезивање линија између истих елемената. То нам омогућује да испратимо како су се елементи померали у току сортирања. Следи пример визуелизације сортирања имена одредишног града употребом *Selection sort* алгоритма. Обратити пажњу на који начин је графички представљена промена (замена) елемената (итерације 1, 4, 10).[2]

Очекивани излаз:



[2] Ово је само један од начина на који је могуће остварити визуелизацију промене елемената у току сортирања. Измена приказаног или дефинисање другачијег решења остављено је на располагању.

Имплементациони захтеви

- Обезбедити посебну класу за руковање са улазним аргументима.
- Завршити дефиницију класе *Flight* која се налази у заглављу Flight.h.
- Обезбедити функцију за поређење два објекта типа *Flight* у односу на дефинисани критеријум сортирања.
- Обезбедити имплементацију алгоритама за сортирање.[3]
- Обезбедити посебну класу за чување прикупљених података.
- Обезбедити класу *MyWindow* која наслеђује класу *Window* по узору на структуру *Simple_window* (видети graph_lib).
- *MyWindow* класа треба обезбеди имплементацију функционалности које се односе на графичку корисничку спрегу.
- Поделити функционалност програма на следећи начин:
    - учитавање и анализа аргумената команде линије
    - учитавање података, сортирање и прикупљање података
    - упис резултата сортирања и прикупљених података у датотеку
    - имплементација основних елемената графичке корисничке спреге
    - визуелизација сортирања

Структура улазне датотеке

Улазна датотека треба да има следећу структуру:

- Прва линија представља заглавље датотеке.
- Све наредне линије представљају улазне податке.
- Унутар линије подаци су раздвојени карактером „;“.
- Подаци су организовани на следећи начин:
    - Дестинација – може да садржи више речи
    - Време поласка – користити 24-часовни формат
    - Број лета – у формату „ссббб“ где ‘*с*’ означава слово а ‘*б*’ број
    - Број излаза– у формату „сб“ где ‘*с*’ означава слово а ‘*б*’ број
- У једној улазној датотеци потребо је дефинисати тачно 10 летова. *Напомена: У сврху тестирања препоручљиво је користити и више од 10 летова.*

[3] Није дозвољено коришћење STL алгоритама за сортирање.

Захтеви улаза и излаза (У/И)

Потребно је омогућити следећу У/И спрегу са корисником:

- Прослеђивање потребних аргумената преко командне линије.
- Својевољно дефинисати структуру и редослед улазних параметара.
- Учитавање података из улазне датотеке.
- Обезбедити испис и чување:
  - сортираних података
  - података прикупљених у току сортирања
- Користећи графичку библиотеку треба обезбедити:
  - могућност уноса путање улазне датотеке
  - могућност уноса путање излазне датотеке
  - могућност одабира или дефинисања критеријума сортирања
  - могућност одабира алгоритма за сортирање
  - визуелизацију сортирања претходно одабраног алгоритма
- Претпоставити да је улазна датотека дефинисана без грешака.
- Произвољно дефинисати формат и структуру излазне датотеке.

Стил кодовања

- У сваком заглављу и модулу са изворним кодом додати кратак опис функционалности, информацију о ауторима, и датум и аутора последње измене.
- Коментарисати најбитније слободне функције, функције чланице и атрибуте класа. За функције обезбедити кратак опис функционалности, листу улазних аргумената, повртану вредност и уколико користи тип изузетка.
- Обратити пажњу на индентацију и форматирање кода, на стил и формат именовања промељивих и функција, као и на дужине линија.
- Код треба да буде прегледан, читљив и да садржи корисне коментаре.
- Више о начину и стилу кодовања доступно је у следећем документу: http://www.stroustrup.com/Programming/PPP-style-rev3.pdf

Излазни резултати

Одабиром релевантних тестних случајева у документацији приложити:

- Табеле (i) броја замена елемената и (ii) броја поређења елемената за сваку итерацију сортирања. *Напомена: Потребно је формирати табеле само за релевантне тестне случајеве. Анализирати оба алгоритма.*
- Табелу која садржи (i) укупан број замена елемената и (i) укупан број поређења елемената на нивоу тестног случаја. *Напомена: Такође анализирати само релевантне тестне случајеве.*
- Коментарисати резултате. Ко је алгоритам ефикаснији и зашто?

Извештај

Треба да садржи следеће:

- Насловну страну са информацијама о аутору.
- Опис и/или анализу:
  - Рада У/И подсистема (учитавање и испис/упис).
  - Списак свих класа, изузатака и слободних функција.
  - Објашњење најбитних атрибута класа и функција чланица, слободних функција и изузетака.
  - Структуре аргумената командне линије и пример коришћења.
  - Структуре излазне датотеке.
  - Одабраног алгоритма за сортирање.
  - Напредних ОО концепата (уколико су коришћени): наслеђивање, преклапање оператора, полиморфизам итд.
  - Графичке спреге (основни елементи и функционалности).
  - Тестних случајева – приложити табелу са улазним подацима и слику која представља стање корисничког програма након завршетка сортирања. Слика треба да обухвати само део графичке корисничке спреге (изоставити позадинске детаље).
  - Ставке специфициране у претходној секцији „Излазни резултати“.
  - Уочени проблеми и ограничења.

Упутство за предају пројекта

- Пројекат треба да буде архивиран и именован на следећи начин:

Indeks_Ime_Prezime.zip
(пример: SW123_Petar_Petrovic.zip)

- Архиву окачити на канвас у предвиђеном временском року.
- Архива треба да садржи следеће директоријуме:
  - **Kodovi** – садржи датотеке са изворним кодом (.cpp и .h/.hpp) и пројектне датотеке (.sln, vcxproj и filters).
  - **Testovi** – садржи све тестне датотеке.
  - **Dokumentacija** – садржи пројектну документацију *Izvestaj.doc*.
- Архива **НЕ СМЕ** садржати следеће датотеке:
    - .sdf
    - .suo
    - .user
    - .obj
    - .lib
    - .exe
    - ipch
    - Debug директоријум
    - Release директоријум
    - **FLTK библиотеку**
  - Приложен је референтни пример решења пројектног задатка.
- За информацију о термину одбране обратити се асистенту.
- На термин одбране потребно је **донети штампану верзију документације**.

НАПОМЕНЕ:

- Израду пројекта наставити у приложеном решењу.
- У циљу квалитетније провере рада програма користити више тестних датотека са различитим летовима.