

Извештај

Објектно оријентисано програмирање 2

Пројектни задатак:

~ Sorting Flights

Студент:

~ Невена Прокић
SW6/2019

Рад У/И подсистема (учитавање, упис)

Имена улазне и излазне датотеке су дефинисане уз помоћ аргумената командне линије. Уколико корисник не унесе име улазне датотеке или унето име није исправно биће извршено учитавање над датотеком која је у аргументима командне линије. Уочени проблем овде је да приликом погрешног уноса корисника програм се даље извршава над подразумеваном датотеком.

Учитавање улазне датотеке се врши под претпоставком да је она добро дефинисана и то се неће проверавати.

Упис у унету или подразумевану датотеку ће бити у формату у ком се налазио и у улазној датотеци. Елементи који се уписују ће се разликовати од учитаних по параметрима по којим су сортирани.

Списак свих класа

- ChooseWindow
- DisplayWindow
- FileWork
- Flight
- MyWindow
- Sort
- PointWindow
- све дефинисане у Graph_lib библиотеци

Објашњење најбитнијих атрибута класа и функција чланица, слободних функција и изузетака

У класи **Sort** која је намењена за сортирање, се налазе класе које је наслеђују **HeapSort** и **SelectionSort**. Ове две класе ће имплементирати апстрактну методу класе Sort, која као параметар прима референцу на низ (вектор) елемената која је потребно сортирати у унапред одређеном поретку. Повратна вредност ове методе ће бити DTO објекат, који садржи сортирани низ по одређеном параметру, низ промена, број итерација, број поређења и број померања.

У **FileWork** класи се налазе две методе `readInputFile` и `writeOutputFile`, које служе за рад са улазним и излазним фајловима, тј. читавање летова и уписивање сортираних.

Класе које се користе за преузимање основних информација од корисника помоћу графичке корисничке спреге су:

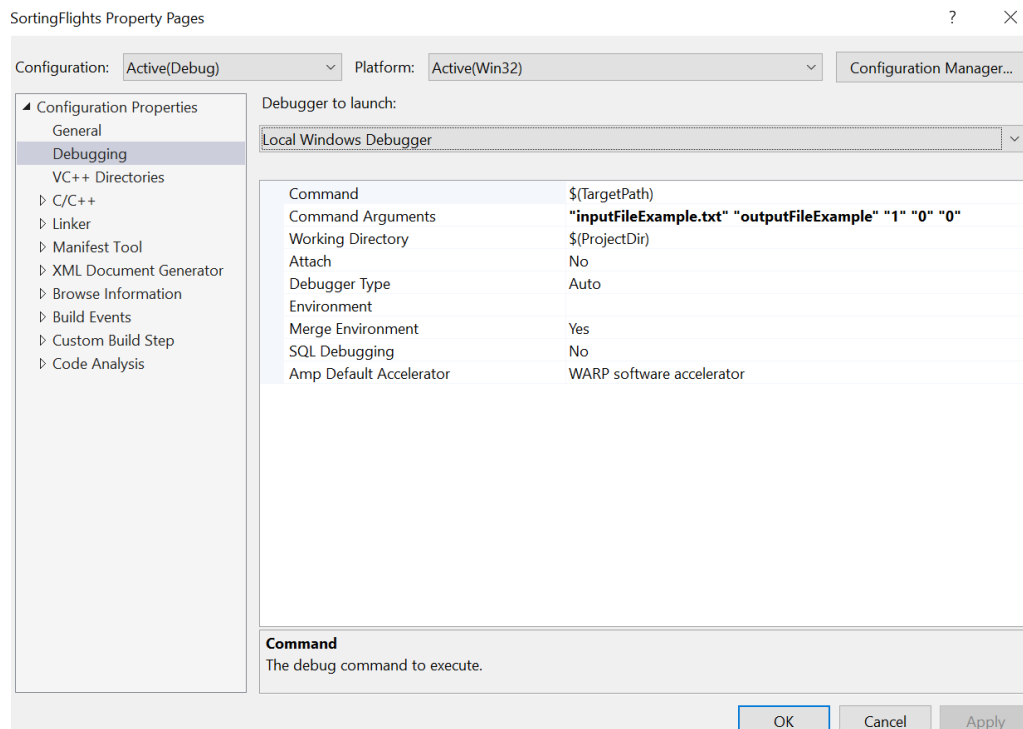
- **ChooseWindow**
- **MyWindow**
- **DisplayWindow**
- **PointWindow**

* детаљније ће бити описане у делу “Елементи и функционалности корисничке спреге”.

Структура аргумената командне линије и пример коришћења

Програм подражава прослеђивање путање улазног и излазног фајла преко аргумената командне линије и помоћу графичког корисничког интерфејса.

Пример коришћења аргумената командне линије:



Структура излазне датотеке

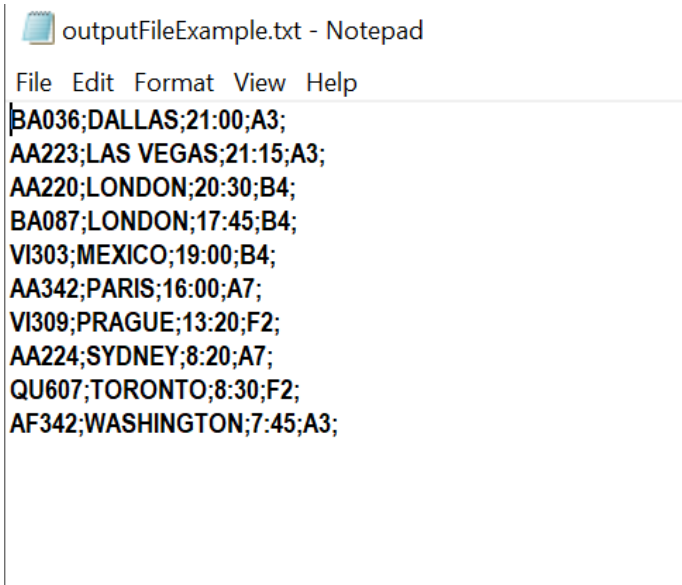
Подаци унутар датотеке раздвојени су „ ; “.

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

FlightNo;Destination;Departure;GateNo;

Приказ излазне датотеке:



outputFileExample.txt - Notepad

File Edit Format View Help

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

Heap sort - структура и опис

Heap sort алгоритам смешта елементе у бинарно стабло.
Овај алгоритам ради по следећем принципу:

1. Формирање heap:
 - Додај елемент у бинарно стабло
 - Изврши unhear који успоставља коректан редослед у стаблу
2. Избацивање елемената из heap-а:
 - Замени корен и последњи чвор
 - Избаци последњи елемент
 - Изврши downhear који успоставља коректан редослед у стаблу

Heap sort ради у $O(n \log n)$ времену.

| | | | | | | | | | | | |
|----|---|---|---|----|---|---|----|---|---|---|---|
| 10 | 4 | 8 | 5 | 12 | 2 | 6 | 11 | 3 | 9 | 7 | 1 |
|----|---|---|---|----|---|---|----|---|---|---|---|

Опис напредних ООП концепата

У изради пројекта су коришћени напредни ООП концепти:

- преклапање оператора
- наслеђивање.

У класи Flight је имплементирано преклапање оператора (`operator<<`, `operator>>`). Ово је урађено због лакшег писања излазне датотеке.

Кроз апстрактну класу Sort је имплементирано наслеђивање. Класе наследнице су HeapSort и SlectionSort. Ове класе садрже скоро исте функције и атрибуте, кључна разлика је сама имплементација одабраног алгоритма.

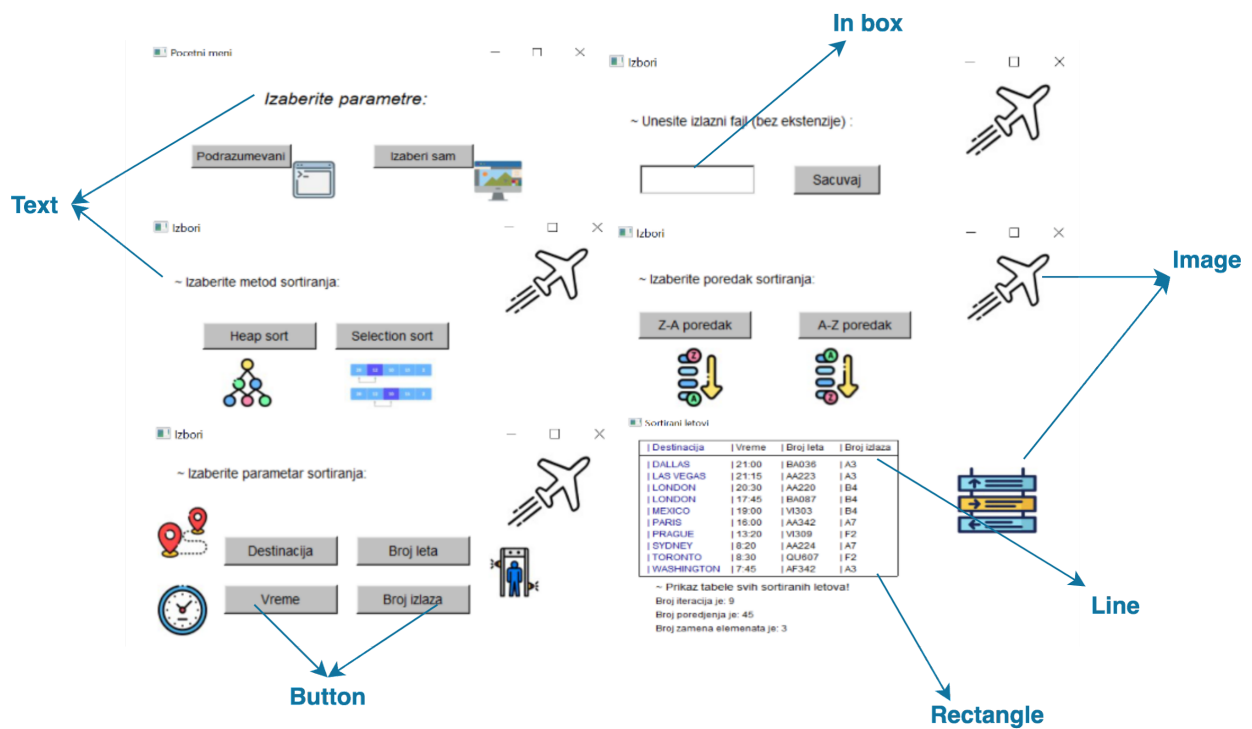
Елементи и функционалности графичке спреге

Класе које су коришћене приликом имплементације графичке корисничке спреге:

- **MyWindow** - кориснику се даје могућност да изабере подразумеване параметре или да их унесе сам.
- **ChooseWindow** - корисник уноси информације о указним и излазним путањама фајла, начину сортирања, поретку сортирања и параметру по ком жели да се елементи сортирају.
- **DisplayWondow** - се налази приказ сортираних летова, број итерација, број поређења, број померања и приказ табеле сортираног елемената кроз итерације.
- **PointWindow** -се налази тачкасти приказ сортираних летова.

Приликом имплементације ових прозора коришћене су следеће структуре из Graph_lib:

- **Text** - коришћен за исписивање упита и информација
- **Button** - коришћен за прикупљање потребних података за сортирање од корисника, за следећу итерацију и повратак
- **In box** - коришћен за унос имена улазне и излазне датотеке
- **Image** - коришћен за постављање слика
- **Rectangle** - коришћен приликом оивичавања табеле приказаних свих летова
- **Line** - коришћен за исцртавање хоризонталне и вертикалне линије како би се подаци табеларно приказали



Тестни случајеви

Табела улазних података:

| Destination | Departure | Flight No | Gate No |
|-------------|-----------|-----------|---------|
| LAS VEGAS | 21:15 | AA223 | A3 |
| DELLAS | 21:00 | BA036 | A3 |
| LONDON | 20:30 | AA220 | B4 |
| MEXICO | 19:00 | VI303 | B4 |
| LONDON | 17:45 | BA087 | B4 |
| PARIS | 16:00 | AA342 | A7 |
| PRAGUE | 13:20 | VI309 | F2 |
| TORONTO | 8:30 | QU607 | F2 |
| SYDNEY | 8:20 | AA224 | A7 |
| WASHINGTON | 7:45 | AF342 | A3 |

~Selection Sort

Графичко кориснички приказ након завршетка сортирања:

Sortirani letovi

| Destinacija | Vreme | Broj leta | Broj izlaza |
|-------------|-------|-----------|-------------|
| DALLAS | 21:00 | BA036 | A3 |
| LAS VEGAS | 21:15 | AA223 | A3 |
| LONDON | 20:30 | AA220 | B4 |
| LONDON | 17:45 | BA087 | B4 |
| MEXICO | 19:00 | VI303 | B4 |
| PARIS | 16:00 | AA342 | A7 |
| PRAGUE | 13:20 | VI309 | F2 |
| SYDNEY | 8:20 | AA224 | A7 |
| TORONTO | 8:30 | QU607 | F2 |
| WASHINGTON | 7:45 | AF342 | A3 |

Broj iteracija je: 9
Broj poređenja je: 45
Broj zamena elemenata je: 3

~ Prikaz tabele svih sortiranih letova!

LAS VEGAS

DALLAS

DALLAS

DALLAS

DALLAS

DALLAS

DALLAS

DALLAS

DALLAS

DALLAS

LAS VEGAS

LAS VEGAS

LONDON

LONDON

LONDON

LONDON

LONDON

LONDON

LONDON

LONDON

LONDON

MEXICO

MEXICO

MEXICO

MEXICO

LONDON

LONDON

LONDON

LONDON

LONDON

MEXICO

MEXICO

MEXICO

MEXICO

LONDON

LONDON

LONDON

LONDON

LONDON

PARIS

PARIS

PARIS

PARIS

PARIS

PARIS

PARIS

PARIS

PARIS

PARIS

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

PRAGUE

TORONTO

TORONTO

TORONTO

TORONTO

TORONTO

TORONTO

TORONTO

TORONTO

TORONTO

SYDNEY

SYDNEY

SYDNEY

SYDNEY

SYDNEY

SYDNEY

SYDNEY

SYDNEY

SYDNEY

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

WASHINGTON

Tackasti prikaz

Sortiraj opet

Sledeca iteracija

Završi

Табеларни приказ:

| | | | | | | | | | |
|------------------------|---|---|---|---|---|---|---|---|---|
| Број итерација | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Број поређења | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Број замене елементима | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

Тачкасти приказ сортирања:

Tackasti prikaz

LAS VEGAS
DALLAS
LONDON
MEXICO
LONDON
PARIS
PRAGUE
TORONTO
SYDNEY
WASHINGTON

DALLAS
LAS VEGAS
LONDON
LONDON
MEXICO
PARIS
PRAGUE
SYDNEY
TORONTO
WASHINGTON

Završi

~Heap Sort

Графичко кориснички приказ након завршетка сортирања:

Sortirani letovi

| Destinacija | Vreme | Broj leta | Broj izlaza |
|-------------|-------|-----------|-------------|
| DALLAS | 21:00 | BA036 | A3 |
| LAS VEGAS | 21:15 | AA223 | A3 |
| LONDON | 20:30 | AA220 | B4 |
| LONDON | 17:45 | BA087 | B4 |
| MEXICO | 19:00 | VI303 | B4 |
| PARIS | 16:00 | AA342 | A7 |
| PRAGUE | 13:20 | VI309 | F2 |
| SYDNEY | 8:20 | AA224 | A7 |
| TORONTO | 8:30 | QU607 | F2 |
| WASHINGTON | 7:45 | AF342 | A3 |

Broj iteracija je: 9
Broj poredjenja je: 46
Broj zamena elemenata je: 23

~ Prikaz tabele svih sortiranih letova!

| | | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| WASHINGTON | TORONTO | SYDNEY | PRAGUE | PARIS | MEXICO | LONDON | LONDON | LAS VEGAS | DALLAS |
| TORONTO | SYDNEY | MEXICO | MEXICO | MEXICO | LONDON | LAS VEGAS | LAS VEGAS | DALLAS | LAS VEGAS |
| PRAGUE | PRAGUE | PRAGUE | PARIS | LONDON | LONDON | LONDON | DALLAS | LONDON | LONDON |
| SYDNEY | MEXICO | LAS VEGAS | LAS VEGAS | LAS VEGAS | LAS VEGAS | DALLAS | LONDON | LONDON | LONDON |
| LONDON | LONDON | LONDON | DALLAS | DALLAS | PARIS | MEXICO | MEXICO | MEXICO | MEXICO |
| PARIS | PARIS | PARIS | DALLAS | DALLAS | PARIS | PARIS | PARIS | PARIS | PARIS |
| LONDON | LONDON | LONDON | LONDON | PRAGUE | PRAGUE | PRAGUE | PRAGUE | PRAGUE | PRAGUE |
| MEXICO | DALLAS | DALLAS | SYDNEY | SYDNEY | SYDNEY | SYDNEY | SYDNEY | SYDNEY | SYDNEY |
| LAS VEGAS | LAS VEGAS | TORONTO | TORONTO | TORONTO | TORONTO | TORONTO | TORONTO | TORONTO | TORONTO |
| DALLAS | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON | WASHINGTON |

Tackasti prikaz

Sortiraj opet

Završi

Табеларни приказ:

| | | | | | | | | | |
|------------------------|---|---|---|---|---|---|---|---|---|
| Број итерација | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Број поређења | 8 | 6 | 4 | 4 | 6 | 6 | 4 | 4 | 2 |
| Број замене елементима | 4 | 3 | 3 | 2 | 3 | 3 | 2 | 2 | 1 |

~Предности и мане

Кључна разлика између ова два алгоритма је време у коме се извршавају:

Selection sort -----> $O(n^2)$

Heap sort -----> $O(n \log n)$

Selection sort :

- додавање n елемената у ред са приоритетом који је имплементиран помоћу несортиране листе - време: $O(n)$
- уклањање n елемената у сортираном редоследу - време: $O(n)$

Heap sort:

- имамо n додавања у heap и n уклањања из heap-а - време: $O(n)$
- додавање и уклањање (downheap, unheap) - време: $O(n \log n)$

Из чега можемо закључити да је због времена које је потребно за извршавање Heap sort бољи.



Проблеми и ограничења

Пошто смо претпоставили да је улазна датотека дефинисана без грешке наш програм неће бити отпоран на грешке приликом лошег дефинисања датотеке и тиме ограничили наш програм. Проблем невалидног уноса улазне датотеке је делимично решен тиме што смо узели име улазне датотеке који је дефинисан у аргументима командне линије. Самим тим кориснику неће бити дата могућност да поново унесе име улазне датотеке.

Следећи проблем који је уочен са heap sort-ом јесте његова визуализација. Приликом табеларног приказа није лако уочити шта су родитељи, а шта деца у стаблу и због тога редослед не изгледа логично. То би се побољшало уколико би се та листа приказала као стабло. Такође, још један проблем је одређивање броја итерација, јер се на више начина може одредити шта представља итерацију у heap-у.