ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 2

2019-2020

- трећи домаћи задатак -

Опште напомене:

1. Домаћи задатак 3 састоји се од једног програмског проблема. Студенти проблем решавају **самостално**, на програмском језику C++.
2. Реализовани програми треба да комуницирају са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
3. Унос података треба омогућити било путем читања са стандардног улаза, било путем читања из датотеке.
4. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
5. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. **Примена рекурзије се неће признати као успешно решење проблема.**
6. За све недовољно јасне захтеве у задатку, студенти треба да усвоје разумну претпоставку у вези реализације програма. Приликом одбране, демонстраторе треба обавестити која претпоставка је усвојена (или које претпоставке су усвојене) и која су ограничења програма (на пример, максимална димензија низа и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
7. Одбрана трећег домаћег задатка ће се обавити у **среду, 25.12.2019. и четвртак, 26.12.2019.** према распореду који ће накнадно бити објављен на сајту предмета.
8. Пре одбране, сви студенти раде тест знања за рачунаром коришћењем система Moodle (http://elearning.rcub.bg.ac.rs/moodle/). **Сви студенти треба да се пријаве на курс пре почетка лабораторијских вежби**. Пријава на курс ће бити прихваћена и важећа само уколико је студент регистрован на систем путем свог налога електронске поште на серверу mail.student.etf.bg.ac.rs.
9. Име датотеке која се предаје мора бити **dz3p1.cpp**
10. Формула за редни број **i** и **j** комбинације проблема коју треба користити приликом решавања задатка је следећа:

    (**R** – редни број индекса, **G** – последње две цифре године уписа):

    i = (R + G) mod 5

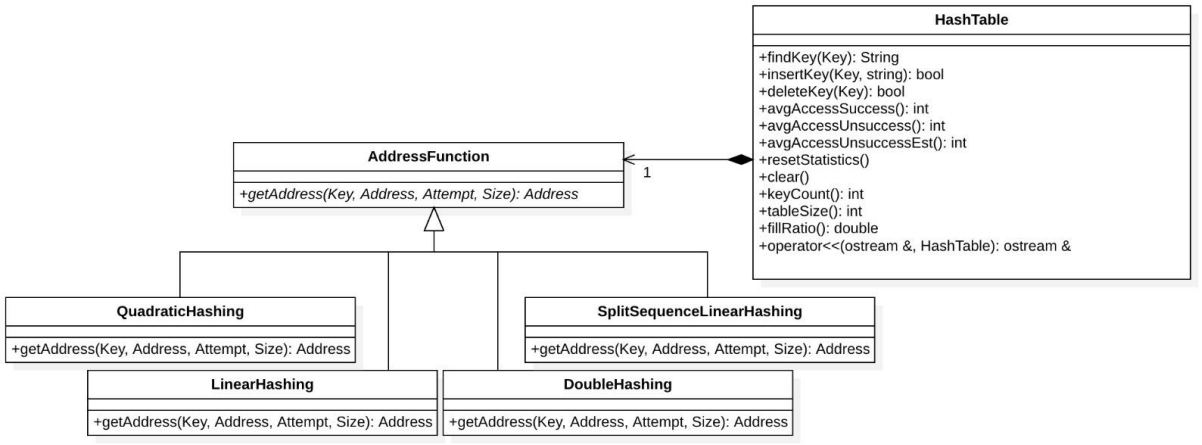
    j = (R + G) mod 4
11. Предаја домаћих ће бити омогућена преко *Moodle* система до **среде, 25.12.2019. у 7:00 часова**. Детаљније информације ће бити благовремено објављене.
12. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака.

17.12.2019. године *Са предмета*

Задатак 1 – Хеш табела [100 поена]

Написати на језику C++ потребне класе за реализацију хеш табеле у коју се умећу објекти који садрже податке о филмовима (*original_language*, *original_title*, *release_date*, *revenue*, *runtime*). Подаци су индексирани (хеширани) на основу назива филма (*original_title*) који је потребно користити као кључ за приступ табели. За разрешавање колизије се примењује техника отвореног адресирања. Пример CSV датотеке која садржи податке о нешто мање од 5000 филмова је дат у прилогу овог документа.

Концептуални дијаграм класа које треба имплементирати је приказан на следећој слици:



Класа која представља апстракцију адресне функције коју ће хеш табела користити за разрешавање колизије се задаје објекту хеш табеле приликом њеног стварања.

[20 поена] Имплементација хеш функције

За потребе мапирања кључева у хеш табелу, потребно је имплементирати одговарајућу хеш функцију. Уколико је потребно, стринг који представља кључ је могуће претворити у нумеричку вредност на начин који осмисли студент.

У зависности од редног броја проблема који се решава **i**, реализовати следећу методу за хеширање кључева:

0. Метод множења
1. Метод склапања
2. Метод конверзије основе
3. Метод средине квадрата
4. Метод анализе знакова (само за првих 5 знакова кључа)

[30 поена] Имплементација хеш табеле

Величина хеш табеле се задаје приликом креирања табеле и не мења се током извршавања. Приликом уметања кључа, додељена адресна функција се позива само ако је матична адреса заузета. Класа **`HashTable`** треба да реализује следеће јавне методе:

- **`string findKey(Key k)`** – проналази задати кључ и враћа показивач на одговарајућу информациони садржај (0 ако се кључ не налази у табели)
- **`bool insertKey(Key k, Info i)`** – умеће кључ и пратећи информациони садржај у табелу и враћа статус (***true*** за успешно уметање, ***false*** за неуспешно). Спречити уметање постојећег кључа.

- **`bool deleteKey(Key k)`** – брише кључ из табеле и враћа статус успеха (***true*** за успешно брисање, ***false*** за неуспешно). Брисање реализовати коришћењем технике полуслободних локација.
- **`int avgAccessSuccess()`** – враћа просечан број приступа табели приликом успешног тражења кључева
- **`int avgAccessUnsuccess()`** – враћа просечан број приступа табели приликом неуспешног тражења кључа израчунат на основу броја (до тог тренутка) неуспешних приступа табели и броја кључева који нису нађени у табели
- **`int avgAccessUnsuccessEst()`** – враћа просечан број приступа табели приликом неуспешног тражења кључа, добијен на основу процене базиране на тренутном степену попуњености табеле
- **`void resetStatistics()`** – поставља све податке потребне за бројање приступа ради одређивања просечног броја приступа за неуспешно тражење кључа на почетну вредност
- **`void clear()`** – празни табелу (брише све кључеве)
- **`int keyCount()`** – враћа број уметнутих кључева
- **`int tableSize()`** – враћа величину табеле
- **`operator<<`** – испис садржаја табеле на стандардни излаз, у сваком реду по један улаз табеле. Празне улазе табеле означити са "**`EMPTY`**". Полуслободне улазе у табели означити са "**`DELETED`**".
- **`double fillRatio()`** – враћа степен попуњености табеле (реалан број између 0 и 1)

Исправна реализација хеш табеле подразумева да, поред наведених метода, постоје друге потребне методе (попут конструктора и деструктора). Студентима се препушта да у класу додају оне методе које сматрају потребним за успешну реализацију.

[30 поена] Имплементација апстрактне класе адресне функције и једне од метода отвореног адресирања

Апстрактна класа декларише јавну методу коју користи класа **`HashTable`** за одређивање наредне адресе приликом разрешавања колизије.

```
Address getAddress(Key k, Address a, Attempt i, Size n);
```

Параметри ове методе су:

`k` – кључ,

`a` – матична адреса,

`i` – редни број покушаја приступа,

`n` – величина хеш табеле.

Метода враћа нову, валидну адресу на којој треба потражити кључ (или локацију где га треба сместити). Изведене класе треба да конкретизују начин одређивања следеће адресе.

У зависности од редног броја проблема који се решава **j**, реализовати следећу методу за решавање колизија:

0. Линеарно претраживање
1. Линеарно претраживање са раздвојеном секвенцом
2. Квадратно претраживање
3. Двоструко хеширање

Класа за линеарно адресирање се параметризује кораком **s** тако да као резултат даје матичну адресу увећану за **i** · **s** као резултат према следећој формули:

$$\texttt{return\_address} = a + s \cdot i$$

Класа за линеарно адресирање са раздвојеном секвенцом се параметризује кораком **s1** и кораком **s2**. Када се за дати кључ **k2** утврди да је његова матична адреса већ заузета неким другим кључем **k1** онда се ова два кључа упореде. Ако је **k2 < k1** за корак у табели се узме константа **s1**, а ако је **k2 > k1** узме се константа **s2** различита од **s1**. Поступак се понавља у сваком кораку хеширања.

$$\texttt{return\_address} = a + s1 \cdot i$$

или

$$\texttt{return\_address} = a + s2 \cdot i$$

Класа за квадратно адресирање се параметризује коефицијентом **c**, а у **i** –том покушају враћа вредност према следећој формули:

$$\texttt{return\_address} = a + c \cdot i^2$$

Класа за двоструко хеширање се параметризује подацима **p** и **q** и враћа следећу вредност:

$$\texttt{return\_address} = a + i \cdot (q + (k \bmod p))$$

[20 поена] Главни програм и функција за тестирање

Функција за тестирање умеће у хеш табелу задате објекте, а затим на основу скупа кључева за претрагу врши претрагу на њих. Након тога исписује резултате (просечан број приступа при успешној претрази, процењен и израчунат број приступа при неуспешној претрази). Табела, скуп објеката који се умећу и скуп кључева за претрагу се задају као параметри функције.

Реализовати главни програм са једноставним интерактивним менијем који кориснику омогућава рад са јавним методама хеш табеле. Такође, главни програм треба да омогући учитавање објеката који се хеширају са стандардног улаза или задате датотеке и позивање описане функције за тестирање за реализовану варијанту хеширања.

Уз поставку задатка је доступна датотека која садржи око 5000 линија са подацима о објектима (филмовима) које треба хеширати која се може користити за тестирање решења, као и неколико тест датотека са кључевима без и са понављањима.