**Хеш-таблици**

*Задача:*

*Към проекта от упражнение 5 да се добави модул, който съдържа клас НashTable, който да реализира Хеш-таблица с отворено адресиране и с компоненти:*

* *Таблица на дескриптори, която е с 4 полета;*

*методи за:*

* *хеш-функция, която да връща индекса на лексическа единица (може да използвате сумиране на ASCII-кодовете на лексическа единица и побитово маскиране на сумата със 127, т.е. разумно е таблицата да бъде със 128 елемента);*
* *търсене на лексическа единица по идентификатор в хеш-таблица (съвет: методът да връща позицията (индекса) на лексическата единица, ако я намери или позицията на която би трябвало да се намира, ако не я намери);*
* *добавяне на лексическа единица;*
* *премахване на лексическа единица от таблицата;*
* *отпечатване на таблицата.*

*Да се актуализира програмата, като вместо обект от класа TТable, се използва обект от новия клас НashTable.*

***Упътване и указанния за работа:***

Хеш таблицата е [структура от данни](https://bg.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BE%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "Структура от данни), съдържаща [ключ](https://bg.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87" \o "Ключ) и [данни](https://bg.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D0%B8), която се характеризира с директен достъп до елементите, независимо от типа им. Елементите ѝ, подобно тези на други структури от данни използвани за търсене, се състоят от ключ и стойност. Ключът е уникален за разлика от стойността – не може да съществуват два елемента с един и същи ключ. Сложността на елементарните операции по ключ (търсене, вмъкване, изтриване и актуализиране) в общия случай е константна, което я прави изключително полезна. Данните в таблицата не са записани последователно, а на позиции, определени от хеш-функцията. Това означава, че ще има участъци от таблицата, в които няма данни, т.е. таблицата не е плътно и равномерно запълнена.

Колизия наричаме ситуация, при която два различни ключа връщат едно и също число за хеш-код.

*Потърсете информация по темата!*

Когато дефинирате методите на класа, имайте предвид че:

1. Прототипите на методите е добре да са същите, както в класа за последователно търсене, за да не се налага да променяте главната функция.
2. Ключ в хеш-таблицата да бъде индексът на обекта от масива, а стойност – самия обект.
3. Добавяне на лексическа единица ( ако това е възможно), на позиция, определена от метода за търсене.
4. При премахване на лексическа единица (ако това е възможно) от таблицата, на съответната позиция полетата от тип char на обекта да бъдат празни стрингове.
5. Методът за търсене е удачно да връща позицията (индекса) на лексическата единица, ако я намери или позицията на която би трябвало да се намира (първата свободна), ако не я намери. Методът ще започва да търси в таблицата от позицията, която хеш-функцията определи до края и от началото до позицията от която е започнал.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ключ  (индекс) | Стойност  (обект) |
|  |  | 0 | lu (alfa) |
|  |  | 1 |  |
|  |  | 2 | lu (gama) |
|  |  | ... |  |
|  |  | 20 |  |
|  |  | ... |  |
| lu(beta) | -> HashFunction(identificator)-> 45 | 45 | lu(beta) |
|  |  | 46 | lu(delta) |
| lu(tabe) | -> HashFunction(identificator)-> 45 | 47 | lu(tabe) |
|  |  |  |  |
|  |  | 127 |  |

На примера е показана ситуация на колизия. При опит да се добави лексическа единица с идентификатор tabe, хеш-функцията ще върне сойност за ключ = 45, но тази позиция вече е заета от лексическа единица с идентификатор beta. Функцията за търсене трябва да върне като резултат -47, т.е. лексическа единица с идентификатор tabe може да се добави (-47<0) на първата свободна позиция (47).

При опит да се премахне лексическа единица с идентификатор tabe, хеш-функцията ще върне сойност за ключ = 45, но тази позиция вече е заета от лексическа единица с идентификатор beta. Затова търсенето отново продължава до първата свободна позиция. Ако търсената лексическа единица е в таблицата, функцията ще я открие и върне 47.

Възможно е таблицата да е запълнена в края, а в началото да има свободни места (където да попадат колизии), затова задължително трябва таблицата да се обхожда от позицията, която хеш-функцията определи до края и от началото до позицията от която сме започнали.

Успешна работа.