

# Мапе и скупови

Љубомир Бановић

Приликом писања програма, програмери се често налазе у дилеми приликом одабира одговарајућих структура података које ће најефикасније решити задати проблем.<sup>1</sup> У овом тексту посветићемо пажњу структурама података званим **мапе** и **скупови**. Мапе и скупови су структуре података које се често користе и представљају део стандардне библиотеке у свим модерним програмским језицима, те постоје и у стандардној библиотеци програмског језика C++.

Мапа је структура података која чува парове кључева и вредности, при чему не постоје два иста кључа и сваки кључ има тачно једну вредност. Неки конкретан пример употребе овакве структуре података би био телефонски именик, где би кључеви били имена људи, а вредности њихови бројеви телефона. У наставку следи пример употребе мапе у програмском језику C++:

```
#include <iostream>
//библиотека која садржи декларацију и дефиницију мапе
#include <map>

using namespace std;

int main() {
    //декларација мапе чији су кључеви стрингови, а вредности целобројни бројеви
    //кључеви представљају имена људи, а вредности њихове бројеве телефона
    map<string, int> brojeviTelefona;

    //кључеви "Pera" и "Mika" се убацују у мапу са одговарајућим вредностима
    brojeviTelefona["Pera"] = 123456;
    brojeviTelefona["Mika"] = 987654;

    //приступ вредностима које одговарају кључевима "Pera" и "Mika"
    cout << "Perin broj telefona je: " << brojeviTelefona["Pera"] << endl;
    cout << "Mikin broj telefona je: " << brojeviTelefona["Mika"] << endl;

    //промена вредности која одговара кључу "Pera"
    brojeviTelefona["Pera"] = 111111;
    cout << "Promenio sam Perin broj telefona, sada je: " << brojeviTelefona["Pera"] << endl;

    //брисање кључа "Mika" из мапе
    brojeviTelefona.erase("Mika");

    //провера да ли кључ "Mika" постоји у мапи
    //метода count враћа број појава кључа у мапи (а то су 0 или 1)
    //ако је резултат методе count 0, кључ не постоји у мапи, а ако је вредност 1 кључ постоји
    if(brojeviTelefona.count("Mika") == 0) {
        cout << "Mikin broj telefona ne postoji u imeniku." << endl;
    }

    //испис броја кључева у мапи
    cout << "Ukupan broj unetih brojeva telefona je: " << brojeviTelefona.size() << endl;

    //кључеви "Zika", "Laza" и "Ceda" се уносе у мапу са одговарајућим вредностима
    brojeviTelefona["Zika"] = 555555;
    brojeviTelefona["Laza"] = 444444;
    brojeviTelefona["Ceda"] = 333333;

    //итерација кроз све парове кључ-вредност у мапи, ова синтакса је подржана од верзије C++17
    for(auto [ime, broj] : brojeviTelefona) {
        cout << ime << ": " << broj << endl;
    }
}
```

<sup>1</sup>Структуре података су начини чувања и организовања података у меморији тако да се ти подаци могу ефикасно користити.

Временска сложеност операција убацивања, приступа и брисања кључева унутар `std::map` је  $O(\log n)$ , где је  $n$  број кључева у мапи. Потребно је напоменути чест баг приликом коришћења `std::map`, а он је да приступ вредности преко кључа, који не постоји у мапи, узрокује додавање тог кључа у мапу са подразумеваном вредношћу за тај тип података. Препорука је да се пре приступа вредности преко кључа провери да ли тај кључ постоји у мапи помоћу методе `count`.

```
#include <iostream>
#include <map>

using namespace std;

int main() {
    map<string, int> brojeviTelefona;

    //операција у следећој линији додаје кључ "ljuba" у мапу са вредношћу 0
    cout << brojeviTelefona["ljuba"] << endl;

    //ова линија исписује 1, јер је кључ "ljuba" додат у претходној линији
    cout << brojeviTelefona.size() << endl;
}
```

Скуп је структура података која чува јединствене елементе (не постоје два иста елемента у истом скупу). Имплементација скупа у стандардној библиотеци програмског језика C++ је `std::set`, која се налази у заглављу `<set>`. У наставку следи пример употребе скупа у програмском језику C++:

```
#include <iostream>
//библиотека која садржи декларацију и дефиницију скупа
#include <set>

using namespace std;

int main() {
    set<int> skup;

    //Уметање елемената 1, 28 и 5 у скуп
    skup.insert(1);
    skup.insert(28);
    skup.insert(5);

    //Елемент 28 се неће уметнути поново у скуп, јер већ постоји елемент 28 у скупу
    skup.insert(28);

    //Испис броја елемената унутар скупа
    cout << "Broj elemenata unutar skupa je: " << skup.size() << endl;

    //Брисање елемента 5 из скупа
    skup.erase(5);

    cout << "Broj elemenata unutar skupa je: " << skup.size() << endl;

    //провера да ли елемент 67 постоји у скупу
    //метода count враћа број појава елемента у скупу
    //ако је резултат методе count 0, елемент не постоји у скупу, а ако је вредност 1 елемент постоји
    if(skup.count(67) == 0) {
        cout << "Element 67 ne postoji u skupu." << endl;
    }

    cout << "Najmanji element skupa je: " << *skup.begin() << endl;

    cout << "Najveci element skupa je: " << *skup.rbegin() << endl;

    skup.insert(-3);
    skup.insert(9);
}
```

```

skup.insert(11);

cout << "Elementi skupa su: " << endl;

//Итерација кроз све елементе скупа
for(auto z : skup) {
    cout << z << endl;
}
}

```

Слично као и код мапа, временска сложеност операција уметања и елемената унутар `std::set` је  $O(\log n)$ , где је  $n$  број елемената у скупу.

Скупове је могуће имплементирати користећи мапе. Наиме, ако желимо да користимо скуп типа  $T$ , можемо да користимо мапу чији су кључеви типа  $T$ , а тип вредности је `bool`. Вредност за одређени кључ означава да ли тај елемент постоји у скупу или не.

## Задаци

1. Дата је ниска  $s$  дужине  $n$ , састављена од малих слова енглеске абеле. Потребно је утврдити да ли је могуће премештањем њених слова добити палиндром. Палиндром је рећ која се исто чита са леве на десно и са десне на лево.

### Пример 1:

Улаз: `s = "anavolimilovana"`

Израз: `da`

Објашњење: Ниска  $s$  је већ палиндром.

### Пример 2:

Улаз: `s = "abab"`

Израз: `da`

Објашњење: Премештањем слова ниске  $s$  могуће је добити ниске `"abba"` и `"baab"` који су палиндроми.

### Пример 3:

Улаз: `s = "ххуууухауа"`

Израз: `ne`

2. Дат је низ  $a$  дужине  $n$ . Потребно је одредити број различитих елемената у том низу.

### Пример 1:

Улаз: `a = [2, 8, 1, 2, 6]`

Израз: `4`

Објашњење: број различитих елемената у низу је 4 и ти елементи су: `{1, 2, 6, 8}`.

### Пример 2:

Улаз: `a = [28, 28, 28]`

Израз: `1`

3. Дат је низ  $a$  дужине  $n$  који се састоји од ненегативних целих бројева. Потребно је одредити минимални ексклузант овог низа - МЕХ. Минимални ексклузант је најмањи ненегативни цели број који не припада низу.

### Пример 1:

Улаз: `a = [1, 2, 0, 13, 4]`

Израз: `3`

Објашњење: 0 припада низу, 1 припада низу, 2 припада низу, али 3 не припада. Дакле, минимални ексклузант је 3.

### Пример 2:

Улаз: `a = [1, 2, 3, 4, 5]`

Израз: `0`

Објашњење: 0 је најмањи број који не припада низу.

4. Дат је низ  $a$  дужине  $n$  и број  $x$ . Потребно је одредити број парова индекса  $(i, j)$ , где је  $i < j$ , таквих да је збир елемената на тим индексима једнак броју  $x$  (тј.  $a[i] + a[j] = x$ ).

**Пример 1:**

Улаз:  $a = [2, 1, 3, 2, 4, 3]$ ,  $x = 5$

Излаз: 5

Објашњење: У питању су парови индекса (индексирани од 1):  $(1, 3)$ ,  $(1, 6)$ ,  $(2, 5)$ ,  $(3, 4)$ ,  $(4, 6)$ .

**Пример 2:**

Улаз:  $a = [2, 2, 2, 2, 2]$ ,  $x = 4$

Излаз: 10

Објашњење: Сваки могући пар индекса је решење.

5. Дат је низ  $a$  дужине  $n$  и број  $x$ . Потребно је одредити број поднихова  $[a_l, a_{l+1}, \dots, a_r]$  чијим је збир елемената једнак  $x$ .

**Пример 1:**

Улаз:  $a = [1, 2, 3]$ ,  $x = 3$

Излаз: 2

Објашњење: У питању су поднихови:  $[1, 2]$  и  $[3]$ .

**Пример 2:**

Улаз:  $a = [1, -1, 2, 3, -2]$ ,  $x = 3$

Излаз: 3

Објашњење: У питању су поднихови:  $[1, -1, 2, 3, -2]$ ,  $[2, 3, -2]$  и  $[3]$ .