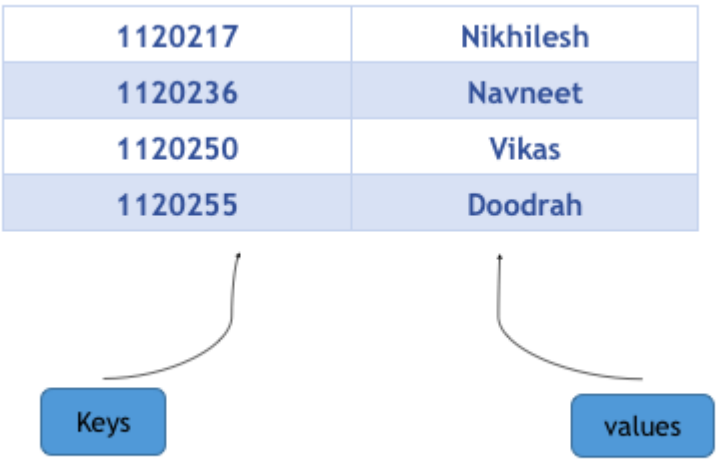
Примена на асоцијативен контејнер мапа (map) – час 3

Мапите се асоцијативни контејнери кои чуваат елементи на мапиран начин. Секој елемент има клучна вредност и мапирана вредност. Ниедни две мапирани вредности не може да имаат иста клучна вредност.

Основни функции поврзани со мапа се:
start () - Враќа итератор на првиот елемент на мапата
end () - враќа итератор на елементот што теоретски го следи последниот елемент на картата
size () - го враќа бројот на елементи во мапата
max_size () - Враќа максимален број елементи што мапата може да ги содржи
empty () - Враќа дали мапата е празна
pair insert(keyvalue, mapvalue) - Додава нов елемент на мапата
erase(iterator position) - го отстранува елементот на позицијата посочена од итераторот
erase (const g) - Ја отстранува клучната вредност ‘g 'од мапата
clear () - ги отстранува **сите елементи од мапата**

Пр. мапа на студенти каде клуч е борјот на индекс на студентот, а вредноста е името на студентот графички може да се претстави на следниов начин:



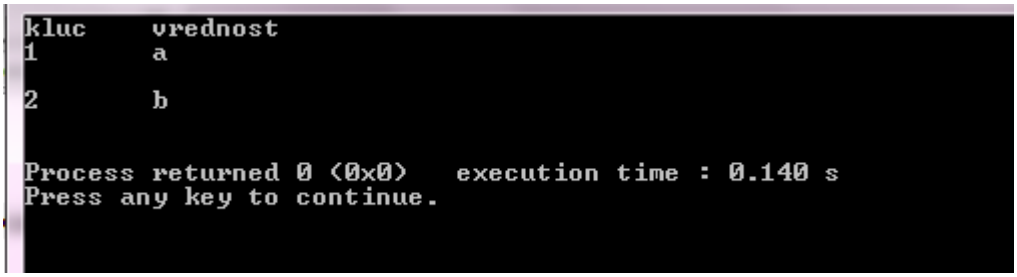
Може да се забележи дека клучевите се подредени по растечки редослед, бидејќи мапите секогаш ги подредуваат клучевите што ги содржат. Ако клучот е од тип string, се подредуваат по абецеда.

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main()
 {
   map<int, char> map1;
   map<int, char>::iterator c;
```

```
    map1[1]='a';
    map1[2]='b';
    cout <<"kluc\tvrednost"<<endl;
    for(c = map1.begin(); c!=map1.end(); c++)
    {
        cout<<c->first;
        cout<< '\t'<<c->second<<'\n'<<endl;
    }
}
```

Излезот е:



Предност на користењето на мапи:

Главна предност на користењето на контејнери за мапи во однос на другите структури на податоци како полиња е ако ги знаете клучевите во мапа контејнерот, тогаш може да го најдете посакуваниот клуч за многу кратко време.

Се избегнува пребарувањето низ цела листа како пример низ полињата и има мала временска сложеност.

Со голем број на елементи, полињата зафаќаат многу мемориски простор за да ги зачуваат мемориските адреси на елементите, но контејнерот мапа зазама само две места за да се зачуваат мамориските адреси.

Функции поврзани со контејнерот мапа:

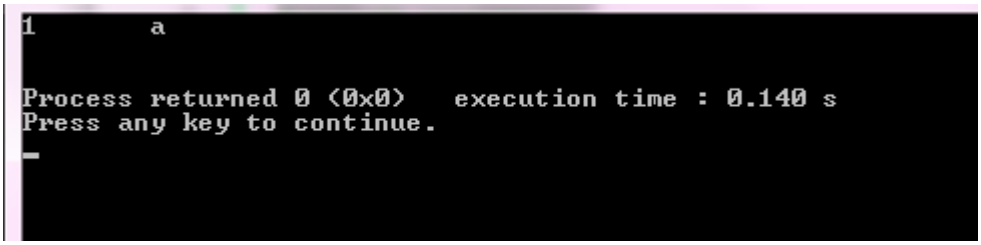
begin() – враќа двонасочен итератор кој покажува на првиот елемент од мапата. Функцијата нема параметри.

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main() {
```

```
    map<int, char> map1;
    map<int, char>::iterator c;
    map1[1]='a';
    map1[2]='b';
    c=map1.begin();
    cout<<c->first;
    cout<< '\t'<<c->second<<'\n'<<endl;
    return 0;
}
```

Излез:



end() -враќа двонасочен итератор кој покажува на елемент по последниот елемент од мапара. Нема влезен параметар. Покажува на навалиден елемент.

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main() {
    map<int, char> map1;
    map<int, char>::iterator c;
    map1[1]='a';
    map1[2]='k';
    map1.end();
    return 0;
}
```

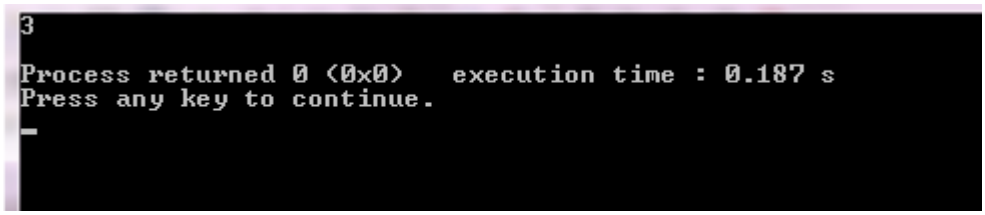
size() - враќа број на елементи на мапата. Важно е да се напомене дека клучната вредност и мапираната вредност се сметаат како единствен елемент.

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main() {
    map<int, char> map1;
    map1[1]='a';
    map1[2]='b';
```

```
    map1[3]='d';
    cout <<map1.size()<<endl;
    return 0;
}
```

Излез:

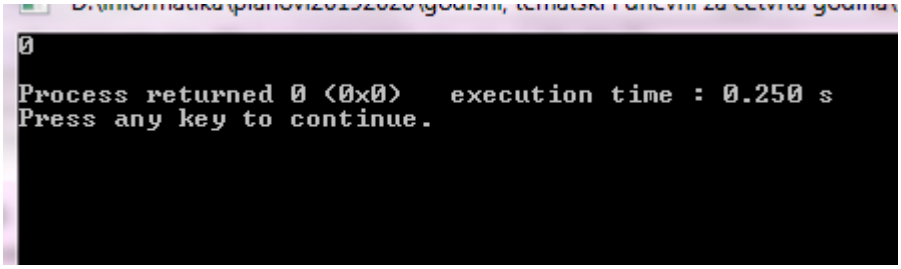


empty() – е логичка функција која се користи за проверка дали мапа контејнерот е празен или не е. Враќа вредност true (1) ако е празен, инаку враќа false (0).

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main() {
    map<int, char> map1;
    map1[1]='a';
    map1[2]='b';
    cout <<map1.empty()<<endl;
    return 0;
}
```

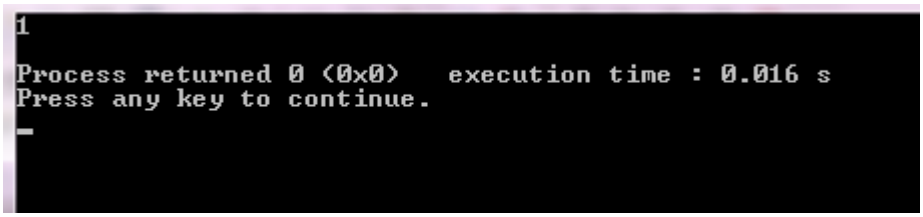
Излезот е:



Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main() {
    map<int, char> map1;
```

```
    cout <<map1.empty()<<endl;
    return 0;
}
```
Излезот е:



insert() – се користи за внесување на елемент во мапата. Потребен е клуч и вредност. Враќа двонасочен итератор покажувајќи на внесената вредност на клучот.

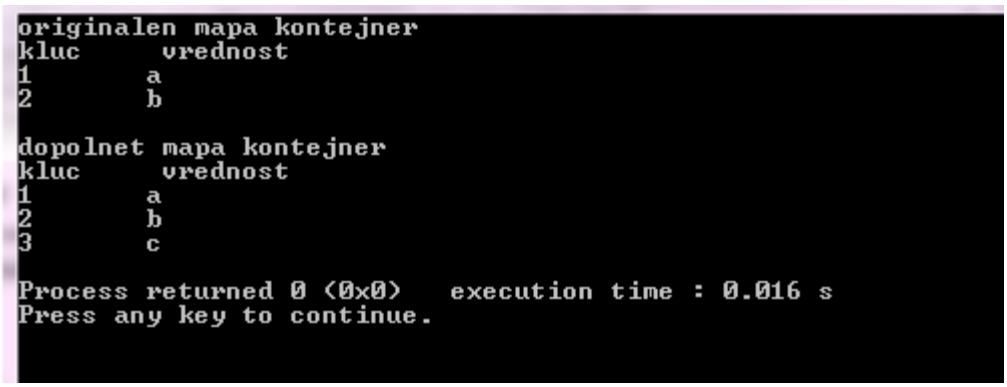
синтакса:

mapcontainer_name.insert({key_value, mapped_value});

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main()
{
    map<int, char> map1;
    map<int, char>::iterator c;
    map1[1] = 'a';
    map1[2] = 'b';
    cout << "originalen mapa kontejner" << endl;
    cout << "kluc\t vrednost" << endl;
    for (c = map1.begin(); c != map1.end(); c++)
    {
        cout << c->first;
        cout << '\t' << c->second<< endl;
    }
    map1.insert({3, 'c'});
    cout<<endl;
    cout << "dopolnet mapa kontejner" << endl;
    cout << "kluc\t vrednost" << endl;
    for (c = map1.begin(); c != map1.end(); c++)
    {
        cout << c->first;
        cout << '\t' << c->second <<endl;
    }
```

Излез:



erase() – се користи за отстранување на пар клуч – вредност од контејнетор на мапата. Ја зема вредноста на клучот или тековната позиција на итераторот како аргумент за да го избрише елементот. Враќа 1 ако елементот е најден и избришан, инаку 0.

Синтакса:

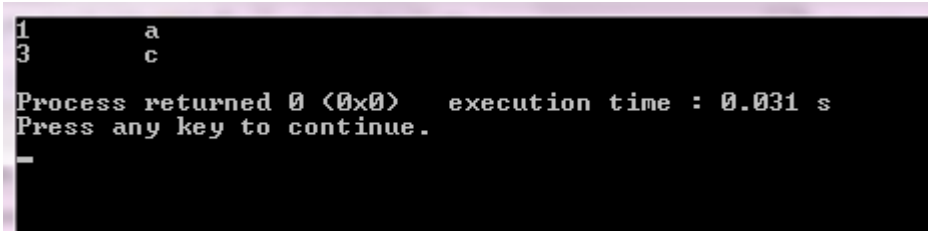
mapcontainer_name.erase(key_value);

или:

mapcontainer_name.erase(iterator_position);
Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main()
{
   map<int, char> map1;
   map<int, char>::iterator c;
   map1[1] = 'a';
   map1[2] = 'b';
   map1.insert({3, 'c'});
   map1.erase(2);
   for (c = map1.begin(); c != map1.end(); c++)
   {
      cout << c->first;
      cout << '\t' << c->second <<endl;
   }
}
```

Излезот е:



clear() – функцијата се користи за да избришат сите елементи од мапа контејнерот. Нема параметри и по нејзината имплементација големината на мапа контејнерот е 0.

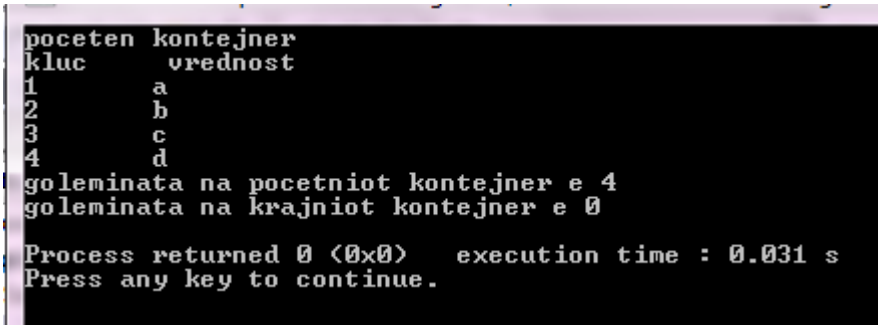
Синтакса:

mapcontainer_name.clear();

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main()
{
   map<int, char> map1;
   map<int, char>::iterator c;
   map1[1] = 'a';
   map1[2] = 'b';
   map1[3] = 'c';
   map1[4] = 'd';

   cout << "poceten kontejner" << endl;
   cout << "kluc\t vrednost" << endl;
   for (c = map1.begin(); c != map1.end(); c++)
   {
      cout << c->first;
      cout << '\t' << c->second << endl;
   }
   cout<<"goleminata na pocetniot kontejner e "<<map1.size()<<endl;
   map1.clear();
   cout<<"goleminata na krajniot kontejner e "<<map1.size()<<endl;
   return 0;
}
```

Излезот е:



max_size() – го враќа најголемиот број на елементи што контејнерот мапа може да го содржи.

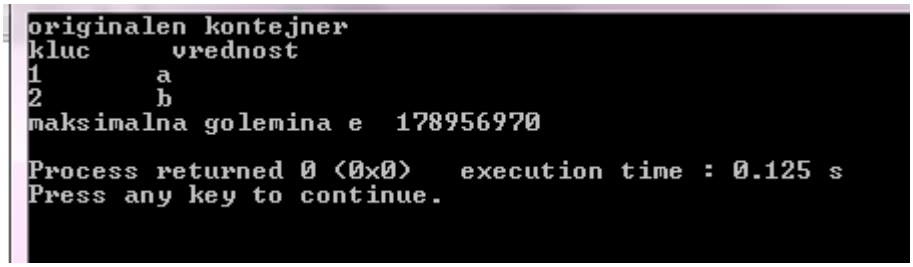
Синтакса:

mapcontainer_name.max_size();

Пр.

```
#include <iostream>
#include <iterator>
#include <map>
using namespace std;
int main()
{
   map<int, char> map1;
   map<int, char>::iterator c;
   map1[1] = 'a';
   map1[2] = 'b';
   cout << "originalen kontejner" << endl;
   cout << "kluc\t vrednost" << endl;
   for (c = map1.begin(); c != map1.end(); c++)
   {
      cout << c->first;
      cout << '\t' << c->second <<endl;
   }
   cout << "maksimalna golemina e  " << map1.max_size() << endl;
   return 0;
}
```

Излез:

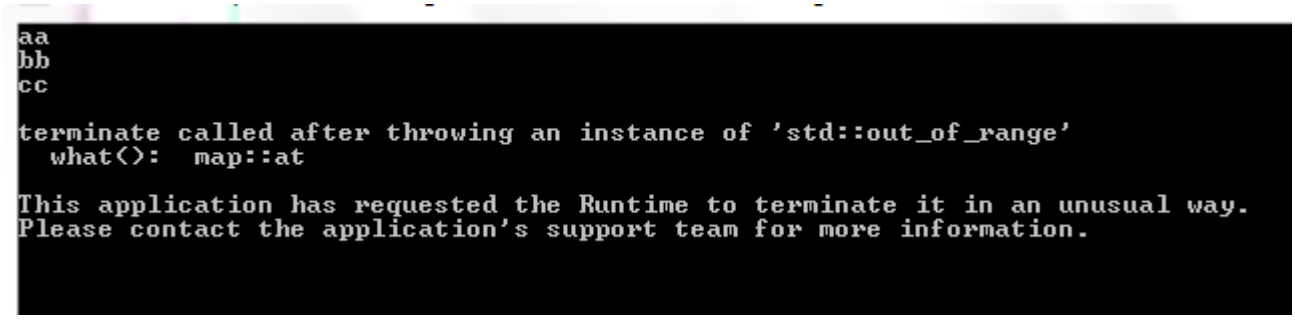


at и [ ] – се користат за пристап до елемент од мапата. Разликата е во тоа што at дава exception ако пристапниот клуч не е присутен во мапата, а операторот [ ] го внесува клучот во мапата ако тој не е присутен.

Пр.

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
   map<int,string> map1;
   map<int, string>::iterator c;
   map1[1] = "aa";
   map1[2] = "bb";
   map1[3] = "cc";
   cout << map1.at(1)<<endl;  // ja pecati vrednosta na kluc 1 t.e. aa
   cout << map1.at(2)<<endl;  // ja pecati vrednosta na kluc 2 t.e. aa

   // parametrite vo funkcijata at() se klucevi a ne indeksi

   cout << map1[3]<<endl; // ja pecati vrednosta na kluc 3 t.e. aa

   map1.at(1) = "ucime";   // ja menuva vrednosta na kluc 1 vo "ucime"
   map1[2] = "za";           // cja menuva vrednosta na kluc 2 vo "za"

   map1[4] = "kontejner";
   /* bidejki nema kluc so vrednost 4 vo mapata,
    go vnesuva parot so kluc 4 i vrednost "kontejner" vo mapata*/
     cout<<endl;
   map1.at(5) = "mapa";
   /* bidejki nema kluc so vrednost 4 vo mapata, throws exception*/
    return 0;
}
```

Излез:



swap() - функцијатта се користи за да ги замени вредностите на две мапи, но мапите мора да бидат од исти тип, а големината може да се разликува.

Пр.

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

using namespace std;

int main()
{
   // prazen mapa kontejner
   map<int, int> map1;
   // vnesuvanje na elementi -dodeluvanje par vrednosti vo mapata
   map1.insert(pair<int, int>(1, 40));
   map1.insert(pair<int, int>(2, 30));
   map1.insert(pair<int, int>(3, 60));
   map1.insert(pair<int, int>(4, 20));
   map1.insert(pair<int, int>(5, 50));
   map1.insert(pair<int, int>(6, 50));
   map1.insert(pair<int, int>(7, 10));

   // pecatenje na sodrzinata na map1
   map<int, int>::iterator itr;
   cout << "\n sodrzinata na map1 e :"<<endl;
   cout << "\tkluc\tvrednost"<<endl;
   for (itr = map1.begin(); itr != map1.end(); ++itr) {
      cout << '\t' << itr->first<< '\t' << itr->second <<endl;
   }
   cout << endl;
   // dodeluvanje na elementite od map1 na map2
   map<int, int> map2(map1.begin(), map1.end());
   // pecatenje na sodrzinata na map2
   cout << "\nsodrzinata na map2 e :"<<endl;
   cout << "\tkluc\tvrednost"<<endl;
   for (itr = map2.begin(); itr != map2.end(); ++itr) {
```

```
        cout << '\t' << itr->first<< '\t' << itr->second <<endl;
    }
    cout << endl;

    // brisenje na elementite so kluc pomal od 3 vo map2
    cout << "\n map2 po brisenje na elementite so kluc pomal od 3 se "<<endl;
    cout << "\tkluc\tvrednost"<<endl;
    map2.erase(map2.begin(), map2.find(3));
    for (itr = map2.begin(); itr != map2.end(); ++itr) {
        cout << '\t' << itr->first<< '\t' << itr->second <<endl;
    }
    cout << endl;

    // brisenje na elementite so kluc 4
    int num;
    num = map2.erase(4);
    cout << "\n brisenje na elementot so kluc 4 : ";
    cout << num << " e izbrisan"<<endl;
   cout << "\tkluc\tvrednost"<<endl;
    for (itr = map2.begin(); itr != map2.end(); ++itr) {
        cout << '\t' << itr->first<< '\t' << itr->second <<endl;
    }
    cout << endl;
    return 0;
}
```

Прашања поврзани со наставните единици може да се испраќаат на email:
anetastojceska@gmail.com