MÖVZU 14

❖ **Vector** (və ya **dinamik massiv**) – özündə elementlər saxlayan bir strukturdur. Vector-a elemenləri həm əlavə etmək mümkündür. Elə buna görə də, vector bir çox dərsliklərdə **dinamik massiv** adlanır. C++ -da vector -u istifadə etmək üçün **#include<vector>** kitabxanasını daxil etmək lazımdır. Vector -un əsas funksiyalarına nəzər yetirək:

Metod	Metodun açıqlaması
`void` **`resize`**`(n)`	vector-a n sayda sıfırlardan ibarət ölçünün verilməsi– **O(n)**
`void` **`push_back`**`(element)`	elementi vector-un sonuna əlavə edir – **O(1)**
`void` **`pop_back`**`(element)`	elementi vector-un sonundan silir – **O(1)**
`front``()`	vector-un əvvəlindəki elementi return edir – **O(1)**
`back``()`	vector-un sonundakı elementi return edir – **O(1)**
`void` **`clear`**`()`	vector-un bütün elementlərini silir – **O(n)**
`size``()`	vector-un ölçüsünü (element sayını) return edir – **O(1)**
`bool` **`empty`**`()`	vector boş olarsa true return edir – **O(1)**

Vector-un təyin olunması

➢ vector-u təyin edərkən **vector<data tip> ad**; şəklində qeyd olunur. Bu zaman vector boş olur:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> v;
}
```

➢ əgər vector-da əvvəlcədən yer ayırmaq istəsək, onda mötərizədə ölçüsünü təyin edə bilərik. Bu zaman vector -un elementləri sıfırlarla doldurulur:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> v(10);
}
```

Vector-un girişə verilməsi

- girişdə daxil edilən siyahını vector-a əlavə etmək üçün **push_back()** metodundan istifadə edilir:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> v;
    int n, x;
    cin>>n;
    for(int i = 0; i < n; i++){
        cin>>x;
        v.push_back(x);
    }
}
```

- əgər vector-un ölçüsünü əvvəlcədən təyin etsək, onda massivdəki kimi indekslə girişə verə bilərik:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    vector <int> v(n);
    for(int i = 0; i < n; i++){
        cin>>v[i];
    }
}
```

Vector-un çıxışa verilməsi

- vector-u çıxışa vermək üçün massivdəki kimi for dövründən istifadə edə bilərik. Bu zaman dövrü (**v.size()-1**) -ə qədər davam etdirməliyik:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> v = {5, 1, 8, 2, 7};

    for(int i = 0; i < v.size(); i++){
        cout<<v[i]<<" ";
    }
}
```

- həmçinin **for-each** dövrünü əks etdirən dövrdən də istifadə edə bilərik. Bu zaman indeks istifadə olunmur. Bu dövrü bütün strukturlara tətbiq etmək mümkündür:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> v = {5, 1, 8, 2, 7};
    for(int x : v){
        cout<<x<<" ";
    }
}
```

Vector pair<tip1, tip2>

- **vector<pair<tip1, tip2> >** strukturu dəyərləri iki-iki, yəni cütlük şəklində saxlamaq üçün istifadə olunur. Ümumiyyətlə pair ayrıcı bir strukturdur və biz bunu istənilən yerdə istifadə edə bilirik:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector<pair<int, int> > v;
    v.push_back({3, 4});
    v.push_back({8, 2});
    v.push_back({5, 5});
    for(int i = 0; i < v.size(); i++){
        cout<<v[i].first<<" "<<v[i].second<<endl;
    }
}
```

```
Çıxış:
3 4
8 2
5 5
```

- həmçinin **for-each** dövründən də istifadə edərək pair-ləri çıxışa verə bilərik:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    vector<pair<int, int> > v;
    v.push_back({3, 4});
    v.push_back({8, 2});
    v.push_back({5, 5});
    for(auto p : v){
        cout<<p.first<<" "<<p.second<<endl;
    }
}
```

vector STL funksiyaları

- v vector-nu artan sıra ilə sıralamaq: **sort(v.begin(), v.end())**
- v vector-nu tərs çevirmək: **reverse(v.begin(), v.end())**
- v vector-nu azalan sıra ilə sıralamaq: **sort(v.rbegin(), v.rend())**

Tapşırıqlar

- azeco.eolymp.space : 8 məsələ (nömrələri: 193-200)