Vektori

Objektno programiranje (C++) - 2. vježbe

dr. sc. Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

13. ožujka 2024. godine

Vektor

- kolekcija objekata istog tipa
- `vector` se često naziva i spremnik (jer sprema druge objekte)
- to nije klasa nego **predložak klase** (*class template*)
- predložak je uputa kompajleru kako dobiti klasu ili funkciju (u tzv. procesu instancijacije) - treba reći koju vrstu klase/funkcije želimo

$\Rightarrow$ za vektore treba navesti tip objekta koji spremamo unutra

Primjer 1.

Tri različita tipa generirana kompajlerom iz `vector` predloška:

- **`vector<int> brojevi;`**
- **`vector<Knjiga> knjige;`**
- **`vector<vector<string>> datoteka;`**

Odredite što sadrži svaki od vektora u gornjem primjeru!

> Neki kompajleri i dalje zahtijevaju stariju sintaksu:
> `vector<vector<string> > datoteka;  //razmak!`

Primjer 2. Inicijalizacija vektora

- **prazan vektor**: `vector<int> iv;`
- kopiranje vektora (svaki element novog vektora je kopija odgovarajućeg elementa iz originalnog vektora):
  `vector<int> iv2(iv);` ✓
  `vector<int> iv3 = iv;` ✓
  `vector<string> sv(iv2);` ✗ `//nisu istog tipa`
- inicijalizacija listom:
  `vector<string> v1{"abc", "ef", "gh"};` ✓
  `vector<string> v2("abc", "ef", "gh");` ✗
- navođenje vrijednosti elementa i broja kopija tog elementa:
  `vector<string> v3(5,"a!");` ✓ `//5 stringova "a!"`
- *defaultna* inicijalizacija (navodimo samo broj elemenata):
  `vector<int> v4(10);` ✓ `//10 nula`
  `vector<string> v5(7);` ✓ `//7 praznih stringova`
  `vector<int> v6 = 7;` ✗ `//NE direktna inicijal.`
- u posljednjem načinu paziti da objekti imaju *defaultni* konstruktor!

Inicijalizacija: ( ) vs. { }

- ( ) → dane vrijednosti služe za konstrukciju objekta
- { } → inicijalizacija listom, no samo ako to nije moguće razmatraju se ostali načini inicijalizacije

Primjer 3.

- `vector<int> v1(10);` → deset elemenata (svaki 0)
- `vector<int> v2{10};` → jedan element (jednak 10)
- `vector<int> v3(10, 1);` → deset elemenata (svaki 1)
- `vector<int> v4{10, 1};` → dva elementa: 10 i 1
- `vector<string> v5{"ab"};` → jedan element `"ab"`
- `vector<string> v6("ab");` → greška
- `vector<string> v7{10};` → deset praznih stringova
- `vector<string> v8{10, "ab"};` → deset stringova `"ab"`

Dodavanje elementa na kraj vektora

- standard zahtjeva da implementacije vektora efikasno dodaju elemente tijekom izvršavanja programa

Primjer 4. Vektor svih učitanih riječi:

```
string rijec;
vector<string> tekst;    //prazni vektor
while (cin >> rijec) {
        tekst.push_back(rijec);
}
```

Primjer 5. Dodavanje u vektor kvadrata brojeva od 1 do 100:


```
vector<int> v;    //prazni vektor
for(int i = 1; i <= 100; ++i)
        v.push_back(i * i);
```

Inicijalizacija vektora pomoću polja - samostalno proučiti

- specificiramo adresu prvog elementa i „jednog iza” zadnjeg elementa koji kopiramo u vektor

Primjer 6.

```
int a[] = {0, 1, 2, 3, 4, 5};
vector<int> v(begin(a), end(a));
for(auto it = v.begin(); it != v.end(); ++it)
    cout << *it;
cout << endl;
```

- funkcije **`begin`** i **`end`** primaju polje i vraćaju pokazivač na prvi element i „prvi nakon zadnjeg” elementa

Pitanje. Što se ispiše ako drugi redak primjera izmijenimo u:

```
vector<int> v(a + 1, a + 4);
```


Primjer 7. Prolazak elementima vektora

```
vector<int> v{1,2,3,4,5,6,7,8,9,10};
for (auto &i : v)
        i *= i;
for (auto i : v)
        cout << i << " ";
cout << endl;
```

Zbog dinamičke prirode vektora, tijelo *range* petlje ne smije mijenjati veličinu vektora kojim se prolazi!

Primjer 8. Ostale operacije s vektorima

- `empty` i `size` su kao kod stringova
- operatore ==, !=, <, <=, > i >= koristimo za **usporedbu vektora**, pri čemu odgovarajuća operacija treba biti definirana na elementima

```
vector<int> v, w = {1,2,3};
if(v.empty())
      cout << "Prazan vektor!" << endl;
auto vel = w.size();
cout << "Broj elemenata: " << vel << endl;
v = {1,2};
v[1] = 3;
if(w < v)
      cout << "Vrijedi w < v." << endl;
```

Napomena: `vel` je tipa `vector<int>::size_type`, a ne `vector::size_type`

Zadatak 1.

Korisnik unosi brojeve između 0 i 100. Pojedini broj predstavlja broj bodova koje je neki student ostvario na nekom kolegiju. Konačna ocjena (između 1 i 5) dobiva se prema formuli

$$\max\left\{\left\lfloor\frac{\text{bodovi}}{15}\right\rfloor - 1,1\right\}.$$

Ispišite koliko je studenata dobilo koju ocjenu.

Primjer.

```
78 90 100 0 34 64 23 9 13 76 75 45 62 60
Ocjena 1: 5
Ocjena 2: 1
Ocjena 3: 3
Ocjena 4: 3
Ocjena 5: 2
```

Napomena: funkcija `std::max` dostupna je u `algorithm` biblioteci.

Pristupanje elementima spremnika: Iteratori

- stringovi i vektori - možemo koristiti **indekse** - općenitije: iteratori
- tipovi koji ih koriste imaju funkcije članice za njihovo dobivanje:
  - `begin` - daje iterator koji označava prvi element/znak
  - `end` - daje iterator koji označava „jedan iza zadnjeg”
  - za prazan spremnik, iteratori koje vraćaju su jednaki
- općenito ne znamo ili ne brinemo za točan tip iteratora
- **valjan** iterator označava element ili „jedan iza zadnjeg” elementa
- element dobivamo **dereferenciranjem** valjanog iteratora

Primjer 9.

```
string s("neki tekst");
if (s.begin() != s.end()) {  //== i != za iter.
     auto it = s.begin();
     *it = toupper(*it);
}
cout << s << endl;
```


Prelazak s trenutnog elementa na sljedeći/prethodni

- prelazak na sljedeći element - inkrementiranje iteratora
- prelazak na prethodni element - dekrementiranje iteratora
- iterator kojeg daje **end** ne označava element pa se on ne može ni inkrementirati ni dereferencirati

Primjer 10.

```
vector<int> v = {1,2,3,4};
vector<int>::iterator it;
for(it = v.begin(); it != v.end(); ++it)
       *it += 5;
```

- umjesto uokvirenog mogli napisati npr. `decltype(v.begin())`
- za string bismo koristili tip `string::iterator`

const_iterator i const vector

Primjer 11.

```
vector<int> v1 = {1,2,3,4};
vector<int>::const_iterator it;
for(it = v1.begin(); it != v1.end(); ++it)
        *it += 5; ✗
```

→ s `it` možemo samo čitati elemente!

Primjer 12.

```
const vector<int> v2 = {1,2,3,4};
vector<int>::iterator it;
for(it = v2.begin(); it != v2.end(); ++it)
        cout << *it;
```

- ne kompajlira se - uokvireni dio treba zamijeniti s `vector<int>::const_iterator`

Pitanja. Kojeg je tipa `it` ako ga definiramo ovako:

- `auto it = v1.begin();`
- `auto it = v2.begin();`
- `auto it = v1.cbegin();`

Primjer 13. Vektor struktura

- tip `video` za nazive videa i njihove preglede:

```
struct video {
        string naziv;
        int pregl;
};
```

- učitamo podatke o pet videa i dodamo ih u vektor videa:

```
vector<video> v;
video t;
for(int i = 0; i < 5; ++i) {
        cin >> t.naziv >> t.pregl;
        v.push_back(t);
}
```

- ispis pomoću iteratora:

```
for(auto it = v.begin(); it != v.end(); ++it) {
    cout << it->naziv << " ima " //(*it).naziv
         << it->pregl << " pregleda." << endl;
}
```

Aritmetika iteratora („pomicanje za 1 ili više”)

- `n` mjesta dalje/prije: `iter + n`, `iter - n`
- uz pridruživanje: `iter += n`, `iter -= n`
- uspoređivanje (označavaju li element prije/poslije): >, >=, <, <=
- `iter1 - iter2` (koliko treba dodati `iter2` za dobiti `iter1`), tzv. udaljenost iteratora (`difference_type` tipa s predznakom)
- u posljednjem iteratori moraju biti za elemente istog spremnika

Primjer 14. Binarno pretraživanje - samostalno proučiti:

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<string> tekst = {"adresa", "adut",
        "akcija", "aleja", "alka", "alt",
        "apel", "arak", "bajt", "bit", "bod"};
```


Nastavak koda iz primjera 30. - samostalno proučiti

```
    string rijec;
    cin >> rijec;
    auto l = tekst.begin(), d = tekst.end();
    auto s = l + (d - l)/2;
    while(s != d && *s != rijec) {
        if(rijec < *s)
            d = s;
        else
            l = s + 1;
        s = l + (d - l)/2;
    }
    if(*s != rijec)
        cout << "nismo ";
    cout << "nasli" << endl;
    return 0;
}
```

Zadatak 2.

Napišite program koji čita linije teksta sa standardnog ulaza ubacujući ih pritom jednu po jednu u vektor. Nakon unosa EOF (*end-of-file*) potrebno je ispisati sadržaj dobivenog vektora, ali tako da su sva prva slova u svim riječima učitanih stringova velika. Ako riječ ne počinje slovom tada ju ne mijenjamo.

Primjer. Ulaz:

```
Nigdar ni tak bilo   da ni nekak bilo,

  pak ni vezda nebu da nam nekak nebu.
```

Izlaz:

```
Nigdar Ni Tak Bilo   Da Ni Nekak Bilo,

  Pak Ni Vezda Nebu Da Nam Nekak Nebu.
```


Zadatak 3.

Napišite program koji čita stringove (riječi!) sa standardnog ulaza sve dok se ne učita EOF, te potom ispisuje koliko se puta pojedini string pojavio na ulazu.

Primjer.

```
jedan dva tri dva pet dva nula jedn
jedan f
   tri jedan
```

```
Ispis podataka:
jedan: 3
dva: 3
tri: 2
pet: 1
nula: 1
jedn: 1
f: 1
```