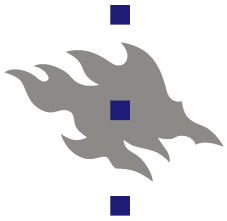


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI



C++-pikakurssi, osa 2/3

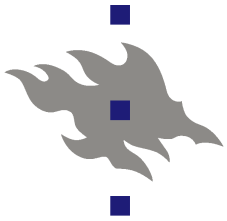
Algoritmit ongelmanratkaisussa, kevät 2018



Vektorit

```
vector<int> v;
v.push_back(1); // [1]
v.push_back(2); // [1,2]
v.push_back(3); // [1,2,3]
```

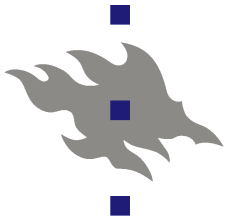
- `vector` on C++:n muuttuvan kokoinen taulukko (kuten Javan `ArrayList`)
- vektoria käyttäessä koodin alussa tulee lukea `#include <vector>`
- `push_back` lisää alkion taulukon loppuun



Vektorit

```
vector<int> v = {1,2,3};
cout << v.size() << "\n"; // 3
cout << v[0] << "\n"; // 1
```

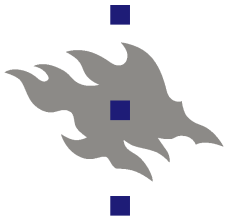
- vektorin määrittelyssä voi myös antaa sen alkusisällön
- `size` kertoo vektorin koon
- vektorin alkioihin voi viitata samalla tavalla kuin taulukossa



Vektorin käsittely

```
vector<int> v = {3,1,5};
v.push_back(4); // [3,1,5,4]
cout << v.back() << "\n"; // 4
v.pop_back(); // [3,1,5]
cout << v.back() << "\n"; // 5
```

- vektoria käsitellään usein sen *lopusta*:
  - `push_back` lisää alkion loppuun
  - `back` palauttaa alkion lopusta
  - `pop_back` poistaa alkion lopusta
- alkion lisääminen ja poistaminen toimivat keskimäärin $O(1)$-ajassa

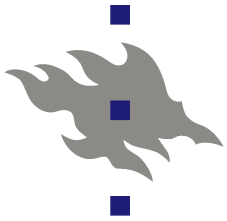


Vektorin läpikäynti

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

```
for (int x : v) cout << x << "\n";
```

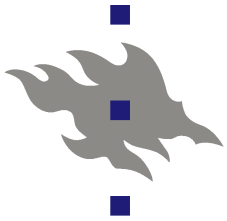
- vektorin voi käydä läpi for-silmukalla kahdella tavalla
- jälkimmäinen tapa on kätevä, jos ei tarvitse tietää alkion indeksiä



Vektori ja merkkijono

```
string s;
s.push_back('a');
s.push_back('b');
s.push_back('c');
s.pop_back();
cout << s << "\n"; // ab
```

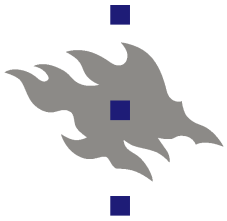
- vektori ja merkkijono toimivat hyvin samalla tavalla C++:ssa



Vektorin kopiointi

```
vector<int> a = {1,2,3};
vector<int> b = a;
a.push_back(4);
// a = [1,2,3,4], b = [1,2,3]
```

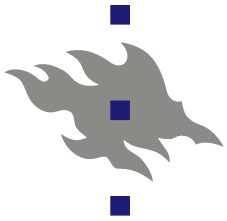
- C++:ssa vektorin sijoittaminen toiseen kopioi sen koko sisällön
- tämä on tärkeä ero verrattuna Javaan, jossa vain viittaus kopioidaan



Järjestäminen

```
vector<int> v = {5,2,4,1};
sort(v.begin(),v.end());
// v = [1,2,4,5]
```

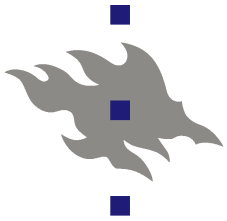
- funktio `sort` järjestää vektorin (tai muun tietorakenteen) sisällön
- koodin alussa tulee lukea `#include <algorithm>`
- `begin` ja `end` tarkoittavat, että tietorakenteen koko sisältö järjestetään



Muita funktioita

```
vector<int> v = {5,2,4,1};
reverse(v.begin(),v.end());
// [1,4,2,5]
random_shuffle(v.begin(),v.end());
// [2,4,5,1]
```

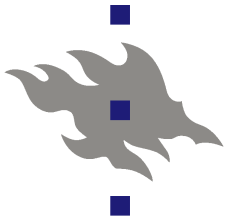
- `reverse` kääntää tietorakenteen sisällön
- `random_shuffle` sekoittaa tietorakenteen sisällön satunnaiseen järjestykseen



Parit

```
pair<int,int> x = {1,2}
cout << x.first << "\n"; // 1
cout << x.second << "\n"; // 2
```

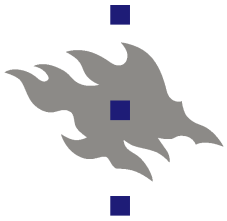
- pari sisältää kaksi muuttujaa yhdessä (niillä voi olla sama tai eri tyyppi)
- `first` viittaa ensimmäiseen muuttujaan ja `second` viittaa toiseen muuttujaan



Parien järjestäminen

```
vector<pair<int,int>> v;
v.push_back({1,3});
v.push_back({3,1});
v.push_back({1,2});
sort(v.begin(),v.end());
// [(1,2),(1,3),(3,1)]
```

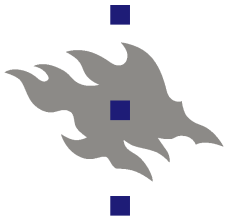
- pareista voi tehdä vektorin, jonka voi järjestää
- parit järjestetään ensisijaisesti ensimmäisen ja toissijaisesti toisen muuttujan mukaan



Merkkijonon järjestäminen

```
string s = "aybabtu";
sort(s.begin(),s.end());
// s = "aabbtuy"
```

- myös merkkijonon voi järjestää `sort`-funktion avulla
- tämä tarkoittaa, että merkit järjestetään aakkosjärjestykseen



Taulukon järjestäminen

```
int a[] = {5,2,4,1};
int n = 4; // taulukon koko
sort(a, a+n);
// a = [1,2,4,5]
```

- myös taulukon voi järjestää `sort`-funktion avulla
- tällöin funktiolle annetaan osoitin taulukon alkuun sekä sen viimeisen alkion jälkeen