1. géptermi gyakorlat – (ezazamaz + vektor)

Az STL `vector`ról

A `vector` a *Standard Template Library* (STL) egyik konténer-fajtája. Ez tehát egy *adatszerkezet*; több, azonos típusú adat összességéből készített összetett adattípus.

Bizonyos tulajdonságait tekintve olyan mint a tömb: a memóriában egy összefüggő területen helyezkedik el, ezáltal könnyen és hatékonyan lehet elemeit indexeléssel elérni. Van mérete, elemei azonos típusúak, azokat le lehet kérdezni és módosítani is lehet őket.

A méretét ellentétben a sima tömbbel, nem külön változóban tároljuk, hanem a `size()` *metódus*ával kérhetjük le. A méretet megadhatjuk a `vector` létrehozásakor, vagy akár később is. Ezen kívül minden a `vector` végére való beszúrás automatikusan eggyel növeli a `vector` méretét.

A többi STL típushoz hasonlóan a vektor is egy *sablontípus*. Azaz létrejöttekör típusparaméterként `<>`-ek között meg kell mondani, hogy milyen fajta elemeket tároljon. Ez a típus természetesen tetszőleges összetett típus is, akár másik `vector` is lehet.

A `vector` végeredményben a *dinamikus* memórián tárolja az adatokat, ezért rugalmasan méretezhető, vagy akár kiüríthető. Használatához a memóriabeli ábrázolását nem kell ismernünk, mutatókkal sem kell találkoznunk, minden számunkra fontos tevékenységet biztosítanak a `vector` metódusai (*enkapszuláció*).

A `vector` a `cout`hoz, `string`hez és még egyéb dolgokhoz hasonlóan az `std` *névtér*ben található.

Néhány szolgáltatása

Létrehozás:

```
vector<típus> név;
```

Pl.: `vector<int> v;` létrehoz egy `v` nevű `int`eket tároló üres vektort.

```
vector<típus> név(méret);
```

Meg lehet adni a kezdeti méretét, ekkor 0-ra inicializálja a benne levő adatokat (ha ezt az adott elemtípussal nem lehet, mert nincs „*default konstruktor*a”, akkor *fordítás* idejű hibát ad).

Beszúrás:

```
v.push_back(elem);
```

Ez beszúrja a `vector` végére, a méretét eggyel növelve.

Figyelem! Ha a `vector`t létrehoztuk `n` elemesnek, majd egy `for` ciklussal egyesével be szeretnénk szúrni ezt az `n` elemet, akkor nem ezt kell csinálni, mert ez az első `n` elem után

kezdené el feltölteni. A megoldás vagy a felülírás, vagy egy kezdetben üres vektorba való `push_back`elés.

Felülírás:

```
v[i] = elem;
```

Ez az `i`-edik elemnek ad értéket. Ha eleve tudom, hogy a tömb `n` elemű lesz, akkor feltöltésének egy lehetséges módja, ha létrehozom `n` eleműnek, majd ilyen módon megadom az egyes elemek értékeit, hasonlóan ahogy egy sima tömbbel csinálnám.

Lekérdezés:

```
elem = v[i];
```

Akár egy tömbben, bár ez *túlindexelés* esetén futás idejű hibát dob (azaz befagy).

Méret módosítás:

```
v.resize(szám);
```

Ha a méretét növelem, akkor a végére szúr 0-kat, ha csökkentem, akkor a végéből veszi el az elemeket. Beszúrással, törléssel módosul.

Méret lekérdezés:

```
n = v.size();
```

Üresség lekérdezés:

```
v.empty();
```

Igaz, ha a vektor üres (`size=0`), hamis különben.

Kiürítés:

```
v.clear();
```

Mint a `resize(0);`

Első/utolsó elem:

```
v.front();
```

Ez az első elem, ugyanaz mint `v[0];`

```
v.back();
```

Ez az utolsó elem, azaz `v[v.size()-1];`

Elem törlése:

```
v.pop_back();
```

Ez kiszedi az utolsó elemet és csökkenti is eggyel a méretet.