

Tapasztalatok az osztály-sablon könyvtár bevezetésével

Gregorics Tibor, Nagy András

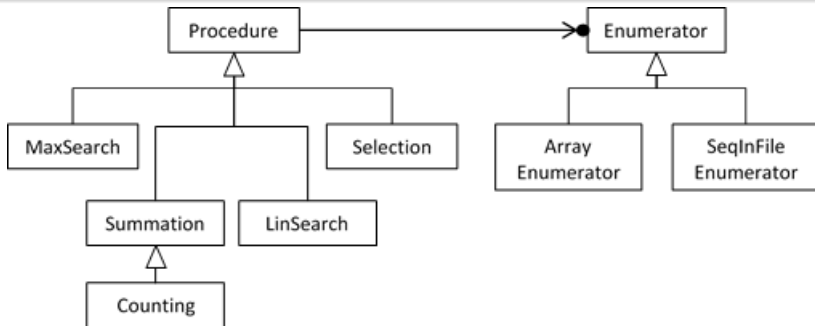
2018. november 24.



Osztálysablon könyvtár bemutatása

- Nevezetes programozási tételeket leíró osztályokat tartalmaz
- Konkrét feladatok megoldása esetén leszármazással dolgozunk, visszavezetve a problémát egy nevezetes tételre
- Az adott feladatra jellemző konkrét műveleteket és értékeket adhatjuk meg. (Pl. összegzésnél a neutrális elemet és az összegző műveletet)
- A feldolgozandó gyűjteményt futás időben adjuk meg (Függőség befecskendezés)

Osztálysablon könyvtár bemutatása



run() metódus

```
init();
for (enor->first(); !enor->end(); enor->next()) {
    body(enor->current());
}
```

Feladat

Számítsuk ki egy C++ vectorban elhelyezett számok összegét.

Az alábbiakat kell meghatározni a megoldásához:

- Összegzéssel oldjuk meg a feladatot
- Összegezendő elemek típusa, valamint az eredmény is egész szám. Az összegezendő elemek típusát leképező függvény az identitás.
- Az összegző művelet egyszerű összeadás, a neutrális elem a 0.
- A felsorolandó gyűjtemény pedig egy vector lesz.

Megoldó osztály

```
class SimpleSum : public Summation<int,int> {  
public:  
    int add(const int& a, const int& b) const {  
        return a + b;  
    }  
    int neutral() const {return 0;}  
    int func(const int& e) {return e;}  
}
```

Osztály használata

```
SimpleSum sum;  
ArrayEnumerator e({1,2,3});  
sum.addProcedure(&e);  
sum.run();
```



Az osztálysablon könyvtár bevezetése az oktatásba

- 2010-ben vezettük be az oktatásba
- A célja az újrafelhasználás gyakorlása volt
- Vegyes fogadtatásban részesült, számos probléma felmerült a használata során, melynek nagy részét orvosoltuk
- A pozitív és negatív tapasztalatokat felhasználva megreformáltuk a könyvtárat



- Robusztus könyvtár, sok feladat megoldására alkalmas
- Objektumorientált technológiák bemutatása, gyakorlása (leszármazás, függőség befeckendezés, stb..)
- Visszavezetési technikák gyakorlása
- Rákényszeríti a hallgatót az elemzésre, tervezésre egyszerű feladatok esetén is
- C++ objektumorientált nyelvi elemek mélyítése, gyakorlása (sablonok, metódusok felül definiálása, stb..)

- Kritikák

- Nem ipari könyvtár
- Nagyobb kód méret, túl erős technológiákra épül
- Rossz konvenciók alapján íródott

- Problémák

- Nem szakszerű használat
- Egyre bővülő szabályrendszert kellett bevezetni, hogy helyesen használják a hallgatók (mit lehet felüldefiniálni, kiből lehet származni..)
- Összegzés tétele túl általános volt

Új összegzés metódusai

```
void  init() override final {_result=neutral();}  
void  body(const Item& e) override final {  
    if(cond(e))_result = add(_result,e);  
}  
void  cond(const Item& e){return true;}  
Value add(const Item& a, const Item& b) const = 0;  
Value neutral() const = 0;  
Value func(const Item& e) const = 0;
```

- Neutrális elem, összeadás művelet, leképező művelet bevezetése
- Az *add* nem volt konstans, nagyon általános volt, ő változtatta meg az eredmény értékét tetszőleges módon



StringStreamEnumerator

- Változó hosszúságú sor beolvasásakor hasznos.
- C++ specifikus
- Példa: Egy sor egy receptet tartalmaz (pl: "tej 1 liter búzadara 13 evőkanál vaj 6 dkg cukor 5 evőkanál") A recept összetevőit szeretnék eltárolni egy tömbben.
- Összegzéssel megoldva: Az add művelet az összefűzés, a semleges elem az üres tömb.

IntervalEnumerator

- Egy $[m, n]$ diszkrét intervallum értékeit sorolja fel
- Példa: Egy n szám faktoriálisa összegzéssel. Ilyenkor az intervallum $[2, n]$, az *add* művelet egy összeszorzás, a semleges elem pedig az 1.

- Final: Nem engedi a leszármazott osztályokban felül definiálni az adott metódust. (pl. az összegzésben a body-t) Nem kell külön kikötni, mit nem definiálhatnak felül a használata során.
- Override: Jelzi, hogy felül definiáljuk egy ősosztály metódusát. Csökkenti a hibázási lehetőségeket a zárthelyiken, ha a hallgatók is használják.
- Barát osztályok bevezetése: Korlátozzuk, hogy a *Procedure* ősosztályból csak a tételek osztályai származhatnak le. (Nem kell külön kikötni, hogy nem származhatnak le az ősosztályból.)
- Template specializáció: Fájl létrehozó metódus specializálása karakterekre. Javít a kód konvencióján, az új összegzésnél elengedhetetlen összefűzések esetén.
- *nullptr*, *pagrma once*..

- Új helyre került a könyvtár bevezetése, a felsorolás programozási tételek után tanulják közvetlenül, gyakorolva a visszavezetési technikákat.
- Javult a könyvtár kódjának tisztasága.
- Az új nyelvi elemekkel elősegítettük a szakszerű használatot, nincs szükség annyira kikötésre.
- A könyvtár hasznos didaktikai eszköznek bizonyult.
(Hangsúlyozni kell a hallgatóknak a könyvtár célját..)

Köszönöm a figyelmet!

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg (EFOP-3.6.3-VEKOP-16-2017-00002).

