Objektum elvű alkalmazások fejlesztése

Programozási tételek újrafelhasználása 1.

Készítette: Gregorics Tibor

Ötlet

- Hozzunk létre a programozási tételeket általánosan leíró olyan kódkönyvtárat (osztály-sablon könyvtárat), amelynek felhasználásával a visszavezetéssel tervezett programjainkat implementálhatjuk.
- Egy feladat megoldása egy tevékenység objektum lesz, amelynek osztályát úgy állítjuk elő, hogy annak
  1. ősét egy megfelelő osztály-sablonból példányosítjuk
  2. néhány örökölt metódusát felüldefiniáljuk
  3. felsoroló objektumot rendelünk hozzá.

Programozási tételek objektum-elvű megfogalmazása

```
bool l = false;
for (t.First(); !t.End(); t.Next()){
    if(!Cond(t.Current())) continue;
    Value val = Func(t.Current());
    if(l)
        if( opt < val ){
            opt = val;
            optitem = t.Current();
        }else{
            l=true;
            opt = val;
            optitem = t.Current();
        }
}
```

```
int db = 0;
for (t.First(); !t.End(); t.Next()){
    if( Cond(t.Current()) ) ++db;
}
```

```
bool l = false;
for (t.First(); !l && !t.End(); t.Next()){
    l = Cond(elem = t.Current());
}
```

```
Init();
for(t.First(); LoopCond(); t.Next()){
    Do(t.Current());
}
```

Programozási tételek ősciklusa

felsorolt elemek típusa

felsoroló objektumra mutató pointer

enor->First()

true

```
template <typename Item>
void Procedure<Item>::Run()
{
   if(enor==NULL) throw MissingEnumerator;

   Init();
   for( First(); LoopCond(); enor->Next())
   {
      Do(enor->Current());
   }
}
```

WhileCond(enor->Current()) && !enor->End()

Programozási tételek ősosztálya

felsorolt elemek típusa

felsoroló objektumra mutató pointer

felüldefiniálható metódusok

ősciklus

```
template <typename Item>
class Procedure
{
   protected:
      Enumerator<Item> *enor;

      Procedure():enor(NULL){}
      virtual void Init()= 0;
      virtual void Do(const Item& current) = 0;
      virtual void First() {enor->First();}
      virtual bool WhileCond(const Item& current) const
         { return true;}
      virtual bool LoopCond() const
         { return !enor->End()&& WhileCond(enor->Current());}

   public:
      enum Exceptions { MissingEnumerator };
      void Run();
      void AddEnumerator(Enumerator<Item>* en)
         { enor = en;}
      virtual ~Procedure(){}
};
```

procedure.hpp

Felsorolók ősosztálya

```
template <typename Item>
class Enumerator
{
    public:
        virtual void First()          = 0;
        virtual void Next()           = 0;
        virtual bool End()      const = 0;
        virtual Item Current() const = 0;
        virtual ~Enumerator(){}
};
```

enumerator.hpp

Tömb elemeinek felsorolója

```
template <typename Item>
class ArrayEnumerator : public Enumerator<Item>
{
   protected:
      std::vector<Item> *vect;
      int   ind;

   public:
      ArrayEnumerator(std::vector<Item> *v):vect(v){}

      void First()         { ind = 0;}
      void Next()          { ++ind;}
      bool End()     const { return ind>=(int)vect->size();}
      Item Current()const { return (*vect)[ind];}
};
```

arrayenumerator.hpp

Szekvenciális inputfájl felsorolója

```
template <typename Item>
class SeqInFileEnumerator : public Enumerator<Item>{
   protected:
      std::ifstream f;
      Item df;
   public:
      enum Exceptions { OPEN_ERROR };
      SeqInFileEnumerator(const std::string& str)
         f.open(str.c_str());
         if(f.fail()) throw OPEN_ERROR;
         if(typeid(Item)==typeid(char))
            f.unsetf(std::ios::skipws);
      }
      ~SeqInFileEnumerator(){ f.close(); }
      void First()         { Next();}
      void Next()          { f >> df;}
      bool End()     const { return f.fail();}
      Item Current() const { return df; }
};
```

#include <typeinfo>

Az Item-re értelmes az operátor>>

seqinfileenumerator.hpp

Általános maximum keresés

```
template <typename Item, typename Value = Item,
          typename Compare = Greater<Value> >
class MaxSearch : public Procedure<Item>
{
   protected:
      bool    l;
      Item    optelem;
      Value   opt;
      Compare better;

      void Init(){ l = false;}
      void Do(const Item& current);
      virtual Value Func(const Item& e) const = 0;
      virtual bool  Cond(const Item& e) const { return true;}
   public:
      bool Found()   const { return l;}
      Value Opt()    const { return opt;}
      Item OptElem() const { return optelem;}
};
```

összehasonlító objektum:
better(a,b) megmutatja, hogy a jobb-e, mint b

maxsearch.hpp

Általános maximum keresés tevékenysége

```
template <typename Item, typename Value,
          typename Compare>
void MaxSearch<Item,Value,Compare>::
Do(const Item& current)
{
   Value val = Func(current);
   if ( !Cond(current) ) return;
   if (l){
      if (better(val,opt)){
         opt = val;
         optelem = current;
      }
   }
   else {
      l = true;
      opt = val;
      optelem = current;
   }
}
```

maxsearch.hpp

Összehasonlító osztályok

```
template <typename Value>
class Greater{
   public:
      bool operator()(const Value& l, const Value& r)
      {
         return l > r;
      }
};

template <typename Value>
class Less{
   public:
      bool operator()(const Value& l, const Value& r)
      {
         return l < r;
      }
};
```

Greater<int> better esetén a better(2,5) azonos a 2>5 értékével.

compare.hpp

1.Feladat

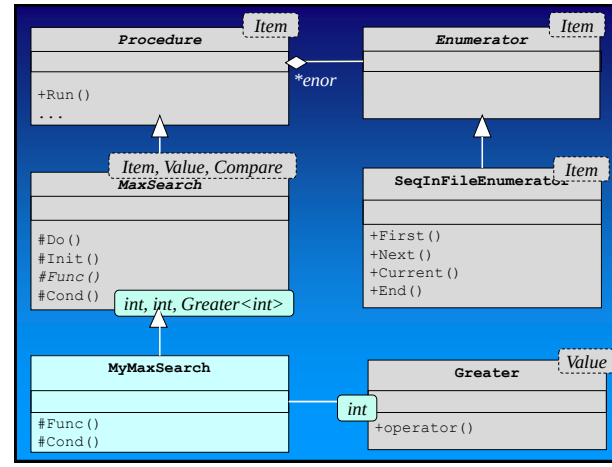
Adott egy egész számokat tartalmazó szöveges fájl, amelyben keressük a legnagyobb páratlan számot!

Megoldás

„Melyik a legnagyobb páratlan szám?”

A feladat visszavezethető a feltételes maximum keresésre

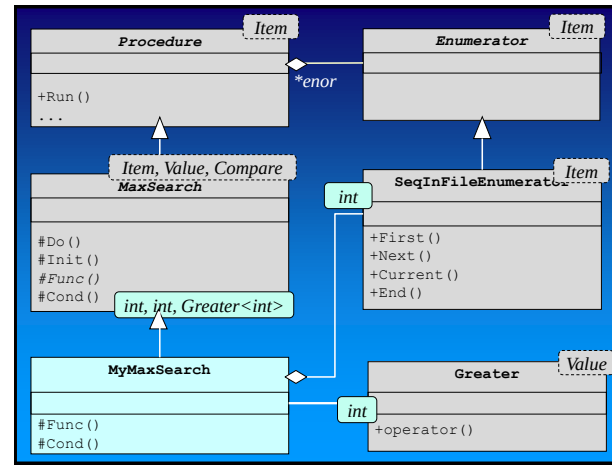
1. Vizsgált elemek (Item): egész számok
2. Összehasonlítandó értékek (Value): egész számok
3. Függvény (Func): identitás.
4. Összehasonlítás (Compare): „nagyobb” reláció (Greater)
5. Feltétel (Cond): páratlan szám
6. Felsoroló: szöveges állomány egész számainak felsorolója



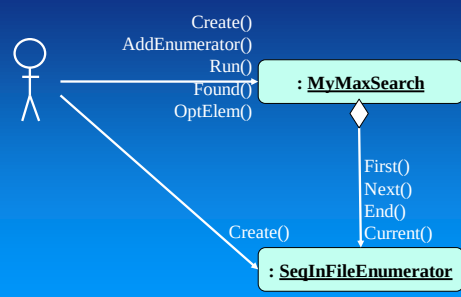
A feladat maximumkeresés osztálya

```
class MyMaxSearch : public MaxSearch<int>
{
   protected:
      int  Func(const int& e) const { return e;}
      bool Cond(const int& e) const { return e%2!=0;}
};
```

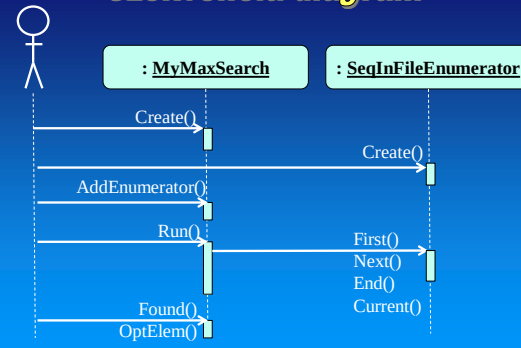
main.cpp



Együttműködési diagram



Szekvencia diagram



Főprogram

Hiányzik a kivételkezelés!

```
int main(){

   MyMaxSearch pr;

   SeqInFileEnumerator<int> file_enum("input.txt");
   pr.AddEnumerator(&file_enum);

   pr.Run();

   if (pr.Found())
      cout << "A legnagyobb páratlan szám:" << pr.OptElem();
   else
      cout << "Nincs páratlan szám";

   return 0;
}
```

tevékenység objektum

felsoroló objektum

működés

main.cpp

2.Feladat

Egy forgalom számlálás során egy hónapon keresztül számolták, hogy az egyes órákban hány utas lép be egy adott metróállomás területére. (A méréseket hétfőn kezdték, de lehet, hogy nem minden nap és órában végeztek megfigyelést.) Az időt egy olyan négyjegyű számmal kódolták, amelynek első két jegye a nap hónapbeli sorszámát, utolsó két jegye az órát mutatja. Egy szöveges fájlban időkód-létszám párok formájában rögzítették a méréseket.
A hétvégi napok közül mikor (nap, óra) léptek be legkevesebben az állomás területére?

Megoldás

„Mikor léptek be legkevesebben a hétvégi időpontok közül?”

A feladat visszavezethető a feltételes minimum keresésre

1. Vizsgált elemek (Item): mérések = időkód-létszám párok
2. Összehasonlítandó értékek (Value): létszám
3. Függvény (Func): mérésből a létszámot adja
4. Összehasonlítás (Compare): „kisebb” (Less) reláció
5. Feltétel (Cond): mérés hétvégére esik
6. Felsoroló: Szöveges állományra épített szekvenciális input fájl méréseinek felsorolója

Elemi típus (Item)

```
struct Pair{
   int time;
   int number;
   int Day()  const { return time/100; }
   int Hour() const { return time%100; }

   friend ifstream& operator>>(ifstream&, Pair&);
};

ifstream& operator>>(ifstream& f, Pair& df)
{
   f >> df.time >> df.number;
   return f;
}
```

Szekvenciális inputfájl `Pair` típusú elemeinek felsorolásához kell

main.cpp

A feladat feltételes minimum keresés osztálya

```
class MyMinSearch: public MaxSearch<Pair, int, Less<int> > {
   protected:
      int  Func(const Pair &e) const { return e.number; }
      bool Cond(const Pair &e) const {
         return (e.Day()%7==6 || e.Day()%7==0;}
};
```

main.cpp

Főprogram

```
int main()
{
   MyMinSearch pr;

   SeqInFileEnumerator<Pair> file_enum("input.txt");
   pr.AddEnumerator(&file_enum);

   pr.Run();

   if (pr.Found()){
      Pair p = pr.OptElem();
      cout << "A " << p.Day() << ". napon a "
      << p.Hour() << "-kor lépett a legkevesebb, "
      << pr.Opt()  << " fő az állomásra" << endl;
   }else cout << "Nincs hétvégi adat" << endl;

   return 0;
}
```

main.cpp


VÉGE