zadanie nr 7 termin realizacji: 18 kwiecień 2005

C++

drzewa AVL (klasy abstrakcyjne) studia dzienne

Drzewo AVL to zrównoważone pod względem wysokości *drzewo binarnych poszukiwań BST*. Drzewa AVL służą do przechowywania elementów z pewnego uniwersum z określonym na tych elementach porządkiem liniowym.

Zadefiniuj abstrakcyjną klasę `ElemPorownywalny`, jako klasę bazową dla elementów jakiegoś zbioru z porządkiem liniowym.

```
class ElemPorownywalny
{
public:
    virtual ~ElemPorownywalny () {}
public:
    // funkcja klonuje obiekt na stercie wykorzystując konstruktor kopiujący
    // funkcję tą trzeba implementować w każdej klasie
    virtual ElemPorownywalny * new_clone () const = 0;
protected:
    // funkcja porownanie(el) zwraca:
    //   +1 gdy *this<el
    //    0 gdy *this==el
    //   -1 gdy *this>el
    virtual int porownanie (const ElemPorownywalny &el) const = 0;
public:
    bool operator < (const ElemPorownywalny &el) const
        { return porownanie(el)>0; }
    bool operator > (const ElemPorownywalny &el) const
        { return porownanie(el)<0; }
    bool operator == (const ElemPorownywalny &el) const
        { return porownanie(el)==0; }
    bool operator != (const ElemPorownywalny &el) const
        { return porownanie(el)!=0; }
    bool operator <= (const ElemPorownywalny &el) const
        { return porownanie(el)>=0; }
    bool operator >= (const ElemPorownywalny &el) const
        { return porownanie(el)<=0; }
};
```

Następnie zdefiniuj abstrakcyjną klasę `Zbior` z podstawowymi operacjami wstawiania, usuwania i wyszukiwania elementów typu `ElemPorownywalny` w zbiorze. Klasa ta będzie bazową dla klasy `DzrzewoAVL` przechowującej elementy zbioru w postaci zrównoważonego drzewa binarnych poszukiwań.

```
class Zbior
{
public:
    virtual ~Zbior ();
public:
    // sprawdzenie, czy podany element jest w zbirze
    virtual bool szukaj (const ElemPorownywalny &el) const = 0;
    // wstawienie nowego elementu do zbioru
    // błąd, gdy element już jest w zbiorze
    virtual void wstaw (const ElemPorownywalny &el) throw (Blad) = 0;
    // usunięcie elementu ze zbioru
    // błąd, gdy elementu nie ma w zbiorze
    virtual void usun (const ElemPorownywalny &el) throw (Blad) = 0;
};
```

Dalej, zdefiniuj klasę `DrzewoAVL` reprezentującą zrównoważone drzewo poszukiwań binarnych AVL. Klasa ta powinna dziedziczyć po abstarkcyjnej klasie `Zbior` i opakowywać rzeczywistą implementację operacji na drzewie AVL, realizowaną w wewnętrznej klasie `DrzewoAVL::WezelAVL`.

```
class DrzewoAVL: public Zbior
{
    class WezelAVL: public Zbior
    {
    protected:
        WezelAVL lewe, prawe;
        ElemPorownywalny element;
    // ...
    };
protected:
    WezelAVL *korzen;
// ...
};
```

Klasa `DrzewoAVL` powinna posiadać kosstruktor domyślny (drzewo puste), konstruktor kopiujący, wirtualny destruktor, operator przypisania kopiującego oraz zaprzyjaźniony operator wypisywania drzewa do strumienia wyjściowego (można sobie wyobrazić jego działanie w postaci przeglądania drzewa w porządku *in-order*). Warto zadbać o to, by w klasie `WezelAVL` także znalazł się zaprzyjaźniony operator pisania do strumienia.

Następnie zdefiniuj klasę `ElemInt` dziedziczącą po `ElemPorownywalny` do pamiętania liczb całkowitych — będzie to nasz podstawowy obiekt do testowania zbioru. Nietrywialną definicję polimorficznej metody `porownanie` przedstawiam poniżej:

```
int ElemInt::porownanie (const ElemPorownywalny &el) const
{
    const ElemInt &e = dynamic_cast<const ElemInt &>(el);
    return wart>e.wart?-1:wart<e.wart?1:0 ;
}
```

Klasę `ElemInt` wykorzystaj w programie testowym, sprawdzającym poprawność wykonywanych na drzewie operacji wstawiania i usuwania elementów.

Uwaga! Klasa `DrzewoAVL` będzie wykorzystywana w następnych zadaniach, a zatem warto napisać ją bardzo starannie.

Uwaga! Opis *drzew AVL* można znaleźć w następującej literaturze:

- L. Banachowski, K. Diks, W. Rytter: *Algorytmy i struktury danych.* WNT, 2003.
- N. Wirth: *Algorytmy + struktury danych = programy.* WNT, 2001.
- `http://www.microsoft.com/poland/developer/jak_zaczac/structures_czesc4.mspx`
- `http://prioris.mini.pw.edu.pl/~awrzek/asd/avl.html`
- `http://www.mini.pw.edu.pl/~kotowski/ASiD/AVL.htm`

Uniwersytet Wrocławski
Instytut Informatyki
ul. Przesmyckiego 20
PL-51-151 Wrocław

Paweł Rzechonek