FEJLETT PROGRAMOZÁSI NYELVEK, 2009
10. GYAKORLAT – Függvényobjektumok

Cél:

- Beépített függvényobjektumok: `greater<T>`
- Felhasználó által készített függvényobjektumok: `PersonCompare, StringCompare, StrangeCompare`

1. Beépített függvényobjektum használata

- Olvasson be a bemenetről gyümölcsneveket és helyezze el ezeket egy csökkenő rendezettséget biztosító halmazban . A csökkenő rendezettséget a `greater<T>` funktor biztosítja.

```
typedef set<string, greater<string> > DecreasingStringSet;
```

- Írassa ki a halmaz tartalmát a szabványos kimenetre.

2. Függvényobjektum készítése – Személyek összehasonlítása

Adott a következő `Person` osztály

```
class Person{
private:
      string fname;
      string lname;
public:
      Person( string fname="", string lname=""):fname(fname), lname(lname){}
      friend ostream& operator<<( ostream& os, const Person& p){//Kod}
};
```

- Készítsen függvényobjektumot, amely személyek összehasonlítását biztosítja (`PersonCompare` osztály)

- Olvasson be a szabványos bementről személyeket és tárolja ezeket egy rendezett halmazban, majd írassa ki a halmaz tartalmát a szabványos kimenetre. Használjon iterátort a halmaz bejárására.

3. Függvényobjektum készítése – Sztringek összehasonlítása

Készítsen egy függvényobjektumot, amely sztringek összehasonlítását képes úgy végezni, hogy nem tesz különbséget kis- és nagybetűk között. A függvényobjektumot lehessen úgy is inicializálni, hogy a megszokott módon végezze a sztringek összehasonlítását és ez legyen az alapértelmezett mód.

```
class StringCompare{
public:
      enum compare_mode { normal, nocase };
private:
  ...
};
```

FEJLETT PROGRAMOZÁSI NYELVEK, 2009
10. GYAKORLAT – Függvényobjektumok

Példa használatra:

```
set<string, StringCompare> s1;//normal mód
set<string, StringCompare> s2(StringCompare::nocase);//nocase mód
```

A `nocase` típusú összehasonlítás megvalósítására felhasználható az `algorithm` fejállományban definiált `lexicographical_compare` függvény.

4. Függvényobjektum készítése – Adott ábécé szerinti összehasonlítás

Adott egy szövegállomány, amelynek első sora egy tetszőleges alfabetikus sorrendet tartalmaz, vagyis az ábécé betűit az aktuális sorrendnek megfelelően. A szövegállomány második sorától kezdődően nevek vannak, minden sorban egy. Állítsuk elő a megadott alfabetikus sorrend szerint a nevek rendezett sorrendjét és írassuk ki egy kimeneti állományba. A szóköz karakter nem jelenik meg az ábécében, az összes többinél kisebbnek tekintendő.

Bemenet:

```
XWYPZRNQMKTJOIGHDBCLAEVUFS
ADAMS JOHN QUINCY
ADAMS JOHN
FRANKLIN BENJAMIN
HANCOCK JOHN
JEFFERSON THOMAS
LEE RICHARD HENRY
LEE ROBERT E
LINCOLN ABRAHAM
```

Kimenet:

```
JEFFERSON THOMAS
HANCOCK JOHN
LINCOLN ABRAHAM
LEE ROBERT E
LEE RICHARD HENRY
ADAMS JOHN
ADAMS JOHN QUINCY
FRANKLIN BENJAMIN
```

FEJLETT PROGRAMOZÁSI NYELVEK, 2009
10. GYAKORLAT – Függvényobjektumok

Megoldási vázlat:

- Egy sajátos rendezettséget kell előállítanunk, ezért függvényobjektumot használunk, amelyet egy `StrangeCompare.h` fejállományban deklarálunk.
- Az állományból a neveket egy olyan halmazba helyezzük, amelyben az összehasonlítást egy `StrangeCompare` típusú függvényobjektum végzi.

```
#ifndef _STRANGECOMPARE_H
#define _STRANGECOMPARE_H
#include <map>
#include <string>
#include <algorithm>
using namespace std;

class StrangeCompare{
private:
    static map<char, int> order;//Karakterek sorrendje
public:
    StrangeCompare( const string& alphabet){//Kod }
    bool operator()( const string& s1, const string& s2 ){//Kod}
    static bool compchar( const char c1, const char c2 ){//Kod}
};
map<char,int> StrangeCompare::order;
#endif      /* _STRANGECOMPARE_H */
```