Tema 2. Problema 2: Els fitxers seqüencials

Esteve Brugulat, Josep M. Ribó

29 de novembre de 2010

1 Objectius

- Presentar els fitxers seqüencials amb els seus diferents modes de treball (lectura/escriptura)
- Presentar els fluxos de C++

Per fer aquest problema, cal que us llegiu:
- Els apunts (secció 2.6)

2 Enunciat de l'activitat

Considera la classe següent:

```
class Student :public Object{

    char nif[10];
    char name[MAXCAR];
    int age;

public:
    Student(){}
    Student (char* pnif,char* pname, int page){
     strcpy(nif,pnif);
     strcpy(name, pname);
     age=page;}
    int getAge(){return age;}
    MyString toString() const
    {
        return MyString(nif)+MyString("   ")+
               MyString(name)+MyString("   ")+
               MyString(age)+MyString("\n");
    }

    void copy(const Object& obj)
    {
          const Student& st =dynamic_cast<const Student&>(obj);
          strcpy(this->nif,st.nif);
          strcpy(this->name,st.name);
          this->age = st.age;
    }
```

```
    bool operator==(const Object& obj) const
    {
         const Student& st =dynamic_cast<const Student&>(obj);
         return (strcmp(st.name,this->name)==0 &&
                 strcmp(st.name,this->name)==0 &&
                 this->age==st.age);
    }

    bool operator<=(const Student& st) const {

        return (strcmp(this->nif,st.nif)<=0);

    }

    Object* clone() const
    {
       return new Student(*this);

    }
};
```

Volem treballar amb fitxers que continguin objectes de la classe `Student`. En particular, el fitxer `estu.dat` que trobareu al projecte és un fitxer que conté objectes d'aquesta classe.

1. Per què creus que hem usat com atribut d'aquesta classe:

   `char name[MAXCAR];`

   en lloc de:

   `char* name;`

   o bé:

   `MyString name;`

2. Escriu una acció:

   ```
   template<class T, class Action>
   void readFile(const MyString& namef)
   ```

   que llegeixi el fitxer d'elements de la classe `Student` que té el nom `namef` i apliqui sobre cada element d'aquest fitxer el tractament descrit a l'operació `operator()` de la classe que substitueixi `Action`.

   Per exemple, una classe que substitueixi `Action` pot ser la següent:

   ```
   template<class T>
   class OutputAction{
   public:
     void operator()(T& x){

           cout<<x.toString()<<endl;
     }
   };
   ```

   Aquesta acció escriu per la sortida estàndar el valor de l'objecte de classe `T` que rep com a paràmetre.

   Pots usar l'operació `toArrayChar` de la classe `MyString` per convertir un `MyString` a `char*`, si ho necessites.

3. Escriu una acció:

```
template<class T,class Action>
void writeFile(const MyString& namef){
```

que genera un fitxer amb elements de tipus `T` i amb nom `nomf`. Cadascun dels elements del fitxer els obtindrà l'operació `operator()` de la classe que instancïi `Action`.

Un exemple de classe que substitueixi `Action` pot ser la següent:

```
class InputAction{
public:
  Student* operator()(){

      char name[MAXCAR];
      char nif[10];
      int age;

      cin>>nif;
      cin>>name;
      cin>>age;

      if (strcmp(nif,"99999")==0) return NULL;
      else return new Student(nif,name,age);
  }
};
```

Clarament, aquesta acció li demana a l'usuari que escrigui per l'entrada estàndar el següent objecte de classe `Student` que cal inserir al fitxer.

4. Escriu una acció:

```
template<class T>
int binarySearch(char* namef, const T& x){
```

que cerqui en un fitxer d'elements de tipus `T` anomenat `namef` un objecte amb el valor de `x`. Si el troba retornarà la posició del fitxer on l'ha trobat. Si no el troba, retornarà -1.

L'acció `binarySearch` ha d'implementar una cerca dicotòmica sobre el fitxer.

Podeu suposar que la classe dels elements del fitxer (exemple: `Student`) té implementades l'operació `<=` que compara dos objectes d'aquella classe.

5. (OPTATIU) Escriu una acció:

```
template<class T>
void merge(char* namef1, char* namef2, char* namef3){
```

que faci la fusió dels fitxers d'elements de tipus `T`: `namef1` i `namef2` per generar un fitxer de nom `namef3` que també estigui ordenat i que contingui els elements de `namef1` i de `namef2` (sense repeticions).