Analyse et Programmation Orientées Objets / C++

Fichiers de données et flux

Les fichiers de données

- **Fonction de mémoire permanente**

  Assurer la persistance des données au-delà de l'exécution du programme qui les a créées et/ou modifiées

  Utilisation des supports magnétiques et/ou optiques

  Extension aux objets

La vue du programmeur (1)

- **Description statique (langage C)**

  Un fichier de données est organisé sous forme d'une séquence **d'enregistrements logiques** consécutifs et ordonnés

  - Chaque enregistrement est une séquence de champs consécutifs et ordonnés
  - Les enregistrements sont de longueur variable
  - Chaque champ peut être un type scalaire ou vectoriel
  - Présence d'une marque de fin d'enregistrement (eol), implicite ou explicite

La vue du programmeur (2)

➡ **Description dynamique (langage C)**

Interface avec le système d'exploitation via un Control Block (C.B.)

- **Opération obligatoire d'ouverture**
- Désignation du fichier et autorisation d'accès
- Adresse du C.B. retournée par ouverture
- Méthode d'accès : séquentiel ou direct
- Type d'opération : lecture, écriture, append
- **Opération conseillée de fermeture**

La vue du programmeur (3)

➡ **Description dynamique (langage C)**

Mémorisation intermédiaire de chaque enregistrement logique dans un buffer local du programme

- Pointeur sur C.B. de type FILE*
- Structure interne du C.B. décrite par <stdio.h>
- Opérations d'échange pilotées par des primitives prédéfinies externes au langage (librairies)
- Echanges entre le buffer local et un buffer système dédié au fichier cible

La vue système (1)

➡ **Description statique**

Un fichier de données est vu comme un ensemble ordonné **d'enregistrements physiques**

- Organisation interne des enregistrements physiques cachée au programmeur
- Chaque enregistrement est une séquence ordonnée d'enregistrements logiques
- Opérations de groupage/dégroupage entièrement masquées au programmeur

La vue système (2)

➔ Description dynamique (1)

Mémorisation de chaque enregistrement physique dans un buffer système (zone de données système) dédié au fichier cible

- Banalisation des accès sous forme de flux
- Mise en place du buffer système à l'ouverture
- Libération du buffer système à la fermeture du fichier
- Primitives d'échange propres au S.E.

La vue système (3)

➔ Description dynamique (2)

Mémorisation des enregistrements physiques sur le media cible

- Structure logique d'une liste chaînée de secteurs
- Projection physique sur des couples (piste, secteur)
- Echange entre le buffer système et les secteurs cibles
- Tête de liste dans la File Allocation Table
- Organisation interne cachée à l'utilisateur par le SGF support (FAT ou NTFS pour Windows)

Les flux de données

- **Définition (langage C++)**

  Un flux est un flot de données échangé entre un producteur (source) et un consommateur (cible)

  Echanges synchronisés par le système d'exploitation

  Flux prédéfinis standard : stdin (cin), stdout(cout), stderr (cerr)

  Flux associé à un fichier de données par une opération d'ouverture du fichier

Classes prédéfinies (1)

- **Gestion des flux (langage C++)**
  - **ios** : paramètres généraux de gestion
  - **streambuf** : gestion des tampons
  - **istream** : flux d'entrée (dérivée de ios)
  - **ostream** : flux de sortie (dérivée de ios)
  - **iostream** : (dérivée istream, ostream, streambuf)

Classes prédéfinies (2)

➡ **Gestion des fichiers (langage C++)**

- **filebuf** : gestion des buffers (dérivée de streambuf)
- **fstreambase** : dérivée de ios
- **ifstream** : dérivée de istream et fstreambase
- **ofstream** : dérivée de ostream et fstreambase

La classe ios

➡ **Paramètres généraux de gestion des flux**

- Indicateurs de status (fin de fichier, …)
- Descripteurs de formats (left, right, dec, hex, …)
- Les manipulateurs (setbase, setprecision, setw, …)
- Les modes d'ouverture (lecture, écriture, creation, …)
- ---

Flux et fichiers de données (1)

➡ **Association d'un flux à un fichier de données**

- Opération obligatoire pour accéder au fichier
- Opération à la charge du programmeur
- **Ouverture implicite** du fichier associé

Flux et fichiers de données (2)

➡**Exemples d'association**

```
# include  <fstream.h>

void main () {
ifstream ciF1 ("F1.dat");
ofstream coF2 ("F2.dat");
- - -

    - - -
}
```

Transfert de données (1)

➡ **Déclaration en langage C++**

Surcharge des opérateurs prédéfinis << et >>

- Opérateur d'extraction (>>)
- Opérateur d'injection (<<)
- Nécessité de la déclaration **" friend "**

friend ofstream& operator << (ofstream&, CX&);
friend ifstream& operator >> (ifstream&, CX&);

Transfert de données (2)

- **Définition en langage C++**

```
# include <fstream.h>

ofstream& operator << (ofstream& cF, CX& obj) {
        ---
        cF << champ1 << champ2 << champ3 ;
        return cF;
}
```

Transfert de données (3)

```
# include <fstream.h>

void main () {
ifstream ciF1 ("F1.dat");
ofstream coF2 ("F2.dat");
char aMessage[80];

    ciF1 >> aMessage;
    coF2<<aMessage;
}
```