Programmation

Cursus

1

Level

1

Algorithmie

Module 3 Partie 1

Sommaire

- Les Fichiers

Références Internet

Introduction à l'algorithmique et à la programmation

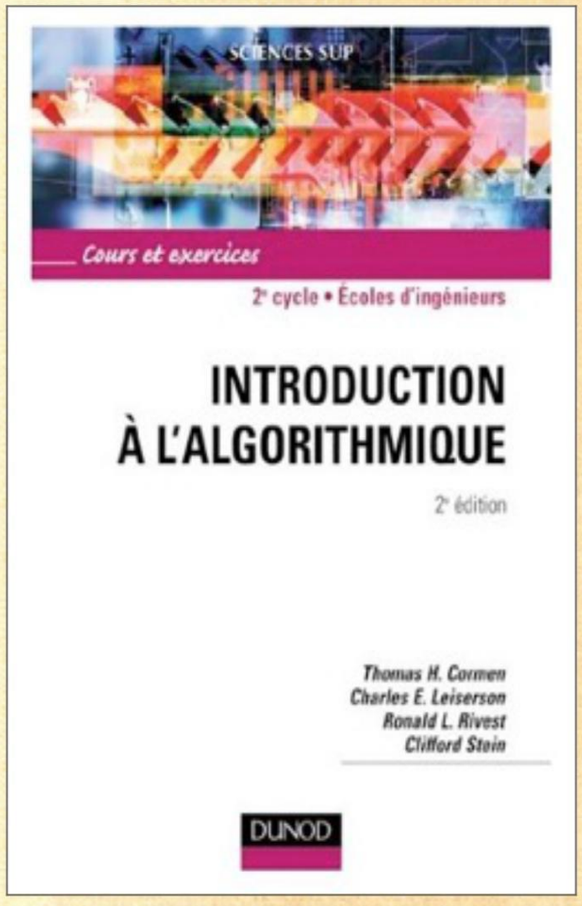
http://perso.citi.insa-lyon.fr/afraboul/imsi/algo-imsi-4.pdf

Références Bibliographiques

Introduction à l'algorithmique :

Cours et exercices

de Cormen, Leiserson, Rivest

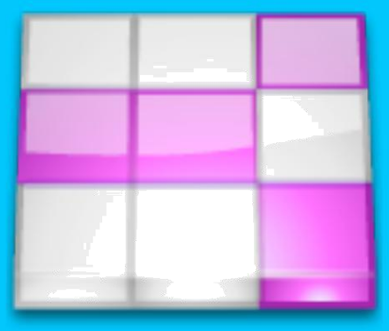


Références Bibliographiques

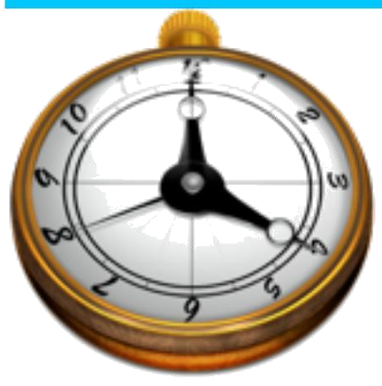
L'algorithmique :

Votre passeport informatique pour la programmation

de Bruno Warin



Liens pédagogiques



Pré-requis:

Historique et modèle de Von Neumann

Nécessaire pour :

Réaliser tout programme.

Algorithmie

Les Fichiers

Manipulation des fichiers

Dans ce chapitre nous verrons qu'on peut sauvegarder de façon pérenne les informations découlant de l'exécution d'un programme. Concomitamment, nous verrons qu'un programme peut exploiter des données qui existent physiquement sur l'ordinateur.

Un fichier est un ensemble d'informations homogènes regroupées sous un même nom. En algorithmique, nous dirons qu'un fichier c'est également un ensemble de données organisées en unités d'accès appelées 'enregistrements' ou 'articles' (tous du même type).

Un fichier est toujours enregistré sur un support externe à la mémoire centrale (disque dur, clé USB, disquette…). C'est pour cela que les informations qu'il contient sont dites **non volatiles**. Les actions les plus courantes sur les fichiers sont :

Création : définir le type du fichier ainsi que son emplacement physique.

Consultation : exploiter les articles du fichier sans en modifier un seul.

Mise à jour : ajouter, modifier ou supprimer des enregistrements ou des champs.

Le mode de représentation physique des fichiers varie d'un ordinateur à un autre. Et pour cause, cette représentation ressortit au Système d'Exploitation. Ainsi, chaque langage de programmation gère les fichiers en fonction du système sur lequel il est installé.

Le type Fichier :

Pascal admet 3 classes : les **fichiers typés**, les **fichiers non typés**, les **fichiers texte**.

Remarques :

- Les fichiers typés et non typés sont des fichiers binaires, c'est-à-dire qu'ils ne contiennent pas des informations directement lisibles par l'être humain.
- La taille maximale d'un fichier est limitée par l'espace disponible sur son support, et sa manipulation dans un programme se fait à l'aide d'un pointeur.

Manipulation des fichiers

Déclaration des fichiers :

Soit **F** un fichier physique existant sur support non volatile. Utiliser **F** dans un algorithme revient à lui associer une variable interne ***f*** par le biais de son chemin d'accès (de **F**).

Il est possible que plusieurs programmes utilisent les données contenues dans un même fichier physique **F**. Dans un tel cas, les noms internes de **F** dans chacun de ces programmes ne seront pas obligatoirement identiques. Par contre leur déclaration, si.

Les fichiers typés

En ce qui concerne les fichiers typés, il y a 2 façons de déclarer une variable interne ***f***.

Syntaxe 1 : **Type** <identificateur_fichier>=**fichier de** <type_des_articles>

Var f : <identificateur_fichier>

Ou plus simplement :

Syntaxe 2 : **Var** f : **fichier de** <type_des_articles>

Ces 2 déclarations permettent au programmeur de définir des fichiers constitués d'articles du type qu'il désire. Par exemple, si on veut définir un fichier qui sera constitué uniquement de nombres entiers, on va faire : **Var** f : **fichier de entier**.

* **Explications**

La déclaration précédente crée une variable interne ***f*** qu'on utilisera pour accéder à un fichier externe **F**. Ce fichier **F** ne pourra contenir qu'une série de nombres entiers tous codés en binaire. La taille maximale de **F** sera fonction de l'espace disponible sur le disque.

NB : On peut définir un fichier constitué d'articles appartenant à un type prédéfini, à l'exception du type Fichier lui-même (un fichier ne peut pas contenir d'autres fichiers).

Manipulation des fichiers

Les fichiers non typés

Ils sont très peu utilisés dans la programmation académique. Ils permettent d'introduire dans un même fichier des articles appartenant à des types différents.

Syntaxe 1 : **Type** <identificateur_fichier>=**fichier**

Var f : <identificateur_fichier>

Ou plus simplement :

Syntaxe 2 : **Var** f : **fichier**

Les fichiers Texte

Un fichier texte est un fichier particulier ne pouvant contenir que des informations de type texte. Il se constitue de plusieurs lignes et on le déclare à l'aide du mot-clé **'Text'**.

Syntaxe 1 : **Type** <identificateur_fichier>=**texte**

Var f : <identificateur_fichier>

Ou plus simplement :

Syntaxe 2 : **Var** f : **texte**

Utilisation des fichiers :

L'assignation

On a vu que pour utiliser un fichier physique **F** dans un algorithme, il fallait que cet algorithme comporte une variable fichier ***f***. *L'association entre* ***f*** et **F** s'effectuera donc au moyen d'un procédé appelé **assignation**, de telle sorte que les modifications apportées à ***f*** dans le programme affecteront directement **F** sur son support.

Syntaxe : **ASSIGNER(f, 'chemin_dacces_F')**

Manipulation des fichiers

Remarques :

- ASSIGNER() est valable pour tous les types de fichiers (binaires ou texte).
- On peut utiliser une même variable ***f*** pour effectuer des modifications sur 2 fichiers physiques **F** et **G** de même type **.** Il suffit pour cela d'effectuer une 1ère Assignation avec le chemin d'accès de **F,** de travailler avec lui, puis de le fermer et d'effectuer une 2nde assignation avec cette fois le chemin d'accès de **G.**

L'ouverture

- **OUVRIR(f)** si **F** est un fichier existant déjà sur disque (ouverture en lecture).
- **REECRIRE(f)** si on veut créer un fichier **F** qui n'existe pas encore sur disque, ou bien qui existe déjà mais dont on veut écraser tout le contenu (ouverture en écriture).
- **AJOUTER(f)** si on veut ouvrir un **fichier texte F** en pouvant y ajouter des données (uniquement en fin de fichier).

Remarques :

- OUVRIR() rend **F** consultable et positionne le pointeur en début de fichier.
- Sur un fichier texte, la primitive OUVRIR() n'autorise que la lecture.
- Dans le cas où **F** n'existe pas sur le disque, REECRIRE() permet de créer un fichier dont le nom et le chemin d'accès sont ceux précisés pendant l'assignation.
- Il peut arriver qu'on ouvre un fichier qui ne contient aucune donnée. Le début de fichier sera alors la fin de fichier. La variable booléenne **Eof**(f) permet à tout moment de vérifier si le pointeur s'y trouve déjà.

Manipulation des fichiers

La lecture et la mise à jour

Après l'ouverture du fichier, plusieurs possibilités sont offertes (lire les informations qu'il contient, en modifier quelques-unes, en supprimer certaines, ou alors en ajouter d'autres). Nous n'utiliserons ces possibilités que sur les fichiers typés et sur les fichiers texte.

Les fichiers typés

On a vu qu'un fichier typé **F** était constitué d'un ensemble d'articles tous appartenant forcément à un même type. La lecture et la mise à j our de **F** dans l'algorithme nécessitent donc la présence d'une variable **p** du type de ces articles.

Syntaxes :

Lire(f, p) pour lire un article de **F** et insérer les données qu'il contient dans la variable **p**.

Ecrire(f, p) pour insérer dans **F** un article ayant les données contenues dans la variable **p**. Il y a écrasement si un autre article occupait déjà le même emplacement dans le fichier.

NB : après la lecture/écriture d'un article, le pointeur se positionne immédiatement sur l'article suivant. S'il n'y en a plus, la position du pointeur devient alors la fin de fichier.

Les fichiers texte

Le procédé est presque le même que précédemment, sauf que le fichier texte est considéré comme un fichier typé constitué exclusivement de lignes de texte. Ainsi, notre variable **p** sera obligatoirement une chaîne de caractères.

Syntaxes :

Lire(f, p) pour lire une ligne de **F** et l'insérer dans la variable **p**.

Ecrire(f, p) pour insérer **à la fin** de **F** une ligne de texte dont la valeur est contenue dans **p**.

Manipulation des fichiers

Remarques :

- **Lire()** n'est possible que si on a ouvert le fichier texte en lecture avec OUVRIR(). Après la lecture, le pointeur se positionne immédiatement sur la ligne suivante. S'il n'y en a plus, la position du pointeur devient alors la fin de fichier.
- **Ecrire()** n'est possible que si on a ouvert le fichier texte en écriture avec AJOUTER().
- Pour éviter les erreurs, il vaut mieux fermer **F** avant de changer le mode d'ouverture.

La fermeture

La fermeture est une opération essentielle dans la manipulation des fichiers. Un bon programmeur ne doit donc jamais la négliger, car elle permet d'éviter les erreurs d'entrée/sortie, de préserver l'intégrité des données d'un fichier, d'optimiser un algorithme en utilisant un minimum de variables internes avec un maximum de fichiers externes.

Syntaxe : **FERMER(f)**

Manipulation des fichiers

EXEMPLE : écrire un algorithme qui permet d'afficher toutes les valeurs contenues dans un fichier d'entiers **F** et qui permet d'ajouter une ligne de texte dans un nouveau fichier texte **G**

```
Algorithme fichiers
Var   p : entier
      ligne : chaîne
       f :fichier de entier
       g : texte
DEBUT
ASSIGNER(f, 'c:\entiers.txt')
ASSIGNER(g, 'c:\texte.doc')
OUVRIR(f)
TANT QUE NON(Eof(f)) FAIRE
      Lire(f, p)
      Ecrire('Nous venons de lire le nombre ', p)
FINTANTQUE
FERMER(f)
```

Manipulation des fichiers

Ecrire(‘Entrez un texte à ajouter dans le second fichier’)

Lire(ligne)

REECRIRE(g)

AJOUTER(g)

Ecrire(g, ligne)

FERMER(g)

FIN