1

Environnement et conventions

PLAN

Un algorithme est un maillon de la chaîne de développement d'une application. Il est le lien indispensable entre l'analyse et le développement final. Ce chapitre présente le rôle d'un algorithme dans cette chaîne de production, et les conventions syntaxiques utilisées pour son écriture. Il aborde également les notions de performances et de génie logiciel, qui servent souvent de guides dans la phase de conception algorithmique.

1. Les étapes de développement d'une application

L'écriture d'un programme dans un langage de programmation n'est que l'étape finale d'un développement qui se déroule en trois phases : *l'analyse*, *l'algorithme* et *la programmation*.

1.1. L'analyse

Le but de tout développement est de fournir une solution informatique à un problème donné. La première étape est *l'analyse du problème*. Elle consiste à explorer le domaine d'application pour en déduire les contraintes et les limites de l'étude, à définir les principales fonctionnalités qui devront être couvertes par le logiciel, à mettre en évidence les problèmes connexes, à faire l'inventaire des acteurs (utilisateurs, gestionnaires, etc.), etc. L'analyse tend à cerner complètement le sujet et à apporter des solutions aux problèmes dévoilés par l'étude.

L'analyse se termine généralement sur des considérations plus techniques qui font le lien avec l'algorithme. Il s'agit de l'*analyse des données* qui aboutit à leur *modélisation*, de la définition des *modules de traitement*, ainsi que leurs

interactions réciproques qui débouchent sur la création des sous-programmes ou des méthodes. L'analyse conditionne le développement futur, car ce sont les solutions proposées à cette étape qui seront programmées. Une mauvaise analyse produira un logiciel médiocre ou qui ne répondra pas aux attentes initiales, quelle que soit la qualité de l'algorithme ou de la programmation.

Voici deux exemples d'analyses partielles. Le premier montre l'incidence de l'étude du domaine d'application sur le résultat de l'analyse. Le second propose une analyse des données et montre comment les structurer en fonction des traitements à effectuer.

L'analyse du premier exemple va évoluer au fur et à mesure que les bonnes questions seront soulevées. Elle produira au final un programme très différent dans sa complexité. Prenons le problème de l'interprétation du numéro de Sécurité sociale (sans la clé finale). La signification des champs de ce numéro est indiquée à la figure 1.1.

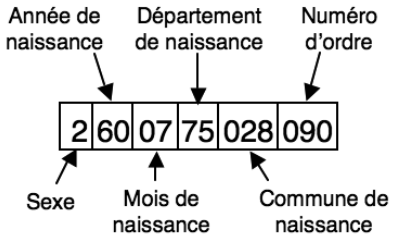

Figure 1.1
L'interprétation d'un numéro de Sécurité sociale.

Une première analyse de ce numéro montre qu'il s'agit d'une *femme*, née en *1960*, au mois de *juillet*, à *Paris*. Or, l'interprétation du quatrième élément de cet exemple est fausse ! Cette personne est née dans le département de la Seine qui, en 1960, regroupait Paris et les départements limitrophes (Val-de-Marne, Val d'Oise, etc.). Cette remarque soulève d'autres questions qui vont compliquer l'analyse, en intégrant un volet historique :

- Comment le découpage des départements a-t-il évolué (Île-de-France, Corse, Outre-Mer, etc.) ?
- Qu'en est-il des personnes nées dans les colonies françaises avant leur indépendance ?
- Comment le problème des centenaires est-il réglé ?

L'analyse simpliste du début aurait conduit à écrire un programme ne traitant qu'une partie de la problématique. Le problème est en réalité beaucoup plus complexe qu'il n'y paraît. L'erreur qu'il ne faut pas faire pendant la phase d'analyse est d'apporter des solutions basées sur ses certitudes, au lieu de se poser d'abord des questions.

Le second exemple montre comment, à partir de l'analyse des données, on peut aboutir à leur modélisation. L'exemple étudié est celui d'une course hippique. La première question à se poser est « qu'est-ce qu'un cheval ? ». La réponse n'est pas liée à la nature de l'animal, mais aux actions à entreprendre pour gérer une course. On peut les résumer à :

- indiquer la liste des chevaux au départ de la course ;
- affecter le temps de chaque cheval à la fin de la course ;
- afficher l'ordre d'arrivée.

On peut déduire qu'une donnée représentant un cheval doit contenir :

- le nom du cheval ;
- le numéro du dossard ;
- le temps d'arrivée ;
- le classement.

On peut également ajouter le nom du jockey, celui du propriétaire, et le montant des gains si on désire ajouter un volet financier à la gestion de la course.

La figure 1.2 présente une modélisation des données de la course hippique. Les chevaux sont gérés comme une liste, où chaque élément est une structure contenant le nom du cheval, le numéro du dossard, le temps effectué durant la course et le classement final. La liste peut être représentée sous la forme d'un tableau, où chaque case contient toute l'information d'un cheval, ou bien sous la forme d'éléments distincts liés entre eux par la connaissance de l'élément précédent et suivant (liste chaînée de structures ou d'objets). La modélisation des données est ensuite suivie par la description de leurs traitements.

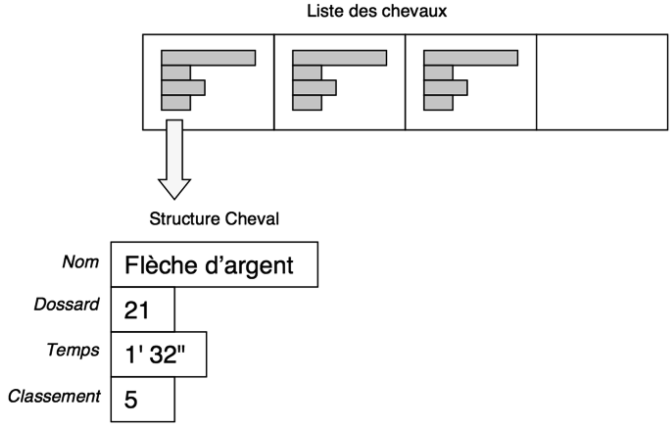

Figure 1.2

Modélisation des données d'une course hippique.

1.2. L'algorithme

Le rôle de l'algorithme est avant tout d'organiser et de structurer l'ensemble des solutions apportées par l'analyse pour aboutir à la solution globale. Il doit surtout présenter la logique aboutissant au résultat final et être dissocié de l'aspect technique des langages informatiques. Il est généralement présenté en pseudo-langage (langage simplifié), ce qui facilite sa traduction dans un langage de programmation. Voici l'algorithme interprétant le numéro de Sécurité sociale, correspondant à une analyse ne traitant pas l'aspect historique.

« PSEUDO-CODE »

```
Programme  Num_Sec_Soc
Déclarations
   Variables Num_Sec,Libellé,Nom_Mois en Chaînes_de_Caractères
   Variables Sexe,Année,Mois,Département,Commune,Num_Ordre en Entier
Début
  Écrire ("Entrez votre numéro de Sécurité sociale :")
  Lire(Num_Sec)
  Sexe ← Conversion_En_Entier(sous_chaîne(Num_Sec,1,1)
  Si (Sexe = 1) alors
     Libellé ← "Monsieur"
  Sinon
     Libellé ← "Madame"
  Finsi
  Année ← Conversion_En_Entier(sous_chaîne(Num_Sec,2,2))
  Année ← Année + 1900
  Mois  ← Conversion_En_Entier(sous_chaîne(Num_Sec,4,2))
  Selon  Mois  Faire
    Cas 1  : Nom_Mois ← "Janvier"
    Cas 2  : Nom_Mois ← "Février"
    Cas 3  : Nom_Mois ← "Mars"
    Cas 4  : Nom_Mois ← "Avril"
    Cas 5  : Nom_Mois ← "Mai"
    Cas 6  : Nom_Mois ← "Juin"
    Cas 7  : Nom_Mois ← "Juillet"
    Cas 8  : Nom_Mois ← "Août"
    Cas 9  : Nom_Mois ← "Septembre"
    Cas 10 : Nom_Mois ← "Octobre"
    Cas 11 : Nom_Mois ← "Novembre"
    Cas 12 : Nom_Mois ← "Décembre"
  FinSelon
  Département ← Conversion_En_Entier(sous_chaîne(Num_Sec,6,2))
  Commune     ← Conversion_En_Entier(sous_chaîne(Num_Sec,6,5))
  Num-Ordre   ← Conversion_En_Entier(sous_chaîne(Num_Sec,11,3))
  Écrire ("Bonjour : ", Libellé)
  Écrire ("Vous êtes né en : ", Année)
  Écrire ("Au mois de : ", Nom_Mois)
  Écrire ("Dans le département : ", Département)
  Écrire ("Dans la commune : ", Commune)
  Écrire ("Avec le numéro d'ordre : ", Num_Ordre)
Fin
```

La syntaxe du pseudo-langage est librement choisie par le développeur. Cependant, elle respecte certaines règles et conventions usuelles qui sont décrites dans ce chapitre.

1.3. L'écriture du programme dans un langage de programmation

Une fois l'algorithme écrit, il faut le traduire en un *programme source*. Cela consiste à écrire un fichier texte contenant la traduction de l'algorithme dans un langage de programmation. Le programme source est ensuite traduit en langage binaire du processeur grâce au compilateur. Le fichier résultat de cette opération est le *programme exécutable*. Dans le cas de certains langages comme PHP et Python, cette phase de compilation n'existe pas. Le programme source est directement *interprété* pour effectuer son exécution. Il n'y a alors aucune génération de programme exécutable.

Dans l'absolu, le développeur doit choisir le langage de programmation le plus adapté au domaine d'application et aux contraintes liées à l'algorithme. En effet, les langages de programmation ont été créés pour répondre à des besoins spécifiques. Par exemple, le langage Fortran est dédié au calcul scientifique, le langage C excelle dans les applications systèmes et réseaux, le langage C++ est très présent dans les applications temps réel, les langages Java et PHP sont conçus pour les développements liés au Web, le langage Python propose des librairies mathématiques, etc. En réalité, le développeur choisit le langage qu'il connaît le mieux ou, de manière plus pragmatique, pour lequel il possède un compilateur ou un interpréteur. Ce n'est donc pas toujours le langage le mieux adapté aux spécificités du développement qui est utilisé.

Afin de permettre au lecteur de tester les programmes sources C, C++, Java, PHP et Python, nous présentons la réécriture des algorithmes dans les cinq langages, les syntaxes de compilation ou d'interprétation de chaque programme source ainsi qu'un exemple d'exécution.

Voici le programme source `numsecsoc.c` écrit en langage C qui correspond à l'algorithme du numéro de Sécurité sociale. La syntaxe du langage C n'admet aucun caractère accentué dans le nom des variables. La variable `Libellé` de l'algorithmique est ainsi écrite `Libelle` dans le programme C.

« C »

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{/* --- Déclarations des variables --- */
 char Num_Sec[14], Libelle[9], Nom_Mois[10], Dept[3], Com[6] ;
 int Sexe, Annee, Mois, Num_Ordre, Departement, Commune      ;
 /* --- Instructions --- */
 /* Saisie du numéro sous la forme d'une chaîne */
 printf("Entrez votre numéro de Sécurité sociale : ") ;
 scanf("%s",Num_Sec) ;
 /* Récupération du numéro 1 ou 2 du Sexe */
 Sexe = Num_Sec[0]-48 ;
 /* Affectation du Libellé */
 if (Sexe == 1)
   strcpy(Libelle,"Monsieur") ;
 else
   strcpy(Libelle,"Madame") ;
```

```
 /* Récupération de l'année de naissance */
 Annee = ((Num_Sec[1]-48)*10) + (Num_Sec[2]-48) ;
 Annee = Annee + 1900 ;
 /* Récupération du mois de naissance */
 Mois = ((Num_Sec[3]-48)*10) + (Num_Sec[4]-48) ;
 switch (Mois)
 {
   case 1  : strcpy(Nom_Mois,"Janvier")  ;
             break ;
   case 2  : strcpy(Nom_Mois,"Février")  ;
             break ;
   case 3  : strcpy(Nom_Mois,"Mars")     ;
             break ;
   case 4  : strcpy(Nom_Mois,"Avril")    ;
             break ;
   case 5  : strcpy(Nom_Mois,"Mai")      ;
             break ;
   case 6  : strcpy(Nom_Mois,"Juin")     ;
             break ;
   case 7  : strcpy(Nom_Mois,"Juillet")  ;
             break ;
   case 8  : strcpy(Nom_Mois,"Août")     ;
             break ;
   case 9  : strcpy(Nom_Mois,"Septembre");
             break ;
   case 10 : strcpy(Nom_Mois,"Octobre")  ;
             break ;
   case 11 : strcpy(Nom_Mois,"Novembre") ;
             break ;
   case 12 : strcpy(Nom_Mois,"Décembre") ;
             break ;
 }
 /* Récupération du département  */
 /* qui reste une chaîne         */
 Dept[0]=Num_Sec[5] ;
 Dept[1]=Num_Sec[6] ;
 Dept[2]='\0'       ;
 Departement=atoi(Dept) ;
 /* Récupération de la Commune */
 Com[0]=Num_Sec[5] ;
 Com[1]=Num_Sec[6] ;
 Com[2]=Num_Sec[7] ;
 Com[3]=Num_Sec[8] ;
 Com[4]=Num_Sec[9] ;
 Com[5]='\0'       ;
 Commune=atoi(Com) ;
 /* Récupération du numéro d'ordre */
 Num_Ordre=((Num_Sec[10]-48)*100)+((Num_Sec[11]-48)*10)+(Num_Sec[12]-
48);
 /* Affichage des résultats */
 printf("Bonjour                %s\n",Libelle)      ;
 printf("Vous êtes né en        : %d\n",Annee)      ;
 printf("Au mois de             : %s\n",Nom_Mois)   ;
 printf("Dans le département    : %d\n",Departement);
 printf("Dans la commune        : %d\n",Commune)    ;
 printf("Avec le numéro d'ordre : %d\n",Num_Ordre)  ;
}
```

La compilation de ce programme est obtenue par la commande suivante :

```
$ make numsecsoc
```

Voici l'exécution du logiciel (fichier exécutable) `numsecsoc` produit par la compilation précédente (le caractère `$` représente le prompt du système UNIX) :

```
$ numsecsoc
Entrez votre numéro de Sécurité sociale : 1650475110019
Bonjour                Monsieur
Vous êtes né en        : 1965
Au mois de             : Avril
Dans le département    : 75
Dans la commune        : 75110
Avec le numéro d'ordre : 19
```

Voici le programme source `numsecsoc.cpp` écrit en langage C++ qui correspond à l'algorithme du numéro de Sécurité sociale. Aucun caractère accentué n'est utilisé dans le nom des variables.

« C++ »

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
int main()
{// --- Déclarations des variables ---
 char Num_Sec[14], Libelle[9], Nom_Mois[10], Dept[3], Com[6];
 // --- Instructions ---
 // Saisie du numéro sous la forme d'une chaîne
 printf("Entrez votre numéro de Sécurité sociale : ") ;
 scanf("%s",Num_Sec) ;
 // Récupération du numéro 1 ou 2 du Sexe
 int Sexe = Num_Sec[0]-48 ;
 // Affectation du Libellé
 if (Sexe == 1)
   strcpy(Libelle,"Monsieur") ;
 else
   strcpy(Libelle,"Madame") ;
 // Récupération de l'année de naissance
 int Annee = ((Num_Sec[1]-48)*10) + (Num_Sec[2]-48) ;
 Annee = Annee + 1900 ;
 // Récupération du mois de naissance
 int Mois = ((Num_Sec[3]-48)*10) + (Num_Sec[4]-48) ;
 switch (Mois)
 {
   case 1  : strcpy(Nom_Mois,"Janvier")   ;
             break ;
   case 2  : strcpy(Nom_Mois,"Février")   ;
             break ;
   case 3  : strcpy(Nom_Mois,"Mars")      ;
             break ;
   case 4  : strcpy(Nom_Mois,"Avril")     ;
             break ;
   case 5  : strcpy(Nom_Mois,"Mai")       ;
             break ;
   case 6  : strcpy(Nom_Mois,"Juin")      ;
             break ;
   case 7  : strcpy(Nom_Mois,"Juillet")   ;
             break ;
   case 8  : strcpy(Nom_Mois,"Août")      ;
             break ;
   case 9  : strcpy(Nom_Mois,"Septembre") ;
             break ;
   case 10 : strcpy(Nom_Mois,"Octobre")   ;
```

```
            break ;
   case 11 : strcpy(Nom_Mois,"Novembre")  ;
            break ;
   case 12 : strcpy(Nom_Mois,"Décembre") ;
            break ;
 }
 // Récupération du département
 // qui reste une chaîne
 Dept[0]=Num_Sec[5] ;
 Dept[1]=Num_Sec[6] ;
 Dept[2]='\0'       ;
 int Departement=atoi(Dept);
 // Récupération de la Commune
 Com[0]=Num_Sec[5] ;
 Com[1]=Num_Sec[6] ;
 Com[2]=Num_Sec[7] ;
 Com[3]=Num_Sec[8] ;
 Com[4]=Num_Sec[9] ;
 Com[5]='\0'       ;
 int Commune=atoi(Com);
 // Récupération du numéro d'ordre
 int Num_Ordre=((Num_Sec[10]-48)*100)+((Num_Sec[11]-
48)*10)+(Num_Sec[12]-48);
 // Affichage des résultats
 printf("Bonjour                %s\n",Libelle)      ;
 printf("Vous êtes né en        : %d\n",Annee)      ;
 printf("Au mois de             : %s\n",Nom_Mois)   ;
 printf("Dans le département    : %d\n",Departement);
 printf("Dans la commune        : %d\n",Commune)    ;
 printf("Avec le numéro d'ordre : %d\n",Num_Ordre)  ;
}
```

La compilation de ce programme est obtenue par la commande suivante (le caractère `$` représente le prompt du système UNIX) :

```
$ make numsecsoc
```

Voici l'exécution du fichier exécutable `numsecsoc` produit par la compilation précédente :

```
$ numsecsoc
Entrez votre numéro de Sécurité sociale : 2621133028088
Bonjour                Madame
Vous êtes né en        : 1962
Au mois de             : Novembre
Dans le département    : 33
Dans la commune        : 33028
Avec le numéro d'ordre : 88
```

Voici le programme source `numsecsoc.java` écrit en langage Java qui correspond à cet algorithme. Aucun caractère accentué n'est utilisé dans le nom des variables. Ce programme utilise la classe `Scanner` du paquetage `java.util` pour lire le numéro de Sécurité Sociale sous la forme d'une chaîne de caractères.

« JAVA »

```
import java.util.*;
class Num_Sec_Soc {
  private String ch ;
  public Num_Sec_Soc() {
   Scanner Entree = new Scanner(System.in);
```