Jean-Michel Léry

Le langage C

Apprendre à programmer

Avec plus de 250 exemples et exercices corrigés

Licence
Master
Autoformation

COMPLÉMENTS
TÉLÉCHARGEABLES

ellipses

1

Les concepts fondamentaux

PLAN

Un langage de programmation est un support syntaxique qui permet d'indiquer à l'ordinateur une suite d'instructions à exécuter. Cette série d'instructions est écrite dans un programme qu'il faut ensuite traduire dans le langage de l'ordinateur. Dans ce chapitre, nous présentons les étapes et les outils de développement qui aboutissent à l'écriture du programme. Nous y étudions également les spécificités du langage C.

1. L'environnement de programmation

1.1. Le langage binaire et les logiciels

Les éléments de l'ordinateur (processeur, mémoire centrale, etc.) sont constitués de circuits logiques basés sur des *transistors* qui gèrent des valeurs numériques 0 ou 1. Ces deux valeurs constituent la base 2, ou *base binaire*. Le langage de l'ordinateur est composé d'une suite de 0 et de 1, c'est le *langage binaire*.

Chaque processeur a été créé pour effectuer une série d'actions élémentaires comme les additions, les soustractions, l'affichage d'un caractère, etc. Chacune de ces actions est une *instruction* écrite dans son langage binaire. Cet ensemble d'instructions se nomme le *jeu d'instructions*.

L'évolution des processeurs produit de nouvelles puces capables d'effectuer des calculs plus complexes. Leur *jeu d'instructions* s'étoffe, grandit, et diffère par conséquent de celui de ses prédécesseurs. Chaque processeur possède un langage binaire qui lui est propre.

Il existe une certaine compatibilité entre les processeurs d'un même constructeur. Le dernier de la gamme comprend le langage binaire de son prédécesseur, mais

l'inverse n'est pas vrai. La *compatibilité* est dite « ascendante ». Ainsi, un logiciel écrit dans le langage d'un processeur ancien sera compris par les processeurs récents du même constructeur, mais non par ceux d'un autre constructeur.

Le logiciel *exécutable* (binaire compris par le processeur) n'est pas *portable*. Il peut être transposé d'un ordinateur à un autre seulement si ces deux machines possèdent des processeurs et des systèmes d'exploitation compatibles. C'est notamment le cas des processeurs Intel et AMD, qui équipent les PC.

Afin de vendre leurs logiciels au plus grand nombre, les éditeurs doivent choisir un langage binaire compréhensible aussi bien par les ordinateurs récents que par les machines anciennes qui sont toujours en service. L'utilisation du langage binaire d'un processeur relativement ancien, compatible avec les plus récents, est donc un gage de bon fonctionnement, et de vente sur l'ensemble du parc informatique installé. Seuls les logiciels dont la performance peut être nettement améliorée par l'emploi des nouvelles instructions (celles des processeurs récents) seront réécrits dans les nouveaux langages binaires. C'est le cas de certains logiciels de traitement d'images ou multimédias.

À NOTER

Un *logiciel* est une suite logique d'instructions écrites dans le langage du processeur.

Le logiciel *exécutable* (binaire compris par le processeur) n'est pas *portable*. Il est transportable d'un ordinateur à un autre si ces deux machines possèdent des processeurs, des architectures matérielles et des systèmes d'exploitation identiques ou compatibles.

1.2. L'évolution des langages

Ce problème de non-portabilité des logiciels écrits en binaire est le point de départ de la création des *langages de programmation*. Le principe retenu consiste à utiliser un langage plus proche du langage humain que du langage binaire, et indépendant de l'ordinateur.

Le langage assembleur

La première évolution fut de nommer les instructions binaires avec des mnémoniques (mots courts représentant un codage) plus explicites pour le programmeur, ce qui a donné naissance au *langage assembleur*. Le programme écrit en assembleur n'est pas interprétable par le processeur, c'est du texte ; il faut le traduire en binaire *exécutable*. Un logiciel particulier effectue la traduction : le *compilateur*. Il existe autant de compilateurs qu'il y a de couples langage de programmation/langage binaire. Cette première évolution était insuffisante, car il existe autant d'assembleurs que de processeurs. Voici un exemple de la somme de 2 et 3 écrite en assembleur.

« Assembleur »

```
mov ax,2
add ax,3
```

Les langages évolués

L'étape suivante a consisté à définir des langages de programmation *évolués* totalement indépendants de l'ordinateur. Le programmeur écrit son application dans ce langage, puis se sert d'un logiciel particulier, un *compilateur*, pour traduire son programme en un fichier binaire exécutable. Ce langage étant indépendant du processeur, le *fichier source* écrit dans le langage évolué est totalement portable d'un ordinateur à l'autre. Une fois installé sur le nouvel ordinateur, il suffit de le traduire dans le langage binaire de cette machine grâce au compilateur (figure 1.1).

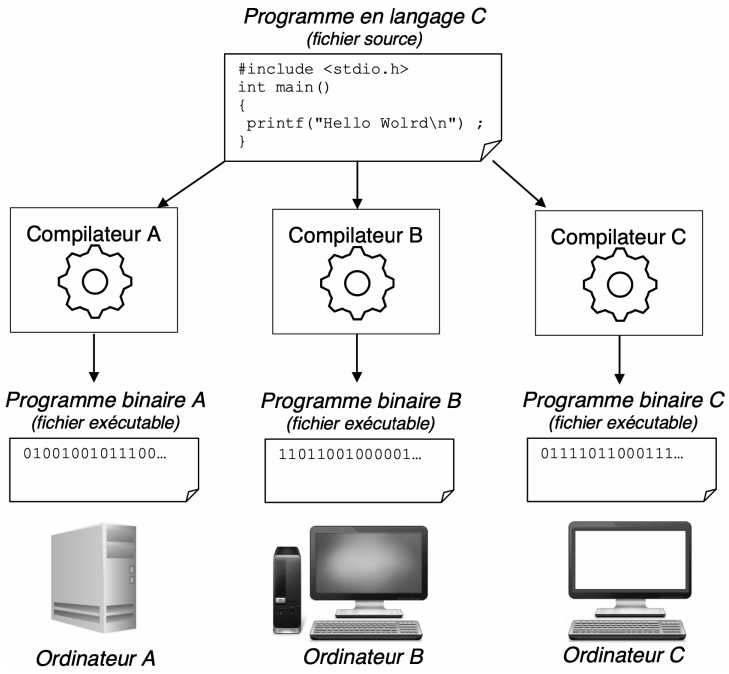

Figure 1.1

Portabilité d'un programme source.

Différents langages de programmation évolués ont vu le jour selon les besoins. Certains sont plus adaptés au développement scientifique, comme le FORTRAN ; d'autres permettent des développements pointus dans les domaines du système ou du réseau, comme le C ; enfin, certains ont été conçus pour faciliter l'écriture de programmes dans l'environnement Internet, comme les langages JAVA ou PHP.

1.3. Les étapes du développement

L'écriture d'un programme n'est que l'étape finale d'un développement qui se déroule en trois phases :

- **L'analyse**. C'est une étape de réflexion sur le ou les problèmes à résoudre. Elle conditionne les traitements futurs.

- **L'algorithme**. C'est l'étape organisationnelle. Les solutions préconisées à l'étape précédente doivent être organisées logiquement.
- **La programmation**. C'est l'étape technique d'écriture dans un langage de programmation.

Cet enchaînement est présenté dans la figure 1.2.

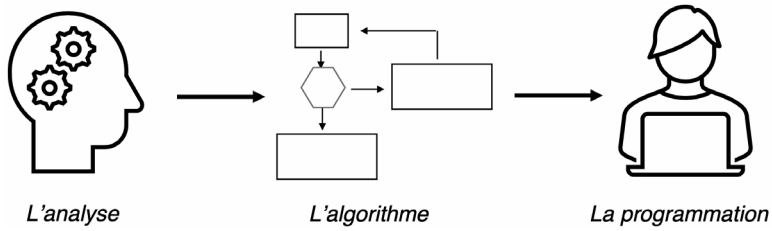

Figure 1.2
Les étapes du développement.

L'analyse

Le but de tout développement est de fournir une solution informatique à un problème donné. Il peut s'agir de la gestion d'un stock de magasin, ou du traitement d'une image satellite. Pour tous ces problèmes, il est nécessaire de cerner le sujet avant d'y apporter une solution. C'est *l'analyse du problème*.

Il existe des méthodes d'analyse, comme MERISE. Elle analyse complètement un problème donné, mais nécessite une mise en œuvre assez lourde. Il est possible d'employer des méthodes plus rapides et adaptées à des problèmes plus simples.

À NOTER

L'analyse propose des solutions qui seront programmées. Une mauvaise analyse produira un programme médiocre ou faux, quelle que soit la qualité de la programmation.

Voici un exemple d'analyse qui évolue au fur et à mesure des questions soulevées, et qui produit à la fin un programme très différent dans sa complexité.

Prenons le problème simple de l'interprétation du numéro de sécurité sociale (sans la clef finale). Le découpage de ce numéro est présenté à la figure 1.3.

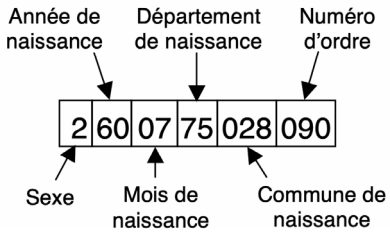

Figure 1.3
Interprétation d'un numéro de sécurité sociale.

Une analyse sommaire permet de faire les constats suivants :

- Il s'agit d'un nombre entier de 13 chiffres (sans la clef finale).
- Le premier chiffre indique le sexe (1=Masculin, 2=Féminin).
- Les deux chiffres suivants représentent l'année de naissance.
- Etc.

À partir de l'exemple de la figure 1.3 on en déduit 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 faux ! Cette personne est née à Créteil dans le département de la Seine qui en 1960 regroupait Paris et les communes limitrophes (Val de Marne, Val d'Oise, …). Il faut donc pousser plus loin l'analyse en intégrant un volet historique et se poser de nouvelles questions :

- Comment a évolué le découpage des départements (Île-de-France, Corse…) ?
- Qu'en est-il des personnes nées dans les anciennes colonies françaises ?
- Comment est réglé le problème des centenaires ?

D'autre part, nous avons supposé que les valeurs du champ sexe sont : 1 (homme) et 2 (femme). Il faut se poser la question de l'existence d'autres valeurs numériques pour homme et femme car il s'agit d'une simple convention de codage.

Enfin l'analyse des numéros de départements montre qu'ils ne sont pas tous constitués de chiffres : les départements corses sont numérotés 2A et 2B. Cette dernière remarque peut changer tout le problème. L'information à traiter qui semblait numérique est peut-être une chaîne de caractères ? Or le traitement d'une chaîne diffère de manière importante du traitement d'un numéro. De même certains numéros de département sont sur 3 chiffres !

L'analyse simpliste du début aurait conduit à écrire un programme ne traitant qu'une partie de la problématique. Le problème est en réalité plus complexe qu'il n'y paraît. L'erreur qu'il ne faut pas commettre lors de l'analyse est d'apporter des solutions basées sur ses certitudes, avant de se poser toutes les questions.

L'algorithme

L'analyse met « à plat » le problème posé et propose des solutions. L'étape suivante consiste à organiser ces solutions, c'est *l'algorithme*.

Il peut être présenté sous la forme de diagrammes ou bien écrit dans un langage de programmation, car c'est déjà un programme ! Sa présentation au travers d'un langage de programmation a l'inconvénient de spécialiser l'écriture des solutions par l'utilisation d'une syntaxe particulière. L'utilisation d'un pseudo-langage est mieux adaptée car la syntaxe y est générique et simplifiée.

À NOTER

Le rôle de l'algorithme est avant tout d'organiser et de structurer l'ensemble des solutions pour aboutir à la solution globale. Il doit présenter la logique aboutissant au résultat final et être dissocié de l'aspect technique des langages informatiques. Une fois écrit, l'algorithme peut être traduit dans n'importe quel langage informatique.

Voici un exemple d'algorithme basé sur l'analyse précédente écrit en pseudo-langage. Seule une petite partie du problème analysé est présentée.

« 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
```

L'écriture et la compilation d'un programme

L'algorithme est déjà le programme. Pour produire le logiciel exécutable il faut :

1. Écrire le *programme source*.

   Il s'agit d'écrire un fichier texte contenant la traduction en C (ou un autre langage) de l'algorithme. Cette phase se nomme *l'édition de texte*.

   Le *fichier source* se nommera par exemple `prog.c`.

2. Produire le *programme objet*.

   Cette étape traduit en binaire le texte précédent grâce à un compilateur.

   C'est l'étape de *compilation*.

   Le *fichier objet* se nommera `prog.o` (sous UNIX/Linux ou MacOS) ou `prog.obj` (sous Windows).

3. Produire le *programme exécutable*.

   Le programme compilé précédent n'est pas encore exécutable.

   En effet, il est possible d'utiliser des sous-programmes déjà traduits en binaire, ce qui évite de tout réécrire. La phase de compilation précédente ne traduit pas ces sous-programmes qui sont stockées sous leur forme binaire dans des fichiers « à tiroirs » qui se nomment des *librairies*. Le fichier résultant de la compilation est incomplet, le code binaire correspondant aux sous-programmes est absent.

   La complétion du fichier objet avec les codes binaires des sous-programmes produit le fichier exécutable, qui est le *logiciel*.

   Cette étape de complétion se nomme *l'édition des liens*.

   Le *fichier exécutable* se nommera alors `prog` (sous UNIX/Linux ou MacOS) ou `prog.exe` (sous Windows).

La figure 1.4 présente ces différentes étapes.

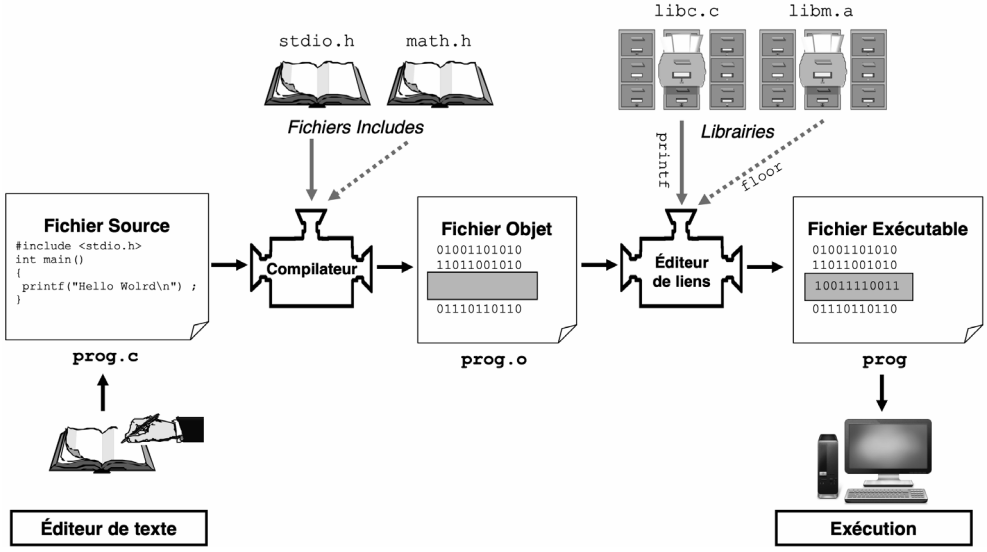

Figure 1.4

L'écriture et la compilation d'un programme.

Nous présentons dans la section 2, les outils de développement dans les trois environnements systèmes : Windows, Linux et MacOS.

L'écriture du programme

Cette phase consiste à écrire un fichier de texte, le *fichier source*, contenant le programme C, via un éditeur de texte (voir section 2).

Le fichier source C `numsecsoc.c` correspond à l'algorithme précédemment.

numsecsoc.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) ;
```