Chap1: Les éléments de base du langage C

3

1) Introduction

Introduction

- L'algorithme est une méthode pour résoudre un problème
- Le programme est le codage lisible par l'ordinateur de cette méthode
- Avant d'écrire un programme, il est nécessaire d'avoir un algorithme

- Les étapes pour résoudre un problème :
  - Analyse → Algorithme → Traduction en langage de programmation

- Le langage de programmation est l'intermédiaire entre l'humain et la machine, il permet d'écrire dans un langage proche de la machine mais intelligible par l'humain les opérations que l'ordinateur doit effectuer.

Introduction au langage C

- Le langage C est un langage évolué et structuré, assez proche du langage machine
- Les compilateurs C possèdent les taux d'expansion les plus faibles de tous les langages évolués (rapport entre la quantité de codes machine générée par le compilateur et la quantité de codes machine générée par l'assembleur et ce pour une même application);
- Les compilateurs C sont remplacés petit à petit par des compilateurs C++.
- Un programme écrit en C est en principe compris par un compilateur C++.
- Le cours qui suit est un cours ce langage C écrit dans un contexte C++.

Ecrire un programme en C (1)

- /* commentaire ignoré par un compilateur */

  ou // généralement réservé pour une ligne de code que l'on veut désactiver temporairement.
- Un programme commence par: **# include <stdio.h>** - une directive de pré-compilation
- Tout programme écrit en C contient exactement une fonction «main».
- Un programme est constitué de fonctions.
- La syntaxe pour une fonction:
  - type de résultat
  - l'identificateur d'une fonction
  - (liste des types de paramètres)

```
int main (void)  // fonction principal
{
}
```

délimitent la fonction (→ { et → }) — *il n'y a pas de paramètres* (↑ void)

Ecrire un programme en C (2)

- Un exemple: écrire bonjour à l'écran

L'en-tête →

Le corps de la fonction →

```
int main (void)  // fonction principal
{
    printf (« bonjour \ n »)  ;
    return 0 ; // provoque la fin du programme
}
```

Ecrire un programme en C (3)

- D'une façon générale, un programme en C se présente de la façon suivante:
  - #..... déclaration de pré-compilation
  - déclaration ou définition d'objets globaux
  - (variables globales)
  - types, prototypes de fonctions
- Des fonctions se présentent de la manière suivante:
  - L'en-tête
  - Corps: déclarations, définitions, objets locaux
- On ne trouve pas de fonctions dans les fonctions!

Etapes pour la création d'un programme exécutable

1- ***Edition du programme source***, à l'aide d'un éditeur (traitement de textes). Le nom du fichier contient l'extension .CPP, exemple: EXI_1.CPP

2- ***Compilation du programme source***, c'est à dire création des codes machine destinés au microprocesseur utilisé. Le compilateur indique les erreurs de syntaxe mais ignore les fonctions-bibliothèque appelées par le programme.

- Le compilateur génère un fichier binaire, non listable, appelé fichier objet: EXI_1.OBJ (commande « compile »).

3- ***Editions de liens***: Le code machine des fonctions-bibliothèque est chargé, création d'un fichier binaire, non listable, appelé fichier exécutable: EXI_1.EXE (commande « build all »).

4- ***Exécution du programme*** (commande « flèche jaune »).

- Les compilateurs permettent en général de construire des programmes composés de plusieurs fichiers sources, d'ajouter à un programme des unités déjà compilées .

=> Le langage C distingue les minuscules, des majuscules. Les mots réservés du langage C doivent être écrits **en minuscules.**

2) Les types de données

Identificateurs

- Permettent de nommer des entités qui seront manipulées par l'ordinateur
  - Variables
  - Fonctions
  - Types
  - Constantes …
- Les identificateurs noms des entités sont écrits à l'aide de caractères suivants:
  - A, b, …z, A, B, … Z
  - 10 chiffres : 0, 1, …. 9
- L'identificateur doit commencer par une lettre
- On ne peut pas utiliser un mot clé de langage comme identificateur (ex: if, main…).

Les types de base

- Toute variable possède un type
- Les types de base sont les suivants:
  - Types « entiers »
    - Char
    - Short int ou short
    - Int
    - Long int ou long
  - Types « réels »
    - Float
    - Double

Les types de données

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32768 à 32767
unsigned short int	Entier court non signé	2	0 à 65535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32768 à 32767 -2147483647 à 2147483647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65535 0 à 4294967295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	2	0 à 4 294 967 295
float	flottant (réel)	4	$3.4*10^{-38}$ à $3.4*10^{38}$
double	flottant double	8	$1.7*10^{-308}$ à $1.7*10^{308}$
long double	flottant double long	10	$3.4*10^{-4932}$ à $3.4*10^{4932}$

Les opérateurs arithmétiques

- Addition: +
- Soustraction: -
- Multiplication: *
- Division: /
- Reste modulo: %
  - La division entre entier donne un coefficient entier

    Ex: int n ;

    n = 7/2 ;

    le résultat est entier: 3
  - Pour avoir 3.5, faire la conversion

    float x ;

    x = (float) 7/2 ;

    la division est une division réelle.

Les opérateurs de comparaison

- Le résultat d'une comparaison est un entier et non un booléen :
  - 0 si le résultat de la comparaison est faux
  - 1 si le résultat de la comparaison est vrai.

  - <
  - >
  - <= pour ≤
  - >= pour ≥
  - = = pour l'égalité
  - ! pour la négation
  - ! = pour la différence

Les caractères

- Pour considérer un caractère en tant que tel on met les apostrophes avant le caractère et après.
  - Ex1: char c ;

```
....
c = 'm' ; ou  c = 109 ;  // ou caractère m est attribué à 109
... printf («exemple %d» , c) ;  // exemple 109 à l'écran
```

imprime la valeur

```
printf(«exemple %c» , c) ;          // exemple m à l'écran
```

imprime le caractère

  - Ex 2: int n, m ;

```
n=2 ; m=3 ;
printf(«bla%d bli %d blu» , n ,m) ;          // bla 2 bli 3 blu
```

Les caractères (2)

- Quelques constantes caractères:
- '\n' interligne
- '\t' tabulation horizontale
- '\v' tabulation verticale
- '\r' retour charriot
- '\f' saut de page
- '\\' backslash
- '\'' cote
- '\"' guillemets

Expressions logiques

- ou : ||
- et : &&
- non: ! (placé avant l'expression à nier)
  - Exemple:

    ....

    int a ; a=2;

    if ((a<3) && (a>1))
- Toute expression possède une valeur.
- Toute **valeur non nulle** est logiquement **assimilée à « vrai »**
- La **valeur nulle** (0) est **assimilée a « faux »**
  - Exemple: l'affectation n=3, possède la valeur 3
- if(n==0) peut s'écrire if (!n)

Exercice:

- Quelle est la valeur de l'expression n==4 dans le programme suivant:

A. int n=3 ;

```
....
if (n==4) … réponse?
if (n) …
n=3;
if (n!=0) …
if (1) …
```

B. int n=3 ;

```
....
if (n=4) …  n=2 ;
printf(« n=%d », n) ;
```

Les opérateurs d'incrémentation

- Les opérateurs d'incrémentation ++ et de décrémentation -- permettent de remplacer les deux instructions suivantes :
  - i = i+1 est remplacée par i ++
  - j = j-1 est remplacée par j--
- Exemple : Que vaut i et j ?
  - i =6 ;
  - j= --i +4 ;
  - j= i++ + 4 ;

Les opérateurs combinés

- Le langage C autorise des écritures simplifiées lorsqu'une même variable est utilisée de chaque côté du signe = d'une affectation. (A éviter pour l'instant)

  - a = a+b; est équivalent à a+= b;
  - a = a-b; est équivalent à a-= b;
  - a = a & b; est équivalent à a&= b;

L'opérateur sizeof()

- L'opérateur sizeof() renvoie la taille en octet de l'opérande ou du type.

```
int i ;
float j ;
sizeof(i); ➔ 2
sizeof(j); ➔ 4
```

- On peut utiliser sizeof avec le nom des types :

```
sizeof(int); ➔ 2
sizeof(float); ➔ 4
```

La conversion de types (1)

- *La conversion implicite ou automatique :*
  - Si on regroupe dans une même expression des types de données différents le compilateur procède à une conversion des types selon la règle suivante:
    - Le type dont le poids est le plus faible est converti au type dont le poids le plus fort.
      Ainsi un char en présence d'un int sera convertit en int (la valeur correspondante sera alors le code ASCII du caractère correspondant),
    - un int en présence d'un float sera convertit en float. Le type dont le poids est le plus fort est le double.
    - Exemple: char a='A';
      int b=32;
      alors a+b sera un int contenant 97 car le char a été convertit en int.
  - Il est aussi important de noter que dans une expression d'affectation le compilateur procède à une conversion implicite en changeant le type de la variable à droite au type de la variable à gauche.
    - Exemple: int a=12;
      float b=a; alors b sera égale à 12.0
      int c=b; alors c sera égale a 12 et la valeur décimale est tronquée.

La conversion de type

- *La conversion explicite ou forcée :*
  - Nous avons vu que le compilateur procède à une conversion quand cette conversion lui paraît logique: int --> float par exemple.
  - Cependant il est tout à fait possible de procéder à la conversion d'un float en un int par exemple.
    Cette conversion est dite une conversion forcée ou explicite.
  - Une conversion forcée peut être achevée en utilisant l'opérateur de cast, la syntaxe d'un cast est la suivante:

(type souhaité) expression

- Exemples:
  - (int)(5.31/5) retournera un int (valeur tronquée), en effet 5.31/5 retourne un float mais puisque un cast est utilisé avant l'expression le float sera converti en int.
  - Pour avoir 3.5, faire la conversion

```
float x ;
 x = (float) 7/2 ;
```

la division est une division réelle.

L'initialisation et les constantes

- Il est possible d'initialiser une variable lors de sa déclaration comme dans :

int n = 15 ;

- Ici, pour le compilateur, n est une variable de type int dans laquelle il placera la valeur 15 ; mais rien n'empêche que cette valeur initiale évolue lors de l'exécution du programme.
  - Notez d'ailleurs que la déclaration précédente pourrait être remplacée par une déclaration ordinaire (int n), suivie un peu plus loin d'une affectation (n=15) ; la seule différence résiderait dans l'instant où n recevrait la valeur 15.

- En fait, il est possible de déclarer que la valeur d'une variable ne doit pas changer lors de l'exécution du programme. Par exemple, avec :

const int n = 20 ;

- on déclare n de type int et de valeur (initiale) 20 mais, de surcroît, les éventuelles instructions modifiant la valeur de n seront rejetées par le compilateur.

- Il est possible d'utiliser dans l'entête de la source:

#define n 20

3) L'affichage

La fonction printf

- Ce n'est pas une instruction du langage C, mais une fonction de la bibliothèque stdio.h. Au début de tout programme qui devra utiliser ces fonctions nous devrons inclure au début la directive suivante: #include <stdio.h>
- *Exemple: affichage d'un texte:*

```
printf("BONJOUR");   /* pas de retour à la ligne du curseur apres l'affichage, */
printf("BONJOUR\n");      /* affichage du texte, puis retour à la ligne du curseur.
                                                                          */
```

- *La fonction printf exige l'utilisation de formats de sortie, avec la structure suivante:*

```
printf("%format",nom_de_variable);
```

- Quelques symboles représentant les formats les plus utilisés :

Symbole	Type pris en charge	Type
%d ou %i	entier	int
%c	caractère	char
%s	chaîne de caractères	char
%f	rationnel	float

Les autres fonctions

- Affichage d'un caractère:
  - La fonction **putchar** permet d'afficher un caractère:
  - c étant une variable de type **char**, l'écriture **putchar(c);** est équivalente à **printf("%c\n",c);**
- Affichage d'un texte:
  - La fonction **puts** permet d'afficher un texte:
  - l'écriture **puts("bonjour");** est équivalente à **printf("bonjour\n");**

4) La saisie

La fonction getchar()

- La fonction getchar permet la récupération d'un seul caractère à partir du clavier. La syntaxe d'utilisation de getchar est la suivante:

```
variable=getchar();
```

- Notez que variable doit être de type char et les parenthèses après getchar sont obligatoires.

Exemple:

```
#include
main()
  {
  char c;
  c=getchar();
  }
```

La fonction scanf()

- Que faire pour pouvoir entrer une chaîne complète ou même un nombre?
  - Solution: la fonction scanf.
  - En effet elle permet de récupérer n'importe quelle combinaison de valeurs numériques, de caractères et de chaînes. La syntaxe d'utilisation de scanf est la suivante:

scanf("chaîne de contrôle", &var1, &var2, ..., &varN);

- " chaîne de contrôle" représente *certaines données de formatage*
- "var1, var2,..., varN" représentent les variables à remplir.

La fonction scanf()

- *La chaine de contrôle:*
  - *entourée de guillemets et constitue le groupe de caractères chaque groupe étant en relation avec* une des variables mentionnées.
  - Le groupe devra commencer par le signe % suivi d'un caractère bien défini (le type) indiquant si la variable est un caractère, un nombre (soit entier soit décimal) ou une chaîne (voir le tableau des formats présentés auparavant).
  - Si la chaîne contient plus qu'un groupe, les formats sont souvent séparés par des espaces et ainsi les données entrées doivent être séparées par des espaces. Le type de chaque variable doit correspondre au type de son groupe et chaque variable doit être précédée du signe & sauf si elle est une chaîne.

La fonction scanf() : Exemples

1) i est un entier. Pour obtenir i de l'utilisateur on écrit:
scanf("%d",&i);
sur le clavier nous tapons: -->> 123
2) a est un caractère:
scanf("%c",&a);
-->> d
3) ville est une chaîne:
scanf("%s",ville);
|---------- pas de & car ville est une chaîne.

-->> beyrouth
4) i et j sont deux entiers. i ne doit pas être plus large que trois chiffres:
scanf("%3d %d",&i,&j);
-->> 231 1024
|---------- notez l'espace séparateur