Introduction au langage C

6 janvier 2010

Table des matières

Chapitre 1

Entrées et sorties de base

Les fonctions d'entrées/sorties assurent la communication entre la machine et le monde extérieur (écran, autre périphériques de sortie). Ces fonctions ne font pas partie intégrante du langage ; elles sont regroupées dans la bibliothèque standard `<stdio>`. Les plus importantes sont :

1.1 Ecriture formatée de données : la fonction printf()

La fonction `printf()` est utilisée pour transférer du texte, des valeurs de variables ou des résultats d'expressions vers le fichier de sortie standard `stdout` (par défaut l'écran). Sa syntaxe est la suivante :

- Ecriture formatée en langage algorithmique :
  - `écrire <Expression1>,<Expression2>, ...`
- Ecriture formatée en langage C : `printf(<format>,<Expr1>,<Expr2>, ...)`
  - <format> : format de représentation
  - <Expr1>,... variables et expressions dont les valeurs sont à représenter

L'expression `<format>` est en fait une chaîne de caractères qui peut contenir :

- du texte ;
- des séquences d'échappement ;
- des spécificateurs de format : les spécificateurs de format indiquent la manière dont les valeurs des expressions <Expr1..N> sont imprimées.

L'expression `<format>` contient exactement un spécificateur de format pour chaque expression <Expr1..N>. Les spécificateurs de format :

- commencent toujours par le symbole % et se terminent par un ou deux caractères qui indiquent le format d'impression ;
- impliquent une conversion d'un nombre en chaîne de caractères. Ils sont encore appelés symboles de conversion.

Exemple 1 :

`La suite d'instructions:`

```
int A = 1234;
int B = 567;
printf("%i fois %i est %li\n", A, B, (long)A*B);
```

va afficher sur l'écran : `1234 fois 567 est 699678`

Les arguments de `printf` sont :

- la partie format : "%i fois %i est %li" ;
  - Le 1er spécificateur (%i) indique que la valeur de A sera imprimée comme entier relatif $\Longrightarrow$ 1234 ;
  - Le 2e spécificateur (%i) indique que la valeur de B sera imprimée comme entier relatif $\Longrightarrow$ 567 ;
  - Le 3e spécificateur (%li) indique que la valeur de (long)A*B sera imprimée comme entier relatif long $\Longrightarrow$ 699678.
- la variable : A ;
- la variable : B ;
- l'expression : (long)A*B.

Exemple 2 :

```
La suite d'instructions:
   char B = 'A';
   printf("Le caractère %c a le code %i !\n", B, B);
```

va afficher sur l'écran : `Le caractère A a le code 65 !` La valeur de B est donc affichée sous deux formats différents :

- %c comme caractère : A
- %i comme entier relatif : 65

1.1.1 spécificateurs de format pour printf()

1.1.2 Arguments du type long

Les spécificateurs %d, %i, %u, %o, %x peuvent seulement représenter des valeurs du type `int` ou `unsigned int`. Une valeur trop grande pour être codée dans deux octets est coupée sans avertissement si nous utilisons %d. Pour pouvoir traiter correctement les arguments du type `long`, il faut utiliser les spécificateurs %ld, %li, %lu, %lo, %lx.

Exemple :

long N = 1500000 ;
- printf("%ld, %lx", N, N) ; $\Longrightarrow$ 1500000, 16e360
- printf("%x, %x" , N) ; $\Longrightarrow$ e360, 16
- printf("%d, %d" , N) ; $\Longrightarrow$ -7328, 22

Spécificateur	**Type**	**Exemple**	**Sortie à l'écran**
c	Caractère	printf("%c",'A')	A
d ou i	Entier décimal	printf("%d",17)	17
e	Notation scientifique	printf("%e",8.1)	81E+01
f	Notation flottante	printf("%f",8.1)	8.1
g	Flottant ou scientifique	printf("%g",8.1)	8.1
o	Entier octal	printf("%o",17)	21
x	Entier hexadécimal	printf("%x",17)	11
s	Chaînes de caractères	printf("%s","hello")	hello
u	Entier non signé	printf("%u",17)	17

1.1.3 Arguments rationnels

Les spécificateurs %f et %e peuvent être utilisés pour représenter des arguments du type `float` ou `double`. La mantisse des nombres représentés par %e contient exactement un chiffre (non nul) devant le point décimal. Cette représentation s'appelle la notation scientifique des rationnels. Pour pouvoir traiter correctement les arguments du type long `double`, il faut utiliser les spécificateurs %Lf et %Le.

Exemple :

float N = 12.1234 ;
double M = 12.123456789 ;
long double P = 15.5 ;
– printf("%f", N) ; $\Longrightarrow$ 12.123400
– printf("%f", M) ; $\Longrightarrow$ 12.123457
– printf("%e", N) ; $\Longrightarrow$ 1.212340e+01
– printf("%e", M) ; $\Longrightarrow$ 1.212346e+01
– printf("%Le", P) ; $\Longrightarrow$ 1.550000e+01

1.1.4 Largeur minimale pour les entiers

Pour les entiers, nous pouvons indiquer la largeur minimale de la valeur à afficher. Dans le champ ainsi réservé, les nombres sont justifiés à droite.

Exemples :

(_ $\leftrightarrow$ position libre)
– printf("%4d", 123) ; $\Longrightarrow$ _123
– printf("%4d", 1234) ; $\Longrightarrow$ 1234

– printf("%4d", 12345) ; $\Longrightarrow$ 2345
– printf("%4u", 0) ; $\Longrightarrow$ _ _ _0
– printf("%4X", 123) ; $\Longrightarrow$ _ _7B
– printf("%4x", 123) ; $\Longrightarrow$ _ _7b

1.1.5 Largeur minimale et précision pour les rationnels

Pour les rationnels, nous pouvons indiquer la largeur minimale de la valeur à afficher et la précision du nombre à afficher. La précision par défaut est fixée à six décimales. Les positions décimales sont arrondies à la valeur la plus proche.

Exemples :

– printf("%f", 100.123) ; $\Longrightarrow$ 100.123000
– printf("%12f", 100.123) ; $\Longrightarrow$ _ _100.123000
– printf("%.2f", 100.123) ; $\Longrightarrow$ 100.12
– printf("%5.0f", 100.123) ; $\Longrightarrow$ _ _100
– printf("%10.3f", 100.123) ; $\Longrightarrow$ _ _ _100.123
– printf("%.4f", 1.23456) ; $\Longrightarrow$ 1.2346

Exercice

```
#include <stdio.h>
main()
{
   int N=10, P=5, Q=10, R;
   char C='S';

   N = 5; P = 2;
   Q = N++ > P || P++ != 3;
   printf ("C : N=%d P=%d Q=%d\n", N, P, Q);

   N = 5; P = 2;
   Q = N++ < P || P++ != 3;
   printf ("D : N=%d P=%d Q=%d\n", N, P, Q);

   N = 5; P = 2;
   Q = ++N == 3 && ++P == 3;
   printf ("E : N=%d P=%d Q=%d\n", N, P, Q);

   N=5; P=2;
   Q = ++N == 6 && ++P == 3;
   printf ("F : N=%d P=%d Q=%d\n", N, P, Q);

   N=C;
```

```
   printf ("G : %c %c\n", C, N);
   printf ("H : %d %d\n", C, N);
   printf ("I : %x %x\n", C, N);
   return 0;
}
```

1. Sans utiliser l'ordinateur, trouvez et notez les résultats du programme ci-dessus.
2. Vérifiez vos résultats à l'aide de l'ordinateur.

1.2 Lecture formatée de données : la fonction scanf()

La fonction `scanf()` est la fonction symétrique à `printf()` ; elle nous offre pratiquement les mêmes conversions que `printf()`, mais en sens inverse. La syntaxe de la fonction est la suivante :

- Lecture formatée en langage algorithmique :
  - `lire <NomVariable1>,<NomVariable2>, ...`
- Lecture formatée en langage C : `scanf("<format>",<AdrVar1>,<AdrVar2>, ...)`
  - `<format>` : format de lecture des données ;
  - `<AdrVar1>`,... : adresses des variables auxquelles les données seront attribuées.

1. La fonction `scanf()` reçoit ses données à partir du fichier d'entrée standard `stdin` (par défaut le clavier) ;
2. La chaîne de format détermine comment les données reçues doivent être interprétées ;
3. Les données reçues correctement sont mémorisées successivement aux adresses indiquées par `<AdrVar1>,...` ;
4. L'adresse d'une variable est indiquée par le nom de la variable précédé du signe & ;
5. La fonction `scanf()` retourne comme résultat le nombre de données correctement reçues (type int).

Exemple :

```
La suite d'instructions:
   int jour, mois, annee;
   scanf("%i %i %i", &jour, &mois, &annee);
```

Lit trois entiers relatifs, séparés par des espaces, tabulations ou interlignes. Les valeurs sont attribuées respectivement aux trois variables jour, mois et annee.

1.2.1 Spécificateurs de format pour scanf()

Le symbole * indique que l'argument n'est pas une variable, mais l'adresse d'une variable de ce type (c.-à-d. : un pointeur sur une variable - voir chapitre sur 'Les pointeurs').

Spécificateur	Type de valeur lue	Type en langage C
c	Caractère	char*
s	Chaînes de caractères	char*
d ou i	Entier décimal	int*
e	Notation scientifique	float*
f	Notation flottante	float*
u	Entier non signé	int*

1.2.2 Le type long

Pour lire une donnée du type `long`, il faut se servir des spécificateurs %ld, %li, %lu, %lo, %lx. (Sinon, le nombre est simplement coupé à la taille de int).

1.2.3 Le type double

Pour lire une donnée du type `double`, il faut se servir des spécificateurs %le ou %lf.

1.2.4 Le type long double

Pour lire une donnée du type `long double`, il faut se servir des spécificateurs %Le ou %Lf.

1.2.5 Indication de la largeur maximale

Pour tous les spécificateurs, nous pouvons indiquer la largeur maximale du champ à évaluer pour une donnée. Les chiffres qui passent au-delà du champ défini sont attribués à la prochaine variable qui sera lue !

Exemple :

Soient les instructions :

```
int A,B ;
scanf("%4d %2d", &A, &B) ;
```

Si nous entrons le nombre 1234567, nous obtiendrons les affectations suivantes :
A=1234 ;
B=56 ;

Le chiffre 7 sera gardé pour la prochaine instruction de lecture.

1.2.6 Les signes d'espacement

Lors de l'entrée des données, une suite de signes d'espacement (espaces, tabulateurs, interlignes) est évaluée comme un seul espace. Dans la chaîne de format, les symboles `\t, \n, \r` ont le même effet qu'un simple espace.

Exemple :

Pour la suite d'instructions

```
int jour, mois, annee;
scanf("\%i/\%i/\%i", \&jour, \&mois, \&annee);
```

Accepte les entrées	**Rejette les entrées**
12/4/1980	12 4 1980
12/04/01980	12 /4 /1980

1.2.7 Nombre de valeurs lues

Lors de l'évaluation des données, `scanf()` s'arrête si la chaîne de format a été travaillée jusqu'à la fin ou si une donnée ne correspond pas au format indiqué. `Scanf()` retourne comme résultat le nombre d'arguments correctement reçus et affectés.

Exemple 1 :

La suite d'instructions :

```
int jour, mois, annee, recu;
recu = scanf("\%i \%i \%i", \&jour, \&mois, \&annee);
```

réagit de la façon suivante (- valeur indéfinie) :

Introduit	**Reçu**	**Jour**	**Mois**	**Année**
12 4 1980	3	12	4	1980
12/4/1980	1	12		
12.4 1980	1	12		
12 4 19.80	3	12	4	19

Exercice

En vous référant aux exemples précédent, écrivez un programme qui lit la date du clavier et écrit les données ainsi que le nombre de données correctement reçues sur l'écran.

Exemple 2 :

Introduisez la date (jour mois année) : 11 11 2009

données reçues : 3
jour : 11
mois : 11
année : 2009

Exercice

- Testez les réactions du programme à vos entrées. Essayez d'introduire des nombres de différents formats et différentes grandeurs ;
- Changez la partie `format` du programme de façon à séparer les différentes données par le symbole '-' .

1.3 Écriture d'un caractère : la fonction putchar('a')

La fonction `putchar('a')` transfère le caractère `a` vers le fichier standard de sortie `stdout`. Les arguments de la fonction putchar sont ou bien des caractères, c'est à dire des nombres entiers entre 0 et 255, ou bien le symbole `EOF` (End Of File).

`EOF` est une constante définie dans `<stdio>` qui marque la fin d'un fichier. La commande `putchar(EOF)` est utilisée dans le cas où `stdout` est dévié vers un fichier.

Type de l'argument

Pour ne pas être confondue avec un caractère, la constante `EOF` doit nécessairement avoir une valeur qui sort du domaine des caractères (en général `EOF` a la valeur `-1`). Ainsi, les arguments de `putchar` sont par définition du type `int` et toutes les valeurs traitées par `putchar` (même celles du type `char`) sont d'abord converties en `int`.

Exemples :

```
char A = 225;
char B = '\a';
int  C = '\a';
putchar('x');   /* afficher la lettre x      */
putchar('?');   /* afficher le symbole ?     */
putchar('\n');  /* retour à la ligne         */
```

```
putchar(65);    /* afficher le symbole du code 65  (ASCII: 'A')  */
putchar(A);     /* afficher la lettre  dont le code est 225 (ASCII: 'ß')  */
putchar(B);     /* beep sonore               */
putchar(C);     /* beep sonore               */
putchar(EOF);   /* marque la fin du fichier */
```

1.4 Lecture d'un caractère : la fonction getchar()

Une fonction plus souvent utilisée que `putchar()` est la fonction `getchar()`, qui lit le prochain caractère du fichier d'entrée standard `stdin`.

Type du résultat

Les valeurs retournées par `getchar()` sont ou bien des caractères compris entre 0 et 255 ou bien le symbole `EOF`. Comme la valeur du symbole `EOF` sort du domaine des caractères, le type résultat de `getchar()` est `int`. En général, `getchar()` est utilisé dans une affectation :

```
int C;
C = getchar();
```

`getchar()` lit les données de la zone tampon de `stdin` et fournit les données seulement après confirmation par pression de lat touche Entrée du clavier. La bibliothèque `<conio>` contient une fonction du nom de `getch()` qui fournit immédiatement le prochain caractère entré au clavier.

La fonction `getch()` n'est pas compatible avec ANSI C et elle peut seulement être utilisée sous MS-DOS.

Exercice1

Ecrire un programme qui lit un caractère au clavier et affiche le caractère ainsi que son code numérique :

– en employant getchar() et printf(),
– en employant getch() et printf().

Exercice2

1. Ecrire un programme qui permute et affiche les valeurs de trois variables A, B, C de type entier qui sont entrées au clavier : A $\Longrightarrow$ B , B $\Longrightarrow$ C , C $\Longrightarrow$ A ;
2. Ecrire un programme qui affiche le quotient et le reste de la division entière de deux nombres entiers entrés au clavier ainsi que le quotient rationnel de ces nombres ;

3. Ecrire un programme qui calcule et affiche l'aire S^2 d'un triangle dont il faut entrer les longueurs des trois côtés. Utilisez la formule : S^2 = P(P-A)(P-B)(P-C) où A, B, C sont les longueurs des trois côtés (type int) et P le demi-périmètre du triangle ;
4. Ecrire un programme qui calcule la somme de quatre nombres du type int entrés au clavier,
   - en se servant de 5 variables (mémorisation des valeurs entrées) ;
   - en se servant de 2 variables (perte des valeurs entrées).
5. – Ecrire un programme qui calcule le prix TTC (type double) d'un article à partir du prix net (type int) et du pourcentage de TVA (type int) à ajouter. Utilisez la formule suivante en faisant attention aux priorités et aux conversions automatiques de type :
   - Ecrire un programme qui calcule le prix net d'un article (type double) à partir du prix TTC (type double) et du pourcentage de TVA (type int) qui a été ajoutée.(Déduisez la formule du calcul de celle indiquée ci-dessus)
6. Ecrire un programme qui calcule et affiche la distance DIST (type double) entre deux points A et B du plan dont les coordonnées (XA, YA) et (XB, YB) sont entrées au clavier comme entiers.