Chapitre 2 : Données (constantes et variables)

Informatique de base

2013-2014

Sup Galilée

Introduction

Un programme en langage C manipule des données. Une donnée possède un type et une valeur, elle peut avoir un nom et peut être modifiable (variable) ou constante. La valeur d'une donnée modifiable peut bien sûr évoluer au cours de l'exécution d'un programme, alors que celle d'une donnée constante ne le peut pas !

Une donnée variable peut être interprétée comme une case de la mémoire accessible à partir du nom de la donnée et qui contient la valeur de cette donnée.

Introduction

Une valeur peut être :

1. Un nombre entier ou un nombre flottant;
2. Une valeur composée (une structure ou un tableau);
3. Un pointeur (adresse d'une case mémoire).

Introduction

Les valeurs sont classées par types. Chaque valeur a un et un seul type.

En langage C il n'existe pas de type spécifique pour les booléens (vrai ou faux), les caractères (`'a'`, `'B'`, `'3'`, `'&'`, ...) et les chaînes de caractères (`"hello"`, ...). Le booléen "faux" est codé par l'entier 0 et le booléen "vrai" par toute valeur non nulle. Un caractère est codé par un nombre entier (code ASCII, par exemple le caractère `'a'` est représenté par 97). Une chaîne de caractères est représentée quant à elle par un tablau de caractères.

Dans ce chapitre nous étudierons les types numériques.

Types numériques : les entiers

1. `char`, codé sur 8 bits, permet de représenter les entiers entre -128 et 127.
2. `unsigned char`, codé sur 8 bits, permet de représenter les entiers entre 0 et 255.
3. `short`, codé sur 16 bits, permet de représenter les entiers entre -32 768 et 32 767.
4. `unsigned short`, codé sur 16 bits, permet de représenter les entiers entre 0 et 65 535.
5. `long`, codé sur 32 bits, permet de représenter les entiers entre -2 147 483 648 et 2 147 483 647.
6. `unsigned long`, codé sur 32 bits, permet de représenter les entiers entre 0 et 4 294 967 296.
7. `int` : rassemble les entiers qui peuvent être implantés de la façon la plus efficace sur la machine. Souvent équivalent au type `long`.

Types numériques : les flottants

Parmi les types de flottants on distingue :

1. float, codé sur 32 bits, permet de représenter les nombres rationnels de $-1,7 \times 10^{38}$ à $-0,29 \times 10^{-38}$ et de $0,29 \times 10^{-38}$ à $1,7 \times 10^{38}$ avec sept chiffres décimaux significatifs.

2. double, codé sur 64 bits, permet de représenter les nombres rationnels de $-0,9 \times 10^{308}$ à $-0,56 \times 10^{-308}$ et de $0,56 \times 10^{-308}$ à $0,9 \times 10^{308}$ avec quinze chiffres décimaux significatifs.

Les constantes littérales

Les nombres positifs, les caractères et les chaînes de caractères possèdent une représentation explicite que l'on appelle constante littérale.

Les nombres négatifs n'ont pas à strictement parler de représentation littérale. Ils sont obtenus en appliquant l'opérateur unaire $-$ à leur valeur absolue (exemple, `-3`).

Les constantes littérales : nombres entiers

Une constante littérale entière est soit 0, soit une suite de chiffres $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$ dont le premier est différent de 0.

Par exemple 0 et 1993.

Une constante littérale entière est la représentation d'une valeur du plus petit des trois types `int`, `long` ou `unsigned long` qui contient cette valeur. Par exemple, 141 946 représente une valeur de type `int` et 3 456 789 012 représente une valeur du type `unsigned long`.

Les constantes littérales : nombres flottants

Une constante littérale flottante est formée :

1. d'une partie entière : une suite de chiffres,
2. suivie d'un point,
3. suivi d'une partie fractionnaire : une suite de chiffres,
4. suivi d'un exposant : la lettre `e` ou `E`, suivie du signe `+` ou `-`, suivi d'une suite de chiffres.

On peut omettre :

1. la partie entière ou la partie fractionnaire, mais pas les deux,
2. le point ou l'exposant, mais pas les deux.

Par exemple `2731.5e-1` représente 273,15, `.15` représente 0,15, et `2e3` représente 2000.

Une constante littérale flottante est de type `double`.

Les constantes littérales : caractères

Une constante littérale a l'une des deux formes suivantes :

1. Un caractère imprimable entre apostrophes. Par exemple : `'a'`, `'1'`, `'?'`, ...

2. Un caractère précédé du caractère \ ("anti-slash" ou "caractère d'échappement") : \n (nouvelle ligne), \t (tabulation), \b (retour arrière), \r (retour chariot), \f (saut de page), \a (signal sonore), et \c (où c est différent de n, t, b, r, f ou a).

Par exemple, le caractère guillemet peut se représenter par `\"`.

Une constante littérale caractère est de type `char`. Par exemple si les caractères sont codés en ASCII, le caractère `'A'` représente l'entier 65 de type `char`.

Les constantes littérales : caractères et code ASCII

Le code ASCII des chiffres de '0' à '9' correspond aux entiers de 48 à 57.

Le code ASCII des lettres majuscules de 'A' à 'Z' correspond aux entiers de 65 à 90.

Le code ASCII des lettres majuscules de 'a' à 'z' correspond aux entiers de 97 à 122.

Les constantes littérales : chaînes de caractères

Une constante littérale chaîne de caractères est formée par la suite de ses caractères, placée entre guillemets.

Par exemple : `"Bonjour !"`, `"Il dit : \"Bonjour !\""`
(Notons dans le dernier exemple que le caractère d'échappement '\' qui permet de distinguer les guillemets appartenant à la chaîne de caractères de ceux servant à la délimiter.)

Une constante littérale chaîne de caractères est la représentation d'une valeur de type " pointeur vers un caractère " `char *` dont on reparlera plus tard...

Noms de variables

Une variable est une donnée modifiable définie par son nom : un identificateur et par le type des valeurs qu'elle peut prendre (que l'on peut lui affecter).

Il y a quelques règles à respecter pour définir un identificateur (un nom) de variable : un identificateur est une suite de caractères dont chacun peut être une lettre (majuscule ou minuscule), un chiffre ou le caractère '_' et dont le premier caractère est une lettre. Par exemple : `x`, `y1`, `ma_variable`

Rajoutons que les mots suivants qui jouent un rôle spécifique dans la syntaxe de C, ne peuvent pas être utilisés comme identificateurs : `auto`, `break`, `case`, `char`, `const`, `continue`, `default`, `do`, `double`, `else`, `enum`, `extern`, `float`, `for`, `goto`, `if`, `int`, `long`, `register`, `return`, `short`, `signed`, `sizeof`, `static`, `struct`, `switch`, `typedef`, `union`, `unsigned`, `void`, `volatile`, `while`.

Déclaration de variables

Les variables manipulées par un programme doivent être déclarées (introduites) avant leur utilisation dans le texte source du programme. **Nous ne traiterons dans ce chapitre que le cas des variables numériques** (la déclaration de variables de type structure, tableau et pointeur sera étudié ultérieurement).

Une déclaration de variables numériques a la forme suivante :

`T ident1 = exp1, ..., identn = expn;`

où

- `T` est l'un des types numériques vus précédemment,
- `ident1, ..., identn` sont des identificateurs (noms de variables),
- `exp1, ..., expn` sont des expressions.

Cette déclaration définit n variables de type `T`, nommées `ident1, ..., identn` et dont les valeurs initiales sont les valeurs des expressions `exp1, ..., expn`.

Notons que l'on peut déclarer une variable sans lui donner de valeur initiale (exemple : `int x;`). Sa valeur est alors indéterminée.

Déclaration de variables : exemples

```
int i, j, j; /*Déclaration de trois variables entières sans valeur initiale*/

float longueur = 10.5, largeur = 6.3;

char une_lettre = 'o';
```

Déclaration de variables : variables constantes (!)

Il est possible de fixer la valeur d'une variable en préfixant sa déclaration par le mot-clef `const`.

Cette variable se comporte alors comme une constante : sa valeur, obligatoirement fournie dans la déclaration, ne pourra pas être modifiée au cours de l'exécution du programme.

Par exemple, la déclaration

```
const float pi = 3.1415926;
```

définit une donnée constante, de nom `pi`, de type `float`, et de valeur `3.1415926`.

Environnement et état

Dans la suite de cours nous appellerons

- environnement, l'ensemble des données visibles en un point d'un programme;
- état, l'ensemble des valeurs des données à un instant de l'exécution d'un programme.

Il est important de noter que l'environnement est de nature statique alors que l'état est de nature dynamique. L'environnement dans lequel sera évaluée une expression ou exécutée une instruction peut être déterminé par la seule lecture du code source. Par contre, l'état d'un programme dépend de l'historique de son exécution.

Environnement et état : exemple

```
(01) #include <stdio.h>
(02) /* Maximum de deux nombres entiers */
(03) main ()
(04)   {
(05)   int x, y, max;
(06)   printf("x ?  ");
(07)   scanf("%d",&x);
(08)   printf("y ?  ");
(09)   scanf("%d",&y);
(10)   if (x >= y)
(11)     max = x;
(12)   else
(13)     max = y;
(14)   printf("max = %d", max);
(15)   }
```

Environnement et état : exemple

L'environnement établi par la ligne 5 est formé des variables `x`, `y` et `max`.

Si les valeurs attribuées à `x` et `y` (lignes 7 et 9) sont 32 et 55, alors l'état du programme après l'exécution de l'instruction de l'instruction `if` est `x` a la valeur 32, `y` a la valeur 55 et `max` a la valeur 55.

Représentation graphique de l'environnement et de l'état

Pour mieux comprendre et étudier le déroulement d'un programme en langage C nous utiliserons une représentation graphique pour l'environnement et l'état.

- une donnée constante de nom `c` et de valeur *v* est représentée par :
  `constante  c  v`
- une donnée modifiable de nom `x` et de valeur *v* est représentée par :
  `variable  x  v`
  Si la variable `x` ne possède pas de valeur, alors on se contente d'écrire
  `variable  x  _`
- une fonction de nom `f` est représentée par `fonction  f`

Dans la suite on pourra également donner le type des données dans un environnement ou un état.

Représentation graphique de l'environnement et de l'état : exemple

Par exemple, l'environnement dans lequel s'exécute le programme du calcul du maximum de deux entiers du premier chapitre est le suivant

Environnement local :

```
variable    max    _
variable    y    _
variable    x    _
```

Environnement global :

```
fonction    main ()
fonction    printf ()
fonction    scanf ()
```

Représentation graphique de l'environnement et de l'état : exemple

La représentation graphique de l'état d'un programme à un instant donné est obtenue en indiquant les valeurs associées à chaque variable à cet instant de l'exécution du programme.

Par exemple l'état du programme de calcul du maximum après l'exécution de l'instruction `if` (en supposant que `x` ait la valeur 32 et `y` la valeur 55) est le suivant :

Environnement local :

```
variable    max    55
variable    y    55
variable    x    32
```

Environnement global :

```
fonction    main ()
fonction    printf ()
fonction    scanf ()
```

Représentation graphique de l'environnement et de l'état : visibilité

Nous verrons plus tard dans le cours et en détail la notion de visibilité des variables. Mais en voici dès à présent quelques idées.

Le langage C offre la possibilité de donner le même nom à des variables distinctes. Ainsi lorsqu'on utilise le nom on doit savoir à quelle variable il se réfère : c'est ce que l'on appelle la visibilité (ou portée) des variables.

On appelle bloc un ensemble d'expressions du langage C écrites entre des accolades { et }.

Représentation graphique de l'environnement et de l'état : visibilité

Les variables déclarées au niveau du fichier source, en dehors de tout bloc, sont dites globales. Elles sont visibles depuis tout point du programme compris entre leur déclaration et la fin du programme, sauf si elles sont masquées par une variable locale ou un argument formel d'une fonction (nous verrons cette notion plus tard). Les variables déclarées dans un bloc sont dites locales à ce bloc. Une variable locale est visible dans le bloc où elle est déclarée (à partir de sa déclaration) ainsi que dans tout bloc inclus sauf si elle est masquée par une variable du même nom dans un sous-bloc.

Les arguments formels d'une fonction (ses paramètres d'entrée) sont considérés comme des variables locales au bloc que constitue le corps de la fonction. Elles sont donc visibles dans tout le corps de cette fonction sauf si elles sont masquées par des variables de même nom.

Représentation graphique de l'environnement et de l'état : visibilité

Une donnée non visible (ou masquée) en un point du programme est représentée comme suit par exemple :

```
variable    x    (155)
```

Autrement dit on écrit sa valeur entre parenthèses pour signifier qu'elle ne peut pas être utilisée à cet instant du programme (la variable est masquée).

Recherche de la valeur d'une donnée

La valeur d'une donnée de nom `n` à un instant donné du déroulement d'un programme est obtenue en recherchant la valeur de la première donnée visible de nom `n`, en parcourant la pile représentant l'état du programme à cet instant, depuis son sommet vers son fond.

Dans l'exemple :

```
variable    x    (158)
variable    y    3
variable    x    155
```

la valeur de la variable `x` est 155 et non 158 (car à cet instant de l'exécution du programme la variable `x` de valeur 158 est masquée).