Université de Nice-Sophia Antipolis
Polytech ELEC3

mardi 14 novembre 2017
Durée: 2h

Variables, blocs et énoncés conditionnels
Travaux Dirigés – Séance n. 3

---

1 Identificateurs

Un identificateur est un nom qui permet de désigner une entité du programme (variables, constante, fonctions, ...). Les noms choisis pour les identificateurs doivent être *significatifs*.

En langage C, les identificateurs peuvent être aussi longs que l'on désire, toutefois le compilateur ne tiendra compte que des 32 premiers caractères. De plus, ils doivent répondre à certaines règles :

- un identificateur doit commencer par une lettre (majuscule ou minuscule) ou un « _ » (pas par un chiffre) ;
- un identificateur peut comporter des lettres, des chiffres et le caractère « _ » (les espaces ne sont pas autorisés !)
- un identificateur ne peut pas être un mot réservé (*e.g.* **if**, **int**, *etc*).

2 La déclaration des variables

Le langage C est un langage typé, c'est-à-dire qu'il impose que les variables, avant leur utilisation, soient déclarées avec le type des valeurs qu'elles pourront désigner.

```
type Nom_de_la_variable;
```

ou bien s'il y a plusieurs variables du même type :

```
type Nom_de_la_variable1, Nom_de_la_variable2, ...;
```

```
int nombre_etudiants;
float delta, racine1, racine2;
```

3 Affectation d'une valeur à une variable

Il est possible d'associer une valeur à une variable grâce à un opérateur d'*affectation*. Le langage C propose plusieurs opérateurs d'affectation. Le plus simple est « = » :

```
nombre_etudiants = 78;
delta = 234.123 / 3.14;
```


3.1 Initialisation d'une variable

La déclaration d'une variable ne fait que « réserver » un emplacement mémoire où ranger une valeur. Tant que l'on ne lui a pas affecté une valeur celle-ci contient ce qui se trouvait précédemment à cet emplacement.

Le langage C permet d'associer une valeur initiale à la variable au moment de sa déclaration, on parle alors d'initialisation :

```
type Nom_de_la_variable = valeur;
```

Par exemple :

```
float longueur = 125.36;
```

4 Blocs d'instructions

Un bloc d'instructions est le groupement de plusieurs instructions entre deux accolades {}. Comme dans une expression, les parenthèses ouvrantes et fermantes permettent de délimiter une sous-expression qui sera vue comme un opérande unique. Un bloc d'instructions entre accolades sera vu comme une instruction unique.

Il est possible de déclarer en tête du bloc des variables qui ne seront visibles que dans celui-ci.

Voyons deux exemples simples pour illustrer cela :

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int a=3;
  int b=4;
  int c=a+b;
  int d=2*c;

  printf("c=%d\n",c);
  printf("d=%d\n",d);
  return EXIT_SUCCESS;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int a=3;
  int b=4;
  { /* définition d'un bloc d'instructions */
        int c=a*b;
        printf("c=%d\n",c);
  }
```

```
  {
    int d=2*c;
    printf("d=%d\n",d);
  }
  return EXIT_SUCCESS;
}
```

Dans le premier exemple, le programme affichera :

```
c=7
d=14
```

Dans le second exemple, le programme ne compilera pas car la variable `d` (en dehors du bloc d'instructions) utilise la variable `c` qui n'est visible que dans le bloc d'instructions précèdent. Vous pouvez d'ailleurs remarquer que vous utilisez la notion de bloc d'instructions depuis vos premiers programmes car le corps de fonction `main` est un bloc d'instructions dans lequel vous écrivez vos programmes !

exercice 1) Écrivez un programme dans lequel vous :
- initialiserez la variable `a=10` ;
- créerez un bloc d'instructions dans lequel vous initialiserez une nouvelle variable `a=100` et la ferez afficher ;
- afficherez la valeur de `a` après le bloc d'instructions.

Que se passe-t-il ? Expliquez.

5 Les énoncés conditionnels

5.1 L'énoncé `if`

Cet énoncé permet d'exécuter ou non un énoncé selon une condition à valeur entière. Sa forme générale est :

if (*B*) E_1 **else** E_2

Si le résultat de l'expression logique B est un entier différent de 0, l'énoncé E_1 sera exécuté, sinon ce sera l'énoncé E_2. Notez que les parenthèses sont obligatoires autour de l'expression B.

L'exemple suivant affecte à une variable `max` le maximum de deux valeurs :

```
if (a>b) max = a; else max = b;
```

Par ailleurs, si la partie *alors* ou la partie *sinon* comportent plusieurs énoncés, il faudra les regrouper dans un bloc d'instructions :


```
/* résolution de l'équation du second degré
   ax²+bx+c=0, avec x ≠ 0
*/
delta = (b*b)-4*a*c;
if (delta>=0) { /* calcul des racines réelles */
   r1 = -(b+sqrt(delta))/(2*a);
   r2 = -b+sqrt(delta)/(2*a);
}
else { /* calcul des racines complexes */
   r1 = r2 = -b/(2*a);
   i1 = sqrt(-delta)/(2*a);
   i2 = -i1;
}
```

D'autre part, comme dans l'exemple suivant, il est possible d'omettre la partie **else** de l'énoncé conditionnel.

```
/* x un entier quelconque */
if (x<0) x = -x;
/* x un entier positif */
```

L'expression B peut être formée à l'aide des opérateurs relationnels habituels comme :

condition	signification
`if (a>b)`	strictement supérieur à
`if (a>=b)`	supérieur ou égal à
`if (a<b)`	strictement inférieur à
`if (a<=b)`	inférieur ou égal à
`if (a==b)`	test d'égalité
`if (a!=b)`	différent de

ATTENTION : Pour effectuer un test d'égalité, il faut mettre deux fois le symbole « = ». Par exemple :

```
if (a==b) x = 0;
```

L'erreur classique aurait été d'écrire :

```
if (a=b) x = 0;
```

L'expression logique peut faire intervenir plusieurs opérateurs logiques comme par exemple :

condition	signification
`a && b`	a ET b doivent être vérifiées
`a || b`	a OU b doivent être vérifiées
`!a`	a ne doit pas être vérifiée

```
if ( (a>b && b>c) || (b>a && c>b) )
   E1
else
   E2
```

exercice 2) Donnez en français, le sens de la condition précédente.

5.1.1 Emboîter des if

Les énoncés placés dans un énoncé conditionnel peuvent être quelconques, et en particulier eux-mêmes des énoncés conditionnels. Il est donc possible d'avoir des énoncés conditionnels emboîtés, comme dans l'exemple suivant :

```
if (x==0) {
   if (y==1)
      z = z + 1;
   else
      w = w + 1;
   y  = 1;
}
else {
   if (a>1 && b<2)
      i = 2;
   else
      j = 1;
   k = 0;
}
c = c + 1;
```

Notez qu'un **else** se rapporte au **if** le plus proche.

```
if (x==0)
   if (y==1)
      z = z + 1;
   else
      w = w + 1;
```

exercice 3) Comment modifier l'exemple précédent pour que le **else** se rapporte au premier **if** ?

exercice 4) Écrivez un programme qui demande à l'utilisateur de saisir un entier relatif x ($x \in \mathbb{Z}$) et qui indique, d'une part, si ce nombre est positif ou négatif, et d'autre part, si ce nombre est pair ou impair.

exercice 5) Écrivez un programme qui demande à l'utilisateur de saisir 3 nombres réels (**double**) au clavier et qui les affiche par ordre croissant.

exercice 6) Écrivez un programme qui résout l'équation $ax + b = 0$ (a et b sont à saisir par l'utilisateur). Bien évidemment, on n'oubliera pas tous les cas particuliers (notamment les cas "tout x est solution" et "pas de solution").

exercice 7) Écrivez un programme qui résout l'équation $ax^2 + bx + c = 0$ (a, b, et c sont à saisir par l'utilisateur) en envisageant tous les cas particuliers.

Remarques

- Calcul du discriminant : $\Delta = b^2 - 4ac$
- Calcul des racines du trinôme :
  - si $\Delta < 0$ deux racines complexes $r_1 = r_2 = \frac{-b}{2a}$ et $i_1 = \frac{\sqrt{-\Delta}}{2a}, i_2 = -i_1$;
  - si $\Delta \geqslant 0$ deux racines réelles $r_1 = \frac{-b+\sqrt{\delta}}{2a}$ et $r_2 = \frac{-b-\sqrt{\delta}}{2a}$ et $i_1 = i_2 = 0$.

Notez que pour le calcul des racines réelles et afin d'éviter des erreurs de calcul, il est nécessaire de calculer d'abord la racine la plus grande en valeur absolue, puis de calculer la seconde à partir du produit $r_2 = \frac{c}{ar_1}$.

Le calcul de la racine carrée se fait via l'utilisation de la fonction `sqrt`. Pour l'utiliser, vous devez inclure le fichier `math.h` et charger la bibliothèque mathématique au moment de la compilation (option `-lm`)

exercice 8) Écrivez un programme qui lit un entier sur l'entrée standard représentant le numéro d'un mois et qui affiche le mois en toutes lettres (janvier=1, février=2, *etc*).

5.2 L'énoncé conditionnel switch

L'énoncé **switch** permet de choisir d'exécuter un ou plusieurs énoncés particuliers parmi d'autres selon la valeur d'une expression de type convertible en entier. Le plus souvent, il offre un raccourci syntaxique pour un ensemble de *si-alors-sinon* emboîtés.

```
char opérateur;
/* calculer c = a op b, avec op = + - x ou /  */
if (opérateur == '+') c = a + b;
else
   if (opérateur == '-') c = a - b;
   else
      if (opérateur == 'x') c = a * b;
      else // opérateur == '/'
           c = a / b;
```

Les énoncés conditionnels précédents s'écrivent avec un **switch** :

```
/* calculer c = a op b, avec op = + - x ou */
switch (opérateur) {
   case '+' :  c = a + b; break;
   case '-' :  c = a - b; break;
   case 'x' :  c = a * b; break;
   case '/' :  c = a / b; break;
}
```

Les valeurs que peut prendre l'expression (ici `opérateur`) sont introduites par le mot-clé **case**. Lorsque les énoncés associés à la valeur sont exécutés, l'achèvement de l'énoncé **switch** doit être explicité à l'aide de l'énoncé **break** [1]

Il existe une valeur spéciale, appelée **default**. À cette dernière, il est possible d'associer un ou plusieurs énoncés, qui sont exécutés lorsque l'expression renvoie une valeur qui ne fait pas partie de la liste des valeurs énumérées.

1. ou **return** dont nous reparlerons plus tard.

```
// calculer c = a op b, avec op = + - x ou /
switch (opérateur) {
   case '+' :  c = a + b; break;
   case '-' :  c = a - b; break;
   case 'x' :  c = a * b; break;
   case '/' :  c = a / b; break;
   default  :  fprintf(stderr,"opérateur inconnu");
}
```

Un énoncé peut être associé à plusieurs valeurs comme le montre l'exemple suivant :

```
switch (opérateur) {
   case '+' :
   case '-' :  printf("op additif\n");
               break;
   case 'x' :
   case '/' :  printf("op multiplicatif\n");
               break;
   default  :  fprintf(stderr,"opérateur inconnu");
}
```

exercice 9) Refaites l'exercice 8 en utilisant l'opérateur l'énoncé **switch**.

exercice 10) Écrivez un programme qui lit sur l'entrée standard le numéro d'un mois et l'année et qui calcule et affiche le nombre de jours dans le mois correspondant. Attention, le mois de février compte 29 jours pour une année bissextile. Une année bissextile est une année divisible par 4 mais pas par 100, ou alors divisible par 400.

6 La date du lendemain

exercice 11) On désire écrire un programme qui affiche la date du lendemain. La date du jour est représentée par trois entiers : jour, mois et année. Le calcul de la date du lendemain ne pourra se faire que sur une date *valide* dont l'année est postérieure à 1600. La date du lendemain sera écrite sur la sortie standard avec le mois en toutes lettres. Pour écrire ce programme, vous suivrez l'algorithme ci-dessous :

```
{ lire le jour, le mois et l'année }
{ vérifier si la date est valide }
si non valide alors
   { signaler l'erreur }
sinon
     { calculer la date du lendemain et }
     { afficher la date du lendemain  }
finsi
```

Remarques :
- Lisez la date au format `09/10/2008` ;
- prenez en compte dans vos calculs les années bissextiles.