Université Paris-Nord — L1
Institut Galilée — Année 2008-2009
Licence 1 - 1er semestre

Travaux dirigés 6 : constantes symboliques ; lecture de données au clavier

Correction. Note aux chargés de TD.

- En cours, ils ont fini les tableaux. Ils ont vu de nouveaux types (char, double) et les conversions possibles avec les int. Le cours s'est fini sur un chapitre Entrées/Sorties qui introduit le scanf et les nouveaux formats de conversion pour printf.
- Les traces ne définissent pas une colonne supplémentaire pour les entrées, ce qui ne permet pas de présenter les subtilités de conversion de scanf. Cela sera fait en S2.
- Bien insister sur la procédure permettant d'attaquer un problème de programmation. Il semble qu'ils ne savent toujours pas l'appliquer. La procédure :
  - on se donne des exemples
  - on trouve un algorithme en francais
  - on traduit l'algorithme en C en s'aidant de commentaires
  - on test sur les exemples qu'on s'est donnes
- L'algorithmique des tableaux parle de cases, et un parcours se présente de la façon suivante :

```
- pour chaque case du tableau :
    - affecter 2 a la case
    - afficher la case
    - ...

    En C, on peut être plus tenté par : pour chaque indice de
    case. Cependant, la plupart des langages modernes (Java, Python
    etc.) ont introduit des conteneurs avec des parcours qui font
    référence aux éléments. On reste sur les cases pour l'instant et
    il faut sans doute expliquer la subtilité.
```

- N.B. : le "programme vide" voit son écriture simplifiée en n'indiquant plus dans le main la déclaration des variables, ni la valeur de retour. Si certains ont encore des problèmes avec ça, il faut repousser.

1 Calcul dans un espace vectoriel

Nous souhaitons faire un certain nombre d'opérations dans un espace vectoriel. Pour cela, nous utilisons des tableaux pour représenter les vecteurs. Un squelette de programme est le suivant :

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
```

```c
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */
#define DIM 5 /* dimension de l'espace vectoriel */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* soient 3 vecteurs x,y et z */
    double x[DIM] = {1.0,-2.3,2.0,2.2,-0.3};
    double y[DIM] = {-1.0,-0.1,1.5,0.4,-0.3};
    double z[DIM];
    int i; /* var. de boucle */

    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* affichage de la composante */
        printf("%g ",x[i]);
    }
    /* i >= DIM */
    printf("\n");

    /* multiplication de x par le scalaire 2.0 */

    /* initialisation de z avec la somme de x et y */

    /* recopie de z dans y */

    /* calcul et affichage du produit scalaire de x et y */

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

1. Que veut dire la directive du pré-processeur : `#define DIM 5` ?

   Correction. À partir de ce point, fais un "chercher/remplacer" de toutes les occurrences de la chaîne de caractères "DIM" par la chaîne de caractères "5". Cette directive est utilisée en C pour définir des constantes symboliques. Par convention, ces constantes sont écrites en majuscules pour les repérer facilement dans le texte. EXIT_SUCCESS en est une autre définie dans le fichier "stdlib.h", cad, ce fichier contient la directive "#define EXIT_SUCCESS 0".

2. Quel est le code de la fonction `main` que traite le compilateur ?

Correction. Rappel : le pré-processeur est avant le compilateur. Il exécute toutes les directives commencant par "#". Ces directives ne sont plus dans le fichier résultat.

```
int main()
{
    double x[5] = {1.0,-2.3,2.0,2.2,-0.3};
    double y[5] = {-1.0,-0.1,1.5,0.4,-0.3};
    double z[5];
    int i;

    for(i = 0;i < 5;i = i + 1)
    {
        printf("%g ",x[i]);
    }
    printf("\n");

    return 0;
}
```

3. Quel est l'intérêt d'utiliser des constantes symboliques dans un programme ?

   Correction. L'intérêt de nommer ces constantes est :
   – d'expliciter leur sémantique : quelle est leur signification, quel est leur contexte d'utilisation, etc.
   – de pouvoir laisser le pré-processeur modifier la valeur de la constante partout dans le programme => pas de risque d'oubli.
   Par exemple PI...

4. Que fait le programme ? Donner son affichage.

   Correction. Le programme :
   – déclare et initialise 2 tableaux de rationnels (double en C) de taille 5, x et y, respectivement avec les valeurs... ;
   – déclare le tableau z de rationnels de taille 5 ;
   – déclare un entier i ;
   – parcours chaque case du tableau x et l'affiche (écrit dans le TD5 comme : affiche les valeurs des variables du tableau ; par abus de langage, on parle de cases d'un tableau).
   Affichage : 1 -2.3 2 2.2 -0.3 (voir doc sur `%g`)

5. Compléter le programme en implantant les problèmes donnés en commentaires dans le code.

   Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */
#define DIM 5 /* dimension de l'espace vectoriel */
```

```
/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* soient 3 vecteurs x,y et z */
    double x[DIM] = {1.0,-2.3,2.0,2.2,-0.3};
    double y[DIM] = {-1.0,-0.1,1.5,0.4,-0.3};
    double z[DIM];
    int i; /* var. de boucle */
    double produit_scalaire; /* le produit scalaire de 2 vecteurs */

    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* affichage de la composante */
        printf("%g ",x[i]);
    }
    /* i >= DIM */
    printf("\n");

    /* multiplication de x par le scalaire 2.0 */
    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* multiplication par 2.0 */
        x[i] = x[i] * 2.0;
    }
    /* i >= DIM */

    /* initialisation de z avec la somme de x et y */
    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* somme des composantes */
        z[i] = x[i] + y[i];
    }
    /* i >= DIM */

    /* recopie de z dans y */
    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* recopie */
        y[i] = z[i];
    }
    /* i >= DIM */

    /* calcul et affichage du produit scalaire de x et y */
    produit_scalaire = 0.0; /* elt neutre pour l'addition */
```

```
    for(i = 0;i < DIM;i = i + 1) /* chaque composante */
    {
        /* multiplication deux à deux */
        produit_scalaire = produit_scalaire + (x[i] * y[i]);
    }
    /* i >= DIM */
    printf("le produit scalaire de x et y est : %g\n",produit_scalaire);

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

2 Lecture de données utilisateur entrées à partir du clavier

La fonction `scanf`, déclarée dans la bibliothèque `stdio` permet d'affecter à des variables des valeurs saisies au clavier par l'utilisateur du programme. Les types de données à lire sont précisés avec les mêmes chaînes de format que pour `printf`, à quelques exceptions près (par exemple "%g" pour afficher un double et "%lg" pour lire un double).

2.1 Rappel

1. Que fait le programme suivant ?

```
1   /* declaration de fonctionnalites supplementaires */
2   #include <stdlib.h> /* EXIT_SUCCESS */
3   #include <stdio.h> /* printf, scanf */
4
5   /* declaration constantes et types utilisateurs */
6
7   /* declaration de fonctions utilisateurs */
8
9   /* fonction principale */
10  int main()
11  {
12      int a;
13      double b;
14      char c;
15
16      printf("Entrez un nombre entier puis un nombre réel puis un caractère : ");
17
18      scanf("%d",&a);
19      scanf("%lg",&b);
20      scanf(" %c",&c);
21
22      printf("Vous avez saisi %d puis %g puis %c.\n",a,b,c);
23
24      return EXIT_SUCCESS;
```

```
25  }
26
27  /* implantation de fonctions utilisateurs */
```

Correction. Le programme :
- déclare 3 variables a,b et c, respectivement de type entier, réel (rationnel) et caractère ;
- demande à l'utilisateur de saisir 3 valeurs, respectivement de type entier, réel (rationnel) et caractère ;
- affecte la valeur entière saisie à la variable entière a (même type sinon quelle est la signification ? le compilateur détecte cette erreur sémantique lors de l'analyse sémantique mais il acceptera de compiler en faisant une conversion automatique de type => source de bugs difficiles à détecter)
- affecte la valeur réelle saisie à la variable réelle/rationnelle b ;
- affecte le caractère saisie à la variable caractère c ;
- affiche les valeurs pour montrer les affectations réalisées (vous pouvez utiliser un tel programme pour vérifier que les représentations sont bornées, cf. cours et TP)

2. Faire la trace du programme en considérant que l'utilisateur saisit au clavier : 1 puis "entrée", 12.2 puis "entrée" et 'c' puis "entrée" .

Correction.

```
ligne          | a      | b      | c      | affichage (sortie/écriture à l'écran)
-------------------------------------------------------------------------------
initialisation | ?      | ?      | ?      |
16             |        |        |        | Entrez un nombre entier puis un nombr
18             | 1      |        |        |
19             |        | 12.2   |        |
20             |        |        | 'c'    |
22             |        |        |        | Vous avez saisi 1 puis 12.2 puis c.
```

Ils l'ont vu en cours : vous pouvez leur faire remarquer que si la lecture du caractère s'était faite avec `"%c"`, la var c contiendrait `'\n'`, caractère qui suit la chaîne "12.2" (due à la mémoire tampon + scanf).

2.2 Calcul de la moyenne d'une série d'entiers saisie par l'utilisateur

Écrire un programme qui demande à l'utilisateur combien d'entiers composent sa série, lit la série d'entiers et affiche la moyenne de valeurs de la série. L'ensemble de la série ne doit pas être stockée en mémoire. Un exemple d'exécution est :

```
Combien d'elements dans la série ? 3
1
10
0
La moyenne des valeurs de cette serie est : 3.66667
```

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    int n; /* taille de la serie a saisir par l'utilisateur*/
    int elt; /* un element de la serie a saisir par l'utilsateur */
    double somme = 0.0; /* somme de la serie a calculer pour afficher la moyenne
                           c'est un double car sinon le C fait une division entiere *
    int i; /* var. de boucle */

    /* demande la taille de la serie a l'utilisateur */
    printf("Combien d'elements dans la série ? ");
    scanf("%d",&n);

    /* saisie serie (n entiers) et calcul incremental de la somme */
    for(i = 0;i < n;i = i + 1) /* chaque entier de la serie */
    {
        /* saisir sa valeur */
        scanf("%d",&elt);

        /* l'ajoute a la somme partielle */
        somme = somme + elt;
    }
    /* i >= n */

    /* somme contient la somme des elements de la serie.
       la moyenne est somme / n */
    printf("La moyenne des valeurs de cette serie est : %g\n",somme / n); /* ce n'est

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

2.3 Initialisation d'un tableau par l'utilisateur

Écrire un programme qui déclare un tableau d'entiers de taille arbitraire TAILLE (une constante symbolique) et qui demande à l'utilisateur de saisir au clavier les valeurs des cases du tableau. Afficher le tableau.

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declaration constantes et types utilisateurs */
#define TAILLE 4 /* taille du tableau utilisateur */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    int tab[TAILLE]; /* tableau a initialiser par l'utilisateur */
    int i; /* var. de boucle */

    /* demande a l'utilisateur de saisir TAILLE entiers*/
    printf("Saisissez %d entiers : ",TAILLE);

    /* saisie des elts du tableau (TAILLE entiers) */
    for(i = 0;i < TAILLE;i = i + 1) /* chaque case du tableau */
    {
        /* saisir sa valeur */
        scanf("%d",&tab[i]);
    }
    /* i >= TAILLE */

    /* affichage du tableau */
    printf("Vous avez saisi le tableau suivant : ");

    for(i = 0;i < TAILLE;i = i + 1) /* chaque case du tableau */
    {
        /* afficher sa valeur */
        printf("%d ",tab[i]);
    }
    /* i >= TAILLE */
    printf("\n");

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

Faire la trace, en donnant à l'avance les valeurs saisies. On considère, pour simplifier dans un premier temps (S1), que "entrée" est utilisée après chaque entier.