Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux dirigés 5 : les tableaux de variables

L'objectif de ce TD est d'apprendre à déclarer, initialiser et utiliser les tableaux de variables (impératives) en C. La boucle `for` sera utilisée pour itérer sur les différentes variables du tableau, en engendrant leur indice. On parle de parcours de tableaux. La feuille d'exercices se poursuit en TP.

Correction. Note aux chargés de TD.

- En cours, ils ont vu les dernières notions de compilation du semestre (notamment, l'utilisation du pré-processeur C pour la définition de constantes symboliques : #define de constantes utilisateurs, #include fichiers déclarant des constantes prédéfinies (INT_MAX etc.)). Les tableaux leur ont été présentés de la façon décrite au paragraphe 1. L'exercice type *calcul du minimium d'un tableau* a déjà été vu en cours.
- Nous poursuivons les exercices type. Ils doivent savoir résoudre/reproduire les exos marqués exercices type et **faire leur trace sur un exemple quelconque**.
- 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 donné
- 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.
```

1 Déclaration et initialisation d'un tableau ; affectation de ses variables

Un tableau de variables est une structure de données permettant de déclarer une collection de variables de même type, dans une zone contigüe en mémoire. Ces variables sont référencées par l'indice de leur place dans le tableau.

1. Que fait le programme suivant ?

```
1  /* declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* declaration constantes et types utilisateurs */
```

```
6
7   /* declaration de fonctions utilisateurs */
8
9   /* fonction principale */
10  int main()
11  {
12      /* declaration et initialisation variables */
13      int tab[3] = {2,-4,8};
14
15      tab[0] = 0;
16      tab[1] = 1;
17      tab[2] = 2;
18
19      printf("tab[0] = %d\n",tab[0]);
20      printf("tab[1] = %d\n",tab[1]);
21      printf("tab[2] = %d\n",tab[2]);
22
23      return EXIT_SUCCESS;
24  }
25
26  /* definitions des fonctions utilisateurs */
```

Correction. Le programme :
- déclare et initialise un tableau, s'appelant `tab`, de 3 variables entières avec les valeurs suivantes : 2,-4 et 8 ;
- affecte 0 à la variable d'indice 0, tab[0] ;
- affecte 1 à la variable d'indice 1, tab[1] ;
- affecte 2 à la variable d'indice 2, tab[2] ;
- affiche les valeurs des variables du tableau ; par abus de langage, on parle de cases d'un tableau.

2. Faire la trace du programme. Qu'affiche le programme ?

Correction.

```
ligne          | tab[0] | tab[1] | tab[2] | affichage (sortie/écriture à l'écran)
-------------------------------------------------------------------------------
initialisation | 2      | -4     | 8      |
15             | 0      |        |        |
16             |        | 1      |        |
17             |        |        | 2      |
19             |        |        |        | tab[0] = 0\n
20             |        |        |        | tab[1] = 1\n
21             |        |        |        | tab[2] = 2\n
```

3. Modifier le programme afin d'effectuer une permutation circulaire vers la droite des valeurs des variables (ou cases) du tableau.

Correction.

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

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */
```

```
/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[3] = {2,-4,8};
    int aux; /* var auxiliaire pour realiser la permutation */

    /* permutation circulaire vers la droite des valeurs des cases */
    aux = tab[2];
    tab[2] = tab[1];
    tab[1] = tab[0];
    tab[0] = aux;

    /* affichage du tableau */
    printf("tab[0] = %d\n",tab[0]);
    printf("tab[1] = %d\n",tab[1]);
    printf("tab[2] = %d\n",tab[2]);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

2 Affichage des éléments d'un tableau

Écrire un programme qui :
- déclare et initialise le tableau de variables entières : 5,2,4,3,0
- pour chaque case du tableau :
  - affiche la case

Correction.

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

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5] = {5,2,4,3,0}; /* tableau a afficher */
    int i; /* var. de boucle */

    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* affichage de la case */
        printf("tab[%d] = %d\n",i,tab[i]);
    }
    /* i >= 5 */

    return EXIT_SUCCESS;
```

```
}

/* definitions des fonctions utilisateurs */
```

3 Exercice type : Initialisation des cases d'un tableau à zéro

Soit un tableau d'entiers, déclaré d'une taille quelconque. Écrire un programme qui :
- affiche le tableau non initialisé
- initialise chaque case à zéro
- affiche le tableau initialisé

Un exemple d'exécution pour un tableau de taille 5 est :

```
affichage du tableau non initialise :
tab[0] = 134513308
tab[1] = -1208832012
tab[2] = 134518316
tab[3] = -1076845768
tab[4] = 134513753
affichage du tableau initialise :
tab[0] = 0
tab[1] = 0
tab[2] = 0
tab[3] = 0
tab[4] = 0
```

Correction. L'exemple plus haut correspond à l'exécution sur un PC 32 bits. Bien insister sur les valeurs arbitraires des variables non initialisées, qui dépendent des programmes qui ont été exécutés avant etc. On les représente par le symbole '?' dans la trace.

```
algo:
- affichage du tableau non initialise
- pour chaque case du tableau :
  - affecte 0 a la case
- affichage du tableau initialise


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

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5]; /* tableau a initialiser */
    int i; /* var. de boucle */

    /* affichage du tableau non initialise */
    printf("affichage du tableau non initialise :\n");
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* affichage de la case */
        printf("tab[%d] = %d\n",i,tab[i]);
```

```
    }
    /* i >= 5 */

    /* initialisation des cases a 0 */
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* initialisation de la case a 0 */
        tab[i] = 0;
    }
    /* i >= 5 */

    /* affichage du tableau initialise */
    printf("affichage du tableau initialise :\n");
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* affichage de la case */
        printf("tab[%d] = %d\n",i,tab[i]);
    }
    /* i >= 5 */

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

4 Exercice type : Calcul de la somme des éléments d'un tableau

Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran la somme des éléments du tableau.

Correction.

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

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5] = {2,-4,8,12,-1};
    int somme = 0; /* element neutre pour l'addition */
    int i; /* var. de boucle */

    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* ajoute la case a la somme partielle */
        somme = somme + tab[i];
    }
    /* i >= 5 */
```

```
    /* somme vaut 0 + 2 - 4 + 8 + 12 - 1 */
    printf("somme = %d\n",somme);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

5 Exercice type : Calcul du minimum d'un tableau

Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran la valeur minimum des éléments du tableau.

Correction.

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

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5] = {2,-4,8,12,-1};
    int min = INT_MAX; /* +infini >= valeurs tableau*/
    int i; /* var. de boucle */

    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        if(tab[i] < min) /* plus petit que min courant */
        {
            /* nouveau min courant */
            min = tab[i];
        }
    }
    /* i >= 5 */

    /* min contient le minimum de tous les elements du tableau */
    printf("le minimum des elements du tableau est : %d\n",min);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

6 Exercices optionnels

Calcul du produit des éléments d'un tableau Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran le produit des éléments du tableau.

Calcul du nombre d'occurrences d'un entier dans un tableau Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques ; soit une variable `n`, initialisée à une valeur quelconque. Écrire un programme qui calcule et affiche à l'écran le nombre d'occurences de la valeur de `n` dans le tableau.