Université Paris-Nord — L1
Institut Galilée — Année 2009-2010
Licence 1 - 1er semestre

Travaux dirigés 6 : 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 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).

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   /* declarations des constantes et types utilisateurs */
6
7   /* declarations des 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  /* definitions des 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).

1.1 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.

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 */
```

1.2 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.

2 Les sujets du contrôle de TP

Sujet 1. Soit un tableau d'entiers relatifs initialisé à des valeurs de votre choix et dont la taille sera fixée par une constante symbolique N. Écrire un programme qui :

1. affiche le contenu du tableau ;
2. affiche la valeur absolue des élements du tableau ;
3. calcule et affiche la somme des valeurs absolues des éléments du tableau.

Le programme doit fonctionner correctement pour n'importe quelle initialisation du tableau.

Sujet 2. Soit une constante symbolique N valant 3. Écrire un programme qui :

1. affiche le carré de côté N suivant :
   ```
   (0,0) (0,1) (0,2)
   (1,0) (1,1) (1,2)
   (2,0) (2,1) (2,2)
   ```
2. puis affiche le carré de côté N suivant :
   ```
   ***** (0,1) (0,2)
   (1,0) ***** (1,2)
   (2,0) (2,1) *****
   ```
3. puis affiche le triangle de côté N suivant :
   ```
   (0,0) (0,1) (0,2)
         (1,1) (1,2)
               (2,2)
   ```

Le programme doit fonctionner correctement pour n'importe quelle valeur positive de N strictement inférieure à 10.

Sujet 3. Soit un tableau d'entiers initialisé à des valeurs de votre choix et dont la taille sera fixée par une constante symbolique N. Soit un entier x initialisé à une valeur de votre choix. Écrire un programme qui :

1. affiche les valeurs du tableau et celle de x ;
2. compte le nombre cases du tableau contenant la valeur de x (c'est à dire le nombre d'occurences de x dans le tableau) et affiche ce nombre à l'écran.

Le programme doit fonctionner correctement pour n'importe quelle initialisation du tableau et n'importe quelle valeur de x.

Sujet 4. Soit un tableau d'entiers relatifs initialisé à des valeurs de votre choix et dont la taille sera fixée par une constante symbolique N. Écrire un programme qui :

1. affiche le contenu du tableau
2. compte le nombre d'entiers positifs ou nuls dans le tableau et affiche ce nombre à l'écran.

Le programme doit fonctionner correctement pour n'importe quelle initialisation du tableau.

Sujet 5. Soit une variable entière n initialisée à une valeur positive. Écrire un programme qui :

1. calcule la factorielle de n (le produit des entiers de 1 à n) et affiche le résultat. Exemple de sortie (pour n = 3) :
   ```
   Factorielle de 3 : 3! = 6.
   ```
2. calcule la somme des factorielles des n premiers entiers positifs. Exemple de sortie (pour $n = 3$) :
   ```
   Somme des factorielles jusqu'a 3 :
   1! + 2! + 3! = 9.
   ```

Le programme doit fonctionner pour n'importe quelle valeur positive de n.

Correction.

Sujet 1

```
...
#define N 5
...
int main()
{
  int t[N]={1,-2,0,30,-25};
  int vabs; /* valeur absolue */
  int somme = 0; /* somme des valeurs absolues */
  int i; /* var de boucle */

  /* affichage du tableau */
  printf("valeurs du tableau : ");
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
    printf("%d ",t[i]); /* affichage de la valeur */
  }
  printf("/n");

  /* affichage des valeurs absolue */
  printf("valeurs absolues du tableau : ");
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
     if (t[i] < 0) /* si la case est négative */
     {
       /* afficher l'inverse de la valeur de la case */
       printf("%d ", -t[i]);
     }
     else /* la case est positive */
     {
       /* afficher la case */
       printf("%d ", t[i]);
     }
  }
  printf("/n");

  /* Calcul de la somme des valeurs absolues */
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
     if (t[i] < 0) /* si la case est negative */
     {
        /* la valeur absolue est l'inverse de la case */
        vabs = -t[i];
    }
     else /* la case est positive */
     {
        /*la valeur absolue est la valeur de la case */
        vabs = -t[i];
```

```
    }
    /* ajout de la valeur absolue a la somme */
    somme = somme + vabs;
  }

  /* affichage de la somme */
  printf("La somme des valeurs absolues est %d\n", somme);

  return EXIT_SUCCESS;
}
```

Sujet 2

```
#define N 3
...
int main()
{
  int ligne; /* numero de ligne (var. de boucle) */
  int colonne;  /* numero de colonne (var. de boucle) */

  /* Affichage du premier carre */
  for (i = 0; i < N; i = i + 1) /* pour chaque ligne */
  {
     for (j = 0; j < N; j = j + j) /* pour chaque colonne */
     {
        /* afficher la paire */
        printf("(%d,%d) ", i, j);
     }
     /* fin de la ligne */
     printf("\n");
  }

  /* Affichage du deuxieme carre */
  for (i = 0; i < N; i = i + 1) /* pour chaque ligne */
  {
     for (j = 0; j < N; j = j + j) /* pour chaque colonne */
     {
        if (i == j) /* sur la premiere diagonale */
        {
           /* afficher 5 etoiles et un espace */
           printf("***** ");
        }
        else /* en dehors de la premiere diagonale */
        {
           /* afficher la paire */
           printf("(%d,%d) ", i, j);
        }
     }
     /* fin de la ligne */
     printf("\n");
  }
```

```
  /* Affichage du triangle */
  for (i = 0; i < N; i = i + 1) /* pour chaque ligne */
  {
     for (j = 0; j < N; j = j + j) /* pour chaque colonne */
     {
        if (i < j) /* sous la premiere diagonale */
        {
           /* afficher 6 espaces */
           printf("      ");
        }
        else /* sur la premiere diagonale et au-dessus */
        {
           /* afficher la paire */
           printf("(%d,%d) ", i, j);
        }
     }
     /* fin de la ligne */
     printf("\n");
  }


  return EXIT_SUCCESS;
}
```

Sujet 3

```
...
#define N 5
...
int main()
{
  int t[N]={1,2,1,1,2};
  int x = 1; /* valeur cible */
  int compteur = 0; /* occurences */
  int i; /* var de boucle */

  /* affichage du tableau */
  printf("valeurs du tableau : ");
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
    printf("%d ",t[i]); /* affichage de la valeur */
  }
  printf("/n");

  /* comptage */
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
     if (x == t[i]) /* si la case contient x */
```

```
    {
       /* incrementer le compteur */
       compteur = compteur + 1;
    }
  }
  /* affichage du decompte */
  printf("La valeur %d apparait %d fois\n", x, compteur);

  return EXIT_SUCCESS;
}
```

Sujet 4

```
...
#define N 5
...
int main()
{
  int t[N]={1,-2,1,-1,0};
  int compteur = 0; /* occurences */
  int i; /* var de boucle */

  /* affichage du tableau */
  printf("valeurs du tableau : ");
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
    printf("%d ",t[i]); /* affichage de la valeur */
  }
  printf("/n");

  /* comptage */
  for (i = 0; i < N; i = i + 1) /* pour chaque case */
  {
     if (0 <= t[i]) /* si la case est positive */
     {
        /* incrementer le compteur */
        compteur = compteur + 1;
     }
  }
  /* affichage du decompte */
  printf("Il y a %d entiers positifs ou nuls\n", compteur);

  return EXIT_SUCCESS;
}
```

Sujet 5

```
int main()
{
  int n = 3;
  int produit = 1; /* accumulateur pour la factorielle */
```

```
  int somme = 0; /* accumulateur pour la somme des factorielles */
  int i; /* var de boucle */
  int j; /* var de boucle */

  /* calcul de la factorielle de n */
  for (i = 1; i <= n; i = i + 1) /* pour chaque entier 1,...,n */
  {
    /* adjoindre l'entier au produit */
    produit = produit * i;
  }
  /* Affichage */
  printf("Factorielle de %d : %d\n", n, produit);


  /* calcul de la somme des factorielles */
  for (j = 1; j < n; j = j + 1)
  {
    /* reinitialisation du produit */
    produit = 1;

    /* calcul de la factorielle de j */
    for (i = 1; i <= j; i = i + 1) /* pour chaque entier 1,...,j */
    {
      /* adjoindre l'entier au produit */
      produit = produit * i;
    }

    /* adjoindre l'entier a la somme */
    somme = somme + produit;
  }

  /* Affichage final */
  printf("Somme des factorielle jusqu'a %d :\n", n);
  /* affichage des termes de la somme */
  printf("1!"); /* premier terme */
  for (i = 2; i <= n; i = i + 1) /* pour chacun des termes suivant */
  {
    /* affichage du terme */
    printf(" + %d!", i);
  }
  /* affichage de la somme */
  printf(" = %d\n", somme);

  return EXIT_SUCCESS;
}
```