

Travaux pratiques n°5 (2h): Procédures et fonctions

Objectifs

- Déclarer et utiliser des procédures et des fonctions en langage C

Partie 1: Compléments pratiques au langage C

Le moyen le plus simple pour écrire un programme avec fonctions et procédures consiste à mettre le corps des fonctions et des procédures dans le fichier qui contient le programme principal. A titre d'exemple, écrire, compiler et exécuter le programme suivant:

```
#include <stdio.h>

/* corps de la fonction mystere */
int mystere (int n)
{
    int res = 1;
    int i;
    for (i=2; i <= n; i++)
        res = i * res;
    return(res);
}

/* programme principal */
int main ()
{
    int nb;
    int res;
    do
    {
        printf("Donnez un nombre entier positif:");
        scanf("%d", &nb);
        getchar();
    }
    while (nb < 0);
    printf("\nVoila un résultat: %d\n", mystere(nb));
    res = mystere(nb-1) * nb;
    printf("\nVoila un autre résultat: %d\n", res);
}
```

Questions:

- Que calcule la fonction mystère?
- Quels sont les deux affichages que fera le programme principal pour nb = 4?

La notion de modularité:

Cependant, dès que l'on programme avec des fonctions et des procédures, il faut créer 3 fichiers suivants (conformément au principe de modularité)

1. un fichier de déclarations *.h* (exemple : *td7.h*) qui contiendra les prototypes des fonctions/procédures à définir ainsi que leurs spécifications
2. un fichier de code *td7.c* qui contiendra le corps de ces fonctions/procédures
3. un fichier de code *pgPrincipaTd7.c* qui contiendra le programme principal permettant de tester les appels aux fonctions de *td7.c*

Chacun des 2 fichiers de code (*td7.c* et *pgPrincipaTd7.c*) devront comporter en dernière ligne des fichiers *include* la ligne *#include <td7.h>* de façon à ce que les prototypes des fonctions soient disponibles ; ainsi, le compilateur peut vérifier la conformité du nombre d'arguments et de leurs types pour chaque fonction appelée ou définie.

Procédez impérativement de la façon suivante :

- Écrire le prototype d'une fonction/procédure dans *td7.h* ainsi que sa spécification
- Écrire le corps de cette fonction/procédure dans *td7.c*,
- Compiler *td7.c* : `gcc -c td7.c -o td7.o ↵`
- Écrire l'appel de la fonction/procédure dans *pgPrincipaTd7.c*
- Compiler *pgPrincipaTd7.c* : `gcc -c pgPrincipaTd7.c -o pgPrincipaTd7.o ↵`
- Faire l'édition des liens entre *pgPrincipaTd7.o* et *td7.o* :
`gcc pgPrincipaTd7.o td7.o -o pgPrincipaTd7 ↵`
- Exécuter *pgPrincipaTd7* : `pgPrincipaTd7`
- Recommencer à l'étape 1 avec une nouvelle fonction/procédure si celle-ci vous donne satisfaction, sinon déboguez celle-ci.

Exemple:

Une fonction *estTrie* dont le corps est mis dans le fichier *exFonct.c*, dont le prototype est mis dans *exemple.h* et un programme principal *exemple.c* qui utilise la fonction *estTrie*.

/* Fichier exemple.h */

```
#define VRAI 1
#define FAUX 0
#define MAX 10
```

```
int estTrie(int t[], int nb);
```

/* Fichier exFonct.c */

```
#include "exemple.h"
```

```
int estTrie(int t[], int nb)
```

```
{
int i=0;
while (i<nb-1 && t[i] <= t[i+1])    // attention aux bornes
    i++;
if                                (i==nb-1)
    return VRAI;                // attention au test de sortie
else    return FAUX;
}
```

/* Programme principal: exemple.c */

```
#include <stdio.h>
```

```
#include "exemple.h"
```

```
int main(void)
{
    int tab[MAX] = {1,2,3,4,5,6};
    int nbEff = 6;
    if (estTrie(tab,nbEff))
        printf("\n OUI\n");
    else
        printf("\n NON");
}
```

Compilation et exécution:

```
gcc -c exFonct.c -o exFonct.o      /* compilation de exFonct.c */
gcc -c exemple.c -o exemple.o      /* compilation de exemple.c */
gcc exemple.o exFonct.o -o exemple /* édition des liens */
./exemple                          /* exécution */
```

Partie 2: Écrire des procédures et des fonctions en langage C

Exercice 1:

- Écrire une fonction/procédure `estTrie` qui teste si un tableau d'entiers est rangé par ordre croissant
- Écrire une fonction/procédure `decaleCircGauche` qui réalise le décalage circulaire vers la gauche d'un tableau de réels .
- Écrire une fonction/procédure `decaleCircDroite` qui réalise le décalage circulaire vers la droite d'un tableau de réels.

Exercice 2: Écrire la procédure ou la fonction (à vous de le décider) en langage C, qui donne en sortie un tableau de caractères de 16 caractères représentant l'écriture en base 2 d'un nombre entier positif (en décimal) passé en argument. On suppose que le plus grand entier positif (codé sur 2 octets) est 32767 ce qui donne en base 2 : 0111111111111111. Le résultat pour 5 sera 00000000000000101.

Exercice 3: Écrire une fonction `geometrique(...)` qui retourne la valeur $1 + r + r^2 + \dots + \dots$ (r réel vérifiant $0 < r < 1$) sachant qu'on n'intégrera dans la somme que les termes strictement supérieurs à un certain seuil fourni (*seuil* réel vérifiant $0 < \text{seuil} < r$).

Exercice 4:

Écrire la procédure qui renvoie le nombre d'entiers pairs et le nombre d'entiers impairs d'un tableau d'entiers.

Exercice 5:

Écrire la procédure *enleveValeurs*:

```
void enleveValeurs( float t[], float r, int* nbElem);
```

spécification : modifie le tableau **t** en enlevant toutes ses valeurs égales à **r**.

Exemples : si en entrée le tableau **t** a les 7 éléments effectifs suivants: 3 5 6 3 8 4 3

Après l'appel à la procédure qui enlève la valeur 3, en sortie il n'aura plus que les 4 éléments suivants : 5 6 8 4.

si en entrée le tableau **t** a les 7 éléments effectifs suivants : 3 5 6 3 8 4 3

Après l'appel à la procédure qui enlève la valeur 9, en sortie le tableau n'aura pas été modifié, il aura toujours les mêmes 7 éléments.

Exercice 6:

Écrire la fonction/procédure **rangePositif** qui permet de ranger dans un tableau **tRes** les entiers positifs ou nuls contenus dans un tableau **t** de **nb** éléments effectifs. Bien réfléchir aux paramètres.