Travaux pratiques n°9 (2h): Procédures et fonctions(2): Pointeurs/paramètres

Objectifs

- maîtriser la notion de pointeur et la gestion de la mémoire lors des passages de paramètres.
- Déclarer et utiliser des procédures et des fonctions en langage C

Documents nécessaires

- sujet du TD9 et notes de TD9
- support de cours sur les procédures et fonctions

Partie 1: Compléments pratiques au langage C

La notion de modularité:

Dès qu'on programme avec des fonctions et des procédures, il faut créer 3 fichiers suivants (conformément au principe de modularité)

- un fichier de déclarations *.h (*exemple *: td7.h)* qui contiendra les prototypes des fonctions/procédures à définir ainsi que leurs spécifications
- un fichier de code *td7.c* qui contiendra le corps de ces fonctions/procédures
- 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 :

1. Écrire le prototype d'une fonction/procédure dans *td7.h* ainsi que sa spécification
2. Écrire le corps de cette fonction/procédure dans *td7.c*,
3. Compiler *td7.c* : *gcc –c td7.c –o td7.o* ↵
4. Écrire l'appel de la fonction/procédure dans *pgPrincipaTd7.c*
5. Compiler *pgPrincipaTd7.c* : *gcc –c pgPrincipaTd7.c –o pgPrincipaTd7.o* ↵
6. Faire l'édition des liens entre *pgPrincipaTd7.o* et *td7.o* :
   gcc pgPrincipaTd7.o td7.o –o pgPrincipaTd7 ↵

---

7. Exécuter *pgPrincipaTd7* : *pgPrincipaTd7*
8. 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: Traduire des corps d'algorithmes détaillés avec procédures et fonctions en langage C

Exercice 1

Écrire et tester les exercices 1 et 5 de ce TD.

Exercice 2: Visualiser la mémoire en utilisant le programme ddd

Ecrire et tester les programmes des exercices 2, 3 et 4.

Rappel:

Pour utiliser ddd il faut donner l'option -g à la compilation.

Exemple:

```
gcc -g monProg.c -o monProg
ddd monProg
```