Exercices courts - 01

notions : pointeurs, prototypes de fonctions

Exercice 1 Prototypes et contrats de fonctions simples

Pour chacune des questions suivantes, définissez soigneusement le prototype de la fonction et réfléchissez à son contrat.

1. Fonction qui renvoie la valeur absolue d'un entier.
2. **Fonction qui renvoie le produit de 3 nombres réels passés en paramètres**
3. Fonction qui calcule la factorielle d'un nombre entier passé en un paramètre
4. Fonction permettant de calculer une puissance entière d'un nombre double
5. **Fonction qui échange le contenu de 2 variables entières**
6. **Fonction qui prend 2 flottant en paramètre et "retourne" la somme et le produit de ces deux flottants.**
7. Fonction qui prend 2 entiers en paramètres, ajoute un au premier et soustrait un au second et "retourne" les nouvelle valeurs de ces deux entiers.
8. Fonction qui prend un tableau de n éléments de type double en paramètre et met tous les éléments à une valeur passée en paramètre.
9. **Fonction qui prend un tableau de n float en paramètre et "retourne" le nombre d'éléments négatifs et le nombre d'éléments positifs.**
10. **Fonction qui résout l'équation du second degré dans $\mathbb{R}$**
11. Fonction qui calcule le pgcd de a et b (elle peut être récursive...)
12. **On dispose de plusieurs fichiers de données au format texte, tous structurés ainsi : la première ligne contient le nombre de valeurs (un entier), chacune des lignes qui suivent contient une valeur flottante. Fonction qui charge l'un de ces fichiers en mémoire, dans un tableau de nombres flottants, et "retourne" ce tableau.**

Exercice 2 Equation du second degré

Dans un module *fonction* (fichiers *fonction.h* et *fonction.c*), écrivez une fonction qui résout une équation du 2nd degré dans $\mathbb{R}$.

Dans un fichier *test.c*, écrivez un programme principal testant cette fonction.

Ecrivez un fichier Makefile pour gérer les étapes de compilation.

Testez.

Exercice 3 QCM appel de fonction

René-Gontran récupère un fichier module *calcul*, constitué des fichiers *calcul.h* et *calcul.c* suivants :

Fichier *calcul.h* :

```
#ifndef  _CALCUL_H_
#define  _CALCUL_H_

// Calcule la racine carree et le logarithme base 2 de x
// et les "retourne"
// au moyen respectivement des pointeurs   psqrt   et   plog
// @return 0 en cas de succes.
// @return 1 en cas d'erreur (x <= 0)
int sqrt_and_log(double x, double *psqrt, double *plog);

#endif
```

Fichier *calcul.c* :

```
#include <math.h>
#include "calcul.h"

int sqrt_and_log(double x, double *psqrt, double *plog) {
  if(x <= 0.) {
    return 1;
  }
  *psqrt = sqrt(x);
  *plog = log(x);

  return 0;
}
```

Pour tester le module, René-Gontran écrit le fichier de test suivant :

Fichier *test.c* :

```
#include "calcul.h"

void main() {
  double x;
  double *p1, *p2;
  int OK;

  printf("donnez x :");
  scanf("%lf", &x);
  OK = sqrt_and_log(x, p1, p2);
  if(OK == 1) {
    printf("Calcul impossible\n");
    exit(EXIT_FAILURE);
  }
  printf("la racine vaut %lf ; et le log vaut %lf\n", *p1, *p2);
  return EXIT_SUCCESS;
}
```

Quelles sont les réponses correctes ?

1. Ligne 1, 2 et dernière ligne du fichier *calcul.h*, on appelle ces primitive de précompilation la "garde" du fichier d'en-tête.
2. Pour compiler, René-Gontran tape dans le Terminal, successivement :

```
gcc -o calcul.c
gcc -o test.c
gcc  -o test test.o calcul .o -lm
```

3. Dans les commandes précédentes, la troisième s'appelle "l'édition des liens"
4. A supposer que les commandes de compilation soient correctes, ce programme compile sans erreur.
5. A l'exécution, si l'utilisteur rentre la valeur 1, le programme écrit : `"la racine vaut 1.000000 ; et le log vaut 0.00000"`
6. A l'exécution, si l'utilisteur rentre la valeur 1, il se produit une erreur de segmentation.
   Dans ce cas, quelle est l'erreur faite par René-Gontran dans son programme principal ?

Exercice 4 Pointeurs et tableaux ; tracer un programme

On considère le fichier source suivant :

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

float * create_tab(int n) {
  float * tab = calloc( n, sizeof(float) );
  float *p;
  for(p = tab ; p < tab + n ; p++) {
    *p = 1.0;
  }
  return tab;
}

void delete_tab(float *tab) {
  free(tab);
}

int main() {
  int n;
  float *t, *p;

  printf("Nb de float ? (>5 SVP) : ");
  scanf("%d", &n);
  t = create_tab(n);

  t[0] = 0;
  * (t+2) = 0;

  for(p = t ; p < t + n ; p++) {
    printf(" %f ", *p);
  }
  printf("\n");
  printf("%d", p-tab);

  delete_tab(t);

  return 0;
}
```

Lorsqu'on exécute ce programme, si l'utilisateur rentre 5, qu'affiche le programme ?