Nantes Université
IUT de Saint-Nazaire
Département Mesures Physiques

Semestre 1
Algorithmique et Informatique
Travaux dirigés

Exercice 1

Ecrire un programme qui demande à quelqu'un son âge et qui en déduit s'il est majeur ou pas. On fera un test de validité de l'âge fourni par l'utilisateur.

———

Notions à travailler :

- *Organigramme d'un algorithme simple et sa traduction en langage C*
- *Déclaration des variables : le type, les noms réservés.*
- *Fonctions scanf et printf de la bibiothèque stdio.h permettant une interaction textuelle utilisateur – machine.*
- *Branchement conditionnel avec if…else.*

Exercice 2

Ecrire un programme qui demande à l'utilisateur de taper 10 nombres entiers, puis affiche leur somme. Le programme ne devra pas utiliser de tableau.

———

Notions à travailler :

- *Organigramme d'un algorithme simple et sa traduction en langage C*
- *Déclaration des variables*
- *Boucle de répétition*
- *Fonctions scanf et printf*

Exercice 3

Un nombre entier naturel p est dit premier s'il n'est divisible que par 1 et par lui-même. Ecrire un programme qui demande à l'utilisateur une valeur de p, puis vérifie si p est un nombre premier ou pas. On sait que si aucun entier i compris entre 2 et $p-1$ ne divise p, alors p est un nombre premier. Au contraire, s'il existe dans cet intervalle un seul entier qui divise p, alors p ne peut pas être un nombre premier.

———

Notions à travailler :

- *Organigramme d'un algorithme simple et sa traduction en langage C*
- *Boucle de répétition paramétrée*
- *Branchement conditionnel avec if…else imbriqué dans la boucle*

Exercice 4

Ecrire un programme qui crée un tableau avec des nombres prédéfinis de votre choix. Ce programme détermine ensuite si un entier saisi par l'utilisateur est contenu dans ce tableau.

———

Notions à travailler :

- *Tableaux 1D, déclaration d'un tableau 1D : le type, la taille*
- *Accès à une ligne d'un tableau*
- *Parcourir un tableau à l'aide d'une boucle de répétition*

Exercice 5

Revenons à l'exercice 2, on a maintenant le droit d'utiliser les tableaux.

Ecrire un programme qui demande à l'utilisateur de taper 10 nombres réels, puis calcule et affiche la somme, la moyenne et l'écart-type des nombres saisis.

On donne les formules suivantes :

$$somme = \sum_{i=0}^{N-1} X_i$$

$$moyenne = \frac{1}{N} \sum_{i=0}^{N-1} X_i$$

$$ecart\_type = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N-1} (X_i - moyenne)^2}$$

Dans ces formules, il faut comprendre que :

- X est un tableau qui stocke des nombres réels
- X_i la ligne numéro i de X
- N le nombre total de lignes du tableau X de la ligne 0 à la ligne $N-1$

———

Notions à travailler :

- *Algorithme pour calculer une formule mathématique*

Exercice 6

Ecrire un programme qui affiche à l'écran les **L** premières lignes du triangle de Pascal, **L** étant un entier saisi par l'utilisateur.

———

Notions à travailler :

- *De l'observation à la généralisation d'un algorithme.*
- *Boucles imbriquées*
- *Tableaux 2D*

Lecture – Programme 1

En pratique, on est souvent demandé de faire des calculs sur les données de mesure stockées dans un fichier texte. Les tableaux sont très utiles pour cette tâche. Le programme ci-dessous permet d'ouvrir un fichier texte `"Mesures.txt"`, lire chaque ligne de ce fichier et mettre toutes les données dans un tableau **X**.

```c
#include <stdio.h>
void main()
{
    // Ouvrir le fichier text "Mesures.txt"
    //   qui est dans le même répertoire que ce programme C
    char    NomFichier[]  = "Mesures.txt";
    FILE    *PointeurFichier ;
    PointeurFichier = fopen(NomFichier, "r");

    // Préparation un tableau pour stocker les données récupérées du ficher text
    int nombre_mesures = 9;  // nombre de mesures qu'on souhaite récupérer
    float X[nombre_mesures]; // tableau de nombre_mesures lignes (ici 9 lignes)

    // valeur est une variable temporaire
    float valeur;

    // on parcourt chaque ligne du fichier avec une boucle for
    int i;
    for (i = 0; i < nombre_mesures; i = i+1)
    {
        // on récupère le nombre trouvé dans chaque ligne du fichier
        //    puis le mettre dans la variable valeur
        fscanf(PointeurFichier,"%f", &valeur);

        // puis on remettre cette valeur dans la ligne i du tableau X
        X[i] = valeur;
    }

    // on ferme le fichier text (OBBLIGATOIRE)
    fclose(PointeurFichier);

    // On affiche toutes les lignes du tableau pour vérifier
    for (i = 0; i < nombre_mesures; i = i+1)
    {
        printf("x[%d] = %f \n", i, X[i]);
    }
}
```

Lecture – Programme 2

Il est souvent demandé de stocker les résultats de calcul dans un fichier texte. Le programme suivant permet de créer un fichier texte "data.txt", puis mettre dans chaque ligne de ce fichier les valeurs numériques contenues dans les lignes correspondantes d'un tableau **D**.

```c
#include <stdio.h>
int main()
{
  // Créer un fichier texte "data.txt"
  //   qui est dans le même répertoire que ce programme C
  char    NomFichier[]  = "data.txt";
  FILE    *PointeurFichier ;
  PointeurFichier = fopen(NomFichier, "w");

  // on parcourt chaque ligne du fichier avec une boucle for
  int i, nbLigne;
  nbLigne = 20; // nbLigne est le nombre de lignes du tableau D
  for (i = 0; i < nbLigne; i = i+1)
  {
    // on écrit dans chaque ligne la valeur dans la ligne i du tableau D
    fprintf(PointeurFichier,"%d \n", D[i]);
  }
  // on ferme le fichier text (OBBLIGATOIRE)
  fclose(PointeurFichier);

  return 0;
}
```

Ecrire des fonctions en C

Les fonctions permettent de regrouper un algorithme ou des algorithmes qui réalisent une tâche précise dans un seul bloc. Le regroupement permet une meilleure organisation des programmes en C. On parle alors de la programmation modulaire.
Une librairie est tout simplement un ensemble de différentes fonctions dont chacune réalise une tâche particulière qu'on peut appeler à n'importe quel endroit dans le programme.

On a déjà vu et utilisé quelques fonctions auparavant. Les fonctions « printf » et « scanf » sont issues de la librairie standard « stdio.h ».

On va voir comment écrire des nouvelles fonctions en C.

Exercice 7
Ecrire une ***fonction*** qui calcule les racines d'un polynôme du second degré et affiche les résultats. Ecrire ensuite un ***programme*** qui demande à l'utilisateur les coefficients du polynôme, puis appelle la fonction ci-dessus pour afficher ses racines.

———

Notions à travailler :
- *Ecrire une fonction en C*
- *Fonction sans valeur retournée*

Exercice 8
Ecrire une ***fonction*** qui effectue un tirage de 2 dés à six faces et qui indique, par une valeur de retour, si le résultat constitue un double. Ecrire ensuite un ***programme*** qui effectue un certain nombre de tirages choisi par l'utilisateur et calcule ensuite le pourcentage des résultats doubles.

———

Notions à travailler :
- *Ecrire une fonction en C*
- *Fonction avec valeur retournée*

Exercices complémentaires

Exercice 9
Ecrire un programme qui exprime un nombre entier de secondes en jours, heures, minutes et secondes. Par exemple 100 000 secondes correspondent à 1 jour, 3 heures, 46 minutes et 40 secondes.

Exercice 10
Ecrire un programme qui calcule les intérêts produits chaque année pendant 30 ans par le placement en banque d'une somme de 1000 euros avec un intérêt de 1.5%.

Exercice 11
Ecrire un programme qui demande trois nombres réels positifs à l'utilisateur, qui indique s'ils peuvent correspondre aux longueurs des trois côtés d'un triangle et, si c'est le cas, qui indique si le triangle est équilatéral, isocèle et/ou rectangle.