Algorithmique

Travaux dirigés : feuille 1

Objectifs pédagogiques et méthode de travail. Cette feuille comporte deux parties.

— La partie 1 "Objectif 10" propose des exercices très proches du cours et des révisions du semestre 1. Avec les QCM en ligne (ENT), cette partie a pour objectif l'assimilation des connaissances de base, en particulier tout ce qui a été vu au semestre 1. Ces notions pleinement acquises permettent d'obtenir la moyenne lors des contrôles de connaissances (contrôle continu et sessions d'examens).

— La partie 2 "Objectif 20" propose des exercices de difficulté croissante sur des notions plus avancées ou des formulations plus abstraites. Les étudiant-e-s informaticien-e-s, et les étudiant-e-s mathématicien-e-s qui souhaitent continuer au delà de la Licence, doivent impérativement traiter cette partie 2.

1 Connaissances

— tableaux 1D et premiers traitements simples
— sous-programmes et sous-programmes avec des tableaux 1D
— un tout petit peu de complexité en temps

2 Objectif 10

Exercice 1.

1. Écrire l'en-tête d'une fonction `max` qui calcule la valeur maximale d'un tableau de taille quelconque de valeurs flottantes.
2. On fixe une taille de tableau. Écrire un algorithme qui réalise les entrées-sorties nécessaires et appelle plusieurs fois la fonction `max`.
3. Écrire le corps de cette fonction en justifiant votre choix de la structure de répétition.
4. Proposer les modifications minimales pour que la fonction `max` devienne la fonction `min`.

Exercice 2.

1. Écrire un algorithme qui calcule la moyenne d'un tableau de n valeurs flottantes.
2. Écrire (l'en-tête, puis l'appel puis le corps d') une fonction `moy` qui calcule et retourne la moyenne d'un tableau de n valeurs flottantes.

Exercice 3.

On rappelle quelques suites numériques classiques.

— la somme des n premiers entiers : $\sum_{k=1}^{n} k = 1 + 2 + \cdots + (n-1) + n$;
— la somme des n premiers carrés : $\sum_{k=1}^{n} k^2 = 1^2 + 2^2 + \cdots + (n-1)^2 + n^2$,
— la somme des n premiers cubes : $\sum_{k=1}^{n} k^3 = 1^3 + 2^3 + \cdots + (n-1)^3 + n^3$.
— Une suite arithmético-géométrique : $u_{n+1} = u_n/3 + 2$ avec $u_0 = 1$.
— La suite de Fibonacci : $u_{n+1} = u_n + u_{n-1}$ et $u_0 = u_1 = 1$.

1. Pour chacune des suites précédentes, écrire l'algorithme qui demande et lit une valeur de n entrée au clavier, calcule le n-ième terme de la suite puis l'affiche à l'écran.
2. Écrire des fonctions (en-tête, appel, corps) qui calculent les n premiers termes des suites précédentes et les stockent dans un tableau. L'appel incluera les phases d'entrées et de sorties de l'algorithme principale précédent.

Exercice 4.

La valeur de factorielle de n, où n est un entier positif est : $n! = 1 \times 2 \times \cdots \times (n-1) \times n$. On choisit $0! = 1$.

1. Écrire un algorithme qui calcule $n!$ pour une valeur de n arbitraire lue au clavier.
2. Écrire (d'abord l'en-tête, puis l'appel puis le corps) d'une fonction `factorielle` qui calcule et retourne $n!$.

Exercice 5.

L'algorithme suivant utilise une fonction `appartient` pour tester si un caractère donné apparait ou non dans la chaîne `mot`.

```
//role : je teste ma fonction appartient
declare
  mot : tableau[10] de caracteres = "Bonjour ! " //chaine a explorer
  c : caracteres                                 //caractere a chercher dans mot
  reponse : booleen                              //il y est ou non ?

debut
  c = 'j'
  reponse = appartient(mot, 10, c)
  afficher(reponse)                              //j'attends Vrai

  c = 'o'
  reponse = appartient(mot, 10, c)
  afficher(reponse)                              //j'attends Vrai

  c = 'a'
  reponse = appartient(mot, 10, c)
  afficher(reponse)                              //j'attends Faux
fin
```

1. Écrire un algorithme qui utilise la fonction `appartient` pour calculer le nombre d'occurrences d'un caractère entré au clavier dans une chaîne de caractères. Cette chaîne est définie dans l'algorithme. Prévoir le cas où le caractère est absent de la chaîne examinée.
   Indication : un mot de longueur 1 est un caractère.
2. Modifier l'algorithme précédent pour calculer et afficher l'indice de la première et de la dernière occurrence de ce caractère. Prévoir aussi le cas où le caractère est absent de la chaîne examinée.
3. Écrire l'en-tête de la fonction `appartient`.
4. Écrire le corps de cette fonction
   (a) avec un `pour`,
   (b) avec un `tantque`.
   Justifier les intérêts respectifs de ces structures de contrôle.
5. Modifier le corps de la fonction précédente pour que si caractère est présent plusieurs fois dans la chaîne, la recherche s'arrête sur :
   (a) la dernière occurrence,
   (b) la première occurrence.

Exercice 6.

Écrire l'algorithme principal qui calcule et affiche les racines d'une équation du second degré. Les coefficients de l'équation seront demandés au clavier et stockés dans un tableau. On veillera à ce que les coefficients entrés définissent bien une équation du second degré. Les racines réelles calculées seront affichées lorsqu'elles existent puis les relations suivantes seront vérifiées et un message de contrôle adapté sera affiché. Pour une équation du second degré $a_2x^2 + a_1x + a_0 = 0$,

— la somme de ses racines est égale à l'opposé du coefficient de degré 1 de l'équation normalisée, soit $x_1 + x_2 = a_1/a_2$

— Le produit des racines d'une équation du second degré est égale au coefficient constant de l'équation normalisée, soit $x_1 \times x_2 = -a_0/a_2$.

Exercice 7.

1. Écrire un algorithme qui calcule un booléen indiquant si un tableau d'entiers donné est trié par ordre croissant. Le tableau sera déclaré et initialisé en début d'algorithme sans solliciter l'utilisateur.
2. Écrire une fonction `trié_ou_pas_trié` (c'est-à-dire son en-tête, puis l'appel puis le corps) qui retourne un booléen indiquant si un tableau d'entiers donné est trié par ordre croissant. L'appel pourra inclure l'entrée d'un nombre fixé de valeurs du tableau.

Exercice 8.

1. Écrire un algorithme qui inverse l'ordre des valeurs d'un tableau donné : le tableau $[a_0, a_1, ..., a_{n-1}]$ sera renversé en $[a_{n-1}, ..., a_1, a_0]$.
2. Écrire une fonction `renverse` (c'est-à-dire son en-tête, puis l'appel puis le corps) qui effectue le traitement précédent sur un tableau passé en paramètre.

3 Objectif 20

Exercice 9.
On reprend l'exercice 3.

1. Écrire un algorithme qui calcule le n-ième terme d'une somme $\sum_{k=1}^{n} k^i$, pour une valeur entière $i = 1, 2, \ldots, p$ quelconque et entrée (lue) au clavier par l'utilisateur. Cet algorithme doit calculer le résultat attendu quelque soit la valeur (positive) entrée i. Il permettra bien sûr de retrouver les calculs des 3 suites de l'exercice 3 pour $i = 1, 2, 3$.
2. Transformer l'algorithme précédent en une fonction.

Exercice 10.
Écrire (d'abord l'en-tête, puis l'appel puis le corps) d'une fonction `Pascal` qui calcule et retourne la i-ème ligne du triangle de Pascal.

Exercice 11. On souhaite évaluer la valeur d'un polynôme $p(x) = \sum_{i=0}^{n} a_i x^i$, de degré donné n en des valeurs x arbitraires.

1. Proposer une structure de donnée pour représenter p.
2. Écrire (d'abord l'en-tête, puis l'appel puis le corps) d'une fonction qui calcule et retourne $p(x)$.
3. Écrire (d'abord l'en-tête, puis l'appel puis le corps) d'une fonction qui calcule et retourne $p(X)$ où $X = (x_k)_{k=1,m}$ est un vecteur de m valeurs d'évaluation.
4. (a) Détailler le principe d'un algorithme qui calcule une racine d'un polynôme sur un intervalle $[a, b]$ donné. La présence d'une racine unique sur $[a, b]$ sera admise MAIS l'existence d'ue racine sera vérifiée par l'algorithme. Introduire un paramètre adapté au caractère approximatif de la valeur calculée.
   (b) Écrire l'en-tête et le corps d'une fonction effectuant ce calcul ainsi qu'un algorithme principal qui traite le problème pour un polynôme p, un degré n et un intervalle $[a, b]$ donnés.

Exercice 12. Écrire un ensemble "algorithme principal d'appel et fonctions d'entrées-sorties et de traitements" qui calcule le produit de deux polynômes p et q de degré respectif n et m quelconque.

Exercice 13.

1. Écrire (d'abord l'en-tête, puis l'appel puis le corps) d'une fonction `est_premier` qui retourne un booléen qui indique si un nombre entier donné n est premier ou non.
2. Écrire un algorithme qui affiche la décomposition en facteurs premiers d'un nombre entier donné.

Exercice 14.

1. Décrire le principe du calcul de la valeur médiane de n valeur données sous la forme d'un tableau.

2. Écrire (l'en-tête, puis l'appel puis le corps d') une fonction `mediane` qui calcule et retourne la médiane d'un tableau de n valeurs entières.
3. Compléter cet algorithme pour décompter le nombre de comparaisons entre éléments du tableau effectuées lors de ce calcul.
4. Que pensez-vous de la complexité en temps de cet algorithme ? Expliciter la complexité dans le pire cas.
5. Coder cet algorithme en C et observer ce nombre selon la valeur de n et du tableau de valeurs. Qu'en pensez vous ?