

Examen de Langage C (*V. Granet*)

Durée : 1h30

Aucun document autorisé

Mobiles interdits

On lit sur l'entrée standard une suite d'entiers quelconques pris sur le type (`int`). Parmi les entiers lus, on veut mémoriser dans un tableau uniquement ceux qui sont positifs et pairs. Le tableau a une taille maximale donnée par la constante `NB_MAX_VALEURS`.

- 1. Écrivez la procédure `lireValeurs` qui fait le traitement précédent. Le nombre d'entiers effectivement mémorisés sera conservé dans le premier élément du tableau. Note : la fonction `scanf` renvoie EOF quand la fin de fichier est atteinte.

Cette procédure possède l'en-tête (que vous devez respecter) suivant :

```
// conséquent : ...
void lireValeurs(int tVal[])



---


/*
 * Antécédent : x un entier quelconque
 * Conséquent : renvoie 1 si x pair et positif, 0 sinon
 */
int valide(const int x) {
    return x>=0 && (x&1)==0;
}

/*
 * Conséquent : tVal contient tVal[0] (<NB_MAX_VALEURS) entiers
 *              positifs et pairs lus sur l'ES
 */
void lireValeurs(int tVal[]) {
    int x, nbVal=0;
    while (scanf("%d", &x)!=EOF) {
        // ∀k ∈ [1;nbVal], tVal[k] contient
        // un entier pair et positif lu sur l'E/S
        if (nbVal<NB_MAX_VALEURS-1 && valide(x))
            tVal[++nbVal] = x;
    }
    // nbVal est le nombre (<NB_MAX_VALEURS) d'entiers valides lus sur l'E/S
    // et enregistrés dans le tableau t => mémoriser nbVal dans tVal[0]
    tVal[0]=nbVal;
}
```

-
- 2. Déclarez le type énuméré `LesStats` qui contient les valeurs, dans cet ordre, `EFFECTIF`, `MIN`, `MAX`, `MOYENNE`, `ECART_TYPE`.

```
enum LesStats {EFFECTIF, MIN, MAX, MOYENNE, ECART_TYPE};
```

.....

Maintenant, on souhaite faire quelques calculs statistiques : conserver l'effectif, et calculer le minimum, le maximum, la moyenne, ainsi que l'écart-type ($\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$) des entiers

mémorisés dans le tableau.

- 3. Écrivez la procédure `stats` qui, à partir d'un tableau (initialisé avec la procédure `lireValeurs` précédente), remplit un second tableau avec les statistiques précédentes aux indices donnés par le type énuméré `LesStats`. Cette procédure possède l'en-tête (que vous devez respecter) suivant :

```
// Antécédent : ...
// Conséquent : ...
void stats(const int tVal[], double tStat[])



---


/*
 * Antécédent : le tableau tVal contient detStat entiers positifs et pairs.
 *               leur nombre, éventuellement 0, est mémorisé dans tVal[0]
 * Conséquent : le tableau tStat contient aux indices données par enum LesStats
 *               l'effectif, le minimum, le maximum, la moyenne et l'écart-type
 *               si l'effectif = 0, les valeurs du minimum, du maximum, de
 *               la moyenne et de l'écart-type sont indéfinies
 */
void stats(const int tVal[], double tStat[]) {
    // mémoriser l'effectif
    tStat[EFFECTIF]=tVal[0];
    if (tVal[0]>0) {
        // calculer les statistiques
        int min, max, som;
        min = max = som = tVal[1];
        // calculer le minimum, le maximum, la moyenne
        for (int i=2; i<=tVal[0]; i++) {
            // ∀k ∈ [1;i-1] min = minimum des tVal[k]
            // max = maximum des tVal[k], et som = somme des tVal[k]
            som+=tVal[i];
            if (tVal[i]<min) min = tVal[i];
            else
                if (tVal[i]>max) max = tVal[i];
        }
        // calculer la moyenne
        double moyenne = som/(double) tVal[0];
        // calculer l'écart-type
        double somDiffCarre=0.0;
        for (int i=1; i<=tVal[0]; i++) {
            somDiffCarre += (moyenne-tVal[i])*(moyenne-tVal[i]);
        }
        double ecartType = sqrt(somDiffCarre/ tVal[0]);
        // mémoriser les statistiques dans tStat
        tStat[MIN]=min;
        tStat[MAX]=max;
        tStat[MOYENNE]=moyenne;
        tStat[ECART_TYPE]=ecartType;
    }
}
```

-
- 4. Écrivez la procédure `ecrireStats` qui écrit sur la sortie standard les statistiques calculées.

```
/*
 * Rôle : écrit sur la sortie standard les statistiques contenues
 *         dans le tableau tStat
 */
```

```

void ecrireStats(const double tStat[]) {
    if (tStat[EFFECTIF]==0)
        printf("Effectif égal à 0 => pas de statistiques\n");
    else {
        printf("statistiques sur %.0f valeurs : ", tStat[EFFECTIF]);
        printf("min = %.0f, max = %.0f, moyenne = %.2f, écart-type = %.2f\n",
            tStat[MIN], tStat[MAX], tStat[MOYENNE], tStat[ECART_TYPE]);
    }
}

```

- 5. Écrivez la fonction `main` qui teste les procédures précédentes. Vous prendrez soin de bien déclarer les tableaux. Par exemple, si l'entrée standard contient les valeurs suivantes :

```

12 34 -39 100 399 432 -8
19 34 56

```

l'exécution du programme fournira le résultat suivant :

```

statistiques sur 6 valeurs : min = 12, max = 432,
                             moyenne = 111.33, écart-type = 145.97

```

```

#define NB_MAX_VALEURS 20
#define NB_STATS 5
int main(void) {

    int lesValeurs[NB_MAX_VALEURS];
    double lesStats[NB_STATS];

    lireValeurs(lesValeurs);
    stats(lesValeurs, lesStats);
    ecrireStats(lesStats);

    return EXIT_SUCCESS;
}

```

Une matrice de m lignes et n colonnes contient des caractères (type `char`). On veut écrire fonction `lgMaxSuite` qui renvoie la taille de la plus longue suite formée de caractères `c` consécutifs sur toutes les lignes de la matrice. Par exemple, la plus longue suite de caractères `'*'` se trouve à la 5^e ligne (en rouge) dans la matrice 7×15 ci-dessous et est égale à 9 :

```

O      *
      ***  O
O  *****
      *
*****
*****  O
      *
      *
*--  *** --*

```

- 6. Écrivez la fonction `lgMaxSuite` comme indiqué précédemment :

```

// Antécédent : m>0 et n>0, c caractère de la plus longue suite dans mat
// Rôle : renvoie la longueur de la plus longue suite de c
int lgMaxSuite(int m, int n, char mat[][n], char c) {

```

```

int lgMaxSuite(const int m, const int n, const char mat[][n], const char c) {
    int lgSuiteMax=0;
    for (int i=0; i<m; i++) {
        // calculer le nombre de caractères c consécutives de la ligne i
        int nbStar=0;
        char pred=mat[i][0];
        for (int j=1; j<n; j++) {
            if (mat[i][j]==c)
                nbStar = (pred!=c) ? 1 : nbStar+1;
            //
            pred = mat[i][j];
        }
        if(nbStar>lgSuiteMax) lgSuiteMax=nbStar;
    }
    return lgSuiteMax;
}

```

.....