Algorithmique et programmation - Séance 2

Révisions - Solutions

Exercice 9.

Un ensemble de n intervalles peut être représenté par un tableau de n lignes et deux colonnes. Chaque ligne représente un intervalle. La première colonne de la ligne contient la borne inférieure de l'intervalle, la seconde colonne sa borne supérieure. On supposera ici que ces bornes sont du type double, que les intervalles sont fermés à gauche et à droite et qu'ils sont triés par ordre croissant de leur borne inférieure.

Exemple .

L'ensemble d'intervalles $\{-1.5, 3.4\}, \{2.3, 6.9\}, \{3.5, 4.7\}, \{7.2, 8.5\}, \{8.5, 9.1\}$ peut être représenté par le tableau suivant :

-1.5	3.4
2.3	6.9
3.5	4.7
7.2	8.5
8.5	9.1

En utilisant cette représentation d'un ensemble d'intervalles, on demande d'écrire une fonction qui pour un ensemble d'intervalles passé en paramètre, renvoie l'ensemble d'intervalles obtenu en réunissant les intervalles qui se chevauchent.

Pour l'exemple précédent, la fonction devra renvoyer le tableau suivant :

-1.5	6.9
7.2	9.1

Correspondant à l'ensemble d'intervalles $\{-1.5, 6.9\}, \{7.2, 9.1\}$.

```
#include <iostream>
using namespace std;

const int MAX = 20;

void unionInterv(double ensInterv[][2], int nbInterv,
                double ensIntervRes[][2], int &nbIntervRes)
{
    double borneInf, borneSup; //Bornes supérieure et inférieure de
                                //l'intervalle en cours de construction.
    int i;

    nbIntervRes = 0;
    if (nbInterv != 0)
    {
        borneInf = ensInterv[0][0];
        borneSup = ensInterv[0][1];
        for (i = 1; i < nbInterv; i++)
        {
            if (ensInterv[i][0] > borneSup)
            {
                //Il n'y a pas chevauchement avec les précédents
                //Ecriture dans le vecteur d'un nouvel intervalle.
                ensIntervRes[nbIntervRes][0] = borneInf;
                ensIntervRes[nbIntervRes][1] = borneSup;
                //Mise à jour des bornes
            }
        }
    }
}
```

```

        borneInf = ensInterv[i][0];
        borneSup = ensInterv[i][1];
        nbIntervRes++;
    }
    else //Chevauchement avec les précédents.
        if (ensInterv[i][1] > borneSup)
            borneSup = ensInterv[i][1];
    }
    //Ecriture dans le vecteur du dernier intervalle.
    ensIntervRes[nbIntervRes][0] = borneInf;
    ensIntervRes[nbIntervRes][1] = borneSup;
    nbIntervRes++;
}
}

void ecrireEnsInterv(double ensInterv[][2], int nbInterv)
{
    int i;
    for (i = 0; i < nbInterv; i++)
        cout << "[" << ensInterv[i][0] << ", " << ensInterv[i][1] << "]" << endl;
}

int main()
{
    double ensInterv[][2] = {{ -1.5, 3.4 }, { 2.3, 6.9 }, { 3.5, 4.7 },
                             { 7.2, 8.5 }, { 8.5, 9.1 }};
    double ensIntervRes[MAX][2];
    int nbIntervRes;

    unionInterv(ensInterv, 5, ensIntervRes, nbIntervRes);
    ecrireEnsInterv(ensIntervRes, nbIntervRes);
}

```

Exercice 10.

Ecrire une fonction qui remplace plusieurs occurrences consécutives d'une même valeur dans un tableau d'entiers par une seule occurrence de cette valeur. La fonction recevra comme paramètre le tableau et son nombre d'éléments significatifs. Celui-ci devra être modifié par la fonction pour refléter les suppressions éventuelles d'éléments.

Exemple.

Si le tableau passé en paramètre à la fonction contient les 10 valeurs significatives suivantes.

2	2	1	2	3	3	3	2	1	1	?	?
---	---	---	---	---	---	---	---	---	---	---	---

Au retour de la fonction le tableau contiendra les 5 valeurs suivantes.

2	1	2	3	2	1	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---	---	---	---

Les "?" représentent des valeurs non significatives.

```

#include <iostream>
using namespace std;

void supprRepet(int tab[], int &n)
{
    int posSvt = 1;
    int prec = tab[0];
    for(int i = 1; i < n; ++i)
        if(tab[i] != prec)
        {
            prec = tab[posSvt] = tab[i];
            ++posSvt;
        }
}

```

```

    }
    n = posSvt;
}

int main()
{
    int tab[] = {2, 2, 1, 2, 3, 3, 3, 2, 1, 1};
    int n = 10;
    supprRepet(tab, n);
    for(int i = 0; i < n; ++i)
        cout << tab[i] << " ";
    cout << endl;
    return 0;
}

```

Exercice 11.

Ecrire une fonction qui supprime les espaces superflus d'une phrase. La fonction recevra comme paramètre une chaîne de caractères contenant la phrase et retournera une chaîne de caractères contenant la même phrase mais dans laquelle on a supprimé les espaces de début et de fin et dont les mots sont séparés par un seul espace. On considère qu'un mot est un ensemble de caractères quelconques bordés soit par un espace, soit par le début ou la fin de la chaîne.

Exemple.

Chaîne fournie à la fonction :

" Cet exercice est très facile "

Chaîne retournée par la fonction :

"Cet exercice est très facile"

```

#include <iostream>
using namespace std;

void supprEspV1(char s[], char res[])
{
    int j = 0; //Position dans res.
    bool espAvant = false; //Indique s'il y a un espace avant
                          //le caractère courant.

    for (int i = 0; s[i] != '\0'; i++)
        if (s[i] != ' ')
        {
            if (espAvant && j != 0)
            {
                //Ajout d'un blanc avant le mot
                //sauf pour le premier mot.
                res[j] = ' ';
                j++;
            }
            res[j] = s[i];
            j++;
            espAvant = false;
        }
        else
            espAvant = true;
        res[j] = '\0';
}

void supprEspV2(char s[], char res[])

```

```

{
  int i, j = 0; //Position dans s et res.
  bool espAvant = false; //Indique s'il y a un espace avant
                        //le caractère courant.

  //Passer les premiers espaces.
  for (i = 0; s[i] != '\0' && s[i] == ' '; i++);

  for (; s[i] != '\0'; i++)
    if (s[i] != ' ')
    {
      if (espAvant)
      {
        res[j] = ' ';
        j++;
        espAvant = false;
      }
      res[j] = s[i];
      j++;
    }
    else
      espAvant = true;
  res[j] = '\0';
}

main()
{
  char s1[] = "  Cet  exercice  est  très  facile  ";
  char s2[100];

  supprEspV1(s1, s2);
  cout << s2 << "*" << endl;
  supprEspV1(s1, s2);
  cout << s2 << "*" << endl;
}

```

Exercice 12.

Ecrire une fonction qui reçoit comme paramètre une chaîne de caractères et compte le nombre de mots écrit dans un tableau la longueur de chaque mot de cette chaîne.

On considère qu'un mot est un ensemble de caractères quelconques bordés soit par un espace, soit par le début ou la fin de la chaîne.

Exemple.

Si la chaîne est

" Une chaîne de caractères "

Le tableau des longueurs de mots contiendra les valeurs

3	6	2	10
---	---	---	----

```

#include <iostream>
using namespace std;

void longMots(char s[], int tab[], int &taille)
{
  taille = 0;
  for(int i = 0; s[i] != '\0';)
  {
    //On passe les espaces.
    for(; s[i] == ' '; ++i);

```

```

    if(s[i] != '\0')
    {
        int debutMot = i;
        //On passe le mot.
        do
            ++i;
        while(s[i] != '\0' && s[i] != ' ');
        tab[taille++] = i - debutMot;
    }
}

void longMotsV2(char s[], int tab[], int &taille)
{
    taille = 0;
    int lg = 0 ;
    for(int i = 0; s[i] != '\0';)
    {
        if (s[i] != ' ')
            lg++;
        else
            if (lg > 0)
            {
                tab[taille++] = lg;
                lg = 0;
            }
    }
}

int main()
{
    int tab[100];
    int taille;
    longMots(" Une chaîne de caractères ", tab, taille);
    for(int i = 0; i < taille; ++i)
        cout << tab[i] << " ";
    cout << endl;
    longMotsV2(" Une chaîne de caractères ", tab, taille);
    for(int i = 0; i < taille; ++i)
        cout << tab[i] << " ";
    cout << endl;
    return 0;
}

```

Exercice 13.

Ecrire une fonction qui supprime d'une chaîne de caractères les répétitions de caractères identiques consécutifs. La fonction remplacera une suite de plusieurs caractères identiques par un seul de ces caractères. La fonction ne doit pas créer une nouvelle chaîne de caractères mais modifier la chaîne qui lui est transmise comme paramètre.

Exemple:

La fonction devra transformer la chaîne "aaabccaaabbaaa" en la chaîne "abcaba".

```

#include <iostream>
using namespace std;

void supprRepetitionsCar(char s[])
{
    char c = '\0';
    int posSvt = 0;

    for (int i = 0; s[i] != '\0'; i++)
    {

```

```
if (s[i] != c)
{
s[posSvt] = s[i];
c = s[i];
posSvt++;
}
}
s[posSvt] = '\\0';
}
```

```
main()
{
char s[100];

cin >> s;
supprRepetitionsCar(s);
cout << s << endl;
}
```