Université Hassan II

Casablanca Faculté des Sciences et Techniques – Mohammedia

Département Informatique

Module : Structures des données

COMPTE RENDU

Manipulation des tableaux statiques

Encadré par :

- Mr *Abdelkrim* BEKKHOUCHA

Réalisé par :

- OUTGOUGA Jalal eddine
- ZADDI Abdelmajid

Année universitaire : 2022-2023

I. Introduction

Dans ce compte rendu nous allons programmer quelques fonctions pour la manipulation des tables statiques, en tant que structure.

II. Analyse et Programme

- **Définition de la structure de notre tableau :**

```c
#include <stdio.h>
#include <stdlib.h>
#define NbElem 25

//Définition de la structure d'un tableau

typedef struct
{
    int Tab[NbElem];
    int IdElem;//Indice du dernier element du tableau
}MaTable;

```

- **Fonction d'initialisation du tableau :**

```c
int InitTab(MaTable *MonTableau)
{
    if(!MonTableau)return((int)0);
    MonTableau->IdElem=-1;//Dire que le tableau est vide
    return (int)1;
}
```

- **Vérifier que le tableau est vide :**

```c
int TabVide(MaTable MonTableau)
{
    if(MonTableau.IdElem==-1)return((int)1);
    return ((int)-1);
}
```

- **Fonction qui vérifie si le tableau est plein :**

```c
int TabSaturee(MaTable MonTableau)
{
    if(MonTableau.IdElem==NbElem)return((int)1);
    return ((int)0);
}
```

- **Fonction d'insertion dans le tableau :**

```
39    int InsertTab(MaTable *MonTableau,int entier)
40    {
41        //Vérifier que la zone memoire est resirver
42        if(!MonTableau)return ((int)-1);
43        //Verifier que le tableau n'est pas saturée
44        if(TabSaturee(*MonTableau))return ((int)0);
45        //Insertion dans le tableau
46        MonTableau->Tab[++MonTableau->IdElem]=entier;
47        return ((int)1);
48    }
49
```

- **Fonction d'insertion dans le tableau dans un indice donné :**

```
52    int InsertPos(MaTable *MonTableau,int entier,int pos)
53    {
54        int i;
55        //Vérifier que la zone memoire est resirver
56        if(!MonTableau)return ((int)-1);
57        //Verifier que le tableau n'est pas saturée
58        if(TabSaturee(*MonTableau))return ((int)0);
59        //Verifier que la position ne depassse pas la taille du tableu
60        if(MonTableau->IdElem<pos)return ((int)2);
61        //Tasser les element du tableau
62        for(i=MonTableau->IdElem;i<pos;i--)
63        {
64            MonTableau->Tab[i]=MonTableau->Tab[i+1];
65        }
66        //Affecter la varible à l'indice donnée
67         MonTableau->Tab[pos]=entier;
68         //Incrementer la taille du tableau
69         MonTableau->IdElem++;
70
71         return ((int)1);
72    }
```

- **Fonction de suppression d'un élément d'indice donné :**

```
80    int SupElem(MaTable *MonTableau,int pos)
81    {
82        int i;
83        //Vérifier que la zone memoire est resirver
84        if(!MonTableau)return ((int)-1);
85            //Verifier que la position ne depassse pas la taille du tableu
86        if(MonTableau->IdElem<pos)return ((int)2);
87        //Tasser les element du tableau
88        for(i=pos ;i<MonTableau->IdElem;i++)
89            MonTableau->Tab[i]=MonTableau->Tab[i+1];
90
91
92           MonTableau->IdElem--;
93
94        return ((int)1);
95    }
```

- **Fonction pour supprimer tous les occurrences d'un élément dans un tableau :**

```
100   int SupOcur(MaTable *MonTableau,int occu)
101   {
102       int indice;
103       for(indice=0;indice<=MonTableau->IdElem;indice++)
104       {
105           if(MonTableau->Tab[indice]==occu) SupElem(MonTableau,indice);
106       }
107   }
```

- **Insertion dans un tableau ordonné :**

```
112   int InsertOrdTab(MaTable *MonTableau,int entier)
113   {
114       int i=0;
115       //Vérifier que la zone memoire est resirver
116       if(!MonTableau)return ((int)-1);
117       //Verifier que le tableau n'est pas saturée
118       if(TabSaturee(*MonTableau))return ((int)0);
119       //chercher l'indice de l'insertion
120       while(MonTableau->Tab[i++]<entier);
121
122       InsertPos(MonTableau,entier,i);
123
124       return ((int)1);
125
126   }
```

- **Recherche d'un élément dans un tableau :**

```
131   int RechercheTab(MaTable MonTableau,int entier)
132   {
133       int i,n=0;
134       for(i=0;i<=MonTableau.IdElem;i++)
135       {
136           if(MonTableau.Tab[i]==entier)n=1;
137       }
138       return ((int)n);
139   }
140
141
```

III. Conclusion

Nous avons vu qu'en C, un tableau fait une réservation de mémoire de taille correspondant à la dimension Déclarée et à la taille des éléments du tableau ce qui pose un petit problème, car à chaque foi on est obligé de Vérifier la taille ce qui n'est pas le cas dans un pointeur .