Université M'Hamed BOUGARA - Boumerdes
Faculté des sciences – Département d'Informatique

Algorithmique et structure de données

Présentée par

Iza lyla

lylaiza@yahoo.fr

Contenu du module

- Introduction
- Historique
- Généralités et concepts de base
- Algorithmes séquentiels simple
- Les structures conditionnelles (en langage algorithmique et en C)
- Les boucles (en langage algorithmique et en C)
- Les tableaux et les chaînes de caractères
- Les types personnalisés

Les structures de données

Plan du cours 5: Les structures de données

- Introduction sur les structures de données
- Tableau unidimensionnel (vecteurs)
- Tableau bidimensionnels (matrices)
- Les chaines de caractères

Structure de données

Une structure de donnée est la façon dont on peut représenter les données en mémoire en générale. Selon le type de données et leur complexité on peut décider de représenter une donnée en utilisant une structure bien déterminée.

Selon le type de données, on peut trouver :

- Les données simple : numérique, logiques ou caractères.
- Les chaines de caractères;
- Les ensembles;
- Les tableaux à une dimension (vecteurs);
- Les tableaux à deux dimensions (matrices);
- Les enregistrements (articles);
- Les fichiers;

Ces structures sont statiques car elles occupent un espace fixe en mémoire. Il existe d'autres structures dynamique.

Structure de données (suite)

1. **Les donnée simple :** sont représentées par des variables simples vu auparavant, où chaque variable pourrait avoir les types simple (entier, reel, char, …);
2. **Les données structurées :** sont les données que l'on peut traiter, ne se présentent pas toujours de façon simple, la représentation d'une liste de valeurs en mémoire ne peut pas se faire à l'aide de variables simples, d'où l'obligation de recourir aux variables de type structurées.
3. **Implémentation en mémoire :** les données du problème doivent être présenter en mémoire pour pouvoir les manipuler. On trouve deux implémentation :
   - **Implémentation statique ou contiguë :** les variables du programme sont déclarées au début et occupe un espace mémoire durant toute l'exécution du programme, ce qui rende la gestion de la mémoire, une tache très difficile.
   - **Implémentation dynamique :** ici les variables sont crées au moment de leur utilisation et supprimées après.

les tableaux

Tableau unidimensionnel (vecteur)

Définition

Un tableau est une variable indicée composée de plusieurs variables de même type, stockage statique en mémoire.

L'espace mémoire occupé est conditionné par le type de donnée que le tableau contient

Un tableau est déterminé par :

- Un ***nom*** unique
- Un ***type*** unique applicable à l'ensemble des éléments du tableau.

Comme toute variable, un tableau doit être déclaré et initialisé avant d'être utilisé

Déclaration d’un tableau unidimensionnel (à une dimension)

La déclaration permet de réserver en mémoire l’espace nécessaire pour ranger tous les éléments du tableau. Il faut donc indiquer :

- Le ***nombre*** d’éléments ***maximum*** pouvant être stocké dans le tableau.
- La ***nature*** des éléments contenus dans le tableau.

La déclare d’un tableau ou vecteur ce fait donc par un ***identificateur***, le mot clé ***tableau***, ***la taille*** du tableau et le ***type de données*** qu’il contient.

Déclaration d'un tableau unidimensionnel (2)

La syntaxe de la déclaration d'un tableau unidimensionnel en algorithmique :

```
var <nom_tab> : tableau [taille] de <type de base>;
```

Où :

- *Nom_tab* : est identificateur qui désigne le nom du tableau.
- *Taille* : est une constante entière positive indiquant le nombre d'éléments du tableau.
- *Un élément* est désigné par <nom_tab> [indice] ou <nom_tab>(indice).
- *L'indice* peut prendre une valeur positive entre 1 et <taille> et entre 0 et (<taille> - 1) en c.

Déclaration d'un tableau unidimensionnel (2)

La syntaxe de la déclaration d'un tableau en c est :

```
<type des éléments du tableau><nom_tableau> [taille];
```

1. **Exemple d'une déclaration en algorithmique :**

```
var tab1 : tableau[100] de reel;
```

2. **Exemple d'une déclaration en c :**

```
int tab1[5];   char caract [100];
```

Remarques

1. La taille d'un tableau doit être connue lors de la compilation
2. Tous les emplacements ne seront pas obligatoirement occupés
3. Les tableau consomment beaucoup de place en mémoire.
4. Impossible d'utiliser une variable pour déclarer la taille d'un tableau : par exemple int n; int tab[n] ; provoquera une erreur

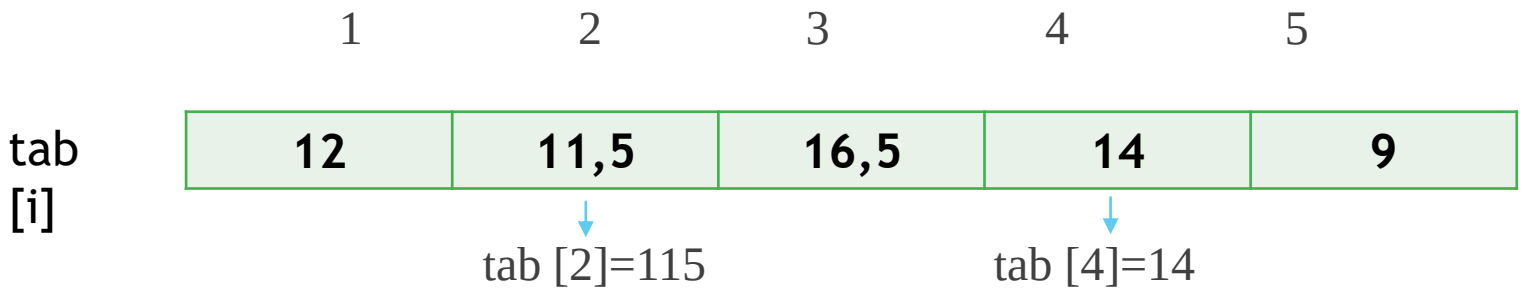
Accéder à un élément d'un tableau

On accède (en lecture ou en écriture) à la i-ème valeur d'un tableau en utilisant la syntaxe suivante :

```
nom de la variable [indice];
```

Par exemple si tab est un tableau de 5 entiers alors sa déclaration est

```
var tab : Tableau [10] de reel;
```



- Les indices du tableau variant de 1 à 5;
- Pour accéder à un élément de ce tableau, il suffit d'indiquer sa position.

Par exemple, `tab [3]` représente le 3éme élément du tableau qui est 16,5; si x est une variable tel que `x← tab[5]` donc `ecrire (x)` donne 9.

Opérations sur un élément d'un tableau

Un élément d'un tableau (repéré par le nom du tableau et son indice) pouvant être manipulé exactement comme n'importe quelle variable ordinaire, on peut donc effectuer des ***opérations*** sur les éléments du tableau : tel que les opérations *d'affectation*, de *lecture* et *d'écriture*.

Exemple :

- Tab [5] ← 25; : c'est affecté 25 à l'élément d'indice 5 du tableau tab.
- Ecrire (tab[5]); : c'est affiché le 5-éme élément du tableau.
- Lire (tab[2]); : c'est d'affecté au 2-éme élément du tableau tab, la valeur lu.

Initialisation d’un tableau :

Avant toute utilisation d’un tableau, il faut tout d’abord qu’il soit initialisé,

1. soit par des valeurs entrées par lecture ;
2. ou bien en lui affectant des valeurs nulle (c.àd. Tab[i] = 0 pour toutes les case du tableau).
3. Soit l’initialisation se fait lors de la déclaration, en plaçant entre accolades les valeurs initiale du tableau séparées par des virgules :

   int t[10]={1,8,9,7,3};

   - Le nombre d’élément entre accolade ne doit pas être supérieur au nombre d’éléments du tableau
   - Les valeurs entre accolades doivent être des constante (l’utilisation des variable provoque des erreurs lors de la compilation)
   - Si le nombre de valeurs entre accolades est inférieur au nombre d’éléments du tableau, les derniers éléments seront initialisés automatiquement à 0.
   - Il doit y avoir au moins une valeur entre accolades, l’instruction suivante permet d’initialiser tous les éléments du tableau à zéro : int T[5]={0};

Exemple :

```
Algorithme initialisation;
Var tab :tableau [10] d’entier;
Var i : entier;
Debut
Pour i allant de 1 à 10 faire
Tab[i] = 0;
Fin pour;
Fin.
```

Remplissage d’un tableau

Le remplissage d’un tableau consiste à entrer les valeurs qu’il doit contenir. Pour ce faire il existe deux méthodes :

1. La première consiste, à entrer directement les éléments du tableau à partir du clavier, par l’utilisation des lectures successives de ses éléments ; comme le montre l’algorithme suivant :

```
Algorithme lire_elem_tab;
Var vect : tableau [n] de <type> // n est une constante.
 var i : entier;
Debut
Pour i allant de 1 à n faire
Lire (vect [i]);
Finpour;
Fin.
```

Exemple 1 :

Ecrire un algorithme qui permet de remplir un tableau (vecteur) avec des valeurs saisies par l'utilisateur.

En algorithmique

```
Algorithme remplir_tab
Var vect : tableau [10] de
reel;
Var i : entier;
Debut
Pour i allant de 1 à 5
faire
Lire (vect [i]);
Fin.
```

En c

```
#include <stdio.h>
 int main(){
float vect [10] ;
Int i ;

for (i=1;i<=10;i++) {
Scanf (’’ %d ’’,&vect [i];
}
return 0;
}
```

Dans ce cas l'algorithme attend que l'utilisateur saisira les 5 nombre réel aux clavier.

Remplissage d’un tableau (suite)

2. La deuxième consiste, à remplir le tableau par le calcule de ses éléments à partir d’une expression; comme le montre l’algorithme suivant :

```
Algorithme remplir_tab_cal;
Var vect : tableau [n] de « type1 »; // n est une constante
Var i : entier;
Element : « type1 »
Debut
Pour i allant de 1 à n faire
Debut
Element ← « expression en fonction  de i »;
Vect  [i] ← element;
Fin pour;
Fin.
```

Exemple 2 :

Entrer les 5 premiers nombres impaire :

En algorithmique :

```
Algorithme
remplir_tab_impaire;
Var tab : tableau
[5]d’entier;
Var i : entier;
Debut
Pour i allant de 1 à 5 faire
Tab[i] ← i*2-1;
Finpour;
Fin.
```

En c :

```
#include <stdio.h>
int main (){
int tab [5],i ;

for (i=1;i<=5;i++) {
tab[i]=i*2-1;
}
return 0;
}
```

Affichage d’un tableau

- L’écriture d’un tableau est l’opération qui permet d’afficher son contenu à l’écran.

```
Algorithme affiche_tab;
Var tab : tableau [n] de « type »; // n est d’une constante.
Var i : entier;
Debut
Pour i allant de 1 à n faire
Ecrire (tab[i]);
Fin pour;
Fin.
```

Exemple :

Ecrire un programme qui permet de remplir un tableau de 5 entiers, et d'afficher son contenu :

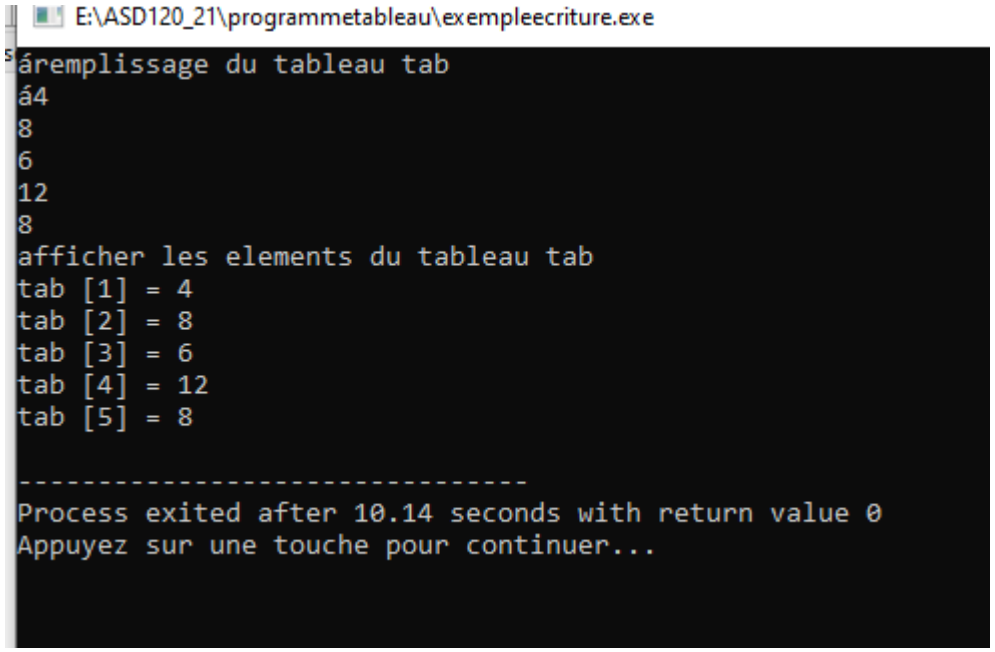
En algorithmique

```
Algorithme remp_affich;
Var tab : tableau [5] d'entier;
Var i : entier;
Debut
Ecrire (’’ remplissage du tableau tab ’’);
Pour i allant de 1 à 5 faire
 lire (tab[i]);
Fin pour;
Ecrire (’’afficher les elements du tableau tab’’);
Pour i allant de 1 à 5 faire
Ecrire (’’tab [’’,i,’’] = ’’, tab [i]);
Fin pour;
Fin.
```

La traduction de l’exemple précèdent en c :

```
#include <stdio.h>
int main () {
int tab [5],i;
printf (" remplissage du tableau tab \n ");
 for (i=1;i<=5;i++){
scanf("%d",&tab[i]);
}
printf ("afficher les elements du tableau tab \n");
 for (i=1;i<=5;i++){
printf ("tab ["); printf("%d",i);
printf("] = %d  \n" , tab [i]);
}
return 0;}
```

L’exécution du programme précèdent :



Rechercher un élément dans un tableau

La recherche dans un tableau non trié : le principe est simple, il suffit de parcourir la tableau du début et comparer l'élément recherché avec tous les éléments du tableau jusqu'à le trouver.

```
Algorithme cherche_tab_ntrié;
Var t : tableau [100] d'entier;
Var i , val: entier;
Debut
Eccrire (''remplir la tableau'');
Pour(i allant de 1 a 100) faire
Lire (t[i]);
Fait;
i←1; lire (val);
Tantque ((i<100) et (t[i]<> val)) faire
i++;
Fintantque;
Si (t[i] = val) alors
Ecrire (''valeur trouver'');
Sinon
Ecrire (''valeur non trouver'');
Fin.
```

La boucle tant que s'arrête quand i=n ou t[i]=val ou les deux à la fois (cas où la valeur se trouve à la fin du tableau).

La recherche dans un tableau non trié en c :

```
#include <stdio.h>
int main(){
      int tab [100],i,val;
      printf("remplir le tableau tab \n");
      for (i=1;i<=5;i++)
      scanf("%d", &tab[i]);
printf("la valeur a rechercher \n")   ;
scanf("%d", &val);
i=1;
int trouver =0;
while ((i<=5)& (trouver == 0)) {
      if (tab[i]==val){
      trouver = 1;
      }
      else
      i++;
      }
if (trouver==1)
printf ("valeur trouver %d \n", i);
else
printf (" valeur non trouver %d \n", i);
return 0;
}
```

La recherche dans un tableau trié : on suppose que le tableau est trié dans l'ordre croissant, on cherche alors une valeur d'indice i tel que T[i]<T[i-1], en suite, il ne reste qu'a vérifier l'égalité « val=t[i] » pour savoir si la valeur existe ou pas dans le tableau.

```
Algorithme cherche_tab_trié;
Var t : tableau [100] d'entier;
Var i , val: entier;
Debut
i←1; lire (val);
Tantque ((i<=100) et (t[i]< val)) faire
i++;
Fintantque;
Si (t[i] = val) alors
Ecrire (''valeur trouver'');
Sinon
Ecrire (''valeur non trouver'');
Fin.
```

La recherche dans un tableau trié en c :

```
#include <stdio.h>
int main () {
int tab [100], i , val ;
printf("remplir le tableau svp %d \n");
for(i=1;i<=5;i++)
scanf("%d", &tab[i]);
i=1;
printf("introduire la valeur a chercher \n");
scanf(" %d", &val);
while ((i<=5) & (tab[i]< val))
i++;
if (tab[i] == val)
printf ("valeur trouver");
else
printf ("valeur non trouver");
return 0;
}
```

La suppression d’un élément dans un tableau

Pour supprimer un élément d’indice k dans un tableau dont le nombre d’éléments est n, nous devons déplacer tous les éléments d’indice k+1 à n d’une case (position – 1).

Exemple

Considérant le tableau tab comportant les trois éléments suivant :

Tab[1]= 10,

Tab[2]=15,

Tab[3]=25,

Si on supprime l’élément de l’indice 2 dans tab alors on aura en résultat le tableaux avec les deux éléments: tab[1]=10 et tab[2]=25

La suppression d’un élément dans un tableau en algorithmique :

```
Algorithme suprimer_ele_tab;
      var tab : tableau [5] d’entier;
      var i,k : entier;
debut
      ecrire("remplir le tableau ");
      pour(i allant de 1 a 5) faire
      lire(tab[i]);
      fait;
      ecrire ("introduire la position de l'element a suprimer");
      lire(k);
      si (k>5) alors ecrire ("la position est introuvable");finsi;
      si non debut
            pour(i allant de k a de 5)faire
            tab[i]← tab[i+1]; fait;

             finsi;
            finsi;
pour(i allant de 1 a 5) faire
      ecrire (tab[i]); fait;
Fin.
```

Suppression d'un élément dans un tableau

```
#include<stdio.h>
int main()
{
     int tab[5],i,k;
     printf("remplir le tableau \n");
     for(i=1;i<=5;i++)
     scanf("%d", &tab[i]);
     printf("introduire la position de l'element a suprimer \n");
     scanf("%d",&k);
     if (k>5) printf ("la position est introuvable\n");
     else {
          for(i=k;i<=5;i++){
          tab[i]=tab[i+1];
          }
          for(i=1;i<5;i++)
     printf("%d  \n", tab[i]);}
return 0;
}
```