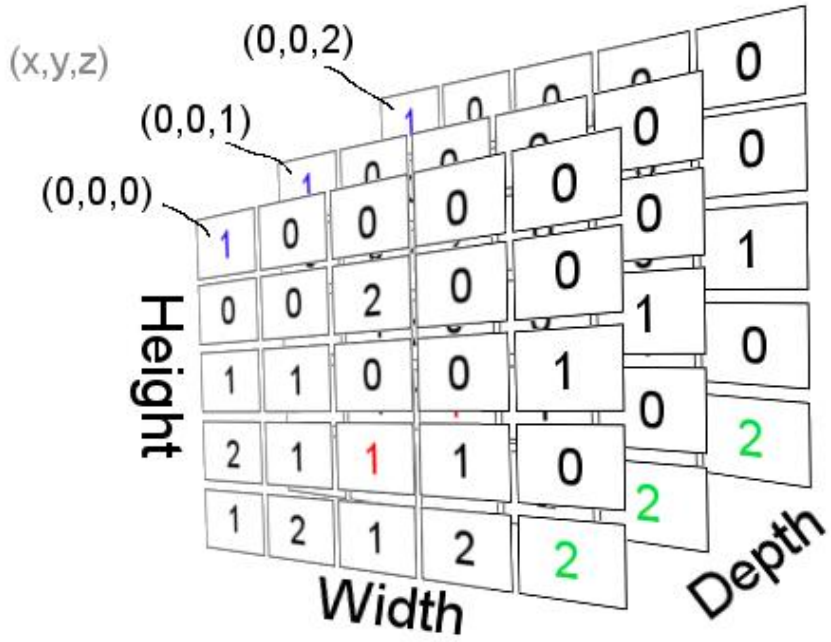
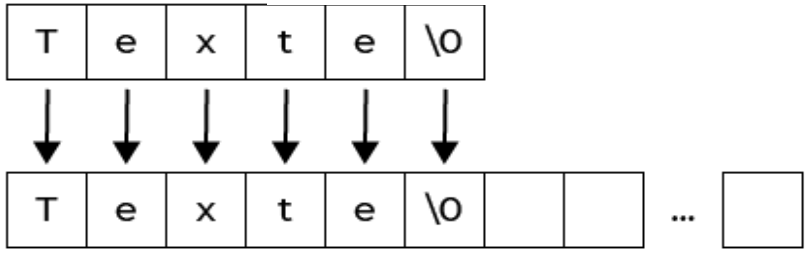
INFORMATIQUE 1

IX. LES TABLEAUX STATIQUES





Eva Ansermin & Romuald Grignon

CY TECH

I. Principe des tableaux

Le problème avec les variables classiques…

- Je veux écrire un algorithme pour gérer les notes du devoir de la promo (les consulter, calculer la moyenne générale, l'écart type, trouver la plus basse etc…).
- Problème : vous êtes 330 !

Le problème avec les variables classiques…

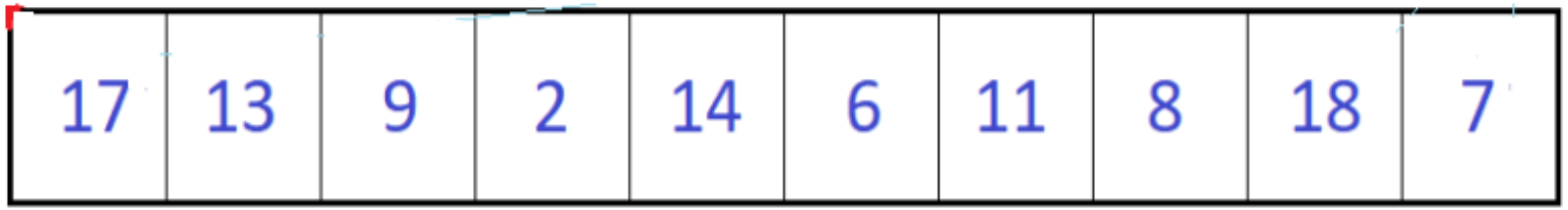
- Je veux écrire un algorithme pour gérer les notes du devoir de la promo (les consulter, calculer la moyenne générale, l'écart type, trouver la plus basse etc…).
- Problème : vous êtes 330 !

```
PROGRAMME note_etudiant
VARIABLE
      moy: réel
     note1 : réel
      note2 : reel
      note3: reel
      …
       note330 : reel
DEBUT:
      moy<- (note1+note2+…+note330)/330
```


Eva ANSERMIN & Romuald GRIGNON

Les tableaux

- Un tableau est un ensemble de variables ordonnées **de même type**.

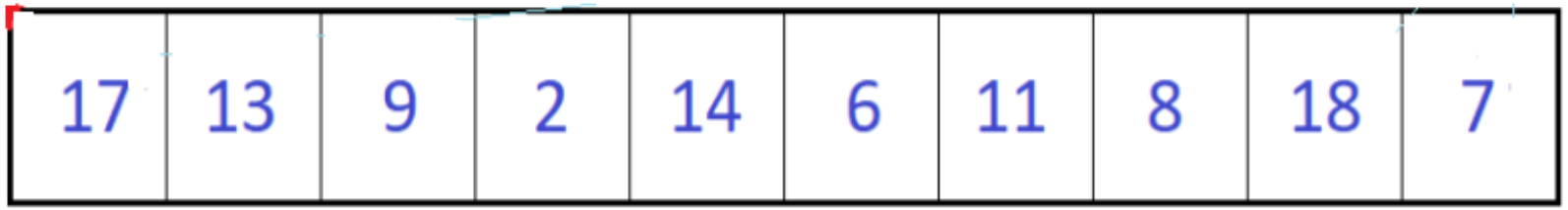


Un Tableau d'entiers

- Manipuler un tableau avec beaucoup de « cases » est plus simple que de manipuler plein de variables séparément.

Les tableaux

- Un tableau est un ensemble de variables ordonnées **de même type**.



Un Tableau d'entiers

- Le nombre de cases d'un tableau est sa **taille.**

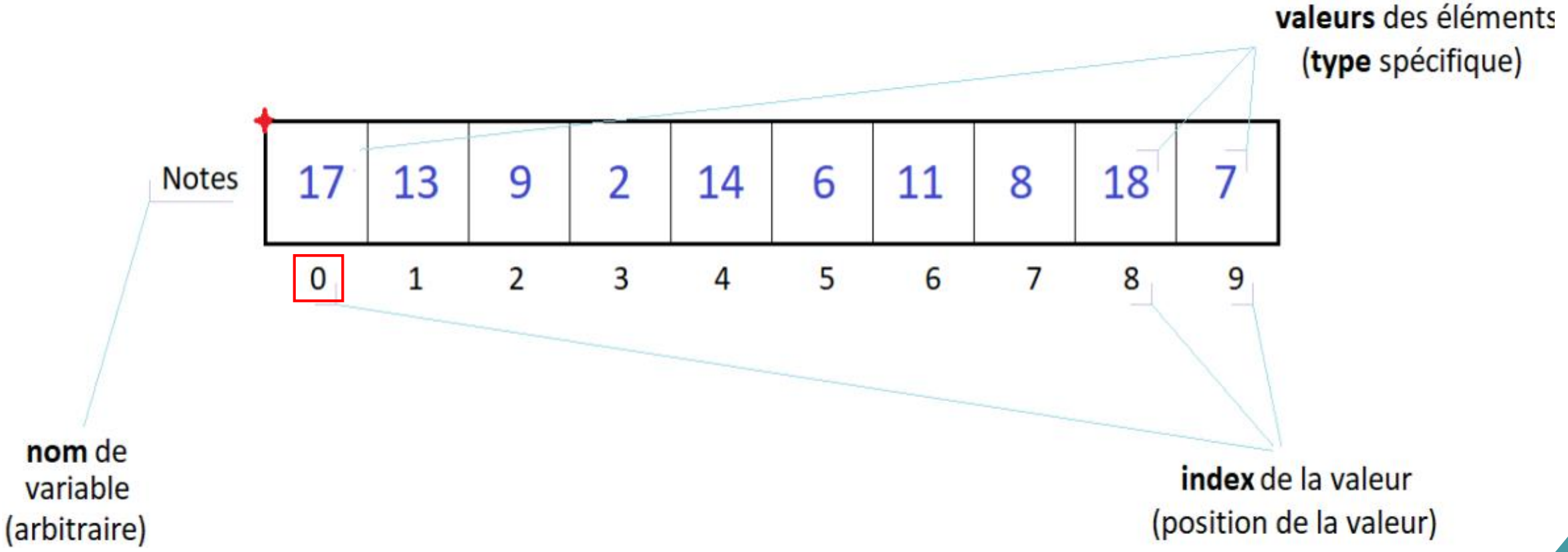
- Dans cet exemple, la taille du tableau est 10.

Les tableaux

- Un tableau est un ensemble de variables ordonnées **de même type**.

Les tableaux

- Un tableau est un ensemble de variables ordonnées **de même type**.



La première case a l'indice 0 !

Les tableaux

- Pour accéder à une case du tableau :

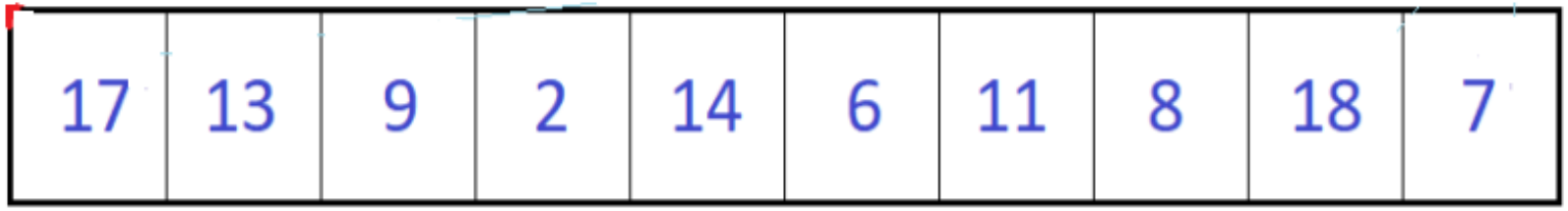
Nom_Tableau[indice de la case]

Les tableaux

- Pour accéder à une case du tableau :

Nom_Tableau[indice de la case]

note



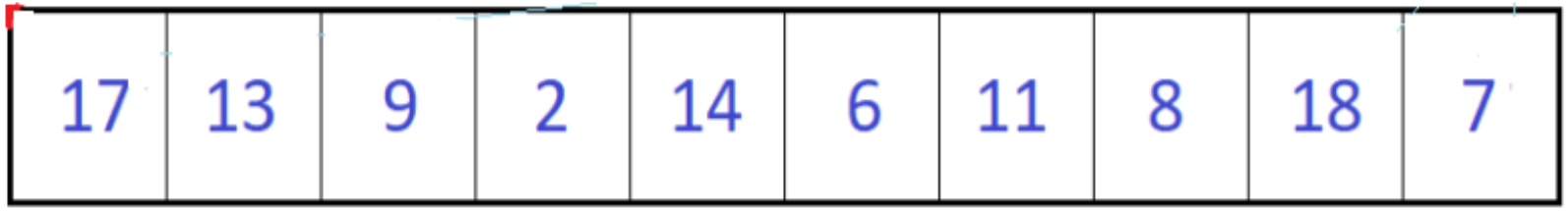
note[1]=
note[4]=
note[10]=

Les tableaux

- Pour accéder à une case du tableau :

Nom_Tableau[indice de la case]

note

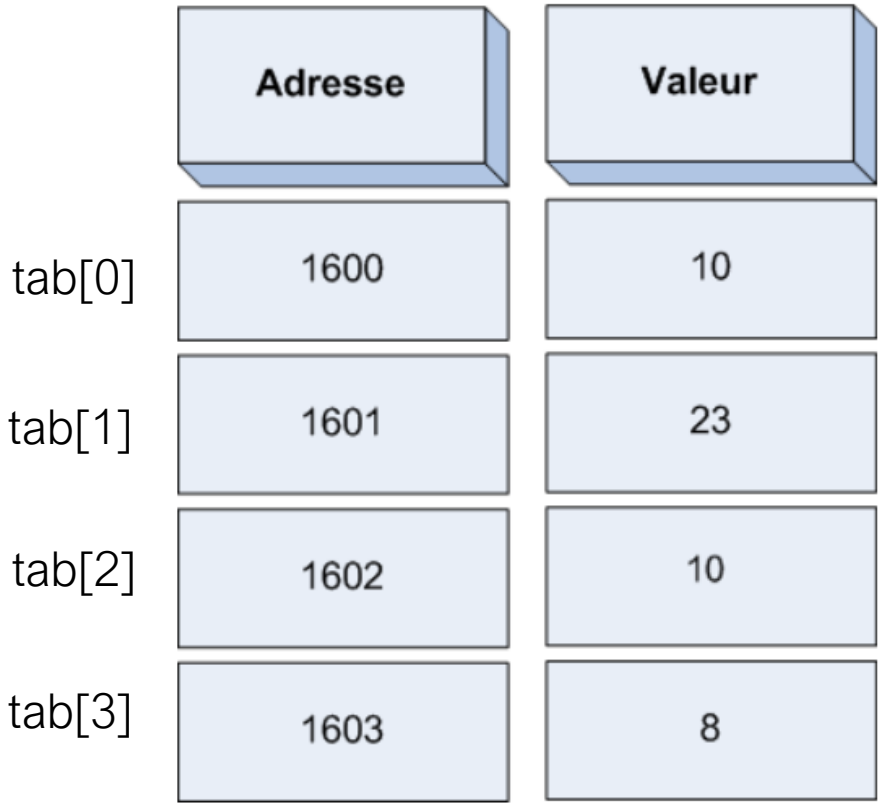


note[1]= 13
note[4]= 14
note[10]= la case n'existe pas!

Le tableau dans la mémoire

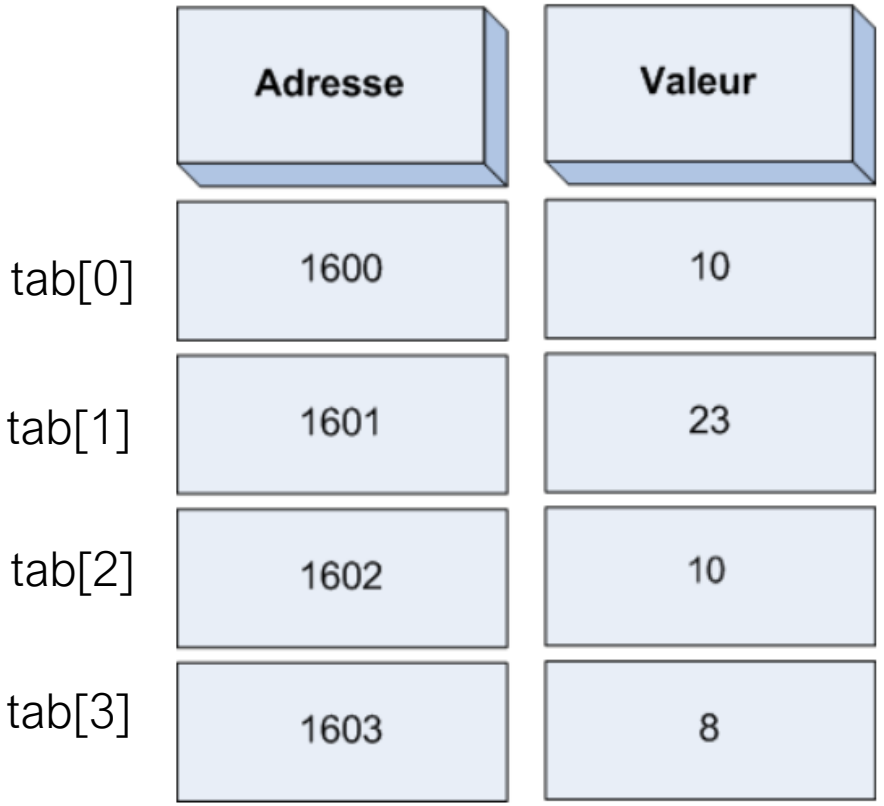
- Les cases d'un tableau sont stockées **à la suite** dans la mémoire.

- Ex : **tab : [10, 23, 10 ,8]** // tableau de taille 4



Le tableau dans la mémoire

- Les cases d'un tableau sont stockées **à la suite** dans la mémoire.



L'adresse de tab[2] est l'adresse de tab[0]+2

L'indice du tableau indique donc où l'ordinateur doit aller chercher la variable dans la mémoire.

Le tableau dans la mémoire

- Les cases d'un tableau sont stockées **à la suite** dans la mémoire.

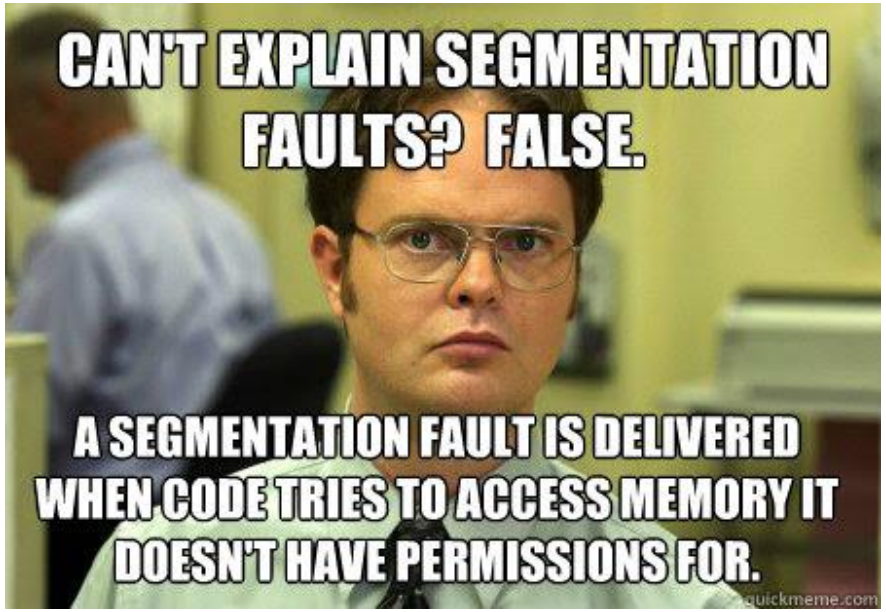
	Adresse	Valeur
tab[0]	1600	10
tab[1]	1601	23
tab[2]	1602	10
tab[3]	1603	8
tab[4]	1604	

tab[4] indique la case mémoire tab[0]+4

tab[4] est « en dehors » de l'espace mémoire réservé pour le tableau!

Attention aux indices des tableaux !

- Si on déclare un tableau de taille N, on ne peut que utiliser des indices entre 0 et N-1.
- Si on utilise un indice incorrect (tableau[-1], tableau[N], tableau[N+10]), on essaye d'accéder à une partie de la mémoire sans en avoir l'autorisation.
- C'est la Segmentation fault.



II. Tableaux en pseudo-code

Déclaration et manipulation d'un tableau : pseudo-code

- Déclaration d'un tableau statique :

  `nomTableau[`***`taille`***`] : TABLEAU DE` *`type`*

  - Le tableau possède *taille* cases contenant une variable de *type*
  - La première case est **nomTableau[0]** la dernière **nomTableau[taille-1]**
- Manipulation :

  Accéder à la (i+1)ème case : `nomTableau[i], 0 ≤ i < taille.`
- Initialisation :

  Par défaut les valeurs ne sont **pas initialisées**

  On peut initialiser un tableau de la manière suivante :

  ```
  nomTableau <- {val1, val2, val3…}
  ```

  Dans ce cas, les cases non indiquées sont mises à zéro.

Déclaration et manipulation d'un tableau : pseudo-code

- Exemple

```
PROGRAMME note_etudiant
VARIABLE
      moy: réel
      note[330]: tableau de réels
DEBUT:
      note <- {0}  // Met toutes les cases à 0 !
      note[0] <- 15
      note[10] <- 13
      ECRIRE("Le premier étudiant a la note de "+note[0])
      …
FIN
```

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- On souhaite afficher son contenu.

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- On souhaite afficher son contenu.

```
ECRIRE(tab)
```

- Il faut afficher toutes les cases du tableau une par une !

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
DEBUT
    ECRIRE(tab[0])
```

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
DEBUT
        ECRIRE(tab[0])
        ECRIRE(tab[1])
```

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
DEBUT
        ECRIRE(tab[0])
        ECRIRE(tab[1])
        ECRIRE(tab[2])
```

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
DEBUT
        ECRIRE(tab[0])
        ECRIRE(tab[1])
        ECRIRE(tab[2])
        …
```

Parcours d’un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d’entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
DEBUT
      ECRIRE(tab[0])
      ECRIRE(tab[1])
      ECRIRE(tab[2])
      …
      ECRIRE(tab[9])
FIN
```

Parcours d'un tableau

- Soit le tableau suivant :

```
tab[10]: tableau d'entiers
```

- Il faut afficher toute les cases du tableau une par une !

```
POUR i de 0  à 10 FAIRE
     ECRIRE(tab[i])
FIN POUR
```

Parcours d'un tableau

- Pour réaliser une opération sur un tableau, il faut parcourir toutes ses cases.

```
POUR i de 0 à taille FAIRE
    instruction
FIN POUR
```

Parcours d'un tableau

- Exemple : remplir un tableau de note aléatoire:

```
PROGRAMME note_etudiant
VARIABLE
      moy: réel
      note[330]: tableau de reels
      i : entier
DEBUT:
      POUR i de 0 à 330   FAIRE
            note[i] <- ALEA( 20 )
      FIN POUR
FIN
```


Eva ANSERMIN & Romuald GRIGNON

Passage de tableaux à une fonction

- On peut directement passer un tableau en argument d'une fonction.

```
PROCEDURE afficher_tab(tab[330]: tableau de réels)
VARIABLE
      i : entier
DEBUT
      POUR i de 0 à 330 FAIRE
            ECRIRE(tab[i])
      FIN POUR
FIN
```

Passage de tableaux à une fonction

- Pour l'appel à la fonction, il suffit de donner le nom du tableau

```
PROCEDURE afficher_tab(tab[330]: tableau de réels)
VARIABLE
      i : entier
DEBUT
      POUR i de 0 à 330 FAIRE
            ECRIRE(tab[i])
      FIN POUR
FIN

VARIABLE
      note[330]: tableau de reels
DEBUT
      afficher_tab(note)
```

Passage de tableaux à une fonction

- Plus général !

```
PROCEDURE afficher_tab(tab: tableau de reels, taille : entier)
VARIABLE
      i : entier
DEBUT
      POUR i de 0 à taille FAIRE
            ECRIRE(tab[i])
      FIN POUR
FIN

VARIABLE
      note[330]: tableau de reels
DEBUT
      afficher_tab(note, 330)
```

Passage de tableaux à une fonction

- Si un tableau passé en argument est modifié dans une fonction, il sera modifié dans le reste du programme sans avoir besoin d'utiliser de return!

```
PROCEDURE increm_tab(tab: tableau de reels, taille : entier)
VARIABLE
      i : entier
DEBUT
      POUR i de 0 à taille FAIRE
            tab[i] <- tab[i]+1
      FIN POUR
FIN

VARIABLE
      note[330]: tableau de reels
DEBUT
       increm_tab(note, 330)      // note sera modifiée !!
```

III. Tableaux en C

Déclaration et manipulation d'un tableau : langage C

- Déclaration d'un tableau statique :

```
type nomTableau[taille]
```

- Manipulation :

  Accéder à la (i+1)ème case : `nomTableau[i], 0 ≤ i < taille.`

- Initialisation :

  On peut initialiser toutes les cases du tableau lors de la déclaration:

```
type nomTableau[taille]={val1, val2, val3, …}
```

Les cases non indiquées sont mises à zéros

Déclaration et manipulation d'un tableau : langage C

- Exemple

```
int main(){
      float note[330]={0}; // met toutes les cases à 0
      note[0]=15;
      note[10]=13;
      printf("La note du premier étudiant est %f",note[0]);
      return 0;
}
```

Parcours d'un tableau : langage C

- Exemple

```
int main(){
      float note[330]={0}; // met toutes les cases à 0
      note[0]=15;
      note[10]=13;
      printf("Les notes des étudiants sont:");
      for (i=0; i<330 ; i++){// boucle pour parcourir le tableau
            printf("%f",note[i]);
      }
      return 0;
}
```

Passage de tableaux à une fonction : langage C

```
void afficher_tab(float tab[], int taille){    // /!\ au []
      int i;
      for (i=0;i<taille;i++){
            printf("%f",tab[i]);
      }
}

int main(){
      float note[330];
      …
      afficher_tab(note, 330);
      return 0;
}
```

IV. Les constantes

Gestion de la taille du tableau

- La taille du tableau est fixe. Il faut donc toujours utiliser la même valeur dans le code.
- 3 possibilités :
    - Ecrire la valeur qui correspond à sa taille tout le temps ! (comme on a fait jusqu'ici)

Gestion de la taille du tableau

- La taille du tableau est fixe. Il faut donc toujours utiliser la même valeur dans le code.
- 3 possibilités :
  - Ecrire la valeur qui correspond à sa taille tout le temps ! (comme on a fait jusqu'ici)
  - Points négatifs :
    - Il faut s'en souvenir
    - Si on souhaite changer entre deux exécutions, il faudra changer la valeur partout !

Gestion de la taille du tableau

- La taille du tableau est fixe. Il faut donc toujours utiliser la même valeur dans le code.
- 3 possibilités :
  - Ecrire la valeur qui correspond à sa taille tous le temps ! (comme on a fait jusqu'ici)
  - Ecrire la taille du tableau dans une variable.

Gestion de la taille du tableau

- La taille du tableau est fixe. Il faut donc toujours utiliser la même valeur dans le code.
- 3 possibilités :
  - Ecrire la valeur qui correspond à sa taille tous le temps ! (comme on a fait jusqu'ici)
  - Ecrire la taille du tableau dans une variable.
    - Points négatifs :
      - C'est dangereux : il ne faut pas modifier la valeur de la variable en cours de code !
      - Certains compilateurs ne l'acceptent pas.

Gestion de la taille du tableau

- La taille du tableau est fixe. Il faut donc toujours utiliser la même valeur dans le code.
- 3 possibilités :
  - Ecrire la valeur qui correspond à sa taille tous le temps ! (comme on a fait jusqu'ici)
  - Ecrire la taille du tableau dans une variable.
  - Utiliser une **constante.**

Les constantes

- Les **constantes** sont des données dont la valeur est **fixée** (au contraire des variables).
- Les constantes sont dites **globales** : elle sont accessibles par l'ensemble du code (au contraire des variables **locales** qui ne sont valables que dans les fonctions dans lesquelles elles sont déclarées).
- Par convention, le nom des constantes en est MAJUSCULE
- Pratique pour les tailles de tableaux qui ne doivent pas être modifiées !

Les constantes

- En pseudo-code

```
PROGRAMME test_tableau
Constante TAILLE1 <- 10
Constante TAILLE2 <- 200


VARIABLE
      tab1[TAILLE1]: tableau de reels
      tab2[TAILLE2] :tableau d'entiers
      …
DEBUT
 POUR i de 0 à TAILLE1 FAIRE
      …
```

Partout dans le code TAILLE1 sera remplacé par 10 et TAILLE2 par 200

Les constantes

- En langage C

```
#include<stdio.h>
#define TAILLE1 10
#define TAILLE2 200

int main(){
      float tab1[TAILLE1];
      int tab2[TAILLE2];
      …
      for (i=0; i<TAILLE1;i++)
      …
```

Partout dans le code TAILLE1 sera remplacé par 10 et TAILLE2 par 200

Définition de constantes en C

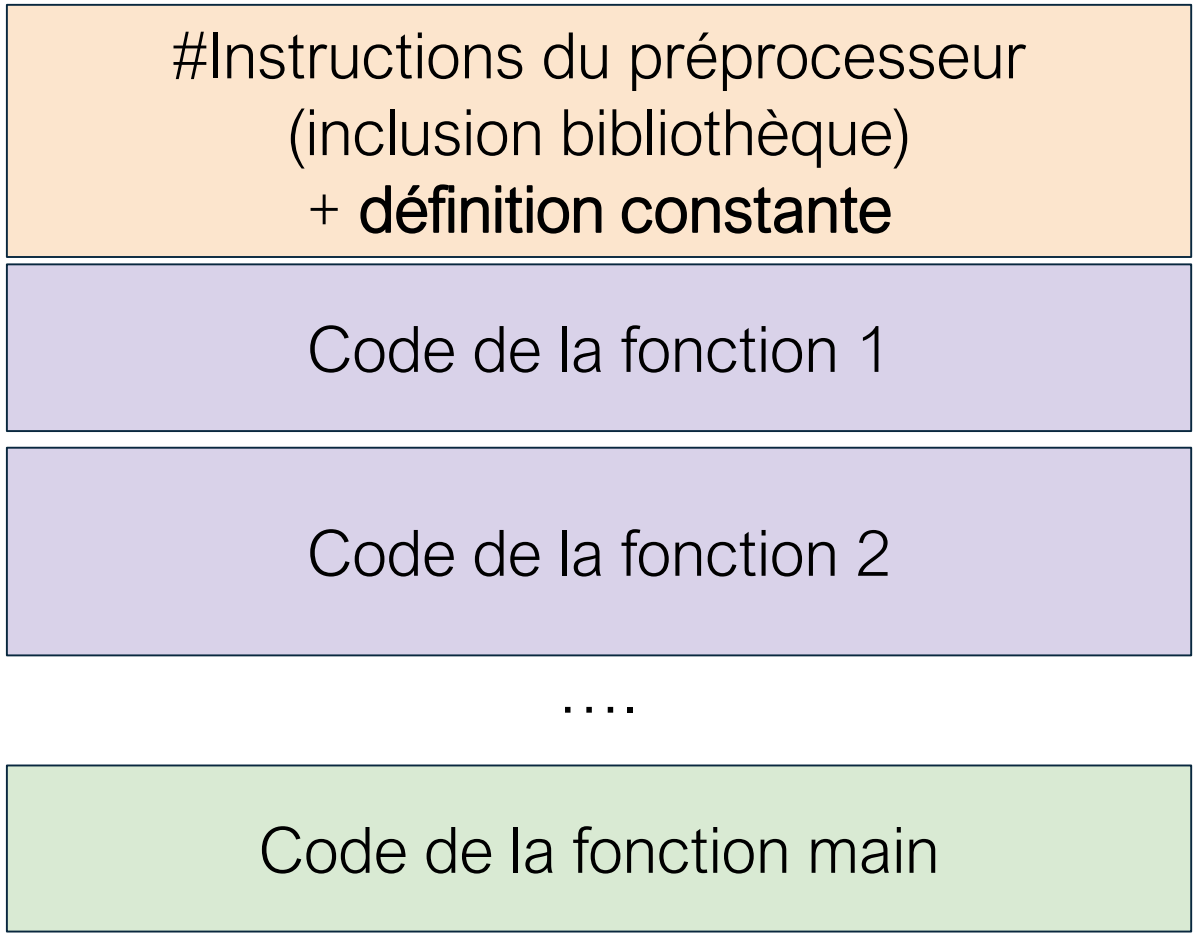
- On peut utiliser #define pour remplacer une expression par une autre.

Syntaxe : `#define symbole expression` :

```
#define N 10
#define VRAI 1
#defince FAUX 0
#define PI 3.141592655359
```

- **Les commandes commençant par # sont toujours en premier dans le code.** Ces commandes sont des directives du compilateur. On les appelle **directives du préprocesseur.**

Structure générale d'un fichier .c



V. Les chaînes de caractères

Tableau en C : cas des chaînes de caractères.

- Le type chaîne de caractères n'existe pas directement en C. Ceci est dû au fait qu'on ne connait pas le nombre de caractères nécessaire à l'avance.

- Une chaîne de caractères est une suite ordonnée de caractères.

Tableau en C : cas des chaînes de caractères.

- Le type chaîne de caractères n'existe pas directement en C. Ceci est dû au fait qu'on ne connait pas le nombre de caractères nécessaire à l'avance.
- Une chaîne de caractères est une suite ordonnée de caractères.
- Pour gérer un ensemble de caractères, on va utiliser **un tableau de caractères.**

B	o	n	j	o	u	r

Tableau en C : cas des chaînes de caractères.

- Si on veut stocker/ afficher un mot on peut donc faire ceci:

```
int i=0;
char chaine[6]; // Tableau de 6 char pour stocker S-a-l-u-t

chaine[0] = 'S’; //toujours mettre les ‘ ‘ quand on manipule des caractères
chaine[1] = 'a';
chaine[2] = 'l';
chaine[3] = 'u';
chaine[4] = ‘t’;
// On peut écrire aussi directement chaine="Salut" !

// afficher le mot:
for(i=0;i<5;i++){
	printf(“%c“, chaine[i]);
}
```

Tableau en C : cas des chaînes de caractères.

- Problème : on ne connait pas toujours le nombre de caractères nécessaire!
- Pour cela, un utilise un caractère de fin de chaîne : **'\0'** ;
- Exemple :

char maChaine[8] = 'Salut';

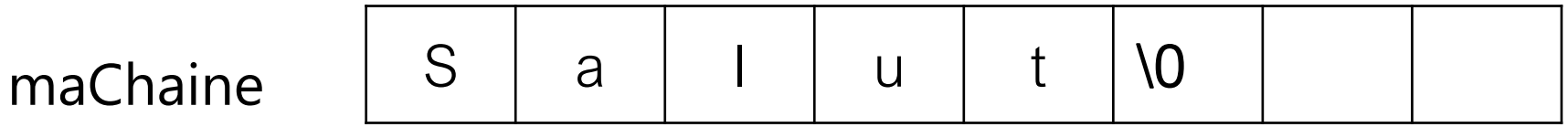


Tableau en C : cas des chaînes de caractères.

- Pour afficher un mot/ phrase sans en connaitre la taille :

```
int i=0;

// afficher le mot:
while(chaine[i] != '\0 '){
        printf("%c", chaine[i]);
        i++
}
```

Tableau en C : cas des chaînes de caractères.

- Heureusement le C peut faire cela automatiquement grâce au format **%s.**

```
char nom[100];     // pourquoi 100 ?
printf("Comment tu t'appelles? ");

scanf("%s",nom); // rempli automatiquement le tableau nom et met \0 à la fin,
pas de & !!!

printf("Bonjour %s", nom); //affiche les cases de nom jusqu'à '\0'

}
```

- La bibliothèque **string.h** comporte des fonctions utiles à la gestion des chaînes de caractères.

VI. Les tableaux à deux dimensions

Tableau à deux dimensions : principe

- Les tableaux peuvent avoir deux ou plusieurs dimensions !
- Les tableau à deux dimensions peuvent être vus comme des matrices.

1	10	15	..	…	…	20
3	…	…	…	…	…	1
6	…	…	…	…	…	7
7	…	…	…	…	…	50
10	…	…	…	…	…	16
18	…	…	…	…	…	7
13	…	…	…	…	…	12

- On manipule alors deux indices : l'un représentant les lignes, l'autre les colonnes.

Tableau à deux dimensions : principe

- Les tableaux peuvent avoir deux ou plusieurs dimensions !
- Les tableau à deux dimensions peuvent être vus comme des matrices.

Indice colonne / Indice ligne	0	1	2	…	…	Nombre colonne -2	Nombre colonne -1
0	1	10	15	..	…	…	20
1	3	…	…	…	…	…	1
2	6	…	…	…	…	…	7
…	7	…	…	…	…	…	50
…	10	…	…	…	…	…	16
Nombre ligne -2	18	…	…	…	…	…	7
Nombre ligne -1	13	…	…	…	…	…	12

- On manipule alors deux indices : l'un représentant les lignes, l'autre les colonnes.

Tableau à deux dimensions : déclaration

- En pseudo –code :
  - Déclaration du tableau :

    `nomTableau [nbLignes, nbColonnes] : TABLEAU DE type`
  - Accéder à la i^{ème} ligne, j^{ème} colonne :

    `nomTableau[i,j], 0 ≤ i < nbLignes, 0 ≤ j < nbColonnes`

- En C :
  - Déclaration :

    ```
    type monTab2D[nbLignes][nbColonnes];
    ```

    exemple : `int monTab2D[3][4] // Tableau de 3 lignes et 4 colonnes ;`
  - Accéder à la i^{ème} ligne, j^{ème} colonne :

    `nomTableau[i][j], 0 ≤ i < nbLignes, 0 ≤ j < nbColonnes`

Tableau à deux dimensions

- Un tableau 2D peut également être vu comme un *tableau de tableaux :* chaque ligne est un sous-tableau.

- C'est ce principe qui est utilisé pour l'initiation d'un tableau:

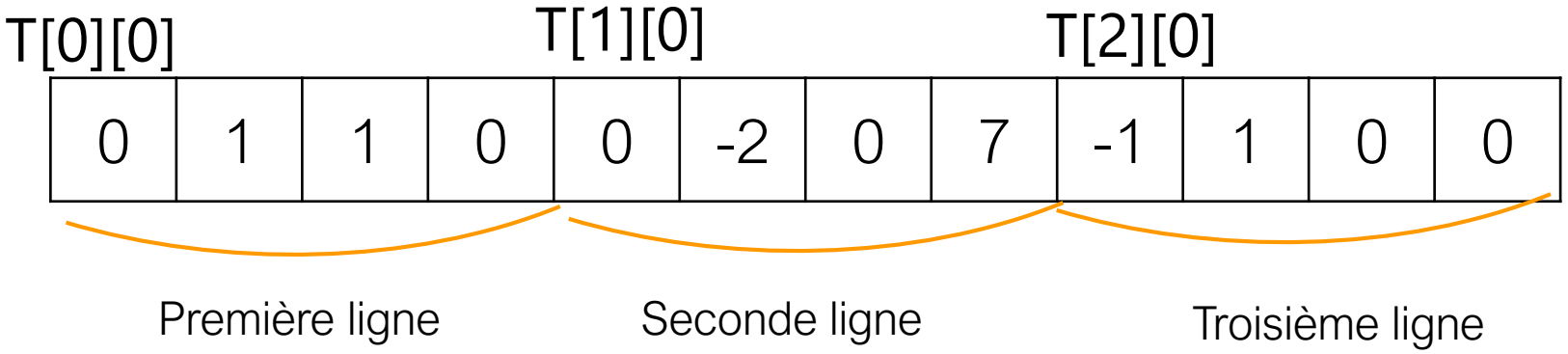
T <- { {0, 1, 1, 0}, {0, -2, 0, 7}, {-1, 1, 0, 0} }

0	1	1	0
0	-2	0	7
-1	1	0	0

Mémoire

- Dans la mémoire le stockage des valeurs est consécutif !

0	1	1	0
0	-2	0	7
-1	1	0	0



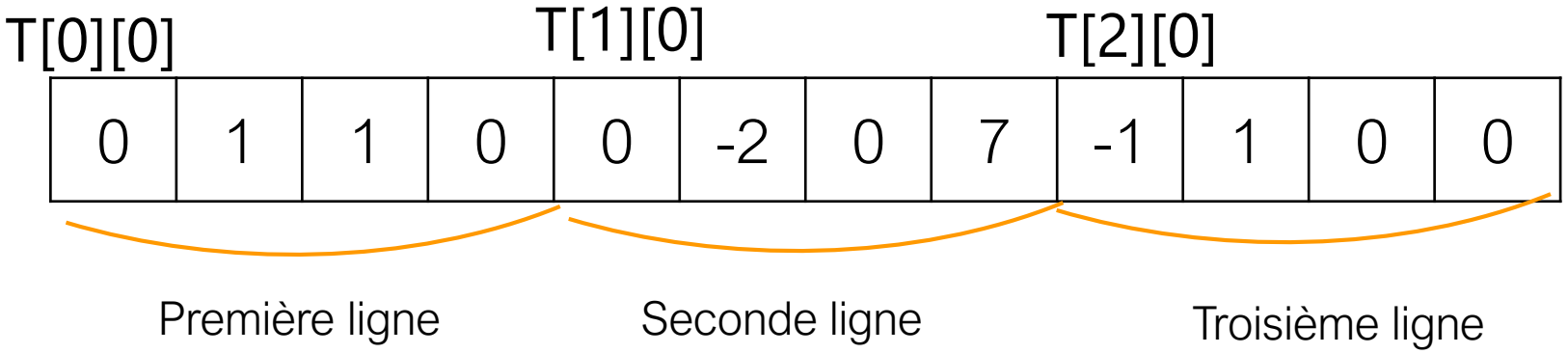
Mémoire

- Un tableau 2D en C est équivalent à un tableau de taille :

  nombre_ligne * nombre_colonne.

- Lors de la manipulation du tableau le calcul suivant est donc effetcué :

tab2D[i][j] ⇔ case du tableau n° **i * nombre_colonne + j**



En résumé

- Les tableaux permettent de stocker un ensemble de variables. Ils peuvent avoir plusieurs dimensions et sont de **taille fixe.**
- Les valeurs du tableau sont stockées les unes à la suite des autres dans la mémoire.
- Attention aux indices !