

Algo & Programmation II

Plan :

- acquis {
 - Structure d'un algorithme
 - variables
 - testes (if)
 - boucles (for, while, do while)
 - printf et scanf
- requis {
 - Tableaux
 - fonctions & procédures
 - passage par valeur et référence
 - ⇕
 - Pointeurs
 - récursivité
- facultatif {
 - Fichiers

I) Tableaux / Array

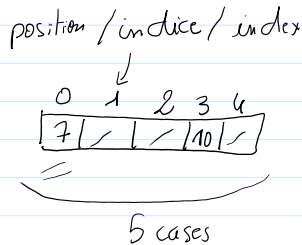
- ⇒ Variables
- ⇒ contient plusieurs cases

Tableau → T [5 | 6 | 70 | 100 | 1]

Case ⇔ Variable

Syntax : type Nom [taille] ;

exemple : int T[5] ; ⇒



⇒ T[0] = 7

T[3] = 10

T[6] = 7 ; ⇐ erreur

exercice 1 :

- écrire un programme qui va demander à l'utilisateur de saisir 5 notes et qui va lui afficher la moyenne.

1) solution 1 : sans Tableau

```
#include <stdio.h>
int main() {
    float note1;
    float note2;
    float note3;
    float note4;
    float note5;
    float moyenne;
    printf("donnez la premiere note"); } x 5
    scanf("%f", &note1);

    moyenne = (note1 + note2 + note3 + note4 + note5) / 5;
    printf("La moyenne est %f", moyenne);
    return 0;
}
```

2) solution en utilisant les tableaux :

```
#include <stdio.h>
int main() {
    float moyenne;
    float notes[5];
    int i;
    for(i=0; i<5; i=i+1)
    {
        printf("donnez la note numero %d", i+1);
        scanf("%f", &notes[i]);
    }

    float somme = 0;
    for(i=0; i<5; i=i+1)
        somme = somme + notes[i];
    moyenne = somme / 5;
    printf("La moyenne est %f", moyenne);
    return 0;
}
```

⇒ Travail à faire : au lieu de calculer la moyenne de 5 notes, demandez à l'utilisateur le nombre de note à gérer.

3) solution avancée :

```
#include <stdio.h>
int main() {
    float note;
    float moyenne;
    float somme;
    somme = 0;

    int i;
    for(i=0; i<5; i=i+1)
        ..
```

```

} printf("do ---- numers %d", i+1);
scanf("%f", &note);
somme = somme + note;
} // on a pas accès aux différentes notes saisies
moyenne = somme / 5;
printf("---- %f", moyenne);
return 0;
}

```

Exercice 2 : - créa un Tableau de 100 cases (entier)

- remplissez ce Tableau de la manière suivante

0	1	2	...	97	98	99
0	10	20	...	970	980	990

- et puis affichez le de la manière suivante :

case 0 = 0
case 1 = 10
⋮
case 99 = 990

boucle

```

1  /* Online C Compiler and Editor */
2  #include <stdio.h>
3
4  int main()
5  {
6      int T[100]; // question 1
7      int i;
8      // question 2
9      for(i=0; i<100; i++)
10         T[i] = i * 10 ;
11      // question 2
12      for(i=0; i<100; i++){
13         printf("case %d = %d\n", i , T[i]);
14     }
15     return 0;
16 }

```

Rappel : Abbreviations \Rightarrow

- $i++ \Rightarrow i = i + 1$
- $++i \Rightarrow i = i + 1$
- $a++ \Rightarrow a = a + 1$
- $i-- \Rightarrow i = i - 1$
- $a+=5 \Rightarrow a = a + 5$
- $a/=b \Rightarrow a = a / b$

Terminologie \Rightarrow

- \Rightarrow tiret
- \Rightarrow underscore
- | \Rightarrow pipe

=> diese / sharp

Fonctions & procédures

=> sous programme qui ont comme objectif : - Simplifier le code
- partage / réutilisation

⇓ on gagne

Temps
lisibilité
⋮

Syntax (Procedure) : void name(parameters) {
// corps de la procédure
// contenu
}

* paramètres : type nom

ex1: void teste1() {
printf("salut\n"); } // Définition (en
// de hors du main)

teste1(); // appel / exécution

teste1();

ex2: void teste2(int x) {
printf("%d", x); } // définition

teste2(7); // appel => x = 7
printf("%d", x);
⇓
7

ex3: void teste3(int x, int y) {
int somme;
somme = x + y;

`printf("%d", somme);`

`teste3(9, 10);` $\Rightarrow x = 9$ et $y = 10$
 $\Rightarrow 19$

```
#include <stdio.h>

void teste3(int x , int y){
    int somme;
    somme = x + y;
    printf("%d\n", somme);
}

int main()
{
    int a = 33;
    teste3(a, 1);
    teste3(a, a);
    teste3(3, 5);
    return 0;
}
```

Portée des variables

Local

Global

```
1 /* Online C Compiler and Editor */
2 #include <stdio.h>
3 int a = 10; // global => elle est reconnue dans toutes les lignes qui
  suivent
4 void teste(){
5     printf("%d dans teste\n", a);
6 }
7 int main()
8 {
9     teste();
10    if(1==1){
11        int x; // une variable locale => reconnue au sein du bloc {}
12        printf("au sein du if %d\n", a);
13    }
14    printf("%d", a);
15
16    return 0;
17 }
```