Algo & Programmation II

Plan :

acquis :
- Structure d'un algorithme
- variables
- testes (if)
- boucles (for, while, do while)
- printf et scanf

requis :
- Tableaux
- fonctions & procedures
- passage par valeur et reference ⇕ Pointeurs
- recursivité

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]; ⇒

position / indice / index

0	1	2	3	4
7	/	/	10	/

5 cases

⇒ T[0] = 7

T[3] = 10

T[6] = 7 ; ⇐ erreur

---

exercice 1 :

- ecrire 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("donne la premiere note");
    scanf("%f",&note1);
```

} x 5

```
    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("donne 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 gerer.

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 ---- numero %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 :

boucle

case 0 = 0
case 1 = 10
:
case 99 = 990

```c
/* Online C Compiler and Editor */
#include <stdio.h>

int main()
{
    int T[100]; // question 1
    int i;
    // question 2
    for(i=0;i<100;i++)
        T[i] = i * 10 ;
    // question 2
    for(i=0;i<100;i++){
        printf("case %d = %d\n", i , T[i]);
    }
    return 0;
}
```

Rappel : Abreviations ⇒ $i++ \Rightarrow i = i+1$
$++i \Rightarrow i = i+1$
$a++ \Rightarrow a = a+1$
$i-- \Rightarrow i = i-1$
$a += 5 \Rightarrow a = a+5$
$a /= b \Rightarrow a = a/b$

Terminologie ⇒ - ⇒ tiret
_ ⇒ underscore
| ⇒ pipe

=> diese / sharp

Fonctions & procedures

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

⇓ on gagne

Temps

lisibilité

⋮

Syntax (Procedure) : `void name(«parametres»){`

`// corps de la procedure`
`// contenu`
`}`

* parametres : type nom

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

`teste1();` // appel/execution
`teste1();`

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

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

ex 3 : `void teste3(int x, int y){`

`int somme;`
`somme = x + y;`

printf("%d", somme);
}

teste3(9,10); => x = 9 et y = 10
=> 19

```c
#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;
}
```

Porté des variables

Local

Global

```c
/* Online C Compiler and Editor */
#include <stdio.h>
int a = 10; // global => elle est reconnu dans toutes les lignes qui suivent
void teste(){
    printf("%d dans teste\n",a);
}
int main()
{
    teste();
    if(1==1){
        int x; // une variable local => reconnu au sein du bloc {}
        printf("au sein du if %d\n",a);
    }
    printf("%d",a);

    return 0;
}
```