



SMIA – S2

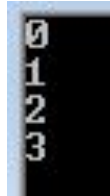
# Corrigé série 3 S2

Informatique 2

2020-2021

# Exercice 1

Qu'affiche le programme suivant :

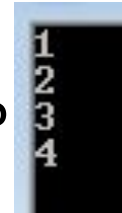


```
0
1
2
3
```

```
1  #include<stdio.h>
2  int main(void){
3      int i=0;
4      int n=0;
5      while(n<10){
6          printf("%d\n",i);
7          i++;
8          n=i*i;
9      }
10 }
```

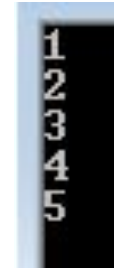
2. Même question si on permute :

a) les instructions de la ligne 6 et de la ligne 7 ?



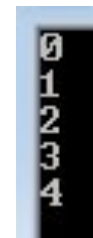
```
1
2
3
4
```

b) les instructions de la ligne 6 et de la ligne 8 ?



```
1
2
3
4
5
```

c) les instructions de la ligne 7 et de la ligne 8 ?



```
0
1
2
3
4
```

# Exercice 2

```
1  #include<stdio.h>
2  int main(void){
3      int n,i,choix,var;
4      float har;
5      do{ printf("entrer: 1 =>somme ; 2 => factoriel ; 3 => somme harmonique:\n");
6          scanf("%d",&choix);
7      }while(choix!=1 && choix !=2 && choix!=3);//controle de saisi
8      do{ printf("entrer un entier > 1 \n");
9          scanf("%d",&n);
10     }while(n<=1); //controle de saisi
11     switch(choix){
12     case 1: // si choix==1 calculer somme 1 à n
13         for(i=1,var=0; i<=n; i++)
14             var+=i;
15         printf("la somme de 1 a %d = %d \n",n,var); break;
16     }
17     case 2:// si choix==2 calculer produit 1 à n: factoriel
18         for(i=1,var=1; i<=n; i++)
19             var*=i;
20         printf("le Factoriel de %d = %d \n",n,var);break;
21     }
22     case 3:// si choix==3 calculer somme 1/1 à 1/n: somme harmonique
23         for(i=1,har=0; i<=n; i++)
24             har+=(float) 1/i;
25         printf("la somme harmonique de 1 a %d = %.2f \n",n,har);break;
26     }
27 }
28 }
```

# Exercice 2

## Remarques:

- N'oublier pas les accolades pour chaque case, car il contient plusieurs instructions
- N'oubliez pas break; vers la fin de chaque case. sinon vous allez exécuter tous les case qui suivent
- N'oubliez pas le contrôle de saisi pour chaque variable que l'utilisateur va saisir
- N'oublier pas que la somme harmonique sera une variable réel ( j'ai choisi float mais vous pouvez mettre double). Il faut alors forcer  $1/i$  pour qu'elle devienne float, en ajoutant l'opérateur du cast (float) avant  $1/i$

# Exercice 3

```
1  #include<stdio.h>
2  int main(void){
3      int p,i,premier;
4      do{ printf("entrer un entier positif > 1  \n");
5          scanf("%d",&p);
6      }while(p<=1);
7      for(i=2,premier=0;i<p && premier==0;i++)
8          if(p%i==0)
9              premier=1;
10     if(premier==0)
11         printf("%d est premier",p);
12     else
13         printf(" %d n est pas premier",p);
14
15 }
```

# Exercice 3

**Remarques:** Il y a plusieurs manière de déterminer si un nombre est premier. Ici:

- je considère que le nombre  $p$  saisi par l'utilisateur est un nombre qui n'est pas premier (je pose la variable  $\text{premier}=0$ )
- Je teste ensuite tous les diviseurs  $i$  compris entre 2 et  $p-1$ . Si je trouve que  $p$  est divisible par un nombre  $i$  ( $p\%i==0$ ) alors je dis que  $p$  est premier ( $\text{premier} = 1$ )
- On exécute une autre itération de la boucle `for` lorsque deux conditions sont vérifiées:
  - Le nombre  $i$  n'a pas encore atteint  $p-1$ :  $i < p$
  - ET  $\text{premier} == 0$ , c'est-à-dire qu'on n'a pas encore trouvé un diviseur de  $p$
- Pour l'affichage, on vérifie la variable  $\text{premier}$ , si  $==0$  alors  $p$  est premier; sinon le  $p$  n'est pas premier
- On n'est pas obligé de mettre des accolades pour la boucle `for` car elle contient une seule instruction `if` (et on n'a pas besoin de mettre des accolades pour `if`, qui elle-même contient une seule instruction). Si vous les mettez, cela reste totalement correcte

# Exercice 4

```
1  #include<stdio.h>
2  int main(void){
3      int n, min, var, i;
4      do{ printf("entrer un entier positif >=2: \t");
5          scanf("%d",&n);
6          printf("\n");
7      }while(n<2);
8      printf("entrer un entier: \t");
9      scanf("%d",&var);
10     min=var;
11     printf("\n");
12     for(i=2;i<=n;i++){
13         printf("entrer un entier: \t");
14         scanf("%d",&var);
15         printf("\n");
16         if(var<min)
17             min=var;
18     }
19     printf("le minimum est %d", min);
20 }
```

**Remarques: Dans cet exercice, on ne doit pas utiliser de tableau (pas encore traité en cours)**

- L'utilisateur saisi le nombre d'éléments n qu'il veut entrer ( $\geq 2$  pour comparer entre au moins 2 éléments)
- Le premier élément saisi sera considéré comme le min
- On introduit une boucle for avec i allant de 2 à n pour comparer tous les éléments saisi
- on ne peut pas utiliser une boucle for qui commence de 1 car il faudra initialiser la variable min avec un élément saisi par l'utilisateur (on ne peut pas initialiser min par 0 ou -1 car on n'a pas idée des valeurs que va saisir l'utilisateur)



# Exercice 5

```
1  #include<stdio.h>
2  #define MinN 0
3  #define MaxN 20
4  int main(void){
5      int n , i;
6      float moy,min, max, note;
7      do{ printf("entrer le nombre de notes >=2: \t");
8          scanf("%d",&n);
9          printf("\n");
10     }while(n<2);
11     min=MaxN;
12     max=MinN;
13     moy=0.0;
14     for(i=1;i<=n;i++){
15         do{ printf("entrer une note entre %d et %d: \t",MinN,MaxN);
16             scanf("%f",&note);
17             printf("\n");
18         }while(note<MinN || note>MaxN);
19         if(note<min)
20             min=note;
21         if(note>max)
22             max=note;
23         moy+=note;
24     }
25     moy/=n;
26     printf("min= %.2f, max=%.2f et moyenne=%.2f", min,max,moy);
27 }
```



# Exercice 5

## Remarques:

- On considère les notes comme des réels (ici float mais vous pouvez utiliser double)
- On définit comme constante symbolique la note maximal MaxN à 20 et MinN à 0, comme ça si on change de système de notation le programme est facilement adaptable (par exemple le système à 100 points)
- Il est bien sûr évident que n'importe quelle note saisie par l'utilisateur doit être maîtrisée par un contrôle de saisie qui n'accepte que les notes entre MinN (ici 0) et MaxN (ici 20)
- Pour les initialisations des notes min et max, on choisit :
  - d'attribuer à min la note maximale MaxN, puisqu'on sait que quelque soit la note saisie par l'utilisateur sera toujours  $\leq \text{MaxN}$
  - et à max la note minimale MinN, puisqu'on sait que quelque soit la note saisie par l'utilisateur sera toujours  $\geq \text{MinN}$

# Exercice 6

```
1  #include <stdio.h>
2  int main(void){
3      int n,lg,sp,j;//lg:numéro ligne ; sp: nombre d'espace
4      do{ printf("entrer le nombre de ligne du triangle (entre 1 et 20) \n");
5          scanf("%d",&n);
6      }while(n<1 || n>20);
7
8      for(lg=1; lg<=n; lg++){//pour chaque ligne
9          sp=n-lg;// calculer nombre espace a gauche et a droite des etoiles
10         for(j=1; j<=sp;j++)
11             printf(" "); // inclure espace à gauche
12         for(j=1; j<=2*lg-1; j++)
13             printf("*"); // inclure *
14         for(j=1; j<=sp; j++)
15             printf(" "); // inclure espace a droite
16         printf("\n");
17     }
18 }
```

# Exercice 6

## Remarques:

- Contrôle de saisi entre 1 et 20 (L4 à L6) sur le nombre de lignes du triangle
- Pour ce genre de problème, on commence par dessiner à la main l'affichage souhaité, on prend un exemple pour  $n=3$  (impair) et puis  $n=4$  (paire) et on essaie de généraliser la règle:

Pour  $n=3$

```
  *
 * * *
* * * * *
```

- $n=3$  et  $lg=1$ , 2 espaces, 1 étoile, 2 espaces
- $n=3$  et  $lg=2$ , 1 espace, 3 étoiles, 1 espace
- $n=3$  et  $lg=3$ , 0 espace, 5 étoiles, 0 espace
- $n=4$  et  $lg=1$ , 3 espaces, 1 étoile, 3 espaces
- $n=4$  et  $lg=2$ , 2 espace, 3 étoiles, 2 espace
- $n=4$  et  $lg=3$ , 1 espace, 5 étoiles, 1 espace
- $n=4$  et  $lg=4$ , 0 espace, 7 étoiles, 0 espace

Pour  $n=4$

```
      *
    * * *
  * * * * *
* * * * * *
```

- **On conclue que les nombre d'espace  $sp= n-lg$  et le nombre d'étoile  $= 2*lg-1$**

- On traite chaque ligne avec la boucle for (de L8 à L18):
  - L9 sert à calculer le nombre d'espace à inclure pour cette ligne
  - La boucle for (L10 et L11) sert à inclure les espaces
  - La boucle for (L12 et L13) sert à inclure les étoiles
  - La boucle for (L14 et L15) est facultative, on peut ne pas la mettre
  - Le retour à la ligne L16 sert à passer à la ligne suivante

# Exercice 7

```
1  #include<stdio.h>
2  #include<math.h>
3  #define deb 150
4  #define fin 410
5  int main(void){
6      int nb,centaine,dizaine,unite,cub;
7      for(nb=deb;nb<=fin;nb++){
8          centaine=nb/100;
9          dizaine=(nb%100)/10;
10         unite=nb%10;
11         cub=pow(centaine,3)+pow(dizaine,3)+pow(unite,3);
12         if(cub==nb)
13             printf("%d est un nombre cubique\n",nb);
14     }
15 }
```

## Remarques:

- On récupère les centaine, dizaine et unité de chaque nombre (nb) entre deb et fin, et on calcule puissance 3 et on somme, si la somme égal à nb c'est un nombre cubique qu'on affiche

# Exercice 8

```
1  #include<stdio.h>
2  #define NbMax 1000
3  int main(void){
4      int nb,somme,div;
5      for(nb=2;nb<=NbMax;nb++){
6          somme=1;//car la somme contient tjr 1
7          for(div=2;div<nb;div++){
8              if(nb%div==0)
9                  somme+=div;
10         if(somme==nb)
11             printf("%d est un nombre parfait\n",nb);
12     }
13 }
```

## Remarques:

- On initialise somme à 1, car 1 est un diviseur de tous les nombres
- $div < n$  car on ne doit pas inclure le nombre lui-même
- Si le nombre a un diviseur, alors ajouter ce diviseur à somme
- Après avoir sommé tous les diviseurs (sauf le nombre lui-même), si  $somme == nb$  alors on affiche que c'est un nombre parfait