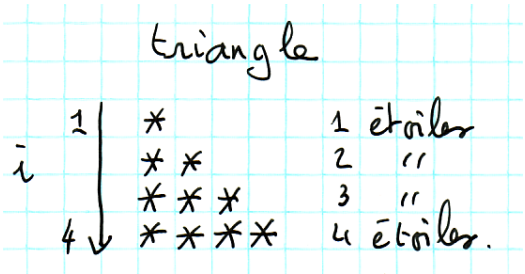
La difficulté provient essentiellement de l'algorithme permettant de réaliser la figure ci dessous.

```
Hauteur du feuillage : 10
Largeur du tronc : 5
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
*******************
        *****
        *****
        *****
        *****
        *****
```

Le texte vous propose de procéder en trois étapes :

A/ Le triangle :

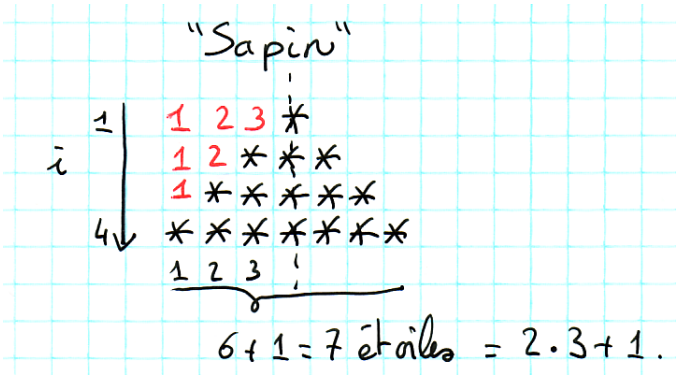


Pour afficher ce triangle, on procédera ligne par ligne. Pour parcourir toutes les lignes, il faut une boucle.
Chaque ligne est composée d'autant d'étoiles que le numéro de la ligne. La ligne sera affichée caractère par caractère en utilisant une boucle. Toutefois la valeur finale de la deuxième boucle dépend du numéro de ligne, c'est à dire de la variable servant d'indice à la première boucle.

Ce qui donne le programme suivant :

```c
// Affiche un triangle d'étoiles a l'écran
#include<stdio.h>

int main(void)
{   int i,j ;
    int L = 7 ; // nombre de lignes d'etoiles,

    for ( i=1 ; i<=L ; i++) // pour toutes les lignes
    {
        for (j=1 ; j<=i ; j++ ) // nbre d'etoiles sur 1 ligne
        {   printf("*");
        }
        printf("\n"); // On passe à la ligne
    }

    return 0;
}
```

B/ Le “feuillage” :

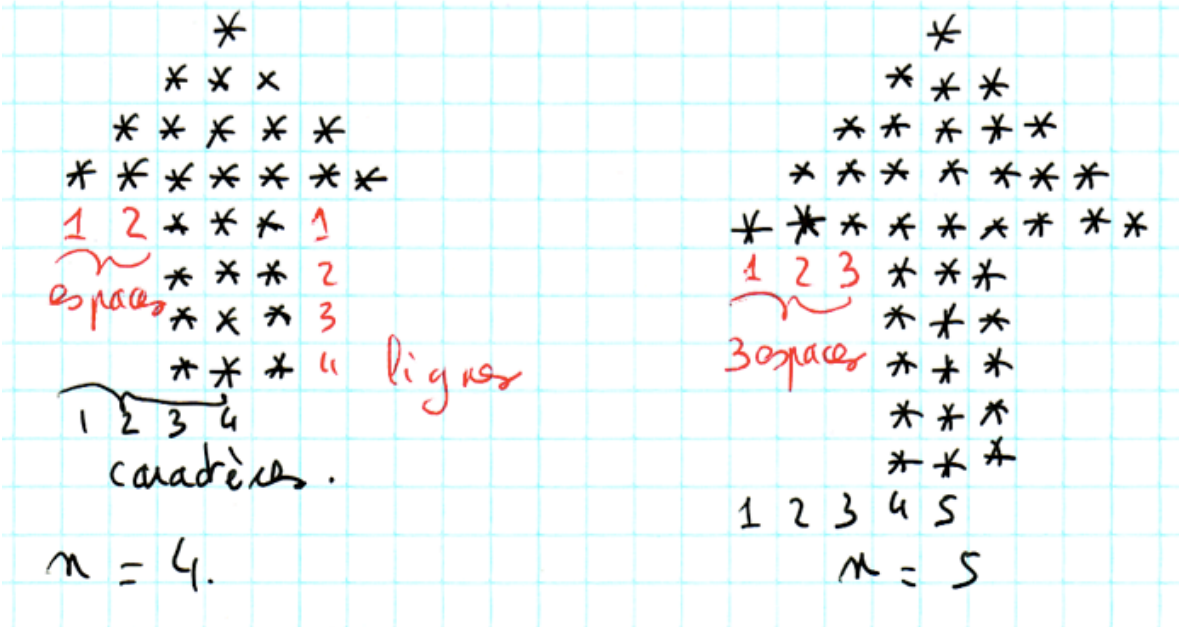


Cette fois, il faut, en plus de placer des étoiles, placer des espaces, numérotés en rouge dans la figure ci-dessus.
On affiche 3, puis 2, puis 1 espaces. Il s’agit donc d’une boucle dont la valeur finale décroit.
Pour les étoiles, on affiche 1, puis 3, puis 5, puis 7 étoiles, autrement dit, chaque ligne est un nombre impair d’étoiles. Or un nombre est impair s’il s’écrit sous la forme 2n+1 ou 2n-1.
Ce qui donne le code suivant :

```c
#include<stdio.h>

int main(void)
{    int i,j;
     int L = 7 ; // nombre de lignes d'étoiles

     for ( i=1 ; i<=L ; i++) // ttes les lignes
     {    /* Affiche des espaces.
           * Notez la limite de la boucle
           * si i va de 1 à L, L-i va de L-1 à zéro
           */
          for ( j=1 ; j<=(L-i) ; j++ )

          {    printf(" ");
          }
          // affichage des étoiles.
          for (j=1 ; j<=(2*i-1) ; j++ )
          {    printf("*");
          }
          printf("\n"); // on passe à la ligne
     }

     return 0;
}
```

C/ Le sapin :

Exemple avec un tronc de 3 étoiles de large.



On considère un tronc fait de T étoiles de large. T est impair (T = 2*t+1), (le programme proposé corrige la valeur de t s'il est entré pair en ligne 17), sinon il est impossible de centrer le tronc sur le feuillage qui lui a une base constituée d'un nombre 2n-1 impair d'étoiles.

Les espaces se répartissent en quantité égale de chaque coté du tronc (sinon celui-ci n'est pas centré). On a donc à répartir : (2n-1 - T) espaces. Il faut donc en placer de chaque coté la moitié L espaces :
L = (2n-1-T)/2 = (2n-1-2t+1)/2 = n-t espaces où t est le quotient entier de T par 2 : t = T/2.

Rappel : Le C adapte les opérateurs aux opérandes, T et 2 étant des entiers, T/2 = t tel que T = 2t+1.
Le programme suivant considère un tronc ayant autant de lignes qu'il est large.

```c
#include<stdio.h>

int main(void)
{   int i,j;
    int f ; // nombre de lignes d'*
    int T,t ; //  nombre de lignes du tronc

    printf("Hauteur du feuillage : ");
    scanf("%d",&f);
    printf("Largeur du tronc : ");
    scanf("%d",&T);

    /*
     * Si le tronc n'est pas de largeur impaire, on augmente
     * sa largeur de 1, ce qui le rend de largeur impaire
     */
    if ((T%2)==0)
        T++;

    t = T/2 ; // si le Tronc fait 3 de large, t = 3/2 = 1.
    /*
     * Affichage du "feuillage"
     */
    for ( i=1 ; i<=f ; i++)
    {   for ( j=1 ; j<=(f-i) ; j++ )
        {   printf(" ");
        }
        for (j=1 ; j<=(2*(i-1)+1) ; j++ ) // nbre d'* sur 1 ligne
        {   printf("*");
        }
        printf("\n");
    }
    /*
     * Affichage du tronc
     */
    for ( i=1 ; i<=T ; i++ ) // ttes les lignes du tronc
    {
        for ( j=1 ; j<(f-t) ; j++ )
            printf(" ");

        for ( j=1 ; j<=T ; j++)
            printf("*");
        printf("\n");
    }

    return 0;
}

```

Notez le signe “<” dans la boucle for ligne 38. Sans cela il faut aller jusqu’à (f-t-1) inclus. Le reste du programme est sans surprises. On peut obtenir le même résultat avec beaucoup moins de code en ne définissant pas les variables intermédiaires, mais le code est

moins lisible.