Université de Strasbourg

Algorithmique et Programmation 2

Licence 1 - UFR Mathématique - Informatique

TD/TP : Mon beau sapin, roi des forêts !

Objectifs & consignes

L'objectif de ce TP est de dessiner un sapin décoré (c'est de saison ☺) dans votre terminal, tel qu'illustré sur la figure 1.

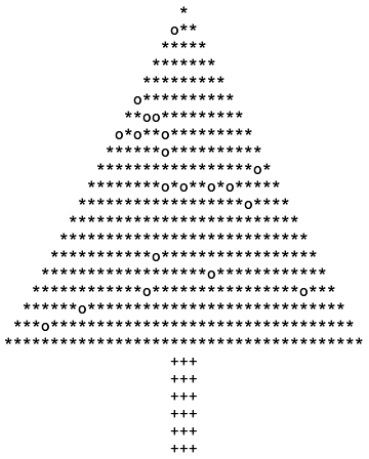

Figure 1 – Exemple de sortie attendue pour un sapin de hauteur $H_b = 20$

Partie 1 : le sapin...

1.1 Description du problème

Votre sapin sera stocké dans un tableau de caractères à 2 dimensions. Les cases représentant les branches seront représentées par le caractère `*` et les cases représentant le tronc du sapin par le caractère `+`. Tout ce qui n'appartient pas au sapin sera représenté dans le tableau par un espace.

La hauteur totale H du sapin est la hauteur de la partie haute (les branches), notée H_b, à laquelle s'ajoute la hauteur du tronc. La hauteur de la partie haute H_b sera fournie par l'utilisateur. La hauteur du tronc, notée H_t, vaut 1/3 de H_b.

La largeur L du sapin correspond à la largeur de la base de la partie haute. Le sommet du sapin comporte une étoile et chaque ligne l comporte ensuite deux étoiles de plus que la précédente.

1.2 Questions préliminaires

1. Que vaut H, la hauteur totale du sapin ?
2. Quelle est la largeur L du sapin en fonction de H_b ?
3. Pour $0 \leq l < H_b$, combien y a-t-il d'étoiles sur la ligne l ?
4. Toujours pour $0 \leq l < H_b$, combien y a-t-il d'espaces avant les étoiles sur la ligne l ?

1.3 Mise en œuvre

1. Définir une fonction `allouer_sapin` permettant d'allouer le tableau nécessaire au stockage de votre sapin. Votre fonction prend en paramètres la hauteur totale et la largeur du sapin et retourne le tableau 2D alloué dynamiquement.
2. Définir une fonction `liberer_sapin` permettant de libérer l'espace alloué pour stocker un sapin. Votre fonction prend en paramètres un tableau 2D de caractères (`char **`) ainsi que le nombre de lignes de ce tableau. La fonction ne retourne rien.
3. (pré-requis : questions préliminaires) Définir une fonction `initialiser_sapin` permettant de remplir par des espaces ou des étoiles un tableau 2D de caractères. Votre fonction prend en paramètres un tableau 2D de caractères (`char **`), la hauteur et la largeur de ce tableau et la hauteur de la partie haute H_b du sapin. La fonction ne retourne rien.
4. Définir une fonction `afficher_sapin`. Votre fonction prend en paramètres un tableau 2D de caractères (`char **`) ainsi que les dimensions de ce tableau et affiche chaque case du tableau. La fonction ne retourne rien.
5. Combien de sapins sont dessinés dans cet énoncé ?
6. Tester l'ensemble de ces fonctions dans un `main`. La hauteur H_b est à demander à l'utilisateur *via* un appel à `scanf`.

Partie 2 : . . . et la décoration

2.1 Description du problème

Vous allez maintenant décorer votre sapin en y plaçant, de manière aléatoire, un certain nombre de boules. Ce nombre sera fourni par l'utilisateur.

Pour identifier la position des boules dans le sapin, vous utiliserez des coordonnées cartésiennes à 2 dimensions (x, y). Il sera nécessaire de vérifier qu'une boule a une position valide dans le sapin : une position est valide si celle-ci fait partie des branches du sapin (une case du tableau ayant pour valeur `'*'`, en excluant donc les cases correspondant au tronc).

2.2 Mise en œuvre

1. Définir un type `position_t` permettant d'identifier une position dans le sapin.
2. Définir une fonction `position_valide` permettant de savoir si une position (x, y) est une position valide dans le sapin. La fonction prend en paramètres une position et un tableau 2D de caractères (`char **`) correspondant au sapin. Elle renvoie 1 si la position est valide (*i.e.* correspond à une branche), 0 sinon.
3. Que fait la fonction `mystere` ci-dessous ?
4. En utilisant la fonction `mystere`, définir une fonction `decorer_sapin` qui prend en paramètres un tableau 2D de caractères (`char **`) correspondant à un sapin, un tableau contenant des positions de décoration valides et la taille de ce tableau. Votre fonction doit modifier le sapin passé en paramètre en remplaçant les `'*'` correspondant aux positions par des `'o'`. La fonction ne renvoie rien.
5. Tester vos fonctions dans votre `main`. Le nombre de boules à placer dans le sapin est fourni par l'utilisateur. Faites une capture d'écran de votre plus beau sapin !

Annexe : fonction mystère

```c
// Generation d'un entier aleatoire compris dans l'intervalle [min;max[
// Note : include stdlib.h (pour rand()) et time.h (pour time())
int aleatoire(int min, int max) {
  // booleen pour savoir si une graine a deja ete fournie :
  static int rand_init = 0;
  if (!rand_init) {
    srand(time(NULL)); // Initialisation de rand avec une graine
    rand_init = 1;
  }
  return rand()%(max-min) + min;
}

void mystere(position_t *deco, int nb_boules,
             const char **sapin, int h_branches, int largeur){
  int ligne = 0, colonne = 0;
  for (int i = 0; i < nb_boules; i++) {
    do {
      ligne   = aleatoire(0, h_branches);
      colonne = aleatoire(0, largeur);
      deco[i].x = ligne; deco[i].y = colonne;
    } while(!position_valide(deco[i], sapin));
  }
  return;
}
```