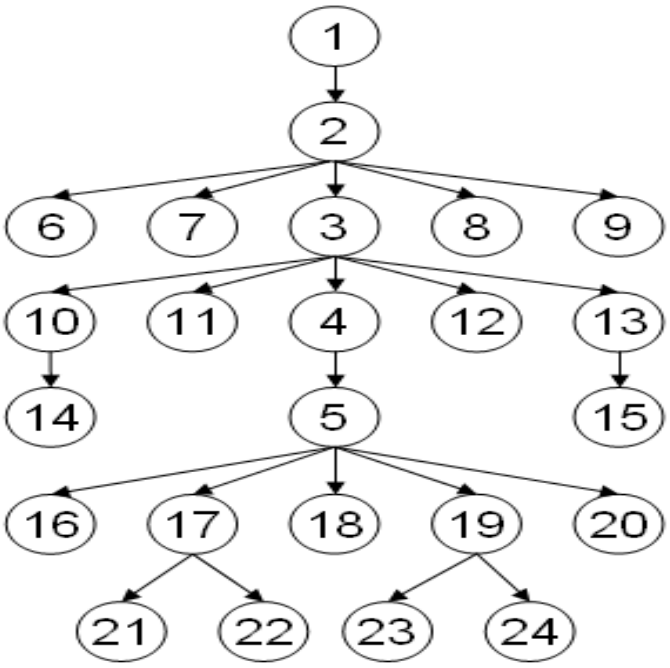
Travaux Dirigés
Exercices corrigés sur les arbres

Exercice 1

Etant donné l'arbre T suivant :



1. Déterminer pour l'arbre T, sa racine, sa taille, sa hauteur, sa profondeur, ses nœuds intérieurs et ses feuilles.
2. Pour le nœud 4, déterminer son parent, ses frères, sa hauteur, sa profondeur, ses ancêtres et ses descendants propres.
3. Déterminer les nœuds qui sont à gauche du nœud 4, et ceux qui sont à sa droite.

Solution de l'exercice 1

1. La racine de l'arbre T est 1, sa taille est 24, sa hauteur est 6, sa profondeur est 6, ses nœuds intérieurs sont : 1, 2, 3, 10, 4, 13, 5, 17, 19, et ses feuilles sont : 6, 7, 8, 9, 11, 12, 14, 15, 16, 18, 20, 21, 22, 23, 24.
2. Pour le nœud 4, son parent est 3, ses frères sont 10, 11, 12, 13, sa hauteur est 3, sa profondeur est 3, ses ancêtres sont : 4, 3, 2, 1et ses descendants propres sont : 5, 16, 17, 18, 19, 20, 21, 22, 23, 24.
3. Les nœuds qui sont à gauche du nœud 4 sont : 10, 11, 14 et ceux qui sont à sa droite sont : 12, 13, 15.

Exercice 2

Soit le tableau suivant qui représente un arbre binaire T en triplets (info, gauche, droit) :

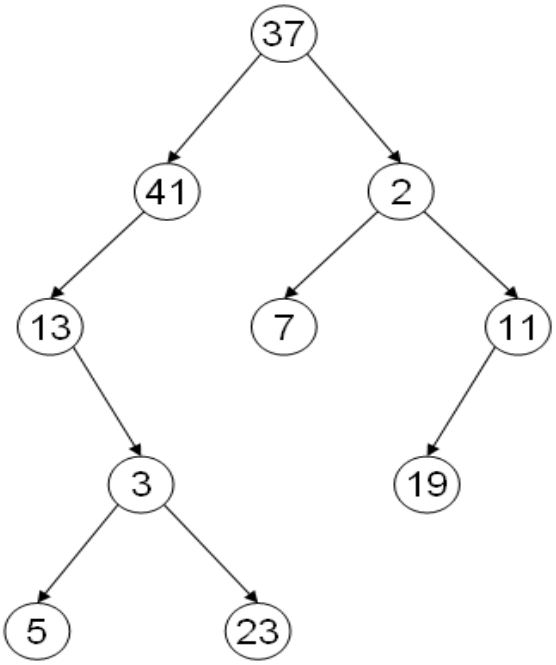
23	2	3	5	7	11	13	37	41	19
-1	5	3	-1	-1	9	-1	8	6	-1
-1	0	0	-1	-1	-1	2	1	-1	-1

La première colonne (indice 0) représente le nœud dont le champ info est 23 (valeur du nœud), le champ gauche est -1 (indice du fils gauche) et le champ droit est -1 (indice du fils droit), La seconde colonne (indice 1) représente le nœud dont le champ info est 2, le champ gauche est 5 et le champ droit est 0, et ainsi de suite. La valeur (-1) indique l'absence d'un fils gauche ou droit.

1. Dessiner l'arbre binaire T et donner sa taille.
2. Donner le code C pour représenter l'arbre T de cette manière.
3. Écrire une fonction qui détermine la racine de l'arbre T.
4. Écrire une procédure qui liste toutes les feuilles de l'arbre T.
5. Donner le résultat du parcours de l'arbre T : (i) en ordre infixe, (ii) en ordre préfixe, et (iii) en ordre postfixe.

Solution de l'exercice 2

1. Dessin de l'arbre binaire T (taille = 10) :



2. Code C :

```
#include <stdio.h>

typedef struct Noeud
{
   int info, gauche, droit;
} Noeud;
```

```
Noeud T[10] = {{23, -1, -1}, {2, 5, 0}, {3, 3, 0}, {5, -1, -1}, {7, -1, -1}, {11, 9, -1}, {13, -1, 2}, {37, 8, 1}, {41, 6, -1}, {19, -1, -1}};
```

3. La racine d'un arbre est le nœud qui ne soit pas un fils d'aucun nœud de cet arbre. Une procédure qui détermine la racine de l'arbre T est la suivante :

```
Noeud obtenirRacine()
{
   int i, L[10] = {0};

  for(i = 0; i < 11; i++)
  {
    if(T[i].gauche != -1)
      L[T[i].gauche] = 1;
    if(T[i].droit != -1)
      L[T[i].droit] = 1;
  }

  for(i = 0; i < 10; i++)
  {
    if(L[i] == 0)
      return T[L[i]];
  }
}
```

Explication : on initialise tous les éléments du tableau L à 0. On parcourt le tableau T pour déterminer les nœuds qui sont des fils : si un nœud d'indice i a un fils gauche (T[i].gauche) ou droit (T[i].droit), il marquer ce dernier dans L. à la fin, le tableau L contient une seule entrée avec la valeur 0 : c'est la racine de l'arbre.

4. Une procédure qui liste toutes les feuilles de l'arbre T :

```
void Feuilles()
{
   int i;

  for(i = 0; i < 10; i++)
  {
    if(T[i].gauche == -1 && T[i].droit == -1)
      printf("%d ", T[i].info)
  }
}
```

5. Résultat en ordre infixe : 5, 3, 23, 13, 41, 37, 7, 2, 19, 11

   Résultat en ordre préfixe : 37, 41, 13, 3, 5, 23, 2, 7, 11, 19

   Résultat en ordre postfixe : 5, 23, 3, 13, 41, 7, 19, 11, 2, 37