Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
IFT-3101

Danny Dubé
Version : Hiver 2011

Exercices reliés au chapitre 7

Exercices

Voici les exercices que je recommande de faire :

– **Exercice supplémentaire 1**. En utilisant les règles de portée de C++, identifiez les déclarations concernées par chaque apparition des noms `a` et `b` dans la figure suivante. La sortie produite par le programme consiste en les entiers 1 à 4. (L'exercice 7.1 dans la 1ère édition est semblable)

```
void b(int u, int v, int x, int y)
{
    struct
    {
        int a; int b;
        void set(int v1, int v2)
        {
            a = v1;
            b = v2;
        }
    } a;

    struct
    {
        int b; int a;
        void set(int w1, int w2)
        {
            a = w1;
            b = w2;
        }
    } b;

    a.set(u, v);
    b.set(x, y);
    printf("%i %i %i %i\n", a.a, a.b, b.a, b.b);
}

int main (int argc, char * const argv[])
{
    b(1,2,3,4);
    return 0;
}
```

Figure. Programme en C++ comportant plusieurs déclarations de `a` et `b`.

– **Exercices 7.2.2**.
Note : utilisez l'algorithme de la figure 7.2 / de l'exemple 7.2

– **Exercices 7.2.3**.

Réponses

Exercice supplémentaire 1

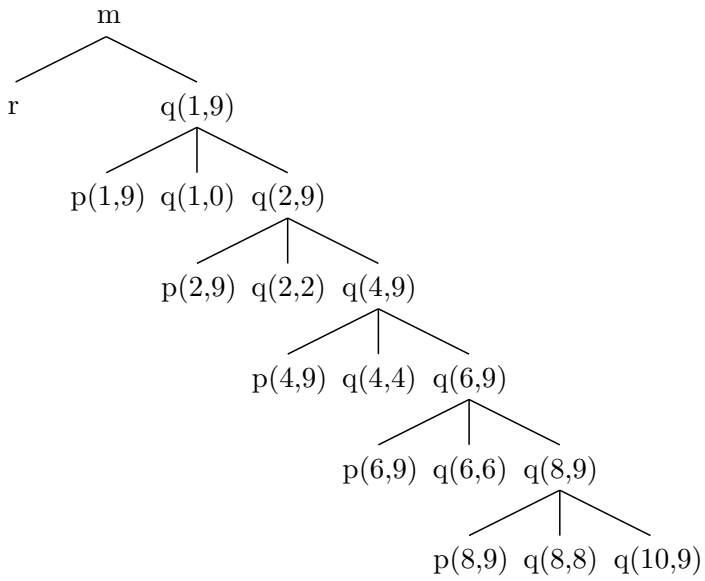
Occurrence	Déclaration
`a = v1;`	`int a; int b;`
`b = v2;`	`int a; int b;`
`a = w1;`	`int b; int a;`
`b = w1;`	`int b; int a;`
`a.set(u,v);`	`struct{ ... } a;`
`b.set(u,v);`	`struct{ ... } b;`
`a.a`	`struct{ ... } a;`
`a.a`	`int a; int b;`
`a.b`	`struct{ ... } a;`
`a.b`	`int a; int b;`
`b.a`	`struct{ ... } b;`
`b.a`	`int b; int a;`
`b.b`	`struct{ ... } b;`
`b.b`	`int b; int a;`
`b(1,2,3,4);`	`void b (...) { ... }`

7.2.2

Tout d'abord, voici une trace de l'algorithme. Le soulignement représente le prochain choix de pivot, et le gras représente un pivot avec lequel on a déjà effectué une partition.

1	3	5	7	9	2	4	6	8
1	3	5	7	9	2	4	6	8
1	2	**3**	5	7	9	4	6	8
1	2	**3**	4	**5**	7	9	6	8
1	2	**3**	4	**5**	6	**7**	9	8
1	2	**3**	4	**5**	6	**7**	8	**9**

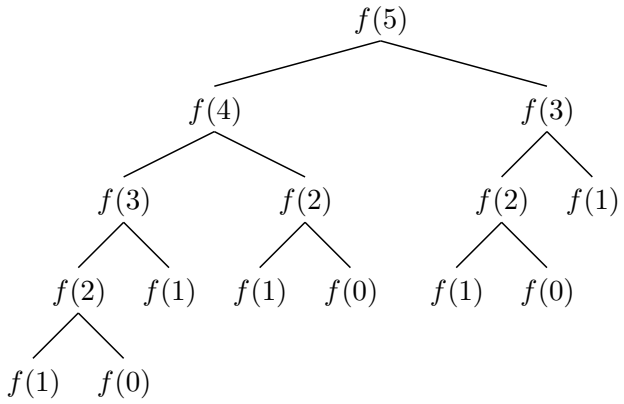
L'arbre d'activation est alors :



Le plus grand nombre de blocs d'activation sur la pile en même temps est de 7.

7.2.3

– (a) L'arbre d'activation est :



– (b) La première fois que $f(1)$ retourne, la pile ressemble à :

Valeur retour	1
n	1
s	–
t	–
Valeur retour	–
n	2
s	–
t	–
Valeur retour	–
n	3
s	–
t	–
Valeur retour	–
n	4
s	–
t	–
Valeur retour	–
n	5
s	–
t	–

Les éléments les plus récemment empilés sont illustrés sur le haut. Une valeur de '–' signifie que cet espace mémoire n'a pas encore été initialisé.

– (c) La cinquième fois que $f(1)$ retourne, la pile ressemble à :

Valeur retour	1
n	1
s	–
t	–
Valeur retour	–
n	3
s	2
t	–
Valeur retour	–
n	5
s	5
t	–

Les éléments les plus récemment empilés sont illustrés sur le haut. Une valeur de '–' signifie que cet espace mémoire n'a pas encore été initialisé.