Licence d'informatique
Module de C/Unix

Examen – Partie C

Jean-Luc LEVAIRE

Septembre 2000

Les documents de cours et TD sont autorisés.

On rendra deux copies séparées pour la partie Unix et pour la partie langage C. La partie C est à rendre sur une *copie de couleur blanche*.

1 Questions de cours

Question 1
Donner la représentation schématique de `i` et de `p` avant exécution de la boucle `for`. Qu'affiche le programme ? Quel est le contenu du tableau `i` à la fin du programme ?

```
void
f (int **i)
{
  *(*i)++ += 1;
}

void
main(void)
{
  int i[5] = {0, 2, 4, 6, 8}, *p = i;

  f(&p);
  printf("%d %d\n", i[0], *p);
}
```

Question 2
En quoi consiste la compilation séparée ? Quels en sont les avantages ?

Question 3
Que signifient les déclarations suivantes :

- `char *(*d1[2])(int);`
- `int (*d2(int (*d[])(int), int))(int);`

2 Manipulation de graphes

Le but de ce problème est d'implémenter des graphes dirigés. On se propose d'utiliser la structure présentée à la Figure 1. Un graphe G est représenté sous la forme d'une structure contenant trois champs : le nombre de nœuds de G, un tableau $suivant$ contenant pour chaque nœud, sa liste de nœuds suivants, et un tableau $precedent$ contenant pour chaque nœud, sa liste de nœuds précédents. Ces deux tableaux, ainsi que les différentes listes de nœuds sont alloués *dynamiquement* lors de la création du graphe. Enfin les nœuds sont référencés dans les listes par un numéro d'ordre choisi quelconque à la création du graphe.

Id : xamc00-sep.tex,v 1.2 2000/11/18 09 :41 :32 marquet Exp

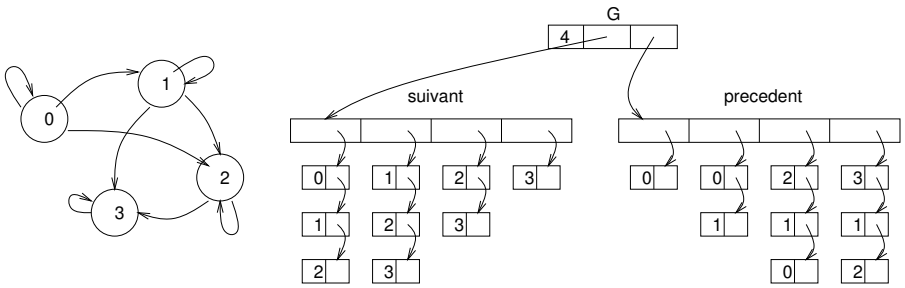

FIG. 1 – Un graphe G et sa représentation interne.

Question 4
Proposer un type `Noeud` pour représenter un noeud dans une liste, et un type `ListeNoeud` pour représenter une liste de nœuds. Proposer alors un type `Graphe` pour représenter un graphe. Proposer enfin un type `Bool` pour représenter le type booléen.

Question 5
Écrire une fonction

```
Bool Coherent(Graphe G) ;
```

qui teste si la représentation G d'un graphe est cohérente. Une représentation d'un graphe est cohérente, si les listes des suivants et des précédents sont cohérentes.

Question 6
Écrire une fonction

```
Bool Transitif(Graphe G) ;
```

qui teste si la relation associée à un graphe `G` est transitive[1].

Question 7
On désire représenter un graphe `G` sous la forme d'une matrice d'adjacence `M`. `M` est une matrice carrée d'entiers, et un élément `M[i][j]` différent de `0` indique qu'il y a une arête du nœud `i` vers le nœud `j`. Écrire une fonction

```
int *Convert(Graphe G) ;
```

qui convertit un graphe `G` en une matrice d'adjacence. La zone contenant cette matrice sera allouée dynamiquement, et la fonction `Convert` retourne un pointeur sur son début.

[1] Un nœud a est en relation avec un nœud b si et seulement si il existe une arête entre a et b dans G. Une relation est transitive si a est en relation avec b et b est en relation avec c, alors a est en relation avec c.