Licence d'informatique
Module de C/Unix

Examen – Partie C

Jean-Luc LEVAIRE

Janvier 2001

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
Un processus sous Unix peut-il ouvrir plusieurs fois simultanément un même fichier ? Pourquoi ? Quel(s) peu(ven)t en être l'(es) intêret(s) et inconvénient(s) ?

Question 2
Que signifient les déclarations suivantes :
- `typedef struct {char *l1; char *l2; char *(*f)(char *)} *ftranslatetab;`
- `char *(*getftranslate(char *, char *, ftranslatetab))(char *).`

2 Lego

Le but du problème est de "jouer au LEGO", c'est-à-dire assembler des briques dans un espace en trois dimensions, et de tester si un assemblage est stable sur un plan. Pour simplifier, une brique est un parallélépipède rectangle toujours orienté parallèlement aux axes du repère (Ox, Oy, Oz). Chaque brique est définie par les coordonnées réélles de son point de coordonnées minimales (x, y et z minimaux), et par les longueurs de ses 3 côtés (lx, ly et lz). Un assemblage de briques est un ensemble de briques, dont le nombre de briques est variable et donc non connu à la compilation. De plus, on suppose naturellement dans toute la suite que les briques ne se recouvrent pas dans un assemblage (elles sont plaçées dans l'espace par l'intermédiaire d'une interface graphique, qui interdit de tels placements).

2.1 Opérations basiques

Question 3
Proposer un type `SPoint` pour représenter un point dans l'espace, un type `Brique` pour représenter une brique, et un type `Lego` pour représenter un assemblage de briques. On rappelle que le nombre de briques dans un assemblage est non borné.

Question 4
Écrire les fonctions

```
float volume_brique(Brique brique);
float volume_lego(Lego lego);
```

qui retournent respectivement le volume d'une brique `brique`, et le volume total (somme des volumes) d'un assemblage de briques `lego`.

Question 5
Écrire une fonction

```
float volume_englobant(Lego lego)
```

qui calcule le volume du plus petit parallélépipède rectangle englobant un assemblage de briques `lego` (par définition des briques, les arêtes de ce parallélépipède sont parallèles aux axes du repère).

Question 6
Écrire les fonctions

```
SPoint cdg_brique(Brique brique);
SPoint cdg_lego(Lego lego);
```

qui retournent respectivement le centre de gravité d'une brique `brique`, et le centre de gravité d'un assemblage de briques `lego`. Le centre de gravité d'une brique se trouve en son centre (les briques étant composées d'un matériau homogène). Le centre de gravité d'un assemblage est le point de coordonnées (xg, yg, zg). On calcule xg en faisant la moyenne des coordonnées `x` de chaque centre de gravité de chaque brique, pondéré par le volume de la brique (idem pour yg et zg).

Question 7
Écrire une procédure

```
void insert(Lego *plego, Brique brique)
```

qui insère la brique `brique` dans un assemblage de briques pointé par `plego`.

2.2 Points de contact

Nous posons un assemblage sur une surface plane perpendiculaire à l'axe Oz. Nous allons d'abord rechercher l'ensemble des points de contact que possède l'assemblage avec cette surface. Cette surface de contact correspond au plan d'équation z = zmin, avec zmin la coordonnée en z minimale parmi les briques de l'assemblage.

Question 8
Écrire une fonction

```
float zmin(Lego lego)
```

qui retourne la coordonnée minimale en z parmi les briques de `lego`.

L'ensemble des points de contact est l'ensemble des points projections sur cette surface des centres de gravité des briques qui ont zmin comme coordonnée en `z` (les briques touchent la surface de contact en leur milieu). Nous travaillerons maintenant dans le plan correspondant à la surface de contact, et n'utiliserons donc que des coordonnées planaires.

Question 9
Proposer un type `PPoint` pour représenter un point dans le plan, et un type `PPointSet` pour représenter un ensemble de points dans le plan.

Question 10
Écrire une fonction

```
PPointSet point_contact(Lego lego)
```

qui retourne l'ensemble des points de contact des briques de l'assemblage `lego` avec la surface de contact définie ci-dessus.

2.3 Polygone de sustentation

Le polygone de sustentation est l'enveloppe convexe de l'ensemble des points de contact. Pour déterminer l'enveloppe convexe d'un ensemble de points, on analyse tous les couples de points possibles de cet ensemble. Pour chaque couple, on détermine la droite passant par ces deux points, et on teste la position de tous les autres points. S'ils sont tous du même côté de cette droite, le couple forme un segment de l'enveloppe convexe. On rappelle que tous les points de coordonnées (x, y) situés du même côté d'une droite AB (A de coordonnées (xa, ya), B de coordonnées (xb, yb)), génèrent une valeur $v = (xa - x)(yb - ya) - (ya - y)(xa - xb)$ de même signe.

Question 11
Écrire une fonction

```
PPointSet sustentation(Lego lego)
```

qui retourne l'ensemble des points du polygone de sustentation de l'assemblage `lego`. On prendra garde, lorsque l'on a trouvé un segment de l'enveloppe convexe, de ne pas insérer deux fois un même point.

Fin du sujet