Examen d'Algorithmique - (2012/2013 - session 1)

Durée : 1h30 heure - Aucuns documents autorisés

Formation Ingénieurs CNAM

On s'attachera à soigner la présentation générale et l'orthographe, et on sera notamment vigilant quant à la syntaxe lors de l'écriture de fragment de code en C. La barême est donné à titre indicatif.

1 Echauffement (4 pts)

On rappelle la structure de données définie pour représenter les arbres binaires d'entiers ainsi que les opérations de base `vide` et `e`. Afin de pouvoir tester notre implantation, on programme également une fonction d'affichage `print`.

```
#include<stdio.h>

typedef struct arbre
{
  int x;
  struct arbre *g, *d;
} Sarbre, *Arbre;

Sarbre *vide() { return (NULL); }

Sarbre *e(Sarbre *ag, int v, Sarbre *ad) /* enracinement */
{
  Sarbre *m=(Sarbre *)malloc(sizeof(Sarbre));
  m->g=ag;
  m->x=v;
  m->d=ad;
  return m;
}

void print(Sarbre *a) /* affichage infixe */
{
  if (a!=NULL)
    {
      print(a->g);
      printf("%d ",a->x);
      print(a->d);
    }
  printf("\n");
}
```

Question 1 Programmer la fonction `fois2` (qui multiple par 2 chaque élément de l'arbre). On procèdera de manière récursive avec des effets de bord (le type de retour de la fonction demandée est `void`).

```
void fois2 (Sarbre *a)
{ ... }
```

Question 2 Programmer de manière récursive une fonction `somme` qui fait la somme de tous les éléments d'une arbre.

```
int somme(Sarbre *a)
{ ... }
```

2 Expressions correctement parenthèsées (8 pts)

On s'intéresse aux expressions de la forme `(())()()`. Le but est de produire une fonction, qui étant donnée une séquence de parenthèses détermine si elle est bien formée ou non. Etre bien formée signifie que toutes les parenthèses ouvertes ont bien été refermées et que l'on n'a pas de parenthèses fermantes superflues.

1- Pour chacune des séquences suivantes, expliquez si elles sont bien formées ou non : `()()()()`, `((()))`, `(((()` et `(()))()(`.

2- Avoir le même nombre de parenthèses ouvrantes et fermantes suffit-il pour être une expression bien formée ?

3- On choisit de représenter les séquences de parenthèses dans un tableau statique de taille fixe N encapsulé dans une structure appelée `seq`. Préciser pourquoi il est nécessaire de placer ce tableau statique à l'intérieur d'une structure. Aurait-on la même contrainte si l'on avait choisi d'utiliser un tableau dynamique ?

4- Proposez un mécanisme d'affichage de cette séquence de parenthèses par affichage un à un des éléments du tableau, si nécessaire avec des séparateurs.

5- Programmez une fonction qui teste si une séquence de parenthèses est bien formée.

3 Avec différents types de parenthèses (8 pts)

On désire maintenant tester la bonne formation d'une séquence de parenthèses qui contient 3 types de parenthèses `()`,`[]` et {}.

6- Expliquez pourquoi la solution naïve fournie dans la première partie ne fonctionne plus.

7- Rappelez comment implanter une pile de caractères sous la forme d'un chaînage (implantation non-contigüe).

8- Indiquez si les séquences suivantes sont bien formées ou non : {{{(()]}, ({[()]}), {{()}}[] et `[[)`.

9- Programmez, en utilisant la structure de données `pile` de la question **7**, une fonction qui teste si une expression représentée dans une structure de type `seq` avec les 3 types de parenthèses est bien formée ou non. Quelle est la condition d'arrêt de cette fonction ?