Université BORDEAUX I

Année 1993-94
Session de février

Maîtrise d'informatique

Modèles et Techniques de Programmation (J.P.Braquelaire)

Durée: 2 heures
Tous documents autorisés

Première partie – 5 points –

Le but de cette partie est d'étudier les mécanismes de liaison et de passage de paramètres implémentés dans le système de traitement de texte TEX. Celui-ci est basé sur un langage de programmation disposant de fonctions prédéfinies et d'un mécanisme de définition de macro-fonctions. Les fonctions (et macro-fonctions) sont insérées directement dans le texte à traiter. Les identificateurs sont formés du caractère `\` suivi d'une suite de lettres, ou d'un caractère quelconque différent d'une lettre. Par exemple la définition

```
\def\vecteur#1{$(#1_1, #1_2, #1_3)$}
```

définit la macro-fonction de nom `vecteur`, avec un paramètre formel noté `#1`, et dont le corps est la suite de caractères entre accolades `$(#1_1, #1_2, #1_3)$`. Il est possible d'utiliser la macro `vecteur` en plaçant son paramètre d'appel entre accolades. Par exemple, le texte :

```
Soient $X$ et $Y$ les vecteurs \vecteur{x} et \vecteur{y}.
```

est récrit en :

```
Soient $X$ et $Y$ les vecteurs $(x_1, x_2, x_3)$ et $(y_1, y_2, y_3)$.
```

qui est ensuite évalué de la façon suivante :

Soient X et Y les vecteurs (x_1, x_2, x_3) et (y_1, y_2, y_3).

Le mécanisme de liaison est *dynamique*, et la validité d'une liaison est celle du bloc dans laquelle elle est placée (un bloc est défini par une paire d'accolades). La stratégie utilisée pour les passages de paramètres est la *stratégie normale* (recherche du *redex* le plus à gauche parmi les plus externes).

Donner, en détaillant les étapes importantes de réécriture et en les commentant, le résultat du traitement par TEX du source suivant :

```
\def\f#1{F\g{#1}F#1}
\def\g#1{G#1}
\def\h#1{#1 , #1 , \k{x}}
\def\k#1{H #1 \def\k##1{#1 ##1 #1}}

\f{\g{ab}}
{ \def\g#1{} \f{1} } \f{2}
\h{\k{y}}
```

On précise qu'une définition n'est pas recopiée dans le texte généré, et que dans le corps d'une macro, le caractère `##` est recopié en `#`.

Deuxième partie – 15 points –

Le but de cette partie est d'étudier un mécanisme de gestion de la mémoire pour des objets définis en langage C standard. On se limitera à un mécanisme simple d'encapsulation, sans sélecteur de méthodes, et dans lequel l'envoi d'un message $\langle mess \rangle$ à un objet $\langle obj \rangle$ d'une classe $\langle cls \rangle$ s'écrit $\langle cls \rangle$_$\langle mess \rangle$($\langle obj \rangle$...).

On attache à chaque classe une liste d'objets libres recevant les objets desalloués. Lors d'une instanciation, le nouvel objet est pris dans la liste libre de la classe si elle n'est pas vide, et alloué dynamiquement sinon.

Tous les objets partagent une même implémentation minimale permettant de les chaîner sur la liste libre. Le type de base `object` décrivant cette implémentation est défini par :

```
struct object
{
    void *link;
};
typedef struct object *object;
```

1) – 2 points – On munit chaque classe d'une structure de donnée, appelée *descripteur de classe*, et comportant la référence à sa liste d'objets libres et la taille d'un objet de la classe (cette dernière information est utile pour automatiser le mécanisme d'allocation d'un objet). Donner en C la déclaration et l'initialisation de cette structure de données.

2) – 6 points – Écrire en C les fonctions d'allocation `new` et de libération `delete` de mémoire pour un objet. La fonction d'allocation reçoit en paramètre un descripteur de classe et retourne un objet. La fonction de libération reçoit un objet et libère la mémoire qu'il occupe en utilisant la liste libre. Ces fonctions sont appelées par les fonctions $\langle cls \rangle$`_new` et $\langle cls \rangle$`_delete` de chaque classe $\langle cls \rangle$.

3) – 2 points – On souhaite attacher à chaque objet sa fonction de libération, sans que cela n'entraine de surcoût notable d'occupation mémoire. Proposer une solution. Comment effectuer dans ce cas la libération d'un objet dont on ne connaît pas le type.

4) – 3 points – On considère maintenant l'exemple d'une classe implémentant un arbre binaire. Les attributs d'un objet de cette classe sont : son fils droit, son fils gauche, et sa valeur. Expliquer, sans détailler l'implémentation, comment définir cet objet et comment s'écrivent les fonctions d'instanciation et de libération.

5) – 2 points – On dispose d'un mécanisme d'exceptions *à la ML* :

- `HANDLE(`$\langle exception \rangle$`,`$\langle expression \rangle$`,`$\langle handler \rangle$`)`
- `RAISE(`$\langle exception \rangle$`)`

Proposer un traitement des erreurs d'allocation dynamique.

FGGabFGab
FF1 FG2F2
H y , y y y , y x y