Traduction et Sémantique
La table des symboles, gestion de l'espace des noms

Luigi Santocanale
LIF, Université de Provence
Marseille, FRANCE

9 avril 2010

Plan

Traduction et Sémantique
La table des symboles, gestion de l'espace des noms

Luigi Santocanale
LIF, Université de Provence
Marseille, FRANCE

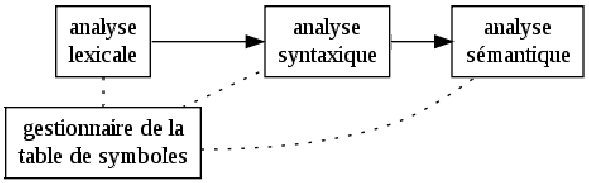
9 avril 2010

Plan

La table des symboles
- Principes
- Un peu plus en profondeur
- Aspects algorithmiques

Gestion de l'espace des noms

Interaction entre les différentes phases et la table



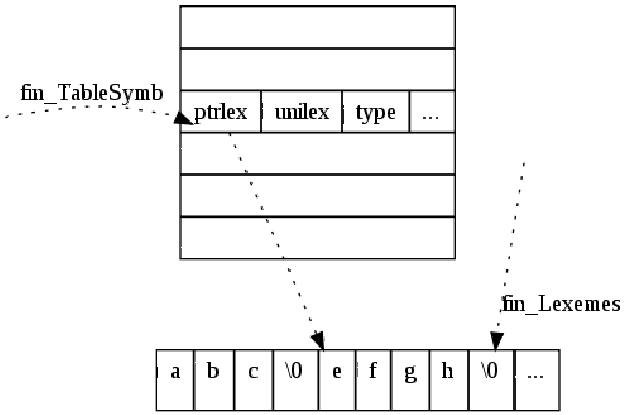
Interaction entre les différentes phases et la table

- Analyse lexicale :
  insertion des noms dans la table,
- Analyses syntaxique et sémantique :
  ajout d'attributs tels que :
  - type,
  - utilisation (procédure, variable, étiquette),
  - adresse en mémoire.

L'interface

- `Inserer(s,t)` :
  rend l'indice de la nouvelle entrée pour
  la chaîne `s` et l'unité lexicale `t`.
- `Chercher(s)` :
  rend l'indice de l'entrée pour la chaîne `s`
  ou `0` si `s` n'est pas trouvable.

Une implantation



Un exemple : l'`Entree` de la table

```
#ifndef GLOBAL_H
#define GLOBAL_H

struct Entree {
  char *ptrlex;
  int unilex;
  /* Utilisés par l'analyseur lexicale */

  int type;
  /* Utilisé par l'analyseur sémantique */
};

#endif /* GLOBAL_H */
```


Un exemple : `chercher`

```
#define T_LEXEMES 999   /* taille de Lexemes */
#define T_TABLESYMB 100 /* taille de la TableSymb */

char Lexemes[T_LEXEMES];
int fin_Lexemes = -1;
/* Denière position utilisée dans Lexemes */

struct Entree TableSymb[T_TABLESYMB];
int fin_TableSymb = 0;
/* Denière position utilisée dans TableSymb */

int chercher(char *symb)
{
  int p;

  for(p = fin_TableSymb; p>0;p--)
    if(strcmp(TableSymb[p].ptrlex,symb) == 0)
      return p;

  return 0;
}
```


Un exemple : `inserer`

```
int inserer(char *symb,int Lex)
{
  int long_symb = strlen(symb);

  /* Vérifier qu'on a assez de place */
  if(fin_TableSymb > T_TABLESYMB)
    Erreur("Table␣des␣symboles␣pleine");
  if(fin_Lexemes + long_symb + 1 > T_LEXEMES)
    Erreur("Table␣des␣lexemes␣pleine");

  /* Faire l'insertion et incrémenter */
  fin_TableSymb++;
  TableSymb[fin_TableSymb].unilex = Lex;
  TableSymb[fin_TableSymb].ptrlex = &Lexemes[fin_Lexemes +1];
  fin_Lexemes += long_symb + 1;
  strcpy(TableSymb[fin_TableSymb].ptrlex,symb);

  return fin_TableSymb;
}
```


Organisation de la table

- stockage des chaines de caractères,
- taille fixe vs. croissance dynamique

L'entrée de la table des symboles

- Entrée $\leftrightarrow$ déclaration (explicite où implicite)
- Création de l'entrée :
  - pendant l'analyse lexicale
  - avant l'analyse lexicale :
    si le mots clès ne sont pas reservées,
  - pendant l'analyse syntaxique :

```
int i;
struct i {int j; float k;};
```

Remarque : les noms ne sont pas nécessairement uniques.

Contenu de l'entrée

- nom
- (pointeurs vers) la chaîne,
- la table des chaines,
- type, utilisation,
- informations sur la mémoire.

Les informations sur la mémoire

Liaison :
association entre un nom et une location en mémoire.

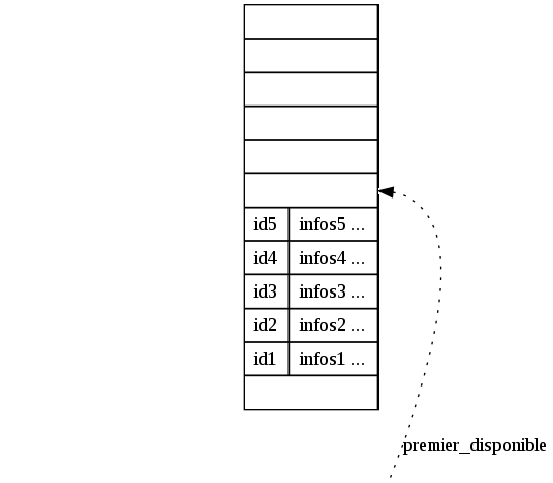
À ne pas confondre avec :
État :
association entre locations de mémoire et leur contenu.

Pour les données statiques,
si la traduction est vers le code machine :

- position de la donnée par rapport à une origine fixe.

Structure de donnée : liste linéaire

Nous l'avons vue auparavant :



Analyse de la performance

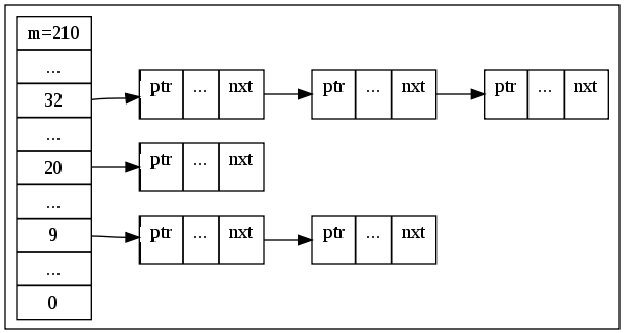
Hypothèses :

- les noms dupliqués ne sont pas permis,
- n symboles à insérer,
- r recherches.

Temps :

$$t = c\frac{n}{2}n + c\frac{n}{2}r = c\frac{n}{2}(n+r)$$

Tables d'adressage dispersés



Ingrédients

- Tableau d'adresses de taille m,
  (possiblement avec m nombre premier)
- m listes chaiénes,
- une focntion d'hachage

$$h : NOMS \longrightarrow \{0, \ldots, m-1\}$$

telle que

$$Prob(h(X) == i)$$

est uniforme.

Le nom s est inséré est cherché dans la liste $h(s)$.

Analyse

Insérer/Chercher une chaine se fait en temps

$$t = c_{n,m} = \frac{n}{m}$$

si h est uniforme.

Pour m proche de n,

- on peut considérer $c_{n,m} = \frac{n}{m}$ une petite constante :
  gain en temps,
- il faut stocker un tableu de m adresses :
  perte en espace.

A la recherche d'un bon hachage

Une meilleure évaluation :
insérer n noms $\{nom_0, \ldots, nom_{n-1}\}$ se fait en temps

$$t = \sum_{j=0}^{m-1} \frac{b_j(b_j+1)}{2}$$

où

$$b_j = \#\{k \mid h(nom_k) = j\} = \text{longueur de la liste } j.$$

L'optimum théorique est de

$$t = (\frac{n}{2m})(n+2m-1).$$

Un exemple

Si $nom = c_1 c_2 \ldots c_n$ on peut calculer

$$\begin{aligned} h_0 &= 0 \\ h_{i+1} &= h_i + \alpha c_{i+1} \\ h(nom) &= h_n \mod m, \end{aligned}$$

où α est une constante entière.

Exercice :
implémentez une table de symboles avec une table d'adressage, dans le projet `tic`.

Plan