Traduction des langages – Placement mémoire

Objectif :
- Définir la nouvelle structure d'arbre obtenue après la passe de placement mémoire
- Définir les actions à réaliser par la passe de placement mémoire

1 Placement mémoire

▷ **Exercice 1** *Donner le placement mémoire des variables du programme suivant :*

```
rat f3 (int a, int b, rat r){
  return [(a + num r) / (b + denom r)];
}
rat f2 (bool b, rat x, rat y){
  int x1 = num x;
  int x2 = denom x;
  rat res = f3(x1, x2, y);
  return res;
}
int f1 (int i, rat r, int n){
  rat r1 = f2(true, r, [i/n]);
  return denom r1;
}
test{
  int x = f1 (13, [4/11], 17);
  rat y = f2 (true, [4/11], [11/4]);
  rat z = f3 (1, 2, [4/11]);
  print ((x+denom y)+num z);
}
```

2 Passe de placement mémoire

Nous rappelons qu'un compilateur fonctionne par passes, chacune d'elle réalisant un traitement particulier (gestion des identifiants, typage, placement mémoire, génération de code,...). Chaque passe parcourt, et potentiellement modifie, l'AST. La troisième passe est une passe de placement mémoire.

2.1 Structure de l'AST après la passe de placement mémoire

La passe de placement calcule l'adresse des variables dans la pile. Cette adresse doit être mise à jour dans les informations associées aux identificateurs.

▷ **Exercice 2** *Définir la structure de l'AST après la passe de placement mémoire.*

2.2 Actions à réaliser lors de la passe de placement mémoire

▷ **Exercice 3** *Définir les actions à réaliser lors de la passe de placement mémoire.*