

Traduction des langages

Les pointeurs

Objectif :

- Etre capable d'ajouter les pointeurs au langage RAT
- Modifier le compilateur en conséquence

1 Les pointeurs

Une variable est physiquement identifiée de façon unique par son **adresse**, c'est-à-dire l'adresse de l'emplacement mémoire qui contient sa valeur.

Un **pointeur** est une variable qui contient l'**adresse** d'un autre objet informatique (une "variable de variable" en somme).

1.1 Déclaration

La déclaration d'un pointeur se fait selon la syntaxe suivante :

```
type* id;
```

Cette instruction déclare une variable de nom id et de type pointeur(type) (pointeur sur une valeur de type type). Par exemple :

```
int* x;
```

déclare une variable x qui pointe sur une valeur de type int.

1.2 Adresse et valeur pointée

Les deux opérateurs particuliers en relation avec les pointeurs sont : & et *.

- & est l'opérateur qui **retourne l'adresse mémoire d'une variable**

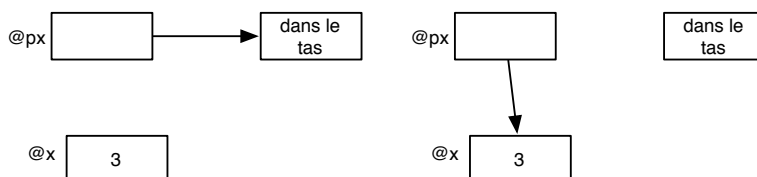
Si x est de type t alors &x est de type Pointeur(t)

Exemple :

```
int * px = (new int);
```

```
int x = 3;
```

```
px = &x;
```



- * est l'opérateur qui **retourne la valeur pointée par une variable pointeur**.

Si x est de type Pointeur (t) alors *x est de type t

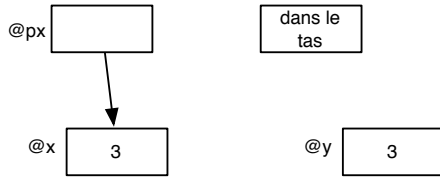
Exemple :

```
int * px = (new int);
```

```
int x = 3;
```

```
px = &x;
```

```
int y = *px;
```



1.3 Exercice

▷ **Exercice 1** Donner le type des variables dans le programme suivant :

```
int * x = (new int);
int * * y = & x;
int z = 4;
*x = z;
*y = &z;
**y = 6;
```

▷ **Exercice 2** Donner le type des variables dans le programme suivant :

```
pointeur{
    int * x = (new int);
    * x = 4;
    int z = 18;
    int *y = & z;
    *y = *x;
}
```

2 Modification de TAM et du compilateur

▷ **Exercice 3**

1. Quels nouveaux terminaux (tokens) faut-il ajouter ?
2. Quelle(s) règle(s) grammaire faut-il modifier ou ajouter pour introduire des pointeurs dans le langage ?
3. Quelles modifications faut-il apporter à l'AST ?
4. Modifier la passe de gestion des identifiants.
5. Modifier la passe de typage.
6. Modifier la passe de placement mémoire.
7. Proposer la traduction en TAM de l'exemple de l'exercice 2. On rappelle l'existence des instructions TAM suivantes, permettant de manipuler des adresses :
 - Empiler une adresse : `LOADA d[r]`
 - Empiler n mots à partir de l'adresse laissée en sommet de pile : `LOADI (n)`
 - Écrire n mots de la pile à l'adresse laissée en sommet de pile : `STOREI (n)`
 - Allocation de mémoire : `SUBR Malloc` (réserve dans le tas une zone de la taille laissée en sommet de pile, l'adresse obtenue est laissée en sommet de pile).
8. Modifier la passe de génération de code.