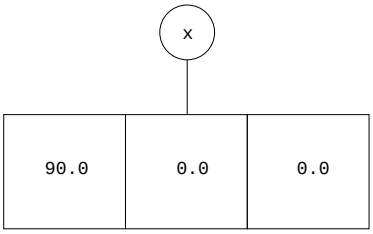
Une variable de type `Point` *ne peut jamais avoir la valeur* `null`*. Elle associe directement, dans l'environnement, la variable* `x` *à une référence* `r` *et, dans la mémoire, la référence* `r` *à l'enregistrement* `{latitude = 90.0, longitude = 0.0, altitude = 0.0}`*. Il y a donc une indirection de moins qu'en Java.*



4.4.3 L'accès aux champs

On peut ensuite accéder aux champs de l'enregistrement

```
printf("%f\n",x.latitude);
```

Si la valeur de l'expression `t` *est un enregistrement, la valeur de l'expression* `t.l` *est le champ* `l` *de cet enregistrement. En particulier, quand* `t` *est une variable* `x`*, sa valeur est* `m(e(x))` *et donc la valeur de l'expression* `x.latitude` *est le champ* `latitude` *de l'enregistrement* `m(e(x))` *et non* `m(m(e(x)))` *comme en Java.*

4.4.4 L'affectation des champs

On peut enfin affecter les champs d'un enregistrement

```
x.latitude = 48.715;
x.longitude = 2.208;
x.altitude = 156.0;
```

Quand on définit une fonction

```
void tropic (struct Point y) {
 y.latitude = 23.45;}
```

et l'appelle par l'instruction

```
tropic(x);
```

la valeur de `x` *n'est pas une référence associée dans la mémoire à un enregistrement comme cela l'était en Java et en Caml, mais l'enregistrement lui-même*