

Compilation

Génération de code

L'objectif de cette séance est de transformer du code écrit dans notre langage Toy en du code C normal.

Analyse statique

Pour commencer, nous allons utiliser le code pour générer des en-têtes qui seront utilisables par un programme C désirant invoquer des fonctions définies en Toy. Ainsi, l'extrait suivant définit deux fonctions, respectivement `f` et `g` :

```
function f : (x,y,a,b) ->
  when x < y -> a
  when x >=y -> b
end
```

```
function g : (x) ->
  print(f(x,1-x,0,1))
```

Pour la suite, nous supposons que les fonctions Toy n'admettent que des arguments de type `float` et renvoient toujours un résultat de type `float`. Pour faire interagir cet extrait avec un programme C, nous aurions besoin d'un fichier d'en-tête tel que :

```
#ifndef __TEST_TOY_H_INCLUDED__
#define __TEST_TOY_H_INCLUDED__

float f(float x, float y, float a, float b);
float g(float x);

#endif __TEST_TOY_H_INCLUDED__
```

Comment générer ce fichier d'en-tête ? Il suffit de

1. procéder à l'analyse syntaxique complète du fichier Toy
2. parcourir l'arbre syntaxique et regarder le nombre et le nom des arguments de chaque fonction.

Le point 1. est déjà géré par `compilateur.c`. À vous de vous occuper du point 2.

Exercice 1. Adaptez la fonction `generate_header()` du fichier `compilateur.c` pour qu'elle construise un fichier d'en-tête du même genre que celui ci-dessus.

Note Il s'agit d'une étape d'analyse statique simplifiée.

Génération de code

Espérons que la génération d'en-tête vous aura servi d'échauffement, puisqu'il s'agit maintenant de générer du vrai code C compilable vers du code machine à l'aide de GCC. Toujours à partir du même extrait, nous voudrions obtenir quelque chose du genre :

```
#include "test.toy.h"

float f(float x, float y, float a, float b)
{
  if(x<y)
```

```

    {
        return a;
    } else if (x>=y) {
        return b;
    }
}

float g(float x)
{
    return print(f(x,1.0000-x,0.0000,1.0000));
}

```

Exercice 2. Compléter la fonction `generate_code()` de `compilateur.c` de manière à ce qu'elle produise quelque chose de ce genre.

Une fois ceci fait, vous pouvez compiler le code Toy en langage C, que vous pouvez utiliser dans vos programmes C. Il vous manque uniquement la fonction `print`, qui n'est pas définie en C, et qui vous empêchera donc d'exécuter vos programmes. Cette fonction pourrait être ajoutée dans une bibliothèque partagée – comme c'est le cas pour les fonctions standard de C – ou dans un en-tête du code généré, c'est-à-dire dans le code produit par `generate_code()`.

Exercice 3. Débrouillez-vous pour que la fonction `generate_code()` produise, en plus de ce qui précède, une fonction `print()` d'affichage des nombres flottants. Le code C ainsi généré doit compiler.

```

/**
 * Affiche une valeur puis renvoie cette même valeur.
 *
 * @param value la valeur à afficher
 * @return value
 */
float print(float value)

```