DOCUMENTATION GLADOS

Louis-Henri Mc Allister
Thibault Castillon
Yuliia Maiorova
Pierre Tran
Petro Floresku

06/03/2023

SOMMAIRE

A PROPOS

Bienvenue dans la documentation de Yuyu, un langage de programmation basé sur Haskell.

INTRODUCTION

Yuyu est un langage de programmation qui a été créé en s'inspirant de Python et d'autres langages interprétés. Il a été conçu pour offrir une syntaxe concise et facile à comprendre, ainsi que des fonctionnalités modernes pour les programmeurs. Yuyu est conçu pour les programmeurs de tous niveaux, qu'ils soient débutants ou avancés.

TYPES DE DONNEES

Yuyu prend en charge différents types de données tels que les entiers, les flottants, les chaînes de caractères, les booléens, les listes et les tableaux. Il utilise une syntaxe similaire à celle de Python pour définir les types de données et les variables.

DEFINITION DE VARIABLES

Les variables dans Yuyu sont définies à l'aide du symbole "->". Par exemple, pour définir une variable "x" avec la valeur 5, il suffit d'écrire "x -> 2". Yuyu utilise un typage dynamique, ce qui signifie que vous n'avez pas besoin de déclarer explicitement le type de la variable.
Voici un exemple d'assignation de variable :

```
a -> 5
b -> 6
```

DEFINITION DE FONCTIONS

Yuyu prend en charge la définition de fonctions en utilisant la syntaxe standard de Haskell. Cependant, il n'est pas possible de redéfinir les fonctions builtins. Par exemple, la fonction "+" est une fonction builtin qui ne peut pas être redéfinie.
Voici un exemple de fonction "add" qui prend deux arguments et renvoie leur somme :

```
add x y -> [+ x y]
```

UTILISATION DES FONCTIONS

Les fonctions dans Yuyu sont appelées en utilisant une syntaxe similaire à celle de Haskell. Par exemple, pour appeler la fonction "add" avec les arguments 3 et 5, il suffit d'écrire "add 3 5".
Voici un exemple d'appel de fonction :

```
test a b -> [- [add a b] 1]
```

Yuyu prend également en charge les fonctions récursives.
Voici un exemple de fonction récursive :

```
fact n -> [Siii [== n 1] 1 [* n [fact [- n 1]]]]
```

CONDITIONS

Yuyu prend en charge l'utilisation de conditions à l'aide de la syntaxe standard de Haskell. Voici un exemple de code qui utilise une condition "if…else" dans une fonction qui vérifie si "a" et "b" sont égaux, si oui, on les rajoute, sinon on les soustrait :

```
test a b -> [Siii [== a b] [add a b] [sub a b]]
```

Une autre condition que Yuyu prend en charge est le "case…of ". La condition "case…of " est interprété par "Prrr…of".
Dans cet exemple la condition "Prrr…of" dans une fonction vérifie si le "symbol" correspond à l'argument "a" ou "b" et renvoi soit la somme de "a" et "b" soit leur différence et s'il n'y a ni de "a" ni de "b" ça renvoi un otherwise (par exemple "De fou hein").

```
test a b -> Prrr symbol of [
     "a" -> [add a b]
     "b" -> [sub a b]
     _ -> "De fou hein"
]
```

MESSAGES D'ERREUR

GLaDOS contient également un système de messages d'erreurs.
Voici un exemple de message d'erreur que l'utilisateur verra lors d'une erreur.
Dans ce cas l'erreur est le message "Holala quel horreur".

```
test a b -> Prrr symbol of [
     1 -> [add a b]
     2 -> "Holala quel horreur"
     _ -> "De fou hein"
]
```

BIBLIOTHEQUE STANDARD

Yuyu dispose d'une bibliothèque standard limitée qui comprend les fonctions et les types de données de base. Il n'est pas possible de redéfinir les fonctions builtins dans la bibliothèque standard.
Les fonctions de base sont les suivantes :
"+", "-", "/", "*" et "%" (addition, soustraction, division, multiplication et modulo).
Les fonctions de comparaison :
"==", "!=", "<", ">", "<=", ">=" (égalité, différence, inférieur, supérieur, inférieur ou égal, supérieur ou égal).
Nous avons également la fonction "display" qui permet d'afficher un caractère à l'écran.
Voici un exemple d'utilisation de la fonction "display" :

```
display "a"
```

GRAMMAIRE DU LANGAGE

Yuyu utilise une grammaire BNF pour définir sa syntaxe.
Ce qui suit ne fait pas partie de langage, mais est utilisé pour faire la grammaire plus lisible. Ils sont utilisés pour indiquer que le token est un mot clé et non un identifiant.
La grammaire de Yuyu est la suivante :

```
<glados>      ::= <stmt_list>

<stmt_list>   ::= <stmt> <stmt_list>
            | <stmt>

<stmt>        ::= <var_assign>
            | <func_decl>
            | <func_call>
            | <if_expr>
            | <case_of>
            | <expr>

<var_assign>  ::= <ident> "->" <expr>

<func_decl>   ::= <ident> <params> "->" <expr>

<params>      ::= <param_list>
            | ""

<param_list>  ::= <ident> " " <param_list>
            | <ident>

<func_call>   ::= <ident> <func_args>

<func_args>   ::= <arg_list>
            | ""

<arg_list>  ::= <expr> " " <arg_list>
            | <expr>

<if_expr>   ::= "Siii" <expr> <expr> <expr>

<expr>      ::= <term add_op> <expr>
```

```
            | <term>

<term>      ::= <factor <mult_op term>
            | <factor>

<factor>    ::= <int_lit>
            | <float_lit>
            | <string_lit>
            | <bool_lit>
            | <ident>
            | <expr>
            | <func_call>
            | <if_expr>
            | <case_of>

<add_op>    ::= "+"
            | "-"

<mult_op>   ::= "*"
            | "/"
            | "mod"

<comp_op>   ::= "=="
            | "!="
            | "<"
            | "<="
            | ">"
            | ">="

<int_lit>   ::= <digit>+

<float_lit> ::= <digit>+ "." <digit>+

<ident>     ::= <alpha> <alphanum>*

<alpha>     ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

<alphanum>  ::= <alpha> | <digit>

<digit>     ::= "0" | "1" | ... | "9"
```

Cette BNF décrit la grammaire de notre langage en utilisant des symboles non-terminaux tels que glados, stmt_list, stmt, var_assign, func_decl, params, param_list, func_call, func_args, arg_list, if_expr, expr, term, factor, add_op, mult_op, int_lit, float_lit, display, et ident.
Les symboles terminaux sont les éléments concrets du langage, tels que les opérateurs, les littéraux de chaînes de caractères et les mots-clés.
En utilisant cette BNF, vous pouvez construire des programmes valides en suivant les règles de la grammaire.

CONCLUSION

Yuyu est un langage de programmation simple et facile à utiliser qui prend en charge les fonctionnalités de base de Haskell et du Python. Il est conçu pour les programmeurs de tous niveaux et peut être utilisé pour développer une variété d'applications. Avec sa syntaxe concise et facile à comprendre, Yuyu peut aider les programmeurs à écrire du code de manière efficace et productive.