INF404 projet : Explication de l'interpréteur :

Marius Mahaie & Pierre Médina :

Fonctionnalités de notre interpréteur :

Langage compris par notre interpréteur :

Opérations Arithmétiques

- Addition (+)
- Soustraction (-)
- Multiplication (*)
- Division (/)

& Opérations de comparaisons

- Égalité (==)
- Inégalité (!=)
- Inférieur à (<)
- Inférieur ou égal à (<=)
- Supérieur à (>)
- Supérieur ou égal à (>=)

Identificateurs et Valeurs :

- N_IDF (représente des variables)
- VALEUR (représente des entiers...)

Instructions :

- N_AFF (Affectations : affecter une valeur à une variable)
- N_LIRE (Lecture : lis une valeur entrée par un utilisateur)
- N_ECRIRE (Ecriture : affiche la valeur d'une expression)
- N_IF (Condition : if, then, else et pour finir fi)
- N_WHILE (Condition : tanque, faire et pour finit fait)

Ainsi que des Séquences d'Instructions :

- N_SPEINST (pour séparer plusieurs instructions par des points-virgules)

Notre interpréteur en quelques mots :

Interprète et affiche un arbre abstrait, construit à l'aide de l'analyse lexicale et syntaxique d'un texte. Notre interpréteur se sert également de table symbole qui stocke les identificateurs (ici des variables).

Exemple d'utilisation de notre interpréteur :

entree48.txt : if then else et fi :

```
x = 1;
y = 2;
if x > y then
w = 1
else
fi
```

Résultat :

```
Analyse du fichier : entree/entree48.txt
Table des symboles initialisée.
```

Analyse syntaxique et construction de l'AST terminées avec succès.

Programme analysé avec succès. AST construit.

--- Lancement Interpréteur ---
DEBUG: Symbole 'x' ajouté avec valeur 1
INTERP: x <- 1
DEBUG: Symbole 'y' ajouté avec valeur 2
INTERP: y <- 2
INTERP: Condition IF fausse, exécution ELSE
INTERP: (Pas de branche ELSE à exécuter)
INTERP: Fin bloc IF
--- Fin Interpréteur ---

État final de la table des symboles :

--- Table des Symboles ---
  x = 1
  y = 2
----

entree49.txt : tanque faire et fait :

```
x = 1;
y = 5;
tanque x < y faire
    x = x + 1
fait
```

Résultat :
Analyse du fichier : entree/entree49.txt
Table des symboles initialisée.

Analyse syntaxique et construction de l'AST terminées avec succès.

Programme analysé avec succès. AST construit.

--- Lancement Interpréteur ---
DEBUG: Symbole 'x' ajouté avec valeur 1
INTERP: x <- 1
DEBUG: Symbole 'y' ajouté avec valeur 5
INTERP: y <- 5
INTERP: Entrée boucle TANQUE
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'x' mis à jour à 2
INTERP: x <- 2
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'x' mis à jour à 3
INTERP: x <- 3
INTERP: Condition TANQUE vraie, exécution du corps

```
DEBUG: Symbole 'x' mis à jour à 4
INTERP: x <- 4
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'x' mis à jour à 5
INTERP: x <- 5
INTERP: Condition TANQUE fausse, sortie de boucle
--- Fin Interpréteur ---
```

État final de la table des symboles :

```
--- Table des Symboles ---
  x = 5
  y = 5
-------------------------
```

entree50.txt : interpréteur avec l'entièreté de son langage :

```
a = 1;
b = 0;
c = 5;
tanque c > 0 faire
    a = a * 2;
    if a > 10 then
        b = b - 1;
    else
        b = b + a;
    fi;
    c = c - 1;
    ecrire(a);
    ecrire(b);
    ecrire(c);
fait;
ecrire(a);
ecrire(b);
```

Résultat :

```
Analyse du fichier : entree/entree50.txt
Table des symboles initialisée.

Analyse syntaxique et construction de l'AST terminées avec succès.

Programme analysé avec succès. AST construit.

--- Lancement Interpréteur ---
DEBUG: Symbole 'a' ajouté avec valeur 1
INTERP: a <- 1
DEBUG: Symbole 'b' ajouté avec valeur 0
INTERP: b <- 0
DEBUG: Symbole 'c' ajouté avec valeur 5
INTERP: c <- 5
```

INTERP: Entrée boucle TANQUE
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'a' mis à jour à 2
INTERP: a <- 2
INTERP: Condition IF fausse, exécution ELSE
DEBUG: Symbole 'b' mis à jour à 2
INTERP: b <- 2
INTERP: Fin bloc IF
DEBUG: Symbole 'c' mis à jour à 4
INTERP: c <- 4
2
2
4
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'a' mis à jour à 4
INTERP: a <- 4
INTERP: Condition IF fausse, exécution ELSE
DEBUG: Symbole 'b' mis à jour à 6
INTERP: b <- 6
INTERP: Fin bloc IF
DEBUG: Symbole 'c' mis à jour à 3
INTERP: c <- 3
4
6
3
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'a' mis à jour à 8
INTERP: a <- 8
INTERP: Condition IF fausse, exécution ELSE
DEBUG: Symbole 'b' mis à jour à 14
INTERP: b <- 14
INTERP: Fin bloc IF
DEBUG: Symbole 'c' mis à jour à 2
INTERP: c <- 2
8
14
2
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'a' mis à jour à 16
INTERP: a <- 16
INTERP: Condition IF vraie, exécution THEN
DEBUG: Symbole 'b' mis à jour à 13
INTERP: b <- 13
INTERP: Fin bloc IF
DEBUG: Symbole 'c' mis à jour à 1
INTERP: c <- 1
16
13
1
INTERP: Condition TANQUE vraie, exécution du corps
DEBUG: Symbole 'a' mis à jour à 32
INTERP: a <- 32

```
INTERP: Condition IF vraie, exécution THEN
DEBUG: Symbole 'b' mis à jour à 12
INTERP: b <- 12
INTERP: Fin bloc IF
DEBUG: Symbole 'c' mis à jour à 0
INTERP: c <- 0
32
12
0
INTERP: Condition TANQUE fausse, sortie de boucle
32
12
--- Fin Interpréteur ---
```

État final de la table des symboles :

```
--- Table des Symboles ---
  a = 32
  b = 12
  c = 0
--------------------------
```

entree_tout_5 : mini-jeu :

```
position = 0;
but = 10;
pas = 0;
obstacle = 7;
bonus = 3;

ecrire(position);
tanque position < but faire
    lire(pas);
    if pas <= 0 then
        ecrire(99);
    else
        position = position + pas;
        ecrire(position);
        if position == obstacle then
            ecrire(88);
            position = position - 2;
            ecrire(position);
        fi;
        if position == bonus then
             ecrire(77);
             position = position + 1;
             ecrire(position);
        fi;
        if position < 0 then
           position = 0;
        fi;
```

```
    fi;
    ecrire(position);
fait;
ecrire(111);
```

Résultat :

```
Analyse du fichier : entree/entree_tout_5.txt
Table des symboles initialisée.

Analyse syntaxique et construction de l'AST terminées avec succès.

Programme analysé avec succès. AST construit.

--- Lancement Interpréteur ---
DEBUG: Symbole 'position' ajouté avec valeur 0
INTERP: position <- 0
DEBUG: Symbole 'but' ajouté avec valeur 10
INTERP: but <- 10
DEBUG: Symbole 'pas' ajouté avec valeur 0
INTERP: pas <- 0
DEBUG: Symbole 'obstacle' ajouté avec valeur 7
INTERP: obstacle <- 7
DEBUG: Symbole 'bonus' ajouté avec valeur 3
INTERP: bonus <- 3
0
INTERP: Entrée boucle TANQUE
INTERP: Condition TANQUE vraie, exécution du corps
Entrez une valeur entière pour pas : 10                 // L'utilisateur entre une valeur pour pas : ici: 10
DEBUG: Symbole 'pas' mis à jour à 10
INTERP: lire(pas) -> 10
INTERP: Condition IF fausse, exécution ELSE
DEBUG: Symbole 'position' mis à jour à 10
INTERP: position <- 10
10
INTERP: Condition IF fausse, exécution ELSE
INTERP: (Pas de branche ELSE à exécuter)
INTERP: Fin bloc IF
INTERP: Condition IF fausse, exécution ELSE
INTERP: (Pas de branche ELSE à exécuter)
INTERP: Fin bloc IF
INTERP: Condition IF fausse, exécution ELSE
INTERP: (Pas de branche ELSE à exécuter)
INTERP: Fin bloc IF
INTERP: Fin bloc IF
10
INTERP: Condition TANQUE fausse, sortie de boucle
111
--- Fin Interpréteur ---

État final de la table des symboles :

--- Table des Symboles ---
```

position = 10
but = 10
pas = 10
obstacle = 7
bonus = 3

Explication

Ce programme simule un déplacement vers un objectif (but) en partant d'une position initiale (position). Il y a des bonus, mais aussi des obstacles déclenché en fonction de la position, il affiche un code de fin (111), si l’objectif est atteint ou dépasser.