Le projet

Site:	Université de Reims Champagne-Ardenne	Imprimé par:	QUENTIN JUILLIARD
Cours:	Langages et compilation	Date:	dimanche 12 mars 2023, 21:45
Livre:	Le projet		

Table des matières

1. Généralités

Le but du projet est de créer un générateur de mondes pour un jeu de plateformes multi-joueurs en réseau. Le jeu correspond au projet des matières INFO0601 et INFO0604, mais ce projet est indépendant.

Un monde est représenté dans un langage spécifique comportant des éléments classiques : des instructions simples (affectation, appel de fonctions/procédures) et des instructions structurées (conditionnelle, boucle TantQue et Pour). Il est possible de définir des variables (de type entier) locales aux fonctions/procédures ou globales, ainsi que des fonctions et des procédures.

Le générateur que vous devez écrire n'est pas un compilateur : il génère les niveaux en interprétant le langage.

2. Description d'un monde

Un monde est constitué de plusieurs niveaux et il est possible de passer d'un niveau à un autre via des portes. L'objectif est de partir de la porte d'entrée et d'atteindre la porte de sortie. Chaque niveau est constitué de 60 cases de large et de 20 de haut. Les cases possibles sont les suivantes :

- Des blocs (*block*)
- Des pièges (*trap*)
- Des vies (*life*)
- Des bombes (*bomb*)
- Des échelles (*ladder*)
- Une porte de début (*start*)
- Une porte de sortie (*exit*)
- Des portes numérotées de 1 à 99 (*door*)
- Des portails (*gate*) numérotés de 1 à 4
- Des clés (*key*) numérotées de 1 à 4 pour ouvrir les portails correspondant
- Des sondes (*probe*)
- Des robots (*robot*)

Vous pouvez récupérer ce code : level. Il permet de représenter un niveau (`level_t`) et le programme propose 3 niveaux de démonstration. Vous n'êtes pas obligés de l'utiliser dans votre projet.

3. Le langage

Constantes et fonctions/procédures prédéfinies

Des constantes permettent de définir des types de blocs constituant les niveaux :

- EMPTY, BLOCK, TRAP, LIFE, BOMB
- DOOR(X) où X est un numéro compris entre 1 et 99
- ENTER et EXIT
- LADDER
- ROBOT, PROBE
- KEY(X) et GATE(X) où X est un numéro compris entre 1 et 4

Les constantes `TRUE` et `FALSE` peuvent être utilisées pour les booléens.

Il existe également les fonctions/procédures suivantes :

- put(x,y,z) : place un bloc à la position (x,y), le type de bloc est spécifié par z
- get(x,y) : retourne le type de bloc situé à la position (x,y)

Les positions sont comprises dans l'intervalle [0; 59] pour x et [0; 19] pour y.

Les instructions

Une affectation utilise le symbole `=`. Elle permet de créer une variable, de l'initialiser ou de modifier sa valeur. Elle est locale à une fonction/procédure, sauf si elle est déclarée en dehors d'une fonction/procédure. Par exemple :

```
x = 1 + 2
```

Les opérateurs classiques sont possibles : "+", "-", "/", "*". Vous pouvez prendre en compte des opérateurs supplémentaires ("++", "+=", etc.). N'oubliez pas de le préciser dans votre rapport.

Pour appeler une procédure, il suffit de spécifier son nom et ses paramètres entre parenthèses. Nous verrons un exemple dans la section suivante.

Il est possible d'utiliser une conditionnelle avec ou sans clause sinon :

```
if(x <= 1 OR y <> 3)
  z = 2
else
  z=3
end
```

La conditionnelle s'écrit avec les opérateurs classiques : AND, OR, NOT, "=", "<>", "<", "<=", ">", ">=". De même, vous pouvez ajouter d'autres opérateurs.

Le langage reconnaît la boucle `while`. Par exemple :

```
while(i < 10)
  i++;
end
```

Le langage reconnaît également la boucle `for`. Les paramètres sont séparés par des points-virgules. Le premier correspond à l'initialisation, le second à la clause de continuité, la troisième à l'incrémentation.

```
for(i=1;i<10;i=i+1)
  put(i,1,BLOCK)
end
```

Procédures, fonctions et niveau

Une procédure débute par le mot clé `prc`, suivie d'un nom et d'une liste de paramètres entre parenthèses. Voici un exemple (cela crée une échelle à une position et d'une hauteur données) :

```
prc ladder(x,y,h)
  for(i=y;i<y+height-1;i=i+1)
    put(x,i,LADDER)
  end
end
```

Pour appeler cette procédure, on peut utiliser l'instruction suivante :

```
ladder(1,1,10)
```

Une fonction débute le mot-clé `fct`. Elle retourne un entier grâce au mot clé `return`.

```
fct hauteur(x,y)
  h = 0
  while(y + h < HEIGHT AND get(x,y + h) == EMPTY)
    h = h + 1
  end
  return h
end
```

Pour l'appeler, on peut utiliser l'instruction suivante :

```
h = hauteur(10, 10)
```

Un nouveau niveau est créé à chaque fois que le code contient la structure suivante (en dehors des fonctions/procédures) :

```
level
  // Instructions
end
```

4. Instructions

Vous devez écrire un programme utilisant *Lex* et *Yacc* pour analyser un fichier de description de monde et générer un fichier correspond au monde (le fichier de sortie peut être un fichier texte, en vous basant du code fourni, ou un fichier binaire que vous pouvez lire avec l'éditeur développé pour le projet INFO0601 / INFO0604).

L'analyse du fichier doit être réalisée en une seule passe. Le code doit être interprété : vous ne devez pas passer par un langage intermédiaire. Les fonctions/procédures doivent être placées en mémoire pour être exécutées à plusieurs reprises.

Vous devez produire un code qui puisse être compilé sur la VM fournie, ainsi que des fichiers d'exemple permettant d'illustrer TOUTES les possibilités de votre langage. Le rapport doit expliquer TOUTES les structures utilisées, ainsi que les règles de la grammaire (expliquez l'intérêt de chaque règle). Attention : vous ne devez pas faire de simples copier/coller de code, vous devez tout décrire.

Le projet est à déposer sur Moodle au plus tard le dimanche 2 avril à 23h30 (retard jusqu'à 23h59 accepté). Si vous dépassez ce délai ou si vous ne déposez pas le code sur Moodle, le projet ne sera pas pris en compte. Pour ces raisons, n'attendez pas le dernier moment pour le déposer.