UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0602 – 2019/2020
Resp. : **Cyril Rabat**

Projet

Robot et billes

Le projet suivant est à réaliser en binôme. Le code (écrit en C et en `lex&yacc`*) devra être compilable à l'aide d'un makefile. Un fichier lisezMoi.txt sera présent pour indiquer l'utilisation du programme. Un rapport est attendu pour expliquer la grammaire, les structures utilisées ainsi que les algorithmes mis en œuvre. L'ensemble sera remis au plus tard le vendredi 10 avril 2020, 20h00,* ***obligatoirement sur Moodle****.*

L'objectif de ce projet est de réaliser un programme qui déplace un robot sur un plateau en fonction d'un algorithme, à l'aide de `ncurses`. Le plateau est décomposé en cases et possède des dimensions variables. Chacune case peut comporter soit du vide (rien), une caisse (un obstacle qui peut être déplacé), un obstacle (qui ne peut être déplacé), une bille ou un trou. Le robot est capable de se déplacer sur les cases vides ou les trous comblés par une bille. Il peut ramasser une bille qui se trouve devant lui ou déposer celle qu'il a ramassé dans un trou ou sur une case vide. S'il se déplace sur une case contenant une caisse ou une bille, l'objet est déplacé d'une case, sauf si un autre objet se trouve derrière (il n'est pas possible de déplacer deux caisses, par exemple).

Le programme s'arrête lorsque tous les trous ont été remplis (même si tout l'algorithme n'est pas terminé), s'il n'y a plus d'actions à effectuer ou si une action ne peut être réalisée (déplacement sur un obstacle ou en dehors des dimensions, ramassage d'une bille inexistante, etc.).

Le programme prend en arguments deux fichiers différents qui doivent être analysés par *Lex&Yacc*. Le premier fichier est une description du plateau et le second correspond à l'algorithme que le robot va exécuter.

La description de la grille

Le plateau est définie dans un fichier au format JSON : les données ne sont pas dans un ordre prédéfini, seule la structure est imposée. Les champs `largeur` et `hauteur` représentent la largeur et la hauteur du plateau. Le champ `debut` définit où se trouve le robot : la position (x, y) (la première case en haut à gauche est $(0, 0)$) et la direction (`BAS`, `HAUT`, `DROITE`, `GAUCHE`). Le champ `cases` décrit le contenu des cases. Chacune est définie par sa position et son type (`TROU`, `BILLE`, `BOITE` et `BLOC`). La cohérence du plateau doit être vérifiée (pas d'objet au même endroit, le nombre de billes doit être supérieur ou égal au nombre de trous, les positions doivent être correctes, *etc.*). Un exemple de fichier est présenté en annexes.

L'algorithme du robot

L'algorithme exécuté par le robot est contenu dans un second fichier. Il comporte des fonctions ou des procédures et au moins une procédure nommée `main`. Elles sont définies comme suit :

```
proc exemple(x : entier, y : bool)
  ...
finproc
```

```
fonc exemple(x : entier, y : bool) : entier
  ...
  retourner 2
finfonc
```

Les instructions sont de 4 types :

- Les affectations
- Les appels à des procédures
- Les boucles `tantque(`*exp*`)` *instructions* `fintantque` avec `exp` une expression booléenne

- Les conditionnelles `si(`*exp*`)` *instructions* `finsi` ou bien `si(`*exp*`)` *instructions* `sinon` *instructions* `finsi` avec `exp` une expression booléenne

Le langage accepte des procédures prédéfinies :

- avance(), recule() : avance ou fait reculer le robot
- droite(), gauche() : change l'orientation du robot
- pose(), prend() : pose la bille ou prend la bille située sur la case devant le robot
- case(x) : indique le contenu de la case où x est HAUT, BAS, DROITE ou GAUCHE ; retourne une constante correspondant au type de la case située en haut, en bas, à droite ou à gauche (en fonction de x)

À tout moment, il est possible d'accéder aux variables suivantes :

- x et y : la position du robot
- direction : la direction du robot

Les constantes suivantes sont également accessibles :

- TROU, CAISSE, BOITE, BILLE : le type de case
- HAUT, BAS, DROITE, GAUCHE : une direction
- LARGEUR et HAUTEUR : la largeur et la hauteur de la grille
- VRAI et FAUX : des valeurs booléennes

Un exemple de fichier est présenté en annexés.

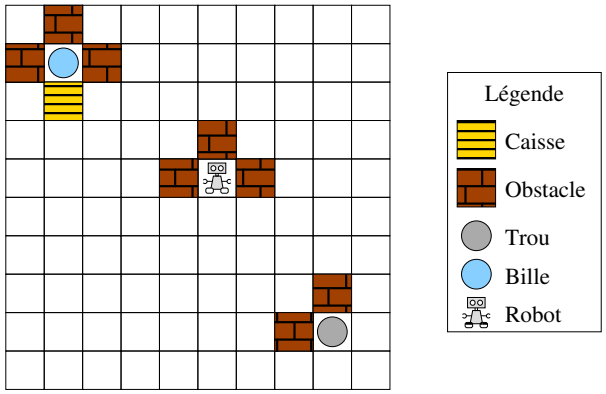
Consignes

Le programme doit pouvoir compiler et s'exécuter dans les conditions des salles de TP. Vous devez déposer sur *Moodle* une archive contenant votre projet (avec un `makefile` et un fichier `readMe.txt`). Des fichiers d'exemple seront mis à disposition : ils serviront à évaluer le projet.

Vous rendrez un rapport (au format PDF) expliquant les choix d'implémentation réalisés. Il expliquera notamment les choix de développement dans l'analyseur lexical et l'analyseur syntaxique. Vous présenterez la grammaire (il est conseillé de l'expliquer partie par partie) ainsi que les structures utilisées.

Annexes

```
{
  "largeur" : 10, "hauteur" : 10,
  "debut" : [
      "x" : 5, "y" : 5,
      "direction" : "BAS"
    ],
  "cases" :
    [
      {
        "x" : 8, "y" : 8,
        "type" : "TROU"
      },
      {
        "x" : 1, "y" : 1,
        "type" : "BILLE"
      },
      {
        "x" : 1, "y" : 0,
        "type" : "BLOC"
      },
      {
        "x" : 0, "y" : 1,
        "type" : "BLOC"
      },
      {
        "x" : 2, "y" : 1,
        "type" : "BLOC"
      },
      {
        "x" : 1, "y" : 2,
        "type" : "CAISSE"
      },
      ...
    ]
}
```



Ce programme permet de résoudre la grille précédente :

```
proc avancer(x : entier)
  i = 0
  tantque(i < x)
    avance()
    i = i + 1
  fintantque
finproc

proc main()
  avance()
  droite()
  avancer(3)
  droite()
  avancer(3)
  gauche()
  avance()
  droite()
  prendre()
  droite()
  droite()
  avance(7)
  gauche()
  avance(7)
  gauche()
  poser()
finproc
```