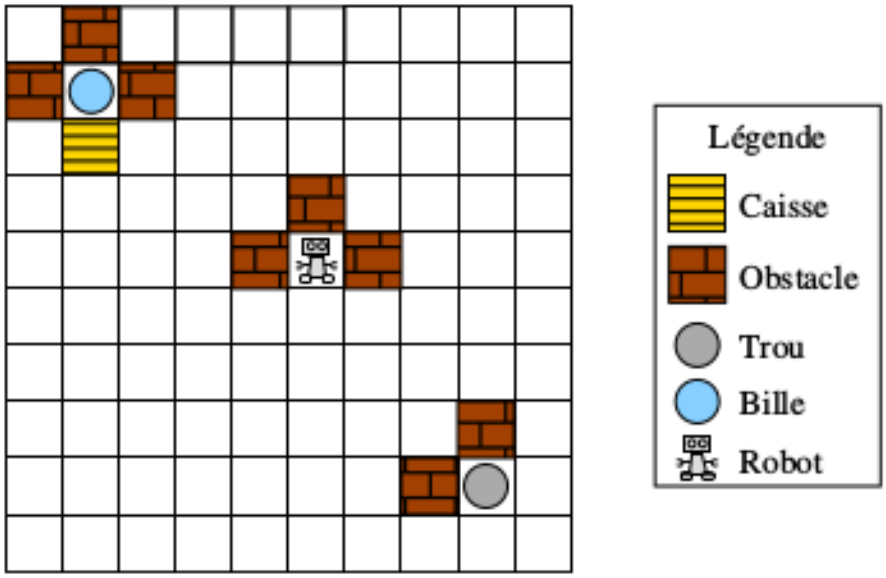
Info0602: Langages et compilation
Responsable:Cyril RABAT

Rapport de projet: Robot et billes



Sommaire:

WARNET Mina
LOISEAU—PINTAUX Antinéa

Grammaire:

ensemble des symboles terminaux
T = { programme, FINP }

ensemble des symboles non terminaux
N = { ENTIER, DEBUTP, MAIN, NOM, PAR, FIN, VAR, WHILE, FINW, IF, ENDI, ELSE, AV, RE, DR, GA, PO, PR, CA, EGAL, DIFF, TESTEG, INF, INFE, SUP, SUPE, ‘,’, ‘:’, ‘+’, ‘-’, parametres, parametre, actions, main, instruction, instructions, affectation, procedure, exp, proc }

ensemble des règles
R = {
programme → main | proc.programme,
main → DEBUTP.MAIN.actions.FINP
proc → DEBUTP.NOM.PAR.parametres.FIN.instructions.FINP
parametres → parametres, parametre | parametre,
parametre → VAR : NOM | ε,
instructions → instructions.instruction | instruction,
instruction → affectation | procedure | WHILE.exp.FIN.instructions.FINW,
instruction → IF.exp.FIN.instructions.ENDI | IF.exp.FIN.instructions.ELSE.instructions.ENDI,
affectation → VAR.EGAL.ENTIER | VAR.EGAL.VAR+ENTIER | VAR.EGAL.ENTIER+VAR,
affectation → VAR.EGAL.VAR-ENTIER | VAR.EGAL.ENTIER-VAR,
exp → VAR.DIFF.VAR | VAR.DIFF.ENTIER | VAR.TESTEG.VAR | VAR.TESTEG.ENTIER | VAR.INF.VAR,
exp → VAR.INF.ENTIER | VAR.INFE.VAR | VAR.INFE.ENTIER | VAR.SUP.VAR | VAR.SUP.ENTIER,
exp → VAR.SUPE.VAR | VAR.SUPE.ENTIER
actions → actions.procedure | procedure,
procedure → AV | RE | DR | GA | PO | PR | CA.ENTIER.FIN
}

symbole de départ
S = programme

Structures:

case_t: c’est une case de la grille sur laquelle se déplace le robot, la structure possède trois attributs:
- int x: le numéro de ligne de la case dans la grille;
- int y: le numéro de colonne de la case dans la grille;
- int type: le type d’objet que contient la case.

robot_t: une structure qui représente le robot que l’on veut déplacer sur la grille, la structure possède quatre attributs:
- int x: le numéro de ligne de la case dans la grille;
- int y: le numéro de colonne de la case dans la grille;
- int dir: la direction vers laquelle le robot est tourné (constantes définies)
- int porte: VIDE (0) le robot ne porte rien,
  CAISSE (1) le robot porte une caisse
  BILLE (3) le robot porte une BILLE

Algorithmes:

algo avance()
Init:
    x, y: entiers (coordonnées de la case ciblée)
Début
    si la case devant le robot est vide alors
        il se déplace sur cette case
    sinon si la case contient une caisse et rien derière
        la caisse est deplacée sur la case de derière et le robot se déplace sur la case
    sinon si la case contient une bille et rien derière
        la caisse est deplacée sur la case de derière et le robot se déplace sur la case
    sinon
        retourner FAUX
    fin si
    retourner VRAI
Fin

algo recule()
Init:
    x, y: entiers (coordonnées de la case ciblée)
Début
    si la case deriére le robot est vide alors
        il se déplace sur cette case
    sinon si la case contient une caisse et rien derière
        la caisse est deplacée sur la case de derière et le robot se déplace sur la case
    sinon si la case contient une bille et rien derière
        la caisse est deplacée sur la case de derière et le robot se déplace sur la case
    sinon
        retourner FAUX
    fin si
    retourner VRAI
Fin

algo droite()
Init:
Début
    la direction du robot est changée de façon à ce qu'il face un quart de tour vers la droite
Fin

algo gauche()
Init:
Début
    la direction du robot est changée de façon à ce qu'il face un quart de tour vers la gauche
Fin

algo pose()
Init:
    x, y: entiers (coordonnées de la case ciblée)
Début
    si la case devant le robot est vide et que le robot porte un objet alors
        il place l'objet sur cette case

```
            retourner VRAI
        sinon si la case devant le robot est un trou et qu'il porte une bille
            il place la bille dans le trou
            retourner BOUCHE
        sinon
            retourner FAUX
        fin si
Fin
```

algo prend()

```
Init:
        x, y: entiers (coordonnées de la case ciblée)
Début
        si la case devant le robot contient un objet déplaçable et que le robot ne porte rien alors
            il prend l'objet
            retourner VRAI
        sinon
            retourner FAUX
        fin si
Fin
```

correspond à la procédure case(x: direction) mais à cause de 'switch case' on ne peut pas garder le nom 'case' pour la fonction.

algo contenu(dir : entier)

```
Init:
        x, y: entiers (coordonnées de la case ciblée)
        type: entier (contenu de la case)
Début
        si dir = HAUT alors
            type = type de la case devant le robot
            retourner type
        sinon si dir = DROITE alors
            type = type de la case à droite du robot
            retourner type
        sinon si dir = BAS alors
            type = type de la case derière le robot
            retourner type
        sinon si dir = DROITE alors
            type = type de la case à gauche du robot
            retourner type
        fin si
Fin
```