Manual de Utilizador - SQUEAK

Ariana Gonçalves((89194) ariana@ua.pt)
Dário Alves((67591) dario.alv@ua.pt)
Diogo Correia((90327) diogo@ua.pt)
João Gonçalves ((80179) joao@ua.pt)
Mariana Pinto((84792) mariana17@ua.pt)

Data de Entrega: 7 de julho de 2019

Neste manual de instruções será explicado como o programador pode utilizar a linguagem desenvolvida para a criação de programas com o resultado pretendido.

NOTA: Para mais informações consultar o relatório do trabalho

TOKENS DISPONÍVEIS

- **START** - dá início ao programa e à main, indica o nome do robot e a sua posição inicial: *START "ROBOT_NAME"position 0*
- **WRITE** - semelhante ao print de java/C, é a função de saída de texto, em que o output é que aparece na consola: *WRITE(output)*
- **END** - identifica o final da main: *END*
- **PICKUP** - recolhe um objeto(por exemplo, um queijo) *PICKUP*
- **RETURN** - retorna ao ponto inicial/partida: *RETURN*
- **GROUND** - retorna o valor inteiro do tipo de chão no qual o robot se encontra, 0 se for chão e 1 até n se for um dos n queijos: *GROUND*
- **WHEELS** - define os valores da velocidade das rodas da esquerda e da direita: *WHEELS(x y)*
- **FINISH** - : envia um sinal em como o robot chegou ao fim do programa: *FINISH*
- **FUNCTION** - idenfica o ínicio de uma função x, com valor de entrada input e retorna um output: *FUNCTION x(input)=>output*
- **ENDFUNCTION** - identica o fim de uma função: *ENDFUNCTION*

- **LOOP** - identifica o incio de um ciclo iterativo/condicional em que uma linha se realiza se a condição se verificar, caso contrário passa para a próxima linha: *LOOP*
- **ENDLOOP** - identifica o fim de um ciclo iterativo/condicional: *ENDLOOP*
- **ACTIONS** - identifica o início de um ciclo condicional em que quando uma condição for verdadeira as restantes são descartadas : *ACTION*
- **ENDACTIONS** - identifica o fim de um ciclo condicional : *ENDACTION*
- **FRONTSENSOR** - devolve o valor da distância do obstáculo mais próximo do sensor da frente do robot: *FRONTSENSOR*
- **BACKSENSOR** - devolve o valor da distância do obstáculo mais próximo do sensor de trás do robot: *BACKSENSOR*
- **LEFTSENSOR** - devolve o valor da distância do obstáculo mais próximo do sensor da esquerda do robot: *LEFTSENSOR*
- **RIGHTSENSOR** - devolve o valor da distância do obstáculo mais próximo do sensor da direita do robot: *RIGHTSENSOR*
- **CHEESEANGLE** - devolve o ângulo entre o robot e o queijo identificado: *CHEESEANGLE 1*
- **COMPASS** - devolve o ângulo do robot em relação ao norte: *COMPASS*
- **STOP** - pára o robot, equivalente a WHEELS(0 0): *STOP*
- **FOWARD** - faz o robot andar para a frente com input velocidade nas rodas: *FOWARD (input)*
- **TURNLEFT** - faz o robot andar para a esquerda com input velocidade nas rodas: *TURNLEFT (input)*
- **TURNRIGHT** - faz o robot andar input valores para a direita: *TURNRIGHT (input)*
- **BACKWARD** - faz o robot andar input valores para trás: *BACKWARD (input)*
- **STARTANGLE** - retorna o ângulo que o robot faz com a posição inicial: *STARTANGLE*
- **STARTDISTANCE** - retorna a distância ao ponto incial/partida: *STARTDISTANCE*

START E END

O START e o END servem para inicializar o programa, sendo obrigatório que a primeira linha deste seja START + STRING que define o nome do robot em uso.

É possível adicionar a posição inicial do robot a partir da palavra POSITION seguida da medida preferível. Caso não seja usado o token POSITION, o compilador entenderá que começa da posição 0.

```
START "robot"POSITION 3
    (...)
END
```

FUNCTIONS

As funções são inicializadas com o token FUNCTION e terminadas com o token ENDFUNCTION. O name será usado no bloco START/END para chamar as mesmas funções para serem usadas pelos robots. O input é o parâmetro de entrada e o output o que a função retorna.

```
FUNCTION name(input)=>output
    (...)
ENDFUCTION
```

Existem quatro tipos de funções:

1. Retorna valor y e tem de parâmetros de entrada x:

   ```
   FUNCTION xpto(x)=>y
   ```

2. Não retorna valores e tem de parâmetros de entrada x:

   ```
   FUNCTION xpto(x)
   ```

3. Retorna valor y e não tem parâmetros de entrada:

   ```
   FUNCTION xpto()=>y
   ```

4. Não tem parâmetros de entrada nem retorna valores:

   ```
   FUNCTION xpto()
   ```

ATRIBUIÇÕES DE VARIÁVEIS

A inicialização de variáveis é feita através de:

```
name <- value
```

Podem ser inicializadas com:

1. um valor/String:

   ```
   variável <- 5
   ```

2. um token que retorne algum tipo de valor:

   variável <- GROUND

CONDIÇÕES

As condições são formadas por expressões matemáticas, usando tokens que retornem valores ou variáveis que irão ser comparados aos valores condicionais em questão.

1. Usando tokens:

   [GROUND=1] => FOWARD(5)

2. Usando variáveis:

   [var=4] => FOWARD(5)

LOOPS

Os blocos de código LOOP servem para adotar estruturas de repetições de instruções (ou conjuntos de instruções) enquanto uma dada condição for satisfeita.

Apresenta a seguinte estrutura:

LOOP
    [CONDIÇÃO] => TOKEN
ENDLOOP

Todas as variáveis têm de ser declaradas fora dos loops.

ACTIONS

Todas as condições que estiverem dentro de actions serão lidas linha a linha até uma delas se verificar. Quando isso acontecer, todas as seguintes são descartadas e a expressão à direita do sinal "=>"é posta em ação, daí o termo "actions".

A sua estrutura consiste em:

ACTIONS
    [CONDIÇÃO1] => TOKEN
    [CONDIÇÃO2] => TOKEN
    TOKEN
ENDACTIONS

NOTA: É possível ter actions dentro de actions. A sua estrutura é a seguinte:

ACTIONS
    [CONDIÇÃO1] => TOKEN
    [CONDIÇÃO2] =>
    ACTIONS
        [CONDIÇÃO21] => TOKEN
        [CONDIÇÃO22] => TOKEN
    ENDACTIONS

TOKEN
ENDACTIONS

Todas as variáveis têm de ser declaradas fora dos actions.

NOTAS ADICIONAIS

A linguagem é case-insensitive, ou seja, não faz distinção entre maiúsculas e minúsculas.