

Porigon-

Apresentação

introdução

Porigon é uma linguagem, focada para usuários aos quais são player especializados no jogo pokémon. Nessa linguagem os jogadores, podem programar uma estratégia para o seu pokémon, em uma batalha, então a partir desses comandos é possível que a batalha aconteça de forma automática, e o jogador apenas assiste o seu pokémon lutando com seu adversário, utilizando a estratégia programada, ate a batalha se finalizada

O nome porigon, foi escolhido pois é um nome de um dos monstros da franquia e é um pokémon que remete a tecnologia e ciência, logo combina com a proposta de ter uma linguagem a ser programada

Curiosidade: o porigon é um dos pokémons mais desconhecidos pelo grande publico, pois sua aparição foi proibida no desenho.

Estrutura e comandos disponíveis

Nossa linguagem funciona da seguinte forma

Pick -> start ->programa->end

É necessário colocar o programa nessa ordem pois se não o programa da erro. Entre os ´ principais comandos estão:

Pick: o comando inicial da linguagem. Simboliza que o pokémon foi escolhido

Start: a batalha começa, toda estratégia sera colocada depois dele.

End: simboliza duas coisas: ou se batalha terminou ou se terminamos um loop ou condição

If: descreve uma condição da nossa linguagem.

Else: contraparte do if

While: descreve uma condição loop.

Attack: condicional indica o ataque do seu pokémon.

Enemie_attack: condicional de ataque do adversario

Life: condicional de vida

Enemie_life: condicional de vida do adversário.

Normal: comando para dar o ataque normal

Special: coamndo para dar o ataque especial

Cure: comando para curar o pokémon.

Nossa bnf:

```

<program>      ::= <pick_cmd> "start" <stmt_list> "end"

<pick_cmd>     ::= "pick"

<stmt_list>    ::= <stmt>
                | <stmt> <stmt_list>

<stmt>         ::= <if_stmt>
                | <while_stmt>
                | <special_cmd>
                | <cure_cmd>
                | <normal_cmd>

<if_stmt>      ::= "if" <condition> ":" <stmt_list>
                | "else" ":" <stmt_list>
                | "end"

<while_stmt>   ::= "while" <condition> ":" <stmt_list> "end"

<special_cmd>  ::= "special"
<normal_cmd>   ::= "normal"
<cure_cmd>     ::= "cure"

<condition>    ::= <operand> <comp_op> <operand>

<operand>     ::= "life"
                | "attack"
                | "enemy_life"
                | "enemy_attack"
                | <number>

<comp_op>     ::= ">"

```

```

<condition>    ::= <operand> <comp_op> <operand>

<operand>     ::= "life"
                | "attack"
                | "enemy_life"
                | "enemy_attack"
                | <number>

<comp_op>     ::= ">"
                | "<"
                | "="
                | ">="
                | "<="

<number>      ::= <digit>
                | <digit> <number>

<digit>       ::= "0" | "1" | "2" | "3" | "4"
                | "5" | "6" | "7" | "8" | "9"

```

Exemplo de uso

Aqui em baixo esta um exemplo de uma estratégia completa para a batalha pokemon.

```
pick
start
  while enemy_life > 0:
    if life <= 40:
      cure
    else:
      if enemy_life >= 30:
        special
      else:
        normal
      end
    end
  end
end
end
```

E aqui esta o resultado da nossa vm

```
Digite um programa Pok|@Wars (Ctrl+D para finalizar):
Iniciando programa Pok|@Wars...
Estado inicial: life=100, attack=20, enemy_life=80, enemy_attack=15
[VM] SPECIAL: dano forte!
[VM] SPECIAL: dano forte!
Estado final:  life=100, attack=20, enemy_life=0, enemy_attack=15
```

Conclusão

Com tudo isso pudemos aprender mais sobre como criar linguagens e compiladores. Agora, espero que aproveitem.

