U-16 指導方針(2018)

~深夜テンションで書きました~

加藤 楓志

# 指導をする上で徹底すること

・変数名,メソッド名は”**意味の通る**”ローマ字

NOT:dog,apple,riko,kayo -私達はあの悪夢を繰り返さない

<https://github.com/KPC-U16/PullReq/blob/master/MissKishi.rb>

・インデントを揃えさせる

・できるかぎりエラー文読ませる

・自分で何をしたいのか説明させる

・メソッド化とその使い回しをさせる

・**逐一実行させる**(動作確認)

これらを徹底しない(させないと)と質問に答えられない、ソースを読んでも何をしているのかわからない、持ち帰って1,日解読に時間をかける等の事案が発生します.

# 大雑把な流れ

1. model.rbやtest3.rb等の用意されてるソースを実際に実行して慣れてもらう

チートシートを見せる

1. getreadyの値を使う処理

ここで基本的な制御構文,変数,配列,演算子,命名規則,インデントなどを教える

1. ②をメソッド化

def,(グローバル変数),引数

1. Look,searchを使った処理

中身、使いどころさんは考えさせる,最初っからメソッド使わせていいかも

## 基本方針

|  |
| --- |
|  |

・時計回り

・アイテムが有れば取りに行く

・斜め戦略は逃げ推奨

・最終目標とりあえずlook

# 詳細な流れ

practice.rb -何もしないプログラムを渡す

|  |
| --- |
| # -\*- coding: utf-8 -\*-  require 'CHaserConnect.rb' #呼び出すおまじない  # 書き換えない  target = CHaserConnect.new("prac") # ()の中好きな名前  values = Array.new(10)  random = Random.new # 乱数生成  #--------ここから--------  loop do # ここからループ  #---------ここから---------  values = target.getReady  if values[0] == 0  break  end  #-----ここまで書き換えない-----  #ここに処理を書く  #---------ここから---------  if values[0] == 0  break  end  end # ループここまで  target.close  #-----ここまで書き換えない----- |

1,実行してみよう

2,チートシートを見てみよう

(省略)

3,歩こう

|  |
| --- |
| #loop do内  hoko = 0 #上0,右3,下6,左9  if hoko == 0  #上を向いて歩こう  values = target.walkUp()  end |

方向系の値は基本時計に沿わせる

変数,変数名の命名規則,代入,if文,コメントを教える

このあとも言えることだが書いたら逐次実行させる

|  |
| --- |
| #loop do内  hoko = 0  case hoko  when 0  #上を向いて歩こう  values = target.walkUp()  when 3  #右を向いて歩こう  values = target.walkRight()  when 6  #下を向いて歩こう  values = target.walkDown()  when 9  #左を向いて歩こう  values = target.walkLeft()  end |

Case文

歩けることを確認する。

|  |
| --- |
| def \_aruku(hoko, values, target) end  #loop do内 #行動しよう  \_aruku(hoko, values, target) |

メソッド化 メソッド名の前に\_(アンダーバー)をつけさせる

メソッド化のメリット,引数の呼び出し

4,前に壁があったら右に曲がろう

|  |
| --- |
| #右に曲がろう  #loop do内  if hoko == 0 #上に行きたい  if values[2] == 2 #上のマスが壁だったら  hoko = 3 #右に曲がる  end  end |

次から途端に難易度が上がり、量が増える

ヤバイ

loop,elsif,elseの説明

|  |
| --- |
| #壁を避けよう  loop do  if hoko == 0 #上に行きたい  if values[2] == 2 #上のマスが壁だったら  hoko = 3 #右向け右  else  break  end  elsif hoko == 3 #右に行きたい  if values[6] == 2 #右のマスが壁だったら  hoko = 6 #下向け下  else  break  end  elsif hoko == 6 #下に行きたい  if values[8] == 2 #下のマスが壁だったら  hoko = 9 #左向け左  else  break  end  elsif hoko == 9 #左に行きたい  if values[4] == 2 #左のマスが壁だったら  hoko = 0 #上向け上  else  break  end  end  end |

次メソッド化

returnの説明

|  |
| --- |
| #壁を避けよう  def \_kabeyoke(hoko, values)  loop do  if hoko == 0 #上に行きたい  if values[2] == 2 #上のマスが壁だったら  hoko = 3 #右向け右  else  break  end  elsif hoko == 3 #右に行きたい  if values[6] == 2 #右のマスが壁だったら  hoko = 6 #下向け下  else  break  end  elsif hoko == 6 #下に行きたい  if values[8] == 2 #下のマスが壁だったら  hoko = 9 #左向け左  else  break  end  elsif hoko == 9 #左に行きたい  if values[4] == 2 #左のマスが壁だったら  hoko = 0 #上向け上  else  break  end  end  end  return hoko  end  #loop do内  hoko = \_kabeyoke(hoko, values)  \_aruku(hoko, values, target) |

5,真横に敵が来たら倒そう

|  |
| --- |
| def \_oku(hoko, values, target)  case hoko  when 0  #上を向いて歩こう  values = target.putUp()  when 3  #右を向いて歩こう  values = target.putRight()  when 6  #下を向いて歩こう  values = target.putDown()  when 9  #左を向いて歩こう  values = target.putLeft()  end  end |

ここから、直にメソッドを作っていく

|  |
| --- |
| def \_mayokoniaitemugaaru(hoko, values)  if values[2] == 3  hoko = 0  elsif values[6] == 3  hoko = 3  elsif values[8] == 3  hoko = 6  elsif values[4] == 3  hoko = 9  end  return hoko  end |

|  |
| --- |
| def \_mayokonitekigairu(hoko, values, target)  ita = 0 #フラグ  #all you needs is kill  if values[2] == 1 #上に敵がいたとき  hoko = 0  ita = 1 #フララ”が立った!  elsif values[6] == 1 #右に敵がいたとき  hoko = 3  ita = 1 #フララ”が立った!  elsif values[8] == 1 #下に敵がいたとき  hoko = 6  ita = 1 #フララ”が立った!  elsif values[4] == 1 #左に敵がいたとき  hoko = 9  ita = 1 #フララ”が立った!  end  puts ita  if ita == 1  ita = 0 #立てたフラグは寝かせましょう  \_oku(hoko, values, target)  end  end  #loop do内  \_mayokonitekigairu(hoko, values, target)  hoko = \_kabeyoke(hoko, values)  \_aruku(hoko, values, target) |

フラグとは

6,真横にアイテムがあったらゲッチュ

|  |
| --- |
| def \_mayokoniaitemugaaru(hoko, values)  if values[2] == 3  hoko = 0  elsif values[6] == 3  hoko = 3  elsif values[8] == 3  hoko = 6  elsif values[4] == 3  hoko = 9  end  return hoko  end  #loop do内  \_mayokonitekigairu(hoko, values, target)  hoko = \_mayokoniaitemugaaru(hoko, values)  hoko = \_kabeyoke(hoko, values)  \_aruku(hoko, values, target) |

同じ要領でげっちゅ!

7,(先を見よう)

|  |
| --- |
| def \_sakiwomiru(hoko, saki, target)  case hoko  when 0  saki = target.searchUp  when 3  saki = target.searchRight  when 6  saki = target.searchDown  when 9  saki = target.searchLeft  end  return saki  end |

|  |
| --- |
| def \_tamanisakiwomiru(hoko, uenosaki, miginosaki, shitanosaki, hidarinosaki, target)  case hoko  when 0  uenosaki = \_sakiwomiru(hoko, uenosaki, target)  hoko = \_nanimonai(hoko, uenosaki)  when 3  miginosaki = \_sakiwomiru(hoko, miginosaki, target)  hoko = \_nanimonai(hoko, miginosaki)  when 6  shitanosaki = \_sakiwomiru(hoko, shitanosaki, target)  hoko = \_nanimonai(hoko, miginosaki)  when 9  hidarinosaki = \_sakiwomiru(hoko, hidarinosaki, target)  hoko = \_nanimonai(hoko, hidarinosaki)  end  return hoko  end |

|  |
| --- |
| #直線上に何もなければ方向転換  def \_nanimonai(hoko, values)  nai = 0  for i in 1..9  if values[i] == 3  nai = 1  end  end  if nai == 0  hoko = (hoko + 3) % 12  end  return hoko  end |

|  |
| --- |
| #Loop do 内  case hoko  when 0  uenosaki = \_sakiwomiru(hoko, uenosaki, target)  hoko = \_nanimonai(hoko, uenosaki)  when 3  miginosaki = \_sakiwomiru(hoko, miginosaki, target)  hoko = \_nanimonai(hoko, miginosaki)  when 6  shitanosaki = \_sakiwomiru(hoko, shitanosaki, target)  hoko = \_nanimonai(hoko, miginosaki)  when 9  hidarinosaki = \_sakiwomiru(hoko, hidarinosaki, target)  hoko = \_nanimonai(hoko, hidarinosaki)  end  hoko = \_mayokoniaitemugaaru(hoko, values)  以下略 |

なんていうかヤバイ

Look,Serchの利用、活用例

以下　見たエラーとその原因を書き連ねて言ってください

後の為にそれを使いエラー一覧を作ろうと思います(このページ回収)