第2回C++プログラミングII課題レポート

S202148 柳澤快

課題２A

* ソースコード

#include <iostream>

#include <vector>

#include <algorithm>

//i行j列の位置座標の定義

struct Pos

{

int i, j;

bool operator==(const Pos p) const { return i == p.i && j == p.j; };

bool operator!=(const Pos p) const { return i != p.i || j != p.j; };

};

//倉庫番のゲームフィールドの定義

class Field

{

private:

//ゲームフィールドの初期設定

const int row{3}; //行数（縦の長さ）

const int column{3}; //列数（横の長さ）

const std::vector<Pos> wall{}; //壁（障害物）0個

const std::vector<Pos> goal{{0, 2}, {1, 0}}; //ゴール2個

Pos player{2, 0}; //プレイヤーの初期位置

std::vector<Pos> box{{1, 2}, {1, 1}}; //箱2個

public:

//フィールドを端末上に表示

void print()

{

for (int i = -1; i <= row; i++)

{

for (int j = -1; j <= column; j++)

{

//フィールドの周囲に壁を表示

if (i < 0 || i >= row || j < 0 || j >= column)

{

std::cout << 'X';

}

//プレイヤー、壁、ゴール、箱を表示

//プレイヤーの場所では'@'を表示

//壁の場所では'X'を表示

//ゴールと箱が重なっている場合は'O'を表示

//ゴールのみの場合は'+'を表示

//箱のみの場合は'$'を表示

//それ以外は空白' 'を表示

else

{

Pos coordinate\_ij{i, j};

if (i == player.i && j == player.j)

std::cout << "@";

else if (std::find(wall.begin(), wall.end(), coordinate\_ij) != wall.end())

std::cout << "X";

else if (std::find(goal.begin(), goal.end(), coordinate\_ij) != goal.end() && std::find(box.begin(), box.end(), coordinate\_ij) != box.end())

std::cout << "O";

else if (std::find(goal.begin(), goal.end(), coordinate\_ij) != goal.end())

std::cout << "+";

else if (std::find(box.begin(), box.end(), coordinate\_ij) != box.end())

std::cout << "$";

else

std::cout << " ";

}

}

std::cout << '\n';

}

}

//ゴールと箱が重なっている場合はゲームクリアと判定

bool

completed()

{

bool judge;

for (int i = 0; i < goal.size(); i++)

(std::find(box.begin(), box.end(), goal[i]) != box.end()) ? judge = true : judge = false;

return judge;

}

//u(up), d(down), r(right), l(left)の文字列を受けてプレイヤーをその方向に移動

//move関数の引数として、Pos型引数を移動方向に変更するラムダ式を与える

//例えば d=='u'の場合は、Pos型変数pを引数として、p.iを1減らして、pを返すラムダ式を与える

void movec(char d)

{

if (d == 'u')

move([](Pos p)

{

p.i--;

return p;

});

else if (d == 'd')

move([](Pos p)

{

p.i++;

return p;

});

else if (d == 'r')

move([](Pos p)

{

p.j++;

return p;

});

else if (d == 'l')

move([](Pos p)

{

p.j--;

return p;

});

}

//Pos型変数を移動する関数nextを受け取りフィールドの状態更新を行う

template <typename T>

void move(T next)

{

//Pos型引数pが移動可能な位置か判定する関数をラムダ式で定義

//pが0～(row-1), 0～(column-1)の座標に収まっているか、wallと重なっていないかを判定

auto can\_move = [=](Pos &p)

{

bool judge\_wall;

(std::find(wall.begin(), wall.end(), p) == wall.end()) ? judge\_wall = true : judge\_wall = false;

return (p.i >= 0 && p.i <= row - 1 && p.j >= 0 && p.j <= column - 1 && judge\_wall);

};

Pos pnext = next(player); //プレイヤーの次の位置の計算

if (can\_move(pnext))

{ //プレイヤーが移動可能な場合のみ更新

//プレイヤーが箱に重ならない場合はそのまま移動

if (std::find(box.begin(), box.end(), pnext) == box.end())

{

player = pnext;

next(player);

}

//プレイヤーが箱に重なる場合は箱の移動判定が必要

else

{

//箱の次の位置を計算

//箱が移動可能な場合はプレイヤーと箱を移動

std::vector<Pos> bnext\_vec;

for (int i = 0; i < box.size(); ++i)

bnext\_vec.push\_back(next(box[i]));

for (int i = 0; i < bnext\_vec.size(); ++i)

{

if (can\_move(bnext\_vec[i]) && pnext == box[i] && std::find(box.begin(), box.end(), bnext\_vec[i]) == box.end())

{

player = pnext;

next(player);

box[i] = bnext\_vec[i];

next(box[i]);

}

}

}

}

};

};

int main()

{

Field f; //フィールドの初期化

char c;

f.print(); //初期フィールドの表示

while (std::cin >> c)

{

// 'q' の場合は終了（リタイア）

if (c == 'q')

{

std::cout << "quit" << '\n';

exit(1);

}

// 'u', 'd', 'r', 'l' の場合はその移動方向に従って更新し、ゲームクリアの場合は終了

if (c == 'u' || c == 'd' || c == 'r' || c == 'l')

{

f.movec(c);

f.print();

if (f.completed())

{

std::cout << "The work is completed!!\n";

break;

}

}

//上記以外の入力の場合は正しい入力候補を表示

else

std::cout << "u(up), d(down), r(right), l(left), q(quit)" << '\n';

}

}

* 自分で考えた異なる初期配置での実行結果の画像

テキスト

自動的に生成された説明

* vectorの要素に含まれているかを判定するのに利用したalgorithm機能の説明

std::findを用いてvectorの最初から最後までのイテレータをみて第３引数の値があるかを判定した。std::find()!=vector.end()ではstd::findがイテレータを返し最後まで行かなければ見つかったことになる。逆にstd::find()==vector.end()ならば見つからなかったことになる。