// 課題3\_ダンジョン生成と脱出AI

// 作成日：2016/6/29

//

// main.cpp

// 脱出経路探索プログラム

//

// 参考ウェブ:

// c言語で迷路：http://studio.s1.xrea.com/prog/maze.html

// ダブルバッファリング：http://tama012.blogspot.jp/2012/11/blog-post.html

// http://vivi.dyndns.org/tech/cpp/DblBuffer.html

#include <fstream>

#include <iostream>

using namespace std;

#include "DoubleBufferConsole.h"

enum BLOCK{

FLOOR, // 床

WALL, // 壁

START, // スタート

GOAL, // ゴール

ROOT, // ダミー

PLAYER // 探索位置

};

const int sizeX = 100;

const int sizeY = 100;

BLOCK map[sizeY][sizeX]; // ダンジョンマップ

DoubleBufferConsole db; // ダブルバッファコンソール

bool clear = false; // ゴールしたかどうか

// 更新速度モード(描画なし、高速、低速)

enum UpdateMode{

NODRAW, FAST, NORMAL

}mode;

// プロトタイプ宣言

void Draw( int, int );

bool Check( int, int );

void Root( int, int );

// マップ描画

void Draw( int posY, int posX ){

if( mode == NODRAW ) return;

db.SetCursorPos( 0, 0 );

int row = 0; // 描画する行数

for( int y = posY-10; y < posY+10; y++ ){ // 引数で指定したy座標の上下10行分を描画

if( y < 0 || y >= sizeY ) for( int i = 0; i < 20;i++ ) db.Write( "×" ); // 外側は×を表示

else{

for( int x = posX-10; x < posX+10; x++ ){ // 引数で指定したx座標の左右10列分を描画

if( x < 0 || x >= sizeX ) db.Write( "×" ); // 外側は×を表示

else {

switch( map[y][x] ){

case FLOOR: db.Write( "　" ); break; // 床

case WALL: db.Write( "■" ); break; // 壁

case START: db.Write( "Ｓ" ); break; // スタート

case GOAL: db.Write( "Ｇ" ); break; // ゴール

case PLAYER: db.Write( "・" ); break; // 探索位置

default: db.Write( "？" ); break; // 念のため

}

}

}

}

// 改行

row++;

db.SetCursorPos( 0, row );

}

if( mode == FAST )Sleep( 10 ); // 1/1000秒待つ

else Sleep( 500 ); // 1/2秒待つ

db.Swap(); // バッファをスワップして画面に表示

}

// 上下左右どれか隣にゴールがあるか

bool CheckAround( int posY, int posX ){

if( map[posY][posX-1] == GOAL || map[posY][posX+1] == GOAL ||

map[posY-1][posX] == GOAL || map[posY+1][posX] == GOAL ){

if( mode != NODRAW )Sleep( 2000 ); // 2秒待つ

return true;

}

else return false;

}

// 探索

void Root( int y, int x ){

if( clear ) return;

if( map[y][x] == FLOOR ) {

map[y][x] = PLAYER;

Draw( y, x );

}

// 現在位置の周りにゴールがない場合

if( CheckAround( y, x ) ){

clear = true;

return;

}else{

// 上下左右それぞれ進める場所があれば進む

if( map[y-1][x] == FLOOR ){

Root( y-1, x );

}

if( map[y+1][x] == FLOOR ){

Root( y+1, x );

}

if( map[y][x-1] == FLOOR ){

Root( y, x-1 );

}

if( map[y][x+1] == FLOOR ){

Root( y, x+1 );

}

}

}

// メイン関数

int main(){

db.Write( "モードを選択\n" );

db.Write( "0:描画なし（ファイル出力のみ）\n1:高速描画モード\n2:通常モード\n " );

db.Swap();

int num;

cin >> num;

switch( num ){

case 0:

mode = NODRAW;

db.Write( "描画なしで実行します。\n" );

break;

case 1:

mode = FAST;

db.Write( "高速描画モードで実行します。\n" );

break;

case 2:

mode = NORMAL;

db.Write( "通常モードで実行します。\n" );

break;

default:

mode = NORMAL;

db.Write( "入力が不正です。通常モードで実行します。\n" );

break;

}

db.Swap();

Sleep( 1000 ); // 1秒待つ

// ファイル読み込み

ifstream ifs;

ifs.open( "maze.txt" );

if( ifs.fail() ){

cout << "ファイルを開けませんでした。" << endl;

return -1;

}

string str; // 読み込んだ文字列を格納するバッファ

int x = 0, y = 0; // マップを生成するマス目の座標

int sx, sy, gx, gy; // スタート、ゴール位置の座標

while( getline( ifs, str ) ){ // 1行ずつ読み込み、ファイルの最後までループ

for( int i = 0; i < str.size(); i++ ){ // 1文字ずつ解析

switch( str[i] ){

case '0': // 床

map[y][x] = FLOOR;

break;

case '1': // 壁

map[y][x] = WALL;

break;

case '2': // スタート

map[y][x] = START;

sx = x; sy = y;

break;

case '3': // ゴール

map[y][x] = GOAL;

gx = x; gy = y;

break;

default:

x--; // 空白を読み込んだ場合一度左へ戻る

}

x++; // 右のマスへ進む

}

x = 0; // 改行

y++;

}

// スタート位置から探索開始

Root( sy, sx );

// 結果テキスト出力

ofstream ofs;

ofs.open( "ans.txt" );

if( ofs.fail() ){

cout << "ファイルが開けません" << endl;

return -1;

}

// マップを片っ端から書き出す

for( int y = 0; y < sizeY; y++ ){

for( int x = 0; x < sizeX; x++ ){

ofs << map[y][x];

ofs << " ";

}

ofs << endl; // 1行終わったら改行

}

ofs.close();

return 0;

}