制作演習_課題 03

パックマンのプログラムのリファクタリングを行います。

1. リファクタリングとは？

ソフトウェアの外部仕様を変更せずに内部仕様を変更することです。

→要するに？

→ユーザーが遊んだ時の、ゲームの動作は一切変更せずに、プログラムの設計や実装だけ変更すること。ユーザーからすると何も変わらない。

→なんのためにするの？

→プログラムの設計や実装を整えることによって、続編の製作や、バージョンアップなどのときに、プログラムの変更を行うのが容易になるから。

2. 今回のリファクタリングの内容

今回の課題で行うリファクタリングは下記の 2 点となります。

・プレイヤーに関係する変数を構造体にする。

・プレイヤーに関する更新処理、描画処理を関数化する。

エネミーや食べ物はすでに、構造体と関数を使用して処理が記述されています。

Enemy.h、Enemy.cpp、Food.h、Food.cpp を参考にしながらリファクタリングを行ってください。なお、Player.h、Player.cpp というファイルは追加してもしなくても、どちらでもよいものとする。

3. 構造体

変数は意味のあるグループとして、まとめることで、プログラムの見通しが良くなります。今回は下記の 3 つの変数を Player 構造体としてまとめてください。

packman/main.cpp(34 行目～)

```
//プレイヤー関係のグローバル変数。
int playerIsDead = 0; //プレイヤーの死亡フラグ。1になったら死亡。
int playerPosX = 7;
int playerPosY = 7;
```

```
struct Player {
    int isDead = 0;    //死亡フラグ。
    int posX = 7; //X座標。
    int posY = 7; //Y座標。
};
```

構造体は、複数の変数をまとめた新しい型を定義するためのものです。上の Player 構造体は型なので、下記のような Player 型の変数を用意する必要があります。

```
Player g_player;
```

プレイヤーの座標は Enemy.cpp から参照されているので、グローバル変数として定義していることに注意してください。

Player 型の g_player が用意出来たら、下記の 3 つの変数は不要となるので、削除してください。

main.cpp

```
//プレイヤー関係のグローバル変数。
//不要なのでコメントアウト。
//int playerIsDead = 0;//プレイヤーの死亡フラグ。1になったら死亡。
//int playerPosX = 7;
//int playerPosY = 7;
```

stdafx.h

```
////////////////////////////////////
////ここからプレイヤー関係。
//// extern宣言も不要なので削除
////////////////////////////////////
//extern int playerIsDead;
//extern int playerPosX;
//extern int playerPosY;
```

これらをコメントアウトしたら、これらの変数を使用していた箇所で、コンパイルエラーが発生すると思います。それらの個所で g_player を使用するように変更して、コンパイルエラーが起きないようにしてください。

4. 関数化

関数には、繰り返し記述されるプログラムを一まとめにして、再利用性を上げることができるという点と、意味のある処理をまとめて、関数化することで、見通しのよいプログラムにすることができるという点があります。今回の関数化は後者の見通しのよいプログラムを作成するというのが目的です。

下記のプレイヤーの更新処理を関数化してください。関数の名前は自由に決めてください。(Enemy.h,cpp や Food.h,cpp を参考にすると、良い名前の関数になります。)

main.cpp(97 行目～148 行目)

```
////////////////////////////////////////////////////////
// プレイヤーの更新処理。
////////////////////////////////////////////////////////
{
	//移動前の座標を保存。
	int oldPosX = playerPosX;
	int oldPosY = playerPosY;
	if (GetAsyncKeyState(VK_LEFT) != 0) {
		//左に移動。
		playerPosX--;
	}
	else if (GetAsyncKeyState(VK_RIGHT) != 0) {
		//右に移動。
		playerPosX++;
	}
	else if (GetAsyncKeyState(VK_UP) != 0) {
		//上に移動。
```

```
            playerPosY--;
        }
        else if (GetAsyncKeyState(VK_DOWN) != 0) {
            //下に移動。
            playerPosY++;
        }
        //移動先をチェック。
        //壁判定
        if (g_map[playerPosY][playerPosX] == 1) {
            //壁なので、座標を戻す。
            playerPosX = oldPosX;
            playerPosY = oldPosY;

        }
        //敵との衝突判定。
        for (int no = 0; no < g_enemyNum; no++) {
            if (g_enemyArray[no].posX == playerPosX
                && g_enemyArray[no].posY == playerPosY
                ) {
                //敵とぶつかった。
                playerIsDead = 1;    //死亡フラグを立てる。
            }
        }
        //食べ物との衝突判定。
        for (int no = 0; no < g_foodNum; no++) {
            if (g_foodArray[no].posX == playerPosX
                && g_foodArray[no].posY == playerPosY
                ) {
                //食べ物とぶつかった。
                //食べ物の死亡フラグを立てる。
                g_foodArray[no].isDead = 1;

            }
        }
}
```

続いて、下記の箇所も関数化してください。関数の名前は自由に決めてください。(こちらも y.h,cpp や Food.h,cpp を参考にすると、良い名前の関数になります。)

min.cpp(161 行目～164 行目)

```
//////////////////////////////////////////////////////
// プレイヤーの描画処理。
//////////////////////////////////////////////////////
kbcDrawMoji(playerPosX, playerPosY, 'P');
```