検証項目

`void out();`
盤面を出力する。配列checkや変数flg3の状態によって数字や記号も出力する。

`char kakunin(int, int);`
メイン関数内でプレイヤーが指定した座標の周りにいくつ爆弾が埋まっているか特定する。もし一つも埋まっていなかった場合は八つの条件ごとに関数

`int set_bom();`

爆弾を埋める座標を指定するための乱数を生成する

`int set_line_bom();`
乱数によって移動爆弾をx軸とｙ軸のどちらに置くか

`void bomber1(int, int);`
周囲に爆弾がいくつ埋まっているか特定する。もし一つも埋まっていなかった場合は0を返す

`void oku_line_bom(int, int);`
set_line_bomで指定した軸と座標に爆弾として１を置き、その他の座標に爆弾の通り道として２を置く。

`void bigbom2();`
2*2の巨大爆弾をセッティングするための関数。乱数で座標を一か所指定した後にそこから右下に広げるイメージで四つの座標に爆弾を埋めて疑似的に巨大爆弾を創り出す

`void bigbom3();`
bigbom2 と同じ要領で 3*3 の巨大爆弾を創り出す

`void c_bigbom();`
bigbom2 と 3 の繰り返しから飛ぶ。巨大爆弾を生成した際に既存の爆弾の位置とかぶってしまうかどうかを検証する関数。

流れ	使用関数	検証
①難易度選択	main()	○
②爆弾セット	set_bom()　bigbom2() big_bom3()　set_line_bom() c_bigbom()　oku_line_bom()	○
③出力	out()	○
④座標入力	main()	○
⑤終了判定	main()	○
⑥爆弾探知	Kakunin()　bomber1()	○
③～⑥繰り返し		○