解答

解答とは言っても一例にすぎません．若干異なっていたとしてもプログラムが動きさえすれば問題ありません．

・穴埋め解答

穴埋め 1：player.vy >= 0

ゲーム画面では，y 軸方向鉛直下向きが＋であることに注意です．ブロックに上からぶつかるか，下からぶつかるかで処理が異なります．

穴埋め 2：player.dire = 1

player.dire は主人公の向きを表しており，0 であれば右，1 であれば左を表します．この向きによって変化するものはなんであるのか，考えてみてください．

穴埋め 3：

(enemy.x < player.x + HERO_SIZE && player.x < enemy.x + enemy.size_x)
&&(enemy.y < player.y + HERO_SIZE && player.y < enemy.y + enemy.size_y)

主人公と敵のあたり判定です．あたり判定についてはシューティングゲームで説明しました．

穴埋め 4：WINDOW_HEIGHT

主人公が画面外に出てしまう（下に落ちる）とは，主人公の y 座標が画面の高さ（下向きに＋であることに注意）よりも大きくなる時です．

穴埋め 5：enemy.life != 0

敵の生存状況は enemy.life で管理されています．0 であれば死，それ以外なら生存です．

穴埋め 6：enemy.dire = 0

穴埋め 7：enemy.dire = 1

穴埋め 8：player.attack

主人公が攻撃中のとき，player.attack は 1 になります．そのため，主人公が攻撃中なら if 文の中身が実行されます．

穴埋め 9：

player.dire == 0 && (enemy.x < player.x + HERO_SIZE + ATTACK_SIZE && player.x + ATTACK_SIZE < enemy.x + enemy.size_x) &&(enemy.y < player.y + HERO_SIZE && player.y < enemy.y + enemy.size_y)

主人公が右を向いているときの，主人公の攻撃と敵のあたり判定です．主人公が左を向いているときとは，少し違うのがわかりますでしょうか．考え方は今までのあたり判定と変わらないので，問題ないはずです．

穴埋め 10：keyState[KEY_INPUT_Z]

keyState には，一つ一つのキーについて押されているかどうかの値が入っています．押されていれば 1，押されていなければ 0 です．KEY_INPUT_Z は DX ライブラリの中でマクロとして定義されています．

マクロとは

#define 名前　数値

という風に定義されています．例えば，

#define BLOCK 32

となっていると，プログラム中にある「BLOCK」というところがすべて「３２」と書いてあるのと同じになります．