```
1 // TreasureHunt.cpp
 2 // Assignment 1.2 By Nathan Graham
4 #include "stdafx.h"
 5 #include <iostream>
 6 #include <string>
7 using namespace std;
9 const int arraySize = 10; // size of the array of rooms
10 const int playerInventorySize = 5; // number of spaces in the players inventory
11 const int roomInventorySize = 50; // number of items that can be in each room
12 const string endLine = "\n"; // makes the end lines more organised
13
14 enum wallDirection { horizontal = 0, vertical = 1 }; // horizontal and vertical
     walls which create a grid over the array of rooms
15 enum door { noDoor = 0, lockedDoor = 1, openDoor = 2, corridor = 3, stairCase =
     4, forest = 5, openArea = 6, tunnel = 7}; //types of walls that can exist on
     the grid
16 enum inventory { emptyItem = 0, skeletonKey = 1, lantern = 2, litLantern = 3,
     food = 4, waterBottle = 5, matches = 6, goldKey = 7 }; //items which can be
     pickedup into the inventory from each room
17
18
19 class inventoryList {
20 public:
21
       inventory inventoryType;
22
       string description;
23
       inventoryList() { inventoryType = inventory::emptyItem, description =
       inventoryList(inventory a, string b ) { inventoryType = a, description =
24
         b; };
25 };
26
27 class wall { // class for the walls which have no doors or openings
28 public:
29
       door doorType;
30
       wall() { doorType = noDoor; };
31 };
32
33 class room { // class for each room in the array
34 public:
       string description;
35
36
       wall *northWall;
       wall *southWall;
37
       wall *eastWall;
38
39
       wall *westWall;
40
       bool visible;
41
       inventory enableVisible; // allows the player to turn visibility on, when a
         room is to dark
42
       inventory enableUnlock; // allows the player to unlock a door, when the door →
         to next room is locked
       bool treasureRoom;
43
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
2
```

```
inventory slots[roomInventorySize];
       room() { description = "Empty room", visible = true, enableVisible =
45
                                                                                        P
          inventory::emptyItem, enableUnlock = inventory::emptyItem, treasureRoom =
                                                                                        P
          false; }; // default options for each room
46
47
       void updateRoom(string name, door northDoor, door southDoor, door eastDoor,
                                                                                        P
          door westDoor, bool roomVisible) { // function which is used for creating
          doors in walls for each room on the array
48
            description = name;
49
            northWall->doorType = northDoor;
            southWall->doorType = southDoor;
50
51
            eastWall->doorType = eastDoor;
52
            westWall->doorType = westDoor;
53
            visible = roomVisible;
54
       }
55
       bool addInventory(inventory inventoryItem) { // function which returns true
56
          for adding inventoryItems to each room if they are the set ones and false
          for anything else
57
            for (int i = 0; i < roomInventorySize; i++) {</pre>
58
                if (slots[i] == inventory::emptyItem) {
                    slots[i] = inventoryItem;
59
60
                    return true;
61
                }
62
            }
63
64
            return false;
65
       }
66
       bool removeInventory(inventory inventoryItem) { // function which returns
67
          true for removing inventoryItems which can only be used once and false for >
          anything else
            for (int i = 0; i < roomInventorySize; i++) {</pre>
68
69
                if (slots[i] == inventoryItem) {
70
                    slots[i] = inventory::emptyItem;
71
                    return true;
72
                }
73
            }
74
75
            return false;
76
       }
77
78
       bool unlockDoor() { // allows the player to unlock the locked doors that
          connect to another room in each direction
79
            bool unlockedDoor = false;
80
            if (northWall->doorType == door::lockedDoor) {
81
82
                northWall->doorType = door::openDoor;
83
                unlockedDoor = true;
84
            }
85
86
            if (southWall->doorType == door::lockedDoor) {
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
3
```

```
southWall->doorType = door::openDoor;
 88
                 unlockedDoor = true;
 89
             }
 90
 91
             if (eastWall->doorType == door::lockedDoor) {
 92
                 eastWall->doorType = door::openDoor;
 93
                 unlockedDoor = true;
 94
             }
 95
 96
             if (westWall->doorType == door::lockedDoor) {
 97
                 westWall->doorType = door::openDoor;
 98
                 unlockedDoor = true;
 99
             }
100
101
             return unlockedDoor;
102
         }
103 };
104
105 class player { // this is the player description, with commands they can do for
       moving around the array, adding and removing items from
                    // inventory and for eating food to maintain energy levels which
    public:
       are reduced each time the player moves room
107
         string name;
108
         int x;
109
         int y;
110
         int energy;
111
         inventory slots[playerInventorySize];
112
         player() { name = "Link:", x = 4, y = 4, energy = 30; };
113
         bool moveNorth(door doorType) {
114
115
             if(y > 0)
                 && doorType != lockedDoor
116
117
                 && doorType != noDoor) {
118
                 y--;
119
                 energy--;
120
                 return true;
121
             }
122
             return false;
123
         }
124
125
         bool moveSouth(door doorType) {
             if (y < (arraySize - 1)</pre>
126
                 && doorType != lockedDoor
127
128
                 && doorType != noDoor) {
129
                 y++;
130
                 energy--;
131
                 return true;
132
             }
133
             return false;
134
         }
135
136
         bool moveEast(door doorType) {
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
4
```

```
137
             if (x < (arraySize - 1)</pre>
138
                  && doorType != lockedDoor
139
                  && doorType != noDoor) {
140
                  x++;
141
                  energy--;
142
                  return true;
143
             }
144
             return false;
145
         }
146
         bool moveWest(door doorType) {
147
148
             if (x > 0)
                  && doorType != lockedDoor
149
150
                  && doorType != noDoor) {
151
                  x--;
152
                  energy--;
153
                  return true;
154
             }
155
             return false;
156
         }
157
         bool addInventory(inventory inventoryItem) {
158
             for (int i = 0; i < playerInventorySize; i++) {</pre>
159
                  if (slots[i] == inventory::emptyItem) {
160
                      slots[i] = inventoryItem;
161
162
                      return true;
163
                  }
164
             }
165
166
             return false;
167
         }
168
         bool removeInventory(inventory inventoryItem) {
169
             for (int i = 0; i < playerInventorySize; i++) {</pre>
170
171
                  if (slots[i] == inventoryItem) {
                      slots[i] = inventory::emptyItem;
172
173
                      return true;
174
                  }
175
             }
176
177
             return false;
178
         }
179
         bool eatFoods() {
180
181
             if (energy >= 30)
182
                  return false;
183
184
             energy += 5;
185
186
             if (energy > 30) {
187
                  energy = 30;
188
             }
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
189
190
             return true;
191
         }
192 };
193
194 void displayDirection(door doorType, string direction) // function which displays →
        the direction of the door type
195 {
196
         switch (doorType) // used to determine which doortype should be used
197
198
             case door::lockedDoor:
                 cout << direction << " there is a locked door." << endLine;</pre>
199
                 break;
200
201
             case door::openDoor:
202
                 cout << direction << " there is an open door." << endLine;</pre>
203
                 break;
204
             case door::corridor:
205
                 cout << direction << " there is a corridor." << endLine;</pre>
206
207
             case door::stairCase:
                 cout << direction << " there is a staircase." << endLine;</pre>
208
209
                 break:
             case door::forest:
210
                 cout << direction << " there is a forest." << endLine;</pre>
211
212
                 break:
213
             case door::openArea:
                 cout << direction << " there is an open area." << endLine;</pre>
214
215
                 break;
216
             case door::tunnel:
217
                 cout << direction << " there is a tunnel." << endLine;</pre>
218
                 break:
219
         }
220 }
221
222 void displayRoom(room *currentRoom, player *currentPlayer) // function which
       displays things about the current room and the player
223 {
224
         // initialise inventory
225
226
         inventoryList availableItems[]{
227
             inventoryList(inventory::emptyItem, "Empty"),
             inventoryList(inventory::skeletonKey, "Bunch of keys including a skeleton →
228
                key"),
             inventoryList(inventory::lantern, "Lantern"),
229
             inventoryList(inventory::litLantern, "Lit Lantern"),
230
231
             inventoryList(inventory::food, "An array of delicious cakes and drinks"),
             inventoryList(inventory::waterBottle, "Water bottle"),
232
233
             inventoryList(inventory::matches, "A box of matches"),
             inventoryList(inventory::goldKey, "A large gold key. This looks very
234
               old."),
235
         };
236
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
                                                                                         6
        cout << currentPlayer->name << " Current Energy: " << currentPlayer->energy
                                                                                         P
           << endLine << "Your location is " << currentRoom->description <<
                                                                                         P
           endLine; // displays current energy levels, current rooms
238
                                                                             //
    description and current items in inventory
        int inventorySize = (sizeof(availableItems)) / sizeof(*(availableItems))); // >
239
           size of array divided by size of element
240
        bool overrideDarkness = false; // statement to say each room is not overrided →
241
            by darkness by default
242
243
        if (!currentRoom->visible) { // If statement saying that if the room is not
          visible then it must be overrided by darkness
244
            for (int i = 0; i < playerInventorySize; i++) {</pre>
245
                 if (currentPlayer->slots[i] == currentRoom->enableVisible) {
246
                     overrideDarkness = true;
                     break;
247
248
249
             }
250
        }
251
        if (currentRoom->visible
252
253
             || overrideDarkness) { // If statement saying that if the room is
               visible, and not overrided by darkness, then display the directions for ₹
                each doorType
             displayDirection(currentRoom->northWall->doorType, "To the north");
254
             displayDirection(currentRoom->southWall->doorType, "To the south");
255
             displayDirection(currentRoom->eastWall->doorType, "To the east");
256
             displayDirection(currentRoom->westWall->doorType, "To the west");
257
             bool titleDisplayed = false;
258
259
            for (int i = 0; i < roomInventorySize; i++) { // If the room inventory is →</pre>
260
                not empty, display what items are in the room
261
                 if (currentRoom->slots[i] != inventory::emptyItem) {
262
263
                     if (!titleDisplayed) {
264
                         cout << "Room item(s) include:" << endLine;</pre>
265
                         titleDisplayed = true;
266
                     }
267
                     for (int s = 0; s < inventorySize; s++) {</pre>
268
269
270
                         if (currentRoom->slots[i] == availableItems[s].inventoryType) >
                           { // If current room has enough slots for available items in→
                          the room inventory, give the available items a slot number >
271
                             cout << "\t" << availableItems[s].description << "[" << i >
```

<< "]" << endLine; // '\t' is done to indent the line</pre>

break;

}

}

272

273

274

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
7
```

```
275
276
             }
277
        }
278
        else
279
             cout << "Everything is dark. You can hear something scurrying around your ➤
                feet." << endLine; //if room is not visible, then output this text</pre>
280
281
        bool titleDisplayed = false; // set as default for items in inventory to not ➤
           be displayed
282
        for (int i = 0; i < playerInventorySize; i++) { // for the players inventory →</pre>
283
            size if there is an item in inventory, then display items in inventory,
           then for items in inventory, give them a slot number
284
             if (currentPlayer->slots[i] != inventory::emptyItem) {
285
286
                 if (!titleDisplayed) {
                     cout << "Your inventory includes:" << endLine;</pre>
287
288
                     titleDisplayed = true;
289
                 }
290
291
                 for (int s = 0; s < inventorySize; s++) {</pre>
292
                     if (currentPlayer->slots[i] == availableItems[s].inventoryType) {
293
294
                         cout << "\t" << availableItems[s].description << "[" << i << →
                          "]" << endLine;
295
                         break;
296
                     }
297
                 }
298
             }
299
        }
300
301
        if (currentPlayer->energy < 5) // if the players energy levels reach 5, this →
           will be displayed
302
             cout << ">>> You are feeling really ill. Maybe you should eat something?" →
                << endLine;
                                        // if the players energy levels reach 10, this >
303
        else
           will be displayed
304
             if (currentPlayer->energy < 10)</pre>
                 cout << ">>> You are feeling weak. Maybe you should eat something?"
305
                   << endLine;</pre>
306 }
307
308 int main()
309 {
        wall walls[2][arraySize + 1][arraySize]; // [horizontal/vertical] [Lines +
310
           1] [walls with a line]
311
         room rooms[arraySize][arraySize]; // [x coordinate] [y coordinate]
312
313
        // initialsie rooms with walls and defaults and empty inventory
314
        for (int x = 0; x < arraySize; x++) {
             for (int y = 0; y < arraySize; y++) {</pre>
315
                 rooms[x][y].northWall = &walls[horizontal][y][x];
316
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
8
```

```
317
                 rooms[x][y].southWall = &walls[horizontal][y + 1][x];
318
                 rooms[x][y].eastWall = &walls[vertical][x + 1][y];
                 rooms[x][y].westWall = &walls[vertical][x][y];
319
320
321
                 for (int i = 0; i < roomInventorySize; i++) {</pre>
                     rooms[x][y].slots[i] = inventory::emptyItem;
322
323
                 }
324
            }
325
        }
326
        // initialise player
327
328
        player currentPlayer;
329
330
        for (int i = 0; i < playerInventorySize; i++)</pre>
331
             currentPlayer.slots[i] = inventory::emptyItem;
332
333
        currentPlayer.addInventory(inventory::matches);
334
335
        // defined rooms
336
337
        rooms[4][4].updateRoom("Home" + endLine + "There is a log fire burning" +
           endLine + "You can smell coffee brewing and everything seems normal",
           door::openDoor, door::openDoor, door::corridor, door::noDoor, true);
338
        rooms[4][4].addInventory(inventory::lantern);
        rooms[4][3].description = "in a field of grass" + endLine + "Mountains can be →
339
            spotted from in the distance";
340
        rooms[4][3].northWall->doorType = openArea;
        rooms[4][2].description = "the footsteps of a mountain pathway" + endLine +
341
           "There is also a mud trail leading into a forest to the west";
342
        rooms[4][2].northWall->doorType = stairCase;
343
        rooms[4][2].westWall->doorType = forest;
344
        rooms[4][2].addInventory(inventory::waterBottle);
        rooms[3][2].description = "inside the forest surrounded by tall trees" +
345
           endLine + "The light being blocked out by them";
346
        rooms[3][2].westWall->doorType = forest;
        rooms[2][2].description = "a forest which is pitch black";
347
348
        rooms[2][2].visible = false;
349
        rooms[2][2].enableVisible = inventory::litLantern;
350
        rooms[2][2].addInventory(inventory::skeletonKey);
351
        rooms[4][1].description = "haendLine way up the mountain pass";
352
        rooms[4][1].northWall->doorType = stairCase;
        rooms[4][0].description = "at the mountains summit" + endLine + "From here
353
          you have a great view of the forest and the fields" + endLine + "As well as →
            the house you came out off";
354
        rooms[4][0].addInventory(inventory::food);
355
        rooms[5][4].description = "a long corridor" + endLine + "There is a family
356
           portrait on the wall, it looks a lot like you.";
357
        rooms[5][4].eastWall->doorType = door::openDoor;
358
        rooms[6][4].description = "You have found your way into a larder" + endLine + →
            "This must be used be storing some food";
        rooms[6][4].visible = false;
359
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
9
```

```
360
        rooms[2][2].enableVisible = inventory::litLantern;
361
        rooms[6][4].addInventory(inventory::food);
362
        rooms[4][5].description = "a staircase" + endLine + "Your uncle used to go
363
          down there to work" + endLine + "back when he was still here";
364
        rooms[4][5].southWall->doorType = stairCase;
        rooms[4][6].description = "a hall at the bottom of the stairs";
365
366
        rooms[4][6].southWall->doorType = lockedDoor;
367
        rooms[4][6].enableUnlock = inventory::skeletonKey;
368
        rooms[4][7].description = "an empty room with a painting of a dog on the
          wall" + endLine + "You don't remember owning a dog";
369
        rooms[4][7].eastWall->doorType = openDoor;
370
        rooms[4][7].westWall->doorType = openDoor;
371
372
        rooms[5][7].description = "a long corridor" + endLine + "You can't quite see →
          the end";
373
        rooms[5][7].eastWall->doorType = corridor;
        rooms[6][7].description = "The long corridor" + endLine + "Torches on the
374
          wall are lighting the way towards the end";
375
        rooms[6][7].eastWall->doorType = corridor;
376
        rooms[7][7].description = "the end of the corridor" + endLine + "The skeleton →
           of a small bird is resting on the floor";
377
        rooms[7][7].southWall->doorType = stairCase;
378
        rooms[7][7].addInventory(inventory::matches);
        rooms[7][8].description = "a labortory" + endLine + "Uncle must have done his →
379
           work in here";
380
        rooms[7][8].eastWall->doorType = openDoor;
381
        rooms[7][8].visible = false;
382
        rooms[2][2].enableVisible = inventory::litLantern;
383
        rooms[8][8].description = "a test chamber" + endLine + "It's a mess" +
          endLine + "looks like a lot of experiments failed in here";
384
        rooms[8][8].eastWall->doorType = openDoor;
        rooms[8][8].visible = false;
385
386
        rooms[2][2].enableVisible = inventory::litLantern;
387
        rooms[9][8].description = "a storage room" + endLine + "Seems like there are →
          all kinds of things stored here" + endLine + "From chemical formulas to
          rations of food";
388
        rooms[9][8].visible = false;
        rooms[2][2].enableVisible = inventory::litLantern;
389
390
        rooms[9][8].addInventory(inventory::food);
391
        rooms[3][7].description = "a natural cave" + endLine + "Enough space for you →
392
          to crouch and walk around";
        rooms[3][7].westWall->doorType = tunnel;
393
394
        rooms[2][7].description = "an enclosed area" + endLine + "Just enough head
          space for you to crawl around" + endLine + "The cave seems to get tighter
          the futher in you go";
395
        rooms[2][7].westWall->doorType = tunnel;
396
        rooms[2][7].southWall->doorType = stairCase;
397
        rooms[1][7].description = "an open area" + endLine + "After squeasing through ➤
           the tunnel you arrived at a door";
        rooms[1][7].westWall->doorType = openDoor;
398
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
399
        rooms[0][7].description = "a singuar room with a opened box";
400
        rooms[0][7].addInventory(inventory::food);
401
        rooms[0][7].addInventory(inventory::goldKey);
402
        rooms[2][8].description = "a dark area" + endLine + "A map of the underground →
           system is on the wall";
403
        rooms[2][8].eastWall->doorType = openArea;
404
        rooms[2][8].visible = false;
405
        rooms[2][2].enableVisible = inventory::litLantern;
406
        rooms[3][8].description = "a thin corridor" + endLine + "There is a large
          vault at the end";
407
        rooms[3][8].southWall->doorType = lockedDoor;
408
        rooms[3][8].enableUnlock = inventory::goldKey;
409
        rooms[3][8].visible = false;
410
        rooms[2][2].enableVisible = inventory::litLantern;
411
        rooms[3][9].description = "inside the vault" + endLine + "You have found your ➤
           family's lost treasures!";
412
        rooms[3][9].treasureRoom = true;
413
414
        while (true) {
            room *currentRoom = &(rooms[currentPlayer.x][currentPlayer.y]); //
415
             Pointer to the room in the array
416
            displayRoom(currentRoom, &currentPlayer);
417
418
           if (currentPlayer.energy <= 0) { // if your energy reaches 0 you will die</pre>
                      << "************ << endLine
419
                       << "** YOU HAVE DIED!! **" << endLine</pre>
420
                       421
422
               break;
423
            }
424
           if (currentRoom->treasureRoom) { // upon reaching the treasure room you
425
             have won the game
                       426
               cout
427
                       << "** Congratulations you have found the family treasue **" >
                       << endLine
                       428
                       << endLine;</pre>
429
               break;
430
            }
431
            string action;
432
433
           cout << "What would you like to do?" << endLine;</pre>
434
            cout << "----" << endLine;</pre>
435
436
           cin >> action;
            cout << "----" << endLine;
437
438
439
440
           if (action == "help") { // after inputting 'help' a list of actions which →
              can be used in the game will be listed
               cout << "Actions are
441
                                                                                P
```

```
'north','south','east','west','pickup','drop','use','quit'" <<</pre>
                   endLine;
442
                 continue;
443
             }
444
445
             // exit game
446
             if (action == "quit") { // action which quits the game
                 cout << "Thank you for playing... See you in game soon." << endLine;</pre>
447
448
                 break;
449
             }
450
451
             // Movement options
             if (action == "north") { // action moves the player north of current
452
                                                                                          P
               room, if the player cannot go north an output will display and the
               player can change their action
453
                 if (!currentPlayer.moveNorth(currentRoom->northWall->doorType))
                     cout ⟨⟨ ">>> Ouch! you are dazed and confused after hitting your >
454
                       head." << endLine;</pre>
455
456
                 continue;
457
             }
458
             if (action == "south") { // action moves the player south of current
459
               room, if the player cannot go north an output will display and the
                                                                                          P
               player can change their action
460
                 if (!currentPlayer.moveSouth(currentRoom->southWall->doorType))
461
                     cout << ">>> Ouch! you are dazed and confused after hitting your →
                       head." << endLine;</pre>
462
463
                 continue;
464
             }
465
             if (action == "east") { // action moves the player east of current room, ➤
466
               if the player cannot go north an output will display and the player can →
                change their action
                 if (!currentPlayer.moveEast(currentRoom->eastWall->doorType))
467
468
                     cout << ">>> Ouch! you are dazed and confused after hitting your ➤
                       head." << endLine;</pre>
469
470
                 continue;
471
             }
472
473
             if (action == "west") { // action moves the player west of current room, →
               if the player cannot go north an output will display and the player can ➤
                change their action
474
                 if (!currentPlayer.moveWest(currentRoom->westWall->doorType))
475
                     cout << ">>> Ouch! you are dazed and confused after hitting your →
                       head." << endLine;</pre>
476
477
                 continue;
478
             }
479
```

```
if (action == "pickup") { // action which picks up item from room
480
                                                                                           P
               inventory and puts it in player inventory
481
                 int pickSlot;
482
483
                 cout << "What do you want to pick up [Number]?" + endLine;</pre>
484
                 cin >> pickSlot;
485
486
                 if (pickSlot > -1
487
                     && pickSlot < roomInventorySize) {
488
                      inventory item = currentRoom->slots[pickSlot];
489
490
                      if (item != inventory::emptyItem) {
491
                          if (currentPlayer.addInventory(item)) {
492
                              currentRoom->removeInventory(item);
493
                              continue;
494
                          }
495
                     }
496
                 }
497
498
                 cout << ">>> Your bags are full or I can't find that item." <</pre>
                                                                                           P
                   endLine; // if there is no room in player inventory, this message
                   will display
499
                 continue;
500
             }
501
502
             if (action == "drop") { // action which drops item from player inventory →
               to room inventory
503
                 int dropSlot;
504
505
                 cout << "What do you want to drop [Number]?" + endLine;</pre>
506
                 cin >> dropSlot;
507
508
                 if (dropSlot > -1
509
                      && dropSlot < playerInventorySize) {
510
                      inventory item = currentPlayer.slots[dropSlot];
511
512
                      if (item != inventory::emptyItem) {
513
                          if (currentPlayer.removeInventory(item)) {
514
                              currentRoom->addInventory(item);
515
                              continue;
516
                          }
517
                     }
518
                 }
519
                 cout << ">>> Your bags are full or I can't find that item." <</pre>
520
                   endLine; // if the player doesn't have the item typed, this message ₹
                    will display
521
                 continue;
522
             }
523
524
             if (action == "use") { // action will use an item in the players
                                                                                           P
               inventory
```

```
E:\TreasureHunt\TreasureHunt\TreasureHunt.cpp
```

```
525
                 int useSlot;
526
                 cout << "What do you want to use [Number]?" + endLine;</pre>
527
528
                 cin >> useSlot;
529
530
                 if (useSlot > -1
531
                      && useSlot < playerInventorySize) {
532
                      inventory item = currentPlayer.slots[useSlot];
533
534
                      if (item == inventory::matches) {
                                                            // if the item is matches
                        and there is a lantern in player inventory, it will light up
                        the lantern
                          for (int i = 0; i < playerInventorySize; i++) {</pre>
535
536
                              if (currentPlayer.slots[i] == inventory::lantern) {
537
                                  currentPlayer.slots[i] = inventory::litLantern;
538
                                  continue;
539
                              }
540
                          }
541
                      }
542
543
                      if (item == inventory::food) { // if the item is food and the
                        player doesn't have max energy, the food will be used up from
                        inventory and energy will increase by 5
544
                          if (!currentPlayer.eatFoods())
                              cout << ">>> You are full and cannot eat any more." <</pre>
545
                          endLine;
546
                          else {
547
                              currentPlayer.removeInventory(inventory::food);
548
                              cout << "Yummy that was delicious." << endLine;</pre>
549
                          }
550
551
                          continue;
552
                      }
553
554
                     if (currentRoom->enableUnlock == item) { // if the item is a key →
                        it will unlock a locked door, if it is the wrong key nothing
                        will happen
555
                          if (!currentRoom->unlockDoor())
556
                              cout << "Click... Nothing happened." << endLine;</pre>
557
558
                              cout << "Click... You've unlocked the door." << endLine;</pre>
559
560
                          continue;
561
                     }
562
                 }
563
                 cout << ">>> You cannot do that right now!" << endLine; // if the</pre>
564
                   action can't be done this message will display
565
                 continue;
566
             }
567
             cout << ">>>> I'm sorry I don't understand what you mean! (help
568
```

```
available)" << endLine; // if a command which is unknown is input, this →
message will display

569 };

570

571 return 0;

572 }

573

574

575

576
```