

```

1  #include "libRecipe.h"
2
3  #include <cctype>
4  #include <ctime>
5
6  using namespace RecipeSpace;
7
8  int main(int argc, char** argv)
9  {
10     srand(time(nullptr));
11
12     if(argc != 7){
13         cerr << "Wrong number of arguments: <Rows> <Cols> <Bush Chance> <Trees> <Flint Stones>
14         <Turns>" << endl;
15         exit(ERROR_COUNT);
16     }
17
18     int intRows = convStrToInt(argv[1]);
19     int intCols = convStrToInt(argv[2]);
20     int intBushChance = convStrToInt(argv[3]);
21     int intNoTrees = convStrToInt(argv[4]);
22     int intNoFlintStones = convStrToInt(argv[5]);
23     int intNoTurns = convStrToInt(argv[6]);
24
25     //Create GAME
26     GameWorld theGame = createGame(intRows, intCols, intNoTrees, intNoFlintStones,
27     intBushChance, intNoTurns);
28
29     char chInput = '\0';
30     bool blnContinue = true;
31
32     do{
33         system("cls");
34         printGame(theGame);
35         cin >> chInput;
36         chInput = tolower(chInput);
37         switch(chInput){
38             case 'a':
39             case 'd':
40             case 'w':
41             case 'x':
42             case 'q':
43             case 'e':
44             case 'z':
45             case 'c':
46             {
47                 movePlayer(theGame, chInput);
48                 break;
49             }
50             case 'j':
51             {
52                 perfromTask(theGame, CHOP_WOOD);
53                 break;
54             }
55             case 'k':
56             {
57                 perfromTask(theGame, CRAFT_ITEM);
58                 break;
59             }
60             case 'l':
61             {
62                 perfromTask(theGame, LIGHT_FIRE);
63                 break;
64             }
65             case 'p':
66             {
67                 blnContinue = false;
68                 break;
69             }
70             default:
71             {
72                 cerr << "Please enter the correct characters" << endl;
73             }
74         }
75
76         //Check Game State
77         checkGameState(theGame);
78
79         if(theGame.state == WON || theGame.state == LOST){
80             blnContinue = false;
81         }
82     }while(blnContinue);

```

```
83 //Check how the game exited
84 cout << "======" << endl;
85 if(theGame.state == WON){
86     cout << "You won the game, you lit the fire" << endl;
87 }else if(theGame.state == LOST){
88     cout << "You lost the game, run out of turns before darkness" << endl;
89 }
90 cout << "======" << endl;
91
92
93 //Free Memory, give it back to the free store
94 deallocMemory(theGame.arrGame, theGame.dimGameWorld.intRows);
95
96 return 0;
97 }
98
```

```

1  #ifndef LIBRECIPE_H_INCLUDED
2  #define LIBRECIPE_H_INCLUDED
3
4  #include <iostream>
5  #include <cassert>
6
7  using namespace std;
8
9  namespace RecipeSpace{
10
11     enum gameStatus{
12         SUCCESS, ERROR_CONV, ERROR_COUNT
13     };
14
15     enum gameState{
16         RUNNING, LOST, WON
17     };
18
19     enum gameFeatures{
20         SPACE, PLAYER, BUSH, TREE, FLINT, FIRE, PLAYER_TREE
21     };
22
23     enum playerTasks{
24         CRAFT_ITEM, CHOP_WOOD, LIGHT_FIRE
25     };
26
27     const char CHARACTERS[7] = {'.', 'P', 'B', 'T', 'E', 'O', 'P'};
28
29     typedef int* OneDArray;
30     typedef OneDArray* TwoDArray;
31
32     struct Position{
33         int intRow;
34         int intCol;
35     };
36
37     struct Dimension{
38         int intRows;
39         int intCols;
40     };
41
42     struct GameWorld{
43         // int intRows;
44         // int intCols;
45         Dimension dimGameWorld;
46
47         Position posPlayer;
48         TwoDArray arrGame;
49         int numSticks;
50         int numFlints; //Collected
51         int numLogs;
52         int numAxes;
53         int numFireKits;
54         int numTurns;
55         gameState state;
56     };
57
58     //Functions
59     int convStrToInt(string strNum);
60     int generateRandomNum(int intLower, int intUpper);
61     GameWorld createGame(int intRows, int intCols, int numTrees, int numFlint, int bushChance,
62 int numTurns);
63     void printGame(const GameWorld& myGame);
64     void movePlayer(GameWorld& myGame, char chInput);
65     void performTask(GameWorld& MyGame, int intInput);
66     void checkGameState(GameWorld& myGame);
67     void deallocMemory(TwoDArray& arrGame, int intRows);
68 }
69 #endif // LIBRECIPE_H_INCLUDED
70

```

```

1  #include "libRecipe.h"
2  #include <sstream>
3
4  namespace RecipeSpace{
5
6      int convStrToInt(string strNum){
7          int intNum = 0;
8          stringstream ss {strNum};
9          ss >> intNum;
10         if(ss.fail()){
11             cerr << "Failed to convert from string to integer" << endl;
12             exit(ERROR_CONV);
13         }
14         return intNum;
15     }
16
17     int generateRandomNum(int intLower, int intUpper){
18         int intRange = intUpper - intLower + 1;
19         return rand() % intRange + intLower;
20     }
21
22     void placeFeature(GameWorld myGame, int intFeature, int intFeatureCount){
23         int intRow = 0;
24         int intCol = 0;
25         for(int i = 0; i < intFeatureCount; i++){
26             intRow = generateRandomNum(0, myGame.dimGameWorld.intRows - 1);
27             intCol = generateRandomNum(0, myGame.dimGameWorld.intCols - 1);
28             while(myGame.arrGame[intRow][intCol] != SPACE){
29                 intRow = generateRandomNum(0, myGame.dimGameWorld.intRows - 1);
30                 intCol = generateRandomNum(0, myGame.dimGameWorld.intCols - 1);
31             }
32             myGame.arrGame[intRow][intCol] = intFeature;
33         }
34     }
35
36     TwoDArray initGame(int intRows, int intCols){
37         TwoDArray arrGame;
38         arrGame = new OneDArray[intRows];
39         for(int r = 0; r < intRows; r++){
40             arrGame[r] = new int[intCols];
41             for(int c = 0; c < intCols; c++){
42                 arrGame[r][c] = SPACE;
43             }
44         }
45         return arrGame;
46     }
47
48     GameWorld createGame(int intRows, int intCols, int numTrees, int numFlint, int bushChance,
49 int numTurns){
50         GameWorld myGame;
51         myGame.dimGameWorld = {intRows, intCols};
52         myGame.state = RUNNING;
53
54         //Allocate Memory and initialize values to space
55         myGame.arrGame = initGame(intRows, intCols);
56
57         //Place Game Characters/Features
58
59         //Player Struct
60         myGame.posPlayer.intRow = intRows/2;
61         myGame.posPlayer.intCol = intCols/2;
62         //Player in Array
63         myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = PLAYER;
64
65         // Trees
66         placeFeature(myGame, TREE, numTrees);
67         // Flints
68         placeFeature(myGame, FLINT, numFlint);
69         // Bushes
70         for(int r = 0; r < intRows; r++){
71             for(int c = 0; c < intCols; c++){
72                 int intChance = generateRandomNum(0, 100);
73                 if(intChance <= bushChance && myGame.arrGame[r][c] == SPACE){
74                     myGame.arrGame[r][c] = BUSH;
75                 }
76             }
77         }
78
79         //Units of items
80         myGame.numSticks = 0;
81         myGame.numFlints = 0;
82         myGame.numLogs = 0;
83         myGame.numAxes = 0;
84         myGame.numFireKits = 0;

```

```

84
85     //Turns left
86     myGame.numTurns = numTurns;
87
88     return myGame;
89 }
90
91 void printGame(const GameWorld& myGame){
92     for(int r = 0; r < myGame.dimGameWorld.intRows; r++){
93         for(int c = 0; c < myGame.dimGameWorld.intCols; c++){
94             int index = myGame.arrGame[r][c];
95             cout << CHARACTERS[index] << " ";
96         }
97         cout << endl;
98     }
99
100     //Menu
101     cout << endl;
102     cout << "=====" << endl;
103     cout << "A) Left D) Right W) uP X) Down Q) Top Left E) Top Right Z) Bottom Left C)
Bottom Right" << endl;
104     cout << "P) Quit Game " << endl;
105     cout << "=====" << endl;
106     cout << "Other actions: J) Chop Wood K) Craft an Item L) Light a Fire" << endl;
107     cout << "=====" << endl;
108     cout << "Items carried => Sticks: " << myGame.numSticks << " Flints: " <<
myGame.numFlints << " Logs: " << myGame.numLogs <<
    " Axes: " << myGame.numAxes << " Fire Kits: " << myGame.numFireKits << endl <<
    "Turns Left: " << myGame.numTurns << endl;
111 }
112
113 bool isValid(int intRows, int intCols, int intDR, int intDC){
114     return (intDR >= 0 && intDR < intRows && intDC >= 0 && intDC < intCols);
115 }
116
117 void movePlayer(GameWorld& myGame, char chInput){
118     int intDR = myGame.posPlayer.intRow;
119     int intDC = myGame.posPlayer.intCol;
120
121     switch(chInput){
122     case 'a':
123     {
124         intDC--;
125         break;
126     }
127     case 'd':
128     {
129         intDC++;
130         break;
131     }
132     case 'w':
133     {
134         intDR--;
135         break;
136     }
137     case 'x':
138     {
139         intDR++;
140         break;
141     }
142     case 'q':
143     {
144         intDR--;
145         intDC--;
146         break;
147     }
148     case 'e':
149     {
150         intDR--;
151         intDC++;
152         break;
153     }
154     case 'z':
155     {
156         intDR++;
157         intDC--;
158         break;
159     }
160     case 'c':
161     {
162         intDR++;
163         intDC++;
164         break;
165     }

```

```

166     }
167
168     //Valid move? In bounds?
169     if (isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols, intDR, intDC)) {
170         //Set the destination feature and update item inventory
171         if (myGame.arrGame[intDR][intDC] == BUSH) {
172             myGame.numSticks++;
173             myGame.arrGame[intDR][intDC] = PLAYER;
174         } else if (myGame.arrGame[intDR][intDC] == FLINT) {
175             myGame.numFlints++;
176             myGame.arrGame[intDR][intDC] = PLAYER;
177         } else if (myGame.arrGame[intDR][intDC] == TREE) {
178             myGame.arrGame[intDR][intDC] = PLAYER_TREE;
179         } else if (myGame.arrGame[intDR][intDC] == SPACE) {
180             myGame.arrGame[intDR][intDC] = PLAYER;
181         }
182
183         //Update the source location
184         if (myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] == PLAYER_TREE) {
185             myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = TREE;
186         } else {
187             myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = SPACE;
188         }
189
190         myGame.posPlayer = {intDR, intDC};
191         myGame.numTurns--;
192     }
193 }
194
195 void performTask(GameWorld& myGame, int intInput) {
196     if (intInput == CHOP_WOOD) {
197         if (myGame.numAxes >= 1) {
198             //Remove trees in a one square radius of player
199             for (int r = myGame.posPlayer.intRow - 1; r <= myGame.posPlayer.intRow + 1; r++) {
200                 for (int c = myGame.posPlayer.intCol - 1; c <= myGame.posPlayer.intCol + 1;
201 c++) {
202                     if (isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols, r,
203 c) && myGame.arrGame[r][c] == TREE) {
204                         //Add one log to players collection for each tree
205                         myGame.numLogs++;
206                         //Remove tree
207                         myGame.arrGame[r][c] = SPACE;
208                     }
209                 }
210             }
211         } else if (intInput == CRAFT_ITEM) {
212             //Must carry right amount of materials
213             if (myGame.numSticks >= 1 && myGame.numFlints >= 1) {
214                 //Create axe
215                 myGame.numAxes++;
216                 //Decrement materials
217                 myGame.numSticks--;
218                 myGame.numFlints--;
219             }
220             if (myGame.numSticks >= 2 && myGame.numFlints >= 1 && myGame.numLogs >= 3) {
221                 //Create fire kit
222                 myGame.numFireKits++;
223                 //Decrement materials
224                 myGame.numSticks -= 2;
225                 myGame.numFlints -= 1;
226                 myGame.numLogs -= 3;
227             }
228         } else if (intInput == LIGHT_FIRE) {
229             if (myGame.numFireKits >= 1) {
230                 //Light fire in gameworld position nearby
231                 // Assumption: Fire can burn anything
232                 int intRow = generateRandomNum(-1, 1) + myGame.posPlayer.intRow;
233                 int intCol = generateRandomNum(-1, 1) + myGame.posPlayer.intCol;
234                 while (!isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols,
235 intRow, intCol)) {
236                     intRow = generateRandomNum(-1, 1) + myGame.posPlayer.intRow;
237                     intCol = generateRandomNum(-1, 1) + myGame.posPlayer.intCol;
238                 }
239                 myGame.arrGame[intRow][intCol] = FIRE;
240
241                 //Maybe, reduce on the firekit numbers
242                 myGame.numFireKits--;
243                 //Won game
244                 myGame.state = WON;
245             }
246         }
247     }
248 }

```

```
247
248     void checkGameState(GameWorld& myGame) {
249         if(myGame.numTurns == 0){
250             myGame.state = LOST;
251         }
252     }
253
254     void deallocMemory(TwoDArray& arrGame, int intRows){
255         assert(arrGame != nullptr);
256         for(int r = 0; r < intRows; r++){
257             delete [] arrGame[r];
258         }
259         delete [] arrGame;
260         arrGame = nullptr;
261     }
262 }
263
```