**Project 1**

**<War Chess>**

**CIS17A-44051**

**Name: Cody Steimle**

**Date: 04/07/17**

**Introduction:**

Title: War Chess

Taking the idea of how a standard Chess game works, I decided to do my project based around that concept but with a twist. Instead of the King being the most important piece with 17 other pieces each having their own movement traits, I have a total of 6 pieces for each player with 3 different classes. Each player, the user and an A.I, will have a total of: 3 Infantry units, 2 Commanders, and 1 General. The General is what acts as the King piece of the game, once that piece is lost the opposing player will win the game.

Each piece’s traits are stored within a structure called “Army.h”, and these traits are: the health remaining, attack power, movement length, and its location relative to the board.

Infantry can move up to three spaces at a time, but only have 1 attack power and 1 health point. Commanders are the balanced units with health, attack, and movement all being 2. Generals have 3 health points, but attack and movement are only 1, so they will need to be protected.

These pieces are setup for each of the players to maintain a balanced game. A turn consists of 1 attack and 1 movement action before ending.

For each turn, the player has the option to: 1) check a unit’s status by printing out data from its structure, 2) select a unit to move and where to, 3) attack an adjacent unit, and 4) the option to end their turn early if so desired. However, movement and attacking has limitations. When moving the player is forced to stay within bounds of the map and will not be allowed to land on top of other unit. Also for both movement and attacking, diagonals are not accepted.

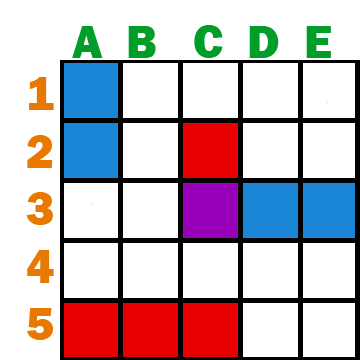
The computer has separate functions for turn, attacking, and moving so the inputs are randomized.

The player will enter their name which will be used throughout the game and also printed to the results file.

There are 2 health packs randomly placed within the map, and if a unit were to land on it, then that exact unit will gain +2 to their remaining health (this means a General could have a total of 7 health if it were to collect both!).

Before the game begins a coin toss is done to find out who will be taking the first turn. After that is decided the game will alternate between players as they take their turns. The turns are kept track of will be written to the results file post-game.

Once the game is over the player will be shown the winner and will also be asked if they would like to print the results to a file.

\*Visual example

**Development Summary:**

Project size: about 1100 Lines

Number of variables: about 70

Number of functions: about 25

Number of structures: 1

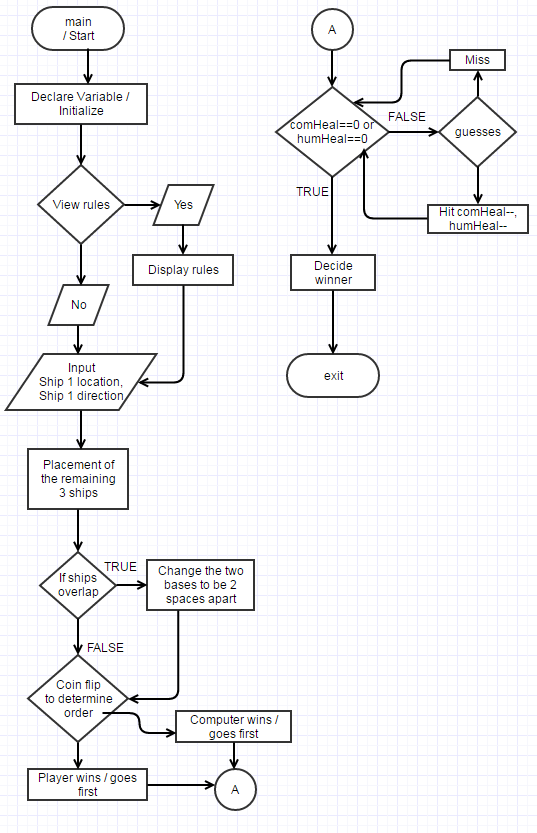
Enter the summary of development here.

Use multiple paragraphs if needed.

**Chapter Concepts:**

|  |  |  |  |
| --- | --- | --- | --- |
| Textbook | Chap.Sec | Concept | Code Location |
| Savitch 9th | 1.3 | #include | Lines 9-13 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Flowchart:**



**Pseudo Code:**

*Initialize*

*Ask if the user would like to view the rules*

*If the user wishes to show rules*

*Show the rules*

*Else continue to the placement of the four ships*

*Fill the game board*

*Prompt the user to set the 1x2 piece.*

*Randomize the placement of the second piece*

*If the piece overlaps*

*Shift the base two spaces apart*

*Else begin setting the Comp pieces*

*Randomize the 1x2 Comp piece*

*Randomize the 1x3 Comp piece until no overlap*

*Coin flip calculation to see if Comp or Player goes first*

*If Comp wins*

*Comp guess first*

*Else Player Wins*

*Skip to Player’s turn*

*Both continue to guess until all pieces of the ships on a single side have been guessed.*

*Calculate and display the winner.*

*Prompt the user to save results to file*

*Exit Program.*

**Variables:**

|  |  |  |  |
| --- | --- | --- | --- |
| Data Type | Name | Location | Description |
| const int | ROWS=5 | global | Number of rows for game board |
| const int | COLS=5 | global | Number of columns for game board |
| string | Line | main | Place holder line |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Functions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Type Name | Purpose | Inputs | Outputs |
| char comAtk | Randomizes computer guess | compCol, compRow | compCol, compRow |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reference:**

1. Textbook (Savitch 9th Edition).
2. Textbook (Gaddis 8th Edition).
3. Mark Lehr GitHub repository.

**Program (main):**

//Set random number seed

srand(static\_cast<unsigned int>(time(0)));

//Declaration of Variables

string line; //Place holder line

char compCol, compRow; //Computer Attack

int shipDir; //Ship direction for random

char chosDir; //Ship direction

char human; //Player inputs

char humCol, humRow; //Player's Attack

char comHeal='5'; //Total computer health

char humHeal='5'; //Total player health

int turn=0; //Tracks turn count

string name; //Player's Name

bool winner; //Winning player (0 - Computer, 1 - Player)

char board[ROWS][COLS]; //The board

//Small ship Human

char hsmlC1, hsmlR1; //1x2 Base

char hsmlC2, hsmlR2; //1x2 Tail

char hsmlDr; //Direction

//Small ship Comp

char csmlC1, csmlR1; //1x2 Base

char csmlC2, csmlR2; //1x2 Tail

char csmlDr; //Direction

//Medium ship Human

char hmedC1, hmedR1; //1x3 Base

char hmedC2, hmedR2; //1x3 Tail

char hmedC3, hmedR3; //1x3 Tail

char hmedDr; //Direction

//Medium ship Comp

char cmedC1, cmedR1; //1x3 Base

char cmedC2, cmedR2; //1x3 Tail

char cmedC3, cmedR3; //1x3 Tail

char cmedDr; //Direction

//Initialize

cout<<"Welcome to the program that emulates a abridged version of the "

"classic board game, Battleship."<<endl;

//Rules

disRule();

//Game Setup

cout<<"What is your desired name? (no spaces)"<<endl;

cin>>name; //Input Name

cout<<"Alright then "<<name<<". Get ready for a game of Battleship!"<<endl;

//Set an empty board

fillAry(board);

//Player's Small Ship

cout<<"Okay, now choose the placement of the smaller ship. (1x2)"<<endl;

do{ //Validate Input

cout<<"Choose a column (A-E, case sensitive):";

cin>>hsmlC1;

}while(hsmlC1<65||hsmlC1>69);

do{ //Validate Input

cout<<"Choose a row (1-5):";

cin>>hsmlR1;

}while(hsmlR1<49||hsmlR1>53);

//Direction of Player's Small Ship

do{ //Validate Input

cout<<"Now choose either Vertical (V) or Horizontal (H):";

cin>>chosDir;

}while(chosDir!='h'&&chosDir!='H'&&chosDir!='v'&&chosDir!='V');

switch(chosDir){

case 'h':

case 'H': {

cout<<"You picked Horizontal."<<endl;

hsmlDr='H';};break;

case 'v':

case 'V': {

cout<<"You picked Vertical."<<endl;

hsmlDr='V';}

}

if(hsmlDr=='V'){ //Vertical

hsmlC2=hsmlC1;

if(hsmlR1+1>53){ //Keep the ship inside the board

hsmlC2=hsmlC1-1;

}

else{

hsmlR2=hsmlR1+1;

}

}

else if(hsmlDr=='H'){ //Horizontal

if(hsmlC1+1>69){

hsmlC2=hsmlC1-1;

}

else{

hsmlC2=hsmlC1+1;

}

hsmlR2=hsmlR1;

}

//Computer's Small Ship

csmlC1=rand()%5+65; //Calls random function, then modifies to A-E

csmlR1=rand()%5+49; //Calls random function, then modifies to 1-5

//Direction of Computer's Small Ship

shipDir=rand()%10+1;

if(shipDir<=5){ //Vertical

csmlDr='V';

csmlC2=csmlC1;

if(csmlR1+1>53){

csmlR2=csmlR1-1;

}

else{

csmlR2=csmlR1+1;

}

}

else if(shipDir>5){ //Horizontal

csmlDr='H';

if(csmlC1+1>69){ //Keep the ship inside the board

csmlC2=csmlC1-1;

}

else{

csmlC2=csmlC1+1;

}

csmlR2=csmlR1;

}

//Computer's Medium Ship

do{

cmedC1=rand()%5+65; //Calls random function, then modifies to A-E

cmedR1=rand()%5+49; //Calls random function, then modifies to 1-5

}while(cmedC1==csmlC1||cmedR1==csmlR1);

//Direction of Computer's Medium Ship

shipDir=rand()%10+1;

if(shipDir<=5){ //Vertical

cmedDr='V';

if(cmedDr==csmlDr){ //If the directions are the same

if(csmlC1==cmedC1){

if(cmedC1+1>69){ //Keep the ship inside the board

cmedC1=cmedC1-1;

}

else{

cmedC1=cmedC1+1;

}

}

if(cmedR1+2>53){ //Keep the ship inside the board

cmedR2=cmedR1-1;

cmedR3=cmedR2-1;

}

else{

cmedR2=cmedR1+1;

cmedR3=cmedR2+1;

}

cmedC2=cmedC1;

cmedC3=cmedC1;

}

else{

if(csmlC1+2>69){ //Keep the ship inside the board

cmedC1=csmlC1-2;

}

else{

cmedC1=csmlC1+2;

}

if(cmedR1+2>53){ //Keep the ship inside the board

cmedR2=cmedR1-1;

cmedR3=cmedR2-1;

}

else{

cmedR2=cmedR1+1;

cmedR3=cmedR2+1;

}

cmedC2=cmedC1;

cmedC3=cmedC1;

}

}

else if(shipDir>5){ //Horizontal

cmedDr='H';

if(cmedDr==csmlDr){ //If the directions are the same

if(csmlR1==cmedR1){

if(cmedR1+1>53){ //Keep the ship inside the board

cmedR1=cmedR1-1;

}

else{

cmedR1=cmedR1+1;

}

}

if(cmedC1+2>69){ //Keep the ship inside the board

cmedC2=cmedC1-1;

cmedC3=cmedC2-1;

}

else{

cmedC2=cmedC1+1;

cmedC3=cmedC2+1;

}

cmedR2=cmedR1;

cmedR3=cmedR1;

}

else{

if(csmlR1+2>53){ //Keep the ship inside the board

cmedR1=csmlR1-2;

}

else{

cmedR1=csmlR1+2;

}

if(cmedC1+2>69){ //Keep the ship inside the board

cmedC2=cmedC1-1;

cmedC3=cmedC2-1;

}

else{

cmedC2=cmedC1+1;

cmedC3=cmedC2+1;

}

cmedR2=cmedR1;

cmedR3=cmedR1;

}

}

//Player's Medium Ship

cout<<"Okay, now the placement of the second ship will be random to prevent"

" overlapping of pieces."<<endl;

do{

hmedC1=rand()%5+65; //Calls random function, then modifies to A-E

hmedR1=rand()%5+49; //Calls random function, then modifies to 1-5

}while(hmedC1==hsmlC1||hmedR1==hsmlR1);

//Direction of Computer's Medium Ship

shipDir=rand()%10+1;

if(shipDir<=5){ //Vertical

hmedDr='V';

if(hmedDr==hsmlDr){ //If the directions are the same

if(hsmlC1==hmedC1){

if(hmedC1+1>69){ //Keep the ship inside the board

hmedC1=hmedC1-1;

}

else{

hmedC1=hmedC1+1;

}

}

if(hmedR1+2>53){

hmedR2=hmedR1-1;

hmedR3=hmedR2-1;

}

else{

hmedR2=hmedR1+1;

hmedR3=hmedR2+1;

}

hmedC2=hmedC1;

hmedC3=hmedC1;

}

else{

if(hsmlC1+2>69){ //Keep the ship inside the board

hmedC1=hsmlC1-2;

}

else{

hmedC1=hsmlC1+2;

}

if(hmedR1+2>53){ //Keep the ship inside the board

hmedR2=hmedR1-1;

hmedR3=hmedR2-1;

}

else{

hmedR2=hmedR1+1;

hmedR3=hmedR2+1;

}

hmedC2=hmedC1;

hmedC3=hmedC1;

}

}

else if(shipDir>5){ //Horizontal

hmedDr='H';

if(hmedDr==hsmlDr){ //If the directions are the same

if(hsmlR1==hmedR1){

if(hmedR1+1>53){

hmedR1=hmedR1-1;

}

else{

hmedR1=hmedR1+1;

}

}

if(hmedC1+2>69){ //Keep the ship inside the board

hmedC2=hmedC1-1;

hmedC3=hmedC2-1;

}

else{

hmedC2=hmedC1+1;

hmedC3=hmedC2+1;

}

hmedR2=hmedR1;

hmedR3=hmedR1;

}

else{

if(hsmlR1+2>53){ //Keep the ship inside the board

hmedR1=hsmlR1-2;

}

else{

hmedR1=hsmlR1+2;

}

if(hmedC1+2>69){ //Keep the ship inside the board

hmedC2=hmedC1-1;

hmedC3=hmedC2-1;

}

else{

hmedC2=hmedC1+1;

hmedC3=hmedC2+1;

}

hmedR2=hmedR1;

hmedR3=hmedR1;

}

}

//Display ship locations

cout<<"-------------------------------------------------------------"<<endl;

cout<<name<<", your ship locations are:"<<endl;

cout<<"Small: "<<hsmlC1<<hsmlR1<<" "<<hsmlC2<<hsmlR2<<endl;

cout<<"Medium: "<<hmedC1<<hmedR1<<" "<<hmedC2<<hmedR2<<" "<<hmedC3<<hmedR3

<<endl;

cout<<"-------------------------------------------------------------"<<endl;

//Game Start!

if(coinFlp()==1){

cout<<name<<" will go first."<<endl;

}

else{

cout<<"Computer will go first."<<endl;

cout<<endl;

turn++;

cout<<" Turn #"<<turn<<endl;

comAtk(compCol, compRow);

if(compCol==hsmlC1&&compRow==hsmlR1){

cout<<"HIT!"<<endl;

humHeal--;

hsmlC1='0';

hsmlR1='0';

}

else if(compCol==hsmlC2&&compRow==hsmlR2){

cout<<"HIT!"<<endl;

humHeal--;

hsmlC2='0';

hsmlR2='0';

}

else{

cout<<"Miss..."<<endl;

}

cout<<name<<"'s remaining health: "<<humHeal<<endl; //Displays User HP

}

do{

//Player's Turn

cout<<endl;

turn++;

cout<<" Turn #"<<turn<<endl;

cout<<"------------------"<<name<<"'s Turn------------------"<<endl;

cout<<"Make your guess."<<endl;

do{ //Validate Input

cout<<"Choose a column (A-E, case sensitive):"<<endl;

cin>>humCol;

}while(humCol<65||humCol>69); //65 is char 'A', 69 is char 'E'

do{ //Validate Input

cout<<"Choose a row (1-5):"<<endl;

cin>>humRow;

}while(humRow<49||humRow>53); //49 is char '1', 53 is char '5'

cout<<name<<" attacks "<<humCol<<"-"<<humRow<<"!"<<endl;

if(humCol==csmlC1&&humRow==csmlR1){

cout<<"HIT!"<<endl;

humRow=humRow-49; //Converts to 0-4

humCol=humCol-65; //Converts to 0-4

board[humRow][humCol]='X'; //Marker

comHeal--;

csmlC1='0';

csmlR1='0';

}

else if(humCol==csmlC2&&humRow==csmlR2){

cout<<"HIT!"<<endl;

humRow=humRow-49; //Converts to 0-4

humCol=humCol-65; //Converts to 0-4

board[humRow][humCol]='X'; //Marker

comHeal--;

csmlC2='0';

csmlR2='0';

}

else if(humCol==cmedC1&&humRow==cmedR1){

cout<<"HIT!"<<endl;

humRow=humRow-49; //Converts to 0-4

humCol=humCol-65; //Converts to 0-4

board[humRow][humCol]='X'; //Marker

comHeal--;

cmedC1='0';

cmedR1='0';

}

else if(humCol==cmedC2&&humRow==cmedR2){

cout<<"HIT!"<<endl;

humRow=humRow-49; //Converts to 0-4

humCol=humCol-65; //Converts to 0-4

board[humRow][humCol]='X'; //Marker

comHeal--;

cmedC2='0';

cmedR2='0';

}

else if(humCol==cmedC3&&humRow==cmedR3){

cout<<"HIT!"<<endl;

humRow=humRow-49; //Converts to 0-4

humCol=humCol-65; //Converts to 0-4

board[humRow][humCol]='X'; //Marker

comHeal--;

cmedC3='0';

cmedR3='0';

}

else{

cout<<"Miss..."<<endl;

}

prntAry(board); //Print the board

if(comHeal=='0')break; //Breaks out of the code when the computer dies

//Computer Turn

cout<<endl;

turn++;

cout<<" Turn #"<<turn<<endl;

comAtk(compCol, compRow);

if(compCol==hsmlC1&&compRow==hsmlR1){

cout<<"HIT!"<<endl;

humHeal--;

hsmlC1='0';

hsmlR1='0';

}

else if(compCol==hsmlC2&&compRow==hsmlR2){

cout<<"HIT!"<<endl;

humHeal--;

hsmlC2='0';

hsmlR2='0';

}

else if(compCol==hmedC1&&compRow==hmedR1){

cout<<"HIT!"<<endl;

humHeal--;

hmedC1='0';

hmedR1='0';

}

else if(compCol==hmedC2&&compRow==hmedR2){

cout<<"HIT!"<<endl;

humHeal--;

hmedC2='0';

hmedR2='0';

}

else if(compCol==hmedC3&&compRow==hmedR3){

cout<<"HIT!"<<endl;

humHeal--;

hmedC3='0';

hmedR3='0';

}

else{

cout<<"Miss..."<<endl;

}

cout<<name<<"'s remaining health: "<<humHeal<<endl; //Displays User HP

}while(humHeal!='0'); //Breaks out of the code when the users dies

//Post Game

cout<<endl;

if((comHeal=='0')&&(humHeal!='0')){

cout<<"\*\*\*\*\*"<<name<<" Won!\*\*\*\*\*"<<endl;

winner=1;

}

else{

cout<<"\*\*\*\*\*You Lose.\*\*\*\*\*"<<endl;

winner=0;

}

cout<<endl;

//Store Winner

if(winner==0){

line='Comp';

}

if(winner==1){

line=name;

}

//Results

savGame(turn, line);

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\* Thanks for playing! \*"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

//Exit Program

return 0;

}