NATIONAL PUBLIC SCHOOL

HSR LAYOUT

BANGALORE

**COMPUTER SCIENCE**

**PROJECT**

COMMAND LINE CHECKERS

Done By : Atharva Raykar && Jaideep Sabharwal

Grade XI A && B

2015-2016

INDEX

1. Overview of C++
2. Need for the project
3. Requirements
4. List of Header Files
5. Classes
6. Function List
7. Program Code
8. Screenshots
9. Limitations and Shortcomings
10. Bibliography

Overview of C++

C++ was developed by **Bjarne Stroustrup**, circa 1979, as an extension and improvement to the C programming language. It was originally referred to as **C with Classes**.

**Evolution of C++**

The motivation for creating a new programming language came from Stroustrup’s experience in programming for his PhD thesis. Stroustrup found that **[Simula](https://en.wikipedia.org/wiki/Simula)** had features that were very helpful for large software development, but the language was too slow for practical use. When Stroustrup started working in [AT&T Bell Labs](https://en.wikipedia.org/wiki/AT%26T_Bell_Labs), he had the problem of analyzing the [UNIX](https://en.wikipedia.org/wiki/Unix) [kernel](https://en.wikipedia.org/wiki/Kernel_(computer_science)) with respect to [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing). Remembering his Ph.D. experience, Stroustrup set out to enhance the [C](https://en.wikipedia.org/wiki/C_(programming_language)) language with **[Simula](https://en.wikipedia.org/wiki/Simula)**-like features. C was chosen because it was general-purpose, fast, portable and widely used.

C++ combines both the high level idea of classes and the low level level machine handling capabilities of C.

It gave the user the best of both worlds and was a reliable and robust programming language.

**Features of C++**

C++ is the multi paradigm, compile, free form, general purpose, statistically typed programming language. This is known as middle level language as it comprises of low level and high level language features. C++ implements the concepts of Object Oriented Programming, OOPS.

The main features of OOPS are:

* Classes and Objects
* Inheritance
* Data abstraction and encapsulation
* Polymorphism
* Dynamic Binding
* Message Passing

In order to fortify portability over a miscellany of domains, C++ follows the ANSI srtandard.

The ANSI standards ensure that a program written on one compiler will work on different compiler and Operating systems.

**Advantages of C++**

C++ has numerous features which make it an excellent programming language. The main features include:

* Classes
* Modularity
* Reusability
* Readabilty
* Portability

**Uses of C++**

C++, being one of the most sought after programming languages, has a variety of uses.

Most packaged software is written in C++. This includes games, operating systems and office applications.

Need for Project

This project is an adaptation of the popular board game, checkers, also called draughts.

The interface aims to be as user friendly and straightforward as possible and is ideal for players new to the game.

The player is prompted only two options: Left and write, and the outcome of the move is determined by the program by analyzing the surrounding blocks

System Requirements

**Hardware:**

128 MB RAM

10 MB Hard Disk Space

**Software:**

Microsoft 95 or higher

MS DOS

Turbo C++ IDE

**User Manual:**

The CD attached contains files needed to install. Follow steps for installation.

1. Open Turbo C++
2. Go to the File Menu
3. Click ‘OPEN’
4. Go to file location
5. Open the file CHECKERS.CPP
6. Run the file in the IDE

List of Header Files

* iostream.h
  + cout
  + cin
* conio.h
  + getch()
* windows.h
  + system()
* process.h()
  + exit()

Classes

class Checkerboard {

private:

char pos[8][8];

int empty;

char turn\_colour;

int winflag;

public:

int liveblack;

int livered;

Checkerboard();

void display();

char get\_turn\_colour();

void change\_turn\_colour();

char get\_piece(int x, int y);

void change\_piece(int oldx, int oldy, int newx, int newy);

void kill\_piece(int x, int y, char dead);

int select\_piece();

int move\_left(int x, int y);

int move\_right(int x, int y);

int checkwin();

int usrcheckwin();

};

Function List

Member Functions of Checkerboard:

* void display() - Updates and displays the board
* char get\_turn\_colour() - Gets turn colour
* void change\_turn\_colour()
* char get\_piece() - Gets piece on given index
* void change\_piece() - Moves the piece
* void kill\_piece() - Attacks the opponent piece
* int select\_piece() - Used to accept user’s move
* int move\_left() - Moves piece left
* int move\_right() - Moves piece right
* int checkwin() - Checks for victory
* int usrcheckwin() - Checks winner if user quits

Program Code

**#include <graphics.h>**

**#include <stdlib.h>**

**#include <dos.h>**

**#include <iostream.h>**

**#include <stdio.h>**

**#include <conio.h>**

**#include <process.h>**

**//class with board display function and pos[][]**

**void bonus();**

**class Checkerboard {**

**private:**

**char pos[8][8];**

**int empty;**

**char turn\_colour;**

**int winflag;**

**public:**

**int liveblack;**

**int livered;**

**Checkerboard();**

**void display();**

**char get\_turn\_colour();**

**void change\_turn\_colour();**

**char get\_piece(int x, int y);**

**void change\_piece(int oldx, int oldy, int newx, int newy);**

**void kill\_piece(int x, int y, char dead);**

**int select\_piece();**

**int move\_left(int x, int y);**

**int move\_right(int x, int y);**

**int checkwin();**

**int usrcheckwin();**

**};**

**//This constructor initializes the default board game positions and copies them to pos[][]**

**//MAKE DEADBLACK AND DEADWHITE ZERO! --done for now**

**Checkerboard::Checkerboard() {**

**liveblack = 12;**

**livered = 12;**

**turn\_colour = 'R';**

**char board\_init[8][8] = {**

**{' ', 'B', ' ', ' ', ' ', 'R', ' ', 'R'},**

**{'B', ' ', 'B', ' ', ' ', ' ', 'R', ' '},**

**{' ', 'B', ' ', ' ', ' ', 'R', ' ', 'R'},**

**{'B', ' ', 'B', ' ', ' ', ' ', 'R', ' '},**

**{' ', 'B', ' ', ' ', ' ', 'R', ' ', 'R'},**

**{'B', ' ', 'B', ' ', ' ', ' ', 'R', ' '},**

**{' ', 'B', ' ', ' ', ' ', 'R', ' ', 'R'},**

**{'B', ' ', 'B', ' ', ' ', ' ', 'R', ' '},**

**};**

**for (int i = 0; i < 8; i++) {**

**for (int j = 0; j < 8; j++) {**

**pos[i][j] = board\_init[i][j];**

**}**

**}**

**}**

**char Checkerboard::get\_piece(int x, int y) {**

**return pos[x][y];**

**}**

**char Checkerboard::get\_turn\_colour() {**

**return turn\_colour;**

**}**

**void Checkerboard::change\_turn\_colour() {**

**if (turn\_colour == 'R') {**

**turn\_colour = 'B';**

**}**

**else {**

**turn\_colour = 'R';**

**}**

**}**

**//changes the location of piece**

**void Checkerboard::change\_piece(int oldx, int oldy, int newx, int newy) {**

**char temp;**

**temp = pos[oldx][oldy];**

**pos[oldx][oldy] = pos[newx][newy];**

**pos[newx][newy] = temp;**

**}**

**//kills the opponent**

**void Checkerboard::kill\_piece(int x, int y, char dead) {**

**pos[x][y] = ' ';**

**if(dead == 'B') {**

**liveblack--;**

**}**

**else {**

**livered--;**

**}**

**}**

**//displays the checkerboard**

**void Checkerboard::display() {**

**//clrscr();**

**cout<<" 1 2 3 4 5 6 7 8 \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 1 | "<<pos[0][0]<<" | "<<pos[1][0]<<" | "<<pos[2][0]<<" | "<<pos[3][0]<<" | "<<pos[4][0]<<" | "<<pos[5][0]<<" | "<<pos[6][0]<<" | "<<pos[7][0]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 2 | "<<pos[0][1]<<" | "<<pos[1][1]<<" | "<<pos[2][1]<<" | "<<pos[3][1]<<" | "<<pos[4][1]<<" | "<<pos[5][1]<<" | "<<pos[6][1]<<" | "<<pos[7][1]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 3 | "<<pos[0][2]<<" | "<<pos[1][2]<<" | "<<pos[2][2]<<" | "<<pos[3][2]<<" | "<<pos[4][2]<<" | "<<pos[5][2]<<" | "<<pos[6][2]<<" | "<<pos[7][2]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 4 | "<<pos[0][3]<<" | "<<pos[1][3]<<" | "<<pos[2][3]<<" | "<<pos[3][3]<<" | "<<pos[4][3]<<" | "<<pos[5][3]<<" | "<<pos[6][3]<<" | "<<pos[7][3]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 5 | "<<pos[0][4]<<" | "<<pos[1][4]<<" | "<<pos[2][4]<<" | "<<pos[3][4]<<" | "<<pos[4][4]<<" | "<<pos[5][4]<<" | "<<pos[6][4]<<" | "<<pos[7][4]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 6 | "<<pos[0][5]<<" | "<<pos[1][5]<<" | "<<pos[2][5]<<" | "<<pos[3][5]<<" | "<<pos[4][5]<<" | "<<pos[5][5]<<" | "<<pos[6][5]<<" | "<<pos[7][5]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" 7 | "<<pos[0][6]<<" | "<<pos[1][6]<<" | "<<pos[2][6]<<" | "<<pos[3][6]<<" | "<<pos[4][6]<<" | "<<pos[5][6]<<" | "<<pos[6][6]<<" | "<<pos[7][6]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**/// cout<<" | | | | | | | | | \n";**

**cout<<" 8 | "<<pos[0][7]<<" | "<<pos[1][7]<<" | "<<pos[2][7]<<" | "<<pos[3][7]<<" | "<<pos[4][7]<<" | "<<pos[5][7]<<" | "<<pos[6][7]<<" | "<<pos[7][7]<<" | \n";**

**// cout<<" | | | | | | | | | \n";**

**cout<<" ------- ------- ------- ------- ------- ------- ------- ------- \n";**

**}**

**int Checkerboard::checkwin() {**

**if(liveblack == 0) {**

**cout<<"Red Wins!";**

**return 0;**

**}**

**if(livered == 0) {**

**cout<<"Black Wins!";**

**return 0;**

**}**

**}**

**int Checkerboard::usrcheckwin() {**

**if (liveblack > livered) {**

**cout << "Black Wins!";**

**}**

**if (livered > liveblack) {**

**cout << "Red Wins!";**

**}**

**else {**

**cout << "It's a draw!";**

**}**

**getch();**

**exit(0);**

**return 0;**

**}**

**//Function to move\_right**

**int Checkerboard::move\_right(int x, int y) {**

**int temp;**

**if (get\_turn\_colour() == 'R') {**

**if (get\_piece(x + 1, y - 1) == ' ') {**

**change\_piece(x, y, x + 1, y - 1);**

**change\_turn\_colour();**

**return 0;**

**}**

**else if (get\_piece(x + 1 ,y - 1) == 'B') {**

**if(get\_piece(x + 2, y - 2) == ' ') {**

**change\_piece(x, y, x + 2, y - 2);**

**kill\_piece(x + 1, y - 1, 'B');**

**return 0;**

**}**

**cout << "Can't do that move. :/";**

**return -1;**

**}**

**else if (get\_piece(x,y) == 'R' || x == 9) {**

**cout << "Can't do that move. Try again. (-\_-)\n";**

**return -1;**

**}**

**}**

**else {**

**if (get\_piece(x + 1, y + 1) == ' ') {**

**change\_piece(x, y, x + 1, y + 1);**

**change\_turn\_colour();**

**}**

**else if (get\_piece(x + 1 ,y + 1) == 'R') {**

**if(get\_piece(x + 2, y + 2) == ' ') {**

**change\_piece(x, y, x + 2, y + 2);**

**kill\_piece(x + 1, y + 1, 'B');**

**return 0;**

**}**

**cout << "Nope. Check your move!";**

**}**

**else if (x == 7) {**

**cout << "Can't do that move. Try again. (-\_-)\n";**

**return -1;**

**}**

**else return -10;**

**}**

**return -20;**

**}**

**//function to move left**

**int Checkerboard::move\_left(int x, int y) {**

**int temp;**

**if (get\_turn\_colour() == 'R') {**

**if (get\_piece(x - 1, y - 1) == ' ') {**

**change\_piece(x, y, x - 1, y - 1);**

**change\_turn\_colour();**

**return 0;**

**}**

**else if (get\_piece(x - 1 ,y - 1) == 'B') {**

**if(get\_piece(x - 2, y - 2) == ' ') {**

**change\_piece(x, y, x + 2, y - 2);**

**kill\_piece(x - 1, y - 1, 'B');**

**return 0;**

**}**

**cout << "Can't do that move. :/";**

**return -1;**

**}**

**else if (get\_piece(x,y) == 'R' || x == 8) {**

**cout << "Can't do that move. Try again. (-\_-)";**

**return -1;**

**}**

**}**

**else {**

**if (get\_piece(x - 1, y + 1) == ' ') {**

**change\_piece(x,y, x - 1, y + 1);**

**change\_turn\_colour();**

**}**

**else if (get\_piece(x - 1, y + 1) == 'R') {**

**change\_piece(x, y, x - 2, y + 2);**

**kill\_piece(x - 1, y + 1, 'R');**

**}**

**else if (x == 0) {**

**cout << "Can't do that move. Try again. (-\_-)";**

**return -1;**

**}**

**else return -10;**

**}**

**return -20;**

**}**

**//function to receive user input to select a piece.**

**int Checkerboard::select\_piece() {**

**int x,y;**

**char dir;**

**cout << "Enter the coordinates of the piece you are moving (x,y): ";**

**cin >> x >> y;**

**if (x == -1) {**

**usrcheckwin();**

**} if (x == -2) {**

**bonus();**

**}**

**x--;**

**y--;**

**if(get\_piece(x,y) != get\_turn\_colour()) {**

**cout << "\nInvalid square. Please try again.\n";**

**return -1;**

**}**

**if (x > 7 || x < 0 || y > 7 || y < 0) {**

**cout << "Out of bounds!\n";**

**return -2;**

**}**

**else {**

**cout << "\nMove left or right? (l/r): ";**

**cin >> dir;**

**if (dir == 'r') {**

**move\_right(x,y);**

**return 0;**

**}**

**if (dir == 'l') {**

**move\_left(x,y);**

**return 0;**

**}**

**else {**

**cout << "Incorrect input. Try again.";**

**return -3;**

**}**

**}**

**}**

**int x=100;**

**int x2 = 500;**

**int score = 0;**

**int midy=getmaxy()/2+280;**

**int y = midy;**

**int y2 = midy;**

**int q = 0;**

**int check=1;**

**void display(){**

**setfillstyle(1,4);**

**fillellipse(x,y,5,5);**

**setfillstyle(1,1);**

**fillellipse(x2,y2,5,5);**

**}**

**void render(){**

**cleardevice();**

**x2 = x2 - 10;**

**if(x2<0){**

**x2 = 400;**

**score++;**

**}**

**display();**

**char sc[10];**

**itoa(score, sc, 10);**

**outtextxy(10,10,sc);**

**line(0,getmaxy()/2+50,getmaxx(),getmaxy()/2+50);**

**if(x2==100&&y==midy){**

**q=1;**

**}**

**delay(75);**

**}**

**void bonus() {**

**int gdriver = DETECT, gmode, errorcode;**

**int left, top, right, bottom;**

**initgraph(&gdriver, &gmode, "");**

**while(!q){**

**char ch;**

**if(kbhit()&&check == 1){**

**ch = getch();**

**if(ch=='s'){**

**check = 0;**

**y=midy+10;**

**render();**

**cleardevice();**

**y=midy+20;**

**render();**

**cleardevice();**

**y=midy+10;**

**render();**

**cleardevice();**

**y=midy;**

**render();**

**cleardevice();**

**check = 1;}**

**if(ch=='w'){**

**check = 0;**

**y=midy-10;**

**render();**

**cleardevice();**

**y=midy-20;**

**render();**

**cleardevice();**

**y=midy-10;**

**render();**

**cleardevice();**

**y=midy;**

**render();**

**cleardevice();**

**check = 1;**

**}**

**if(ch=='q') q = 1;**

**}**

**else{**

**render();**

**}**

**}**

**if(q==1)**

**{**

**cleardevice();**

**rscr();**

**outtextxy(50,50,"you lose (haha!!!!!1!!)");**

**getch();**

**}**

**//MAIN**

**int main() {**

**clrscr();**

**Checkerboard board;**

**cout << "\t\t\tWelcome to checkers!\nIf you hit a deadlock, or want to quit, enter the coordinates as (-1,-1)\n";**

**getch();**

**system("cls");**

**while(board.checkwin() != 0) {**

**system("cls"); //cls for windows and clear for linux**

**board.display();**

**cout << "Turn: " << board.get\_turn\_colour() << "\n";**

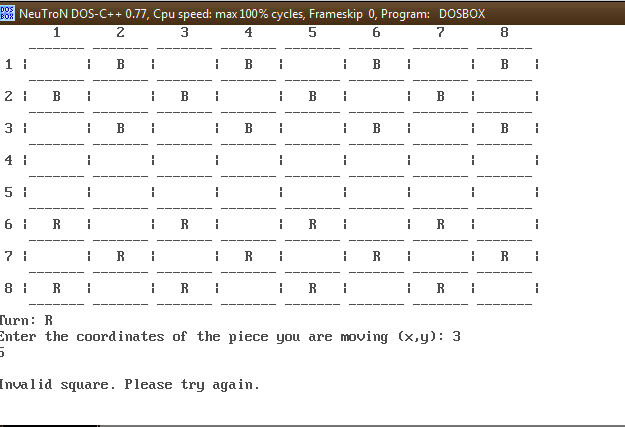
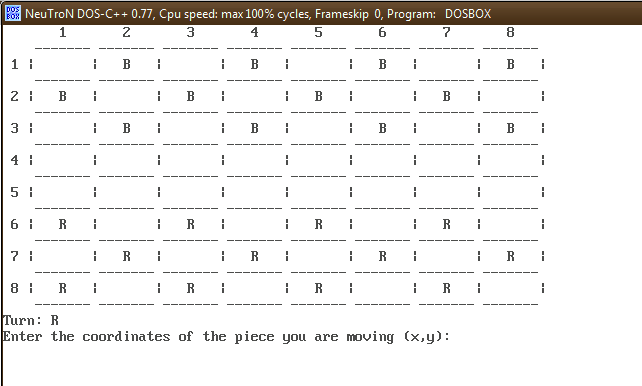
**board.select\_piece();**

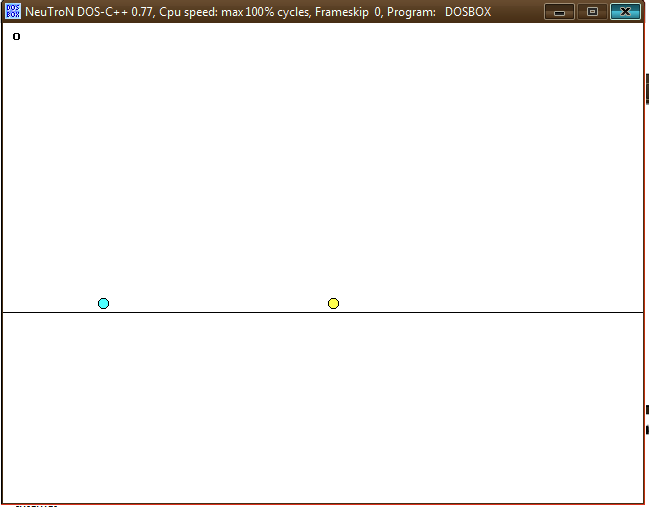
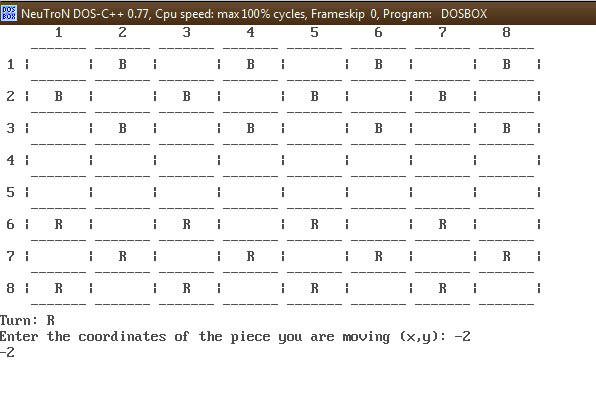
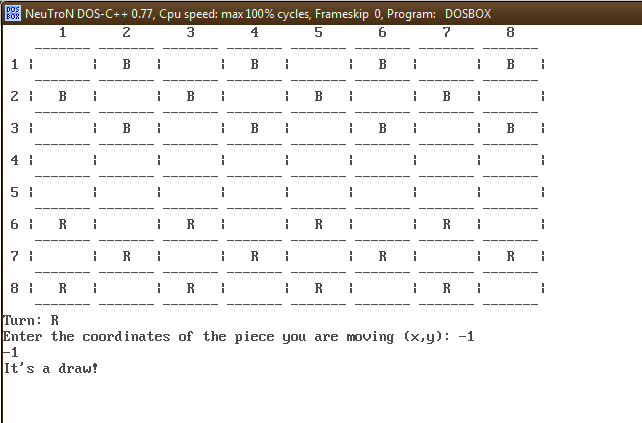
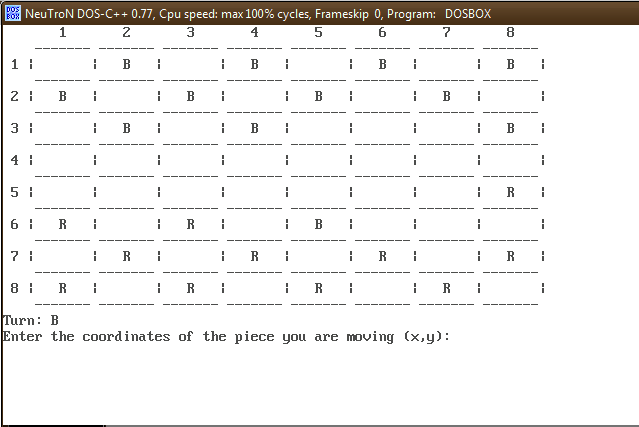
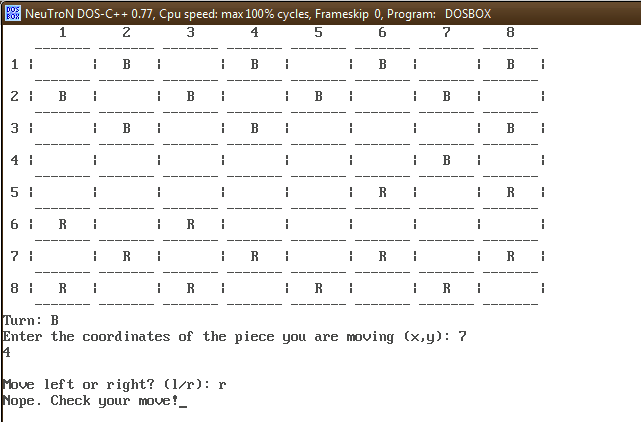
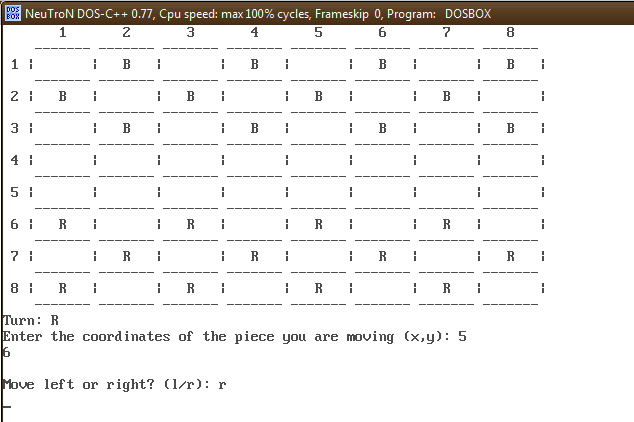
**getch();**

**getch();**

**}**

**}**

Screenshots



Shortcomings and Limitations

The project, in its present state, suffers from some shortcomings and limitations. These are stated as follows:

1. Does not support the “Queen” rule, wherein if a checker makes it to the other end, it can move backwards also.
2. Does not support a Single Player Mode

Bibliography

1. <https://en.wikipedia.org/wiki/C%2B%2B>
2. <http://www.cplusplus.com/doc/tutorial/>