Connect Four

Project 1

CSC- 17C – 48942

Tsz,Kwan

11 – November – 2015

Content

1.Introduction……………………………………………3  
Rule and Gameplay  
Thoughts after Program

2. Development………………………..………………3-4  
Approach Strategy

3. Variables list…………………………….……..……...4

4. Topic Covered (Checklist)…………………….….......5

5. Libraries included……………………………………..5

6. Pseudo Code…………………………………………..6

7. Flowchart……………………………………………...7

8. Code…………………………………………….....8-34

1. Introduction

**Rules and Gameplay**

Connect four is a two-player connection game. The grid size is 6x7. Two players take turn to drop colored discs from the top into the grid. The game ends when one player has four in a row, or the grid is fully filled. Instead of color disc, I used “X” and “O.” Every time after checking the validation, the program will check the boxes around the disc to see if any four in a row exist.

**Thoughts after Program**

Setting up the board by using linked list was not too hard if I only had had one pointer that pointed to next node. I made the linked list that looked like 2D array. In the structure of the node, I have four pointers which are up, down, previous, and next. However, only nodes that are in the first column will use the up and down pointers. The down pointer make the program runs faster because overloaded operator will modified by column to get the number of row, which can decrease the number of loops.

2. Development

Approach Strategy

Because the project requires recursive function, I tried to make the checkWin (checking four in a row) recursive. It seemed easy, but it was not. I checked all eight directions inside the function. I needed to pass in the original coordinates to recovery if the current coordinates are out ranged, empty, or not same disc. I wrote the iterative function first and changed it to recursive. I need to include all the possible situations in the recursive function, and it became much longer than I expected. Also, it was difficult to debug because it call itself so many times, and it was hard to keep track on it.

3. Variables list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **File** | **Line** |
| int | index | node index (debug purpose) | Connect\_four.h | 18 |
|  | size | board size | Connect\_four.h | 27 |
|  | row | board rows | Connect\_four.h | 28 |
|  | col | board columns | Connect\_four.h | 29 |
|  | count | numbers in a row | Connect\_four.h | 34 |
| string | temp | temp to store name | main.cpp | 25 |
|  | p1 | player 1 name | Connect\_four.h | 36 |
|  | p2 | player 2 name | Connect\_four.h | 37 |
| bool | valid | validation flag for temp | main.cpp | 26 |
|  | first | flag for player 1 and 2 | main.cpp | 27 |
|  | repeat | flag for play again | main.cpp | 28 |
|  | win | flag for win | Connect\_four.h | 35 |
|  | valid | flag for p1 and p2 | Connect\_four.h | 38 |
| Node\* | next | point to next node | Connect\_four.h | 20 |
|  | prev | point to previous node | Connect\_four.h | 21 |
|  | up | point to node above | Connect\_four.h | 22 |
|  | down | point to node below | Connect\_four.h | 23 |
|  | head | point to first node | Connect\_four.h | 30 |
|  | last | point to last node | Connect\_four.h | 31 |
|  | temp | temporary node pointer | Connect\_four.h | 32 |
|  | work | temporary node pointer | Connect\_four.h | 33 |
| T | data | board data | Connect\_four.h | 19 |
| C4<char> | test(6,7) | Game object | main.cpp | 29 |
| fstream | io | file io object | Connect\_four.h | 154 |

4. Topic Covered (Checklist)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chapter** | **type** | **code** | **File** | **line** |
| Linked list |  | struct Node { | Connect\_Four.h | 17 |
|  |  | int index; | Connect\_Four.h | 18 |
|  |  | T data; | Connect\_Four.h | 19 |
|  |  | Node \*next; | Connect\_Four.h | 20 |
|  |  | Node \*prev; | Connect\_Four.h | 21 |
|  |  | Node \*up; | Connect\_Four.h | 22 |
|  |  | Node \*down; | Connect\_Four.h | 23 |
|  |  | Node() : index(0), data(' '),next(NULL), prev(NULL), up(NULL), down(NULL) {} | Connect\_Four.h | 24 |
|  |  | } | Connect\_Four.h | 25 |
| Recursion |  | void C4<T>::chkWin(int cRow, int cCol, int oRow, int oCol,  int dir, char a, bool opp){ | Connect\_Four.h | 302-567 |
|  |  | chkWin(cRow, cCol, oRow, oCol, dir, a, opp); | Connect\_Four.h | 373 |
|  |  | chkWin(cRow, cCol, oRow, oCol, dir, a, opp); | Connect\_Four.h | 488 |
|  |  | } | main | 57 |
| Assign operator |  | C4<T> &operator=(const C4<T>&); | Connect\_Four.h | 79 |
| overload [] |  | T &operator[](const int &index); | Connect\_Four.h | 70 |
| STL sort function |  | sort(temp.begin(), temp.end()); | Connect\_Four.h | 644 |
| Quick sort |  | void C4<T>::quickSort(int start, int end){ | Connect\_Four.h | 610 |

5. Libraries included

* <bits/stdc++.h>
* “Connect\_Four.h”

6. Pseudo Code

Create game object

Create board by using linked list

Get players name

do{

While( !win){

first = true

player 1 turn

check win

if( !win)

first = false

player 2 turn

check win

}

if (first)

player 1 wins

else

player 2 wins

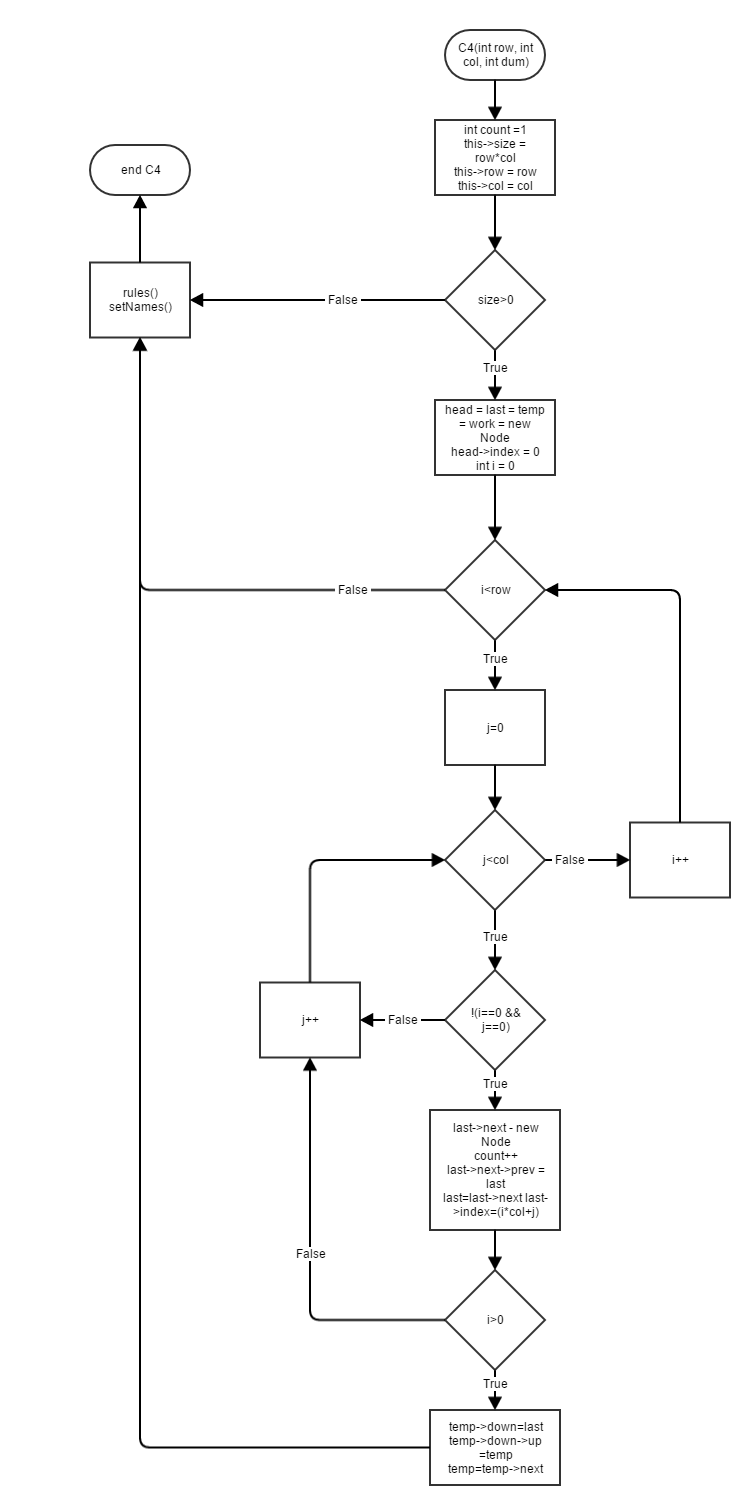
ask for play again

if (yes) reset board

}while (yes)

program ends

7. Constructor Flowchart



8. Code – main.cpp

/\*

\* File: main.cpp

\* Author: Himkw\_000

\*

\* Created on October 1, 2015, 10:38 PM

\*/

/\*

\* connect 4 2 players

\* board size = 6\*7

\* checkWin = recursion

\*/

#include <bits/stdc++.h>

using namespace std;

//user libraries

//#include "Minesweeper.h"

#include "Connect\_Four.h"

//function prototype

bool valid(string);

//Program starts

int main(int argc, char\*\* argv) {

//declare vars

string temp;

bool valid;

bool first=false;

bool repeat=false;

C4<char> test(6,7);

do{

repeat=false;

first=false;

test.show();

while(!test.getWin()){

first?first=false:first=true;

do{

getline(cin, temp);

// cout<<temp;

}while(!test.setValid(temp));

// cout<<"setboard\n";

test.setBoard(temp[0]-65, first);

test.show();

}

test.clr();

cout<<test.getName(first)<<" wins!\n";

do{

valid=true;

cout<<"Play again ? (Y/N)";

getline(cin,temp);

if(temp.length()!=1)valid=false;

else if(temp[0]!='Y' && temp[0]!='N')valid=false;

else if(temp[0]=='Y')repeat=true;

}while(!valid);

}while(repeat);

return 0;

}

bool valid(string temp){

if(temp.length()!=1){

return false;

}

else if(temp[0]<'A' || temp[0]>'G'){

return false;

}

return true;

}

**Connect\_Four.h**

/\*

\* File: Connect\_Four.h

\* Author: Himkw\_000

\*

\* Created on October 12, 2015, 11:53 AM

\*/

#ifndef CONNECT\_FOUR\_H

#define CONNECT\_FOUR\_H

#include <bits/stdc++.h>

using namespace std;

template <class T>

class C4{

private:

struct Node {

int index;

T data;

Node \*next;

Node \*prev;

Node \*up;

Node \*down;

Node() : index(0), data(' '),next(NULL), prev(NULL), up(NULL), down(NULL) {}

};

int size;

int row;

int col;

Node \*head;

Node \*last;

Node \*temp;

Node \*work;

int count;

bool win;

string p1;

string p2;

bool valid;

T top\_left(int index){return operator[](index-col-1);}

T top(int index){return operator[](index-col);}

T top\_right(int index){return operator[](index-col+1);}

T left(int index){return operator[](index-1);}

T right(int index){return operator[](index+1);}

T bottom\_left(int index){return operator[](index+col-1);}

T bottom(int index){return operator[](index+col);}

T bottom\_right(int index){return operator[](index+col+1);}

void rules();

void setNames();

public:

//default constructor

C4(){size=row=col=0;}

/\*\*

\* only first col has up and down links

\* @param row

\* @param col

\*/

C4(int row, int col);

/\*\*

\* all nodes links to up, down, prev, next

\* @param row

\* @param col

\* @param dum

\*/

C4(int row, int col, int dum);

~C4();

int getSize(){return size;}

int getRow(){return row;}

int getCol(){return col;}

T &operator[](const int &index);

void setBoard(int col, bool first);

bool setValid(string);

bool checkWin(int row, int col);

void show();

void chkWin(int cRow, int cCol, int oRow, int oCol, int dir, char a, bool opp);

bool getWin(){return win;}

void clr();

string getName(int a){return (a==1?p1:p2);}

C4<T> &operator=(const C4<T>&);

string getName(bool first){return (first?p1:p2);}

void quickSort(int start, int end);

void stlSort();

};

template <class T>

C4<T>::C4(int row, int col){

this->size = row\*col;

this->row = row;

this->col = col;

head = last = temp = work = new Node;

head->index = 0;

for (int i = 1; i < row \* col; i++) {

// cout<<"for loop i = "<<i<<"\n";

last->next = new Node;

// cout<<"Create Node "<<last->index+1<<endl;

last->next->prev = last;

last = last->next;

last->index = i;

// cout<<"last move to "<<last->index<<endl;

if ((i) % col == 0) {

// cout<<"i+1%col==0\n";

// cout<<last->index<<" connect to "<<temp->index<<endl;

temp->down = last;

temp->down->up = temp;

temp = temp->down;

}

}

// cout<<"call rules\n";

rules();

setNames();

}

template <class T>

C4<T>::C4(int row, int col, int dum) {

int count=1;

this->size = row\*col;

this->row = row;

this->col = col;

if(size>0){

head = last = temp = work = new Node;

head->index = 0;

// cout << "last = " << last->index << endl;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

//j=0

if (!(i == 0 && j == 0)) {

last->next = new Node;

count++;

last->next->prev = last;

last=last->next;

last->index=(i \* col + j);

// cout<<"i = "<<i<<" j = "<<j<<endl;

// cout<<"last = "<<last->index<<"\n";

if (i > 0) {

// cout<<"temp = "<<temp->index<<"\n\n";

// cout<<"last = "<<last->index<<endl;

temp->down=last;

temp->down->up=temp;

temp = temp->next;

}

}

}

}

}

rules();

setNames();

// cout<<count<<endl;

}

template <class T>

void C4<T>::rules(){

// cout<<"rules function\n";

fstream io;

string temp;

io.open("Rules.dat", ios::in);

if(io){

// cout<<"open file\n";

while(getline(io,temp)){

cout<<temp;

}

}

else{

cout<<"ERROR: Failed to open rules file\n";

}

}

template <class T>

void C4<T>::setNames(){

do{

valid=true;

cout<<"Player 1 name : ";

getline(cin, p1);

if(p1.length()<3 || p1.length()>8){

cout<<"Invalid input: input has to be between 3 and 8 characters\n";

valid=false;

}

else{

for(int i=0;i<p1.length() && valid;i++){

if(!isalpha(p1[i])){

valid=false;

cout<<"Invalid input: input has to be characters\n";

}

}

}

}while(!valid);

do{

valid=true;

cout<<"Player 2 name : ";

getline(cin, p2);

if(p2.length()<3 || p2.length()>8){

cout<<"Invalid input: input has to be between 3 and 8 characters\n";

valid=false;

}

else{

for(int i=0;i<p2.length() && valid;i++){

if(!isalpha(p2[i])){

valid=false;

cout<<"Invalid input: input has to be characters\n";

}

}

}

}while(!valid);

}

template <class T>

C4<T>::~C4(){

if(size>0){

work=head;

for(int i=0;i<size-1;i++){

head=work;

work=head->next;

delete head;

}

delete work;

}

}

template <class T>

T &C4<T>::operator [](const int &index){

// cout<<"call index function\n";

work = head;

for (int i = 0; i < index; i++) {

work = work->next;

}

// cout<<"in function "<<work->index<<endl;

return work->data;

}

template <class T>

void C4<T>::setBoard(int col, bool first){

// cout<<"set board function\n";

// first?cout<<"true\n":cout<<"false\n";

int i=0;

char a;

while(operator[](i\*this->col+col)!=' ')i++;

// if(setValid(i, col)){

if(first){

operator[](i\*this->col+col)='X';

a='X';

}

else{

operator[](i\*this->col+col)='O';

a='O';

}

// }

chkWin(i, col, i, col, 1, a, false);

}

template <class T>

bool C4<T>::setValid(string tempS){

// cout<<"call setValid function\n";

if(tempS.length()!=1){

// cout<<"length!=1\n";

return false;

}

else if(tempS[0]<'A' || tempS[0]>'G'){

// cout<<"out range\n";

return false;

}

else{

int i=0;

int q=tempS[0]-65;

while(operator[](i\*col+q)!=' ')i++;

// cout<<"i = "<<i<<endl;

if(i>row)return false;

return true;

}

}

template <class T>

void C4<T>::show(){

cout<<" ";

for(int i=0;i<col;i++){

char temp=i+65;

cout<<temp<<" ";

}

cout<<endl;

cout<<endl;

for(int i=row-1;i>=0;i--){

cout<<" | ";

for(int j=0;j<col;j++){

cout<<operator[](i\*col+j)<<" | ";

}

cout<<endl;

cout<<" ---";

for(int k=0;k<row;k++){

cout<<"----";

}

cout<<endl;

}

cout<<" ";

for(int i=0;i<col;i++){

char temp=i+65;

cout<<temp<<" ";

}

cout<<endl;

}

template <class T>

void C4<T>::chkWin(int cRow, int cCol, int oRow, int oCol,

int dir, char a, bool opp){

// cout<<"beginning\n";

// cout<<cRow<<" "<<cCol<<" "<<oRow<<" "<<oCol<<" "<<dir<<" "<<a<<" "<<opp<<endl;

if(cRow==oRow && cCol==oCol && opp==false){

// cout<<"same x-y"<<endl;

// cout<<cRow<<" "<<cCol<<" "<<oRow<<" "<<oCol<<" "<<dir<<" "<<a<<" "<<opp<<endl;

//reset

count=1;

win=false;

//水平

if(dir==1){

// cout<<"dir=1\n";

if(cCol-1>=0){

// cout<<"opp=false\n";

cCol--;

opp=false;

}

else{

// cout<<"opp=true\n";

cCol++;

opp=true;

}

}

//左上右下

else if(dir==2){

// cout<<"dir=2\n";

if(cRow-1>=0 && cCol-1>=0){

// cout<<"opp=false\n";

cRow--;

cCol--;

opp=false;

}

else{

// cout<<"opp=true\n";

cRow++;

cCol++;

opp=true;

}

}

//上下

else if(dir==3){

// cout<<"dir=3\n";

if(cRow-1>=0){

// cout<<"opp=false\n";

cRow--;

opp=false;

}

else{

// cout<<"opp=true\n";

cRow++;

opp=false;

}

}

//右上左下

else{

// cout<<"dir=4\n";

if(cRow-1>=0 && cCol+1<col){

// cout<<"opp=false\n";

cRow--;

cCol++;

opp=false;

}

else{

// cout<<"opp=false\n";

cRow++;

cCol--;

opp=true;

}

}

// cout<<cRow<<" "<<cCol<<" "<<oRow<<" "<<oCol<<" "<<dir<<" "<<a<<" "<<opp<<endl;

chkWin(cRow, cCol, oRow, oCol, dir, a, opp);

}//first move / change direction

//rest move

else{

//match

// cout<<"match\n";

if(operator[](cRow\*col+cCol)==a){

count++;

//terminate win

if(count==4){

win=true;

return ;

}

if(dir==1){

//左上右下

if(!opp){

if(cCol-1>=0){

cCol--;

}

else{

cCol=oCol+1;

opp=true;

}

}

else{

if(cCol++<col){

cCol++;

}

else{

cCol=oCol;

opp=false;

dir=2;

}

}

}

//左上右下

else if(dir==2){

if(!opp){

if(cRow-1>=0 && cCol-1>=0){

cRow--;

cCol--;

}

else{

cRow=oRow+1;

cCol=oCol+1;

opp=true;

}

}//!opp

else{

if(cRow+1<row && cCol+1<col){

cRow++;

cCol++;

}

else{

dir=3;

opp=false;

cRow=oRow;

cCol=oCol;

}

}

}//dir==2

//上下

else if(dir==3){

//向上

if(!opp){

//向上不越界

if(cRow-1>=0){

cRow--;

}

//向上越界，掉頭向下

else{

cRow=oRow+1;

opp=true;

}

}//!opp

//已掉頭向下

else{

if(cRow+1<row){

cRow++;

}

//向下越界 dir->4

else{

cRow=oRow;

cCol=oCol;

dir=4;

opp=false;

}

}//opp

}//dir==3

//右上左下

else{

if(!opp){

if(cRow-1>=0 && cCol+1<col){

cRow--;

cCol++;

}

else{

opp=true;

cRow=oRow+1;

cCol=oCol-1;

}

}

else{

if(cRow+1<row && cCol-1>=0){

cRow++;

cCol--;

}

else{

win=false;

return ;

}

}

}

chkWin(cRow, cCol, oRow, oCol, dir, a, opp);

}//rest move match

//rest move doesn't match

else{

// cout<<"not match\n";

if(dir==1){

// cout<<"dir=1\n";

if(!opp){

// cout<<"!opp\n";

if(oCol+1<col){

// cout<<"opp=true\n";

opp=true;

cCol=oCol+1;

}

else{

// cout<<"dir=2\n";

oRow=oRow;

cCol=oCol;

dir=2;

}

}

else{

// cout<<"opp\n";

// cout<<"dir=2, opp=false\n";

opp=false;

dir=2;

cRow=oRow;

cCol=oCol;

}//dir->2

}//dir=1

else if(dir==2){

if(!opp){

//掉向

if(oRow+1<row && oCol+1<col){

cRow=oRow+1;

cCol=oCol+1;

opp=true;

}

else{

//dir->3

cRow=oRow;

cCol=oCol;

dir=3;

}

}

else{

opp=false;

dir=3;

cRow=oRow;

cCol=oCol;

}

}//dir=2

//上下

else if(dir==3){

if(!opp && oRow+1<row){

cRow=oRow+1;

opp=true;

}

else{

dir=4;

opp=false;

cRow=oRow;

cCol=oCol;

}

}//dir=3

else{

if(!opp && oRow+1<row && oCol-1>=0){

opp=true;

cRow=oRow+1;

cCol=oCol-1;

}

else{

win=false;

return ;

}

}

chkWin(cRow, cCol, oRow, oCol, dir, a, opp);

}//rest move doesn't match

}

}

template <class T>

C4<T> &C4<T>::operator =(const C4<T>& obj){

size=obj.getSize();

row=obj.getRow();

col=obj.getCol();

if(obj.getSize()>0){

head = last = temp = work = new Node;

head->index=0;

for(int i=1;i<row\*col;i++){

last->next = new Node;

last->next->prev = last;

last = last->next;

last->next = i;

if((i) % col == 0){

temp->down = last;

temp->down->up = temp;

temp = temp->down;

}

}

last->next=NULL;

work=head;

obj.work=obj.head;

for(int i=0;i<size;i++){

work.data=obj.work->data;

work=work->next;

obj.work=obj.work->next;

}

}

return \*this;

}

template <class T>

void C4<T>::clr(){

work = head;

for(int i=0;i<size;i++){

work->data = ' ';

work = work->next;

}

}

template <class T>

void C4<T>::quickSort(int start, int end){

if(start>=end) return;

int mid = operator[](end);

int left = start;

int right = end - 1;

while(left<right){

while(operator[](left)<mid && left<right){

left++;

}

while(operator[](right)>=mid && left<right){

right--;

}

T temp = operator[](left);

operator[](left) = operator[](right);

operator[](right) = temp;

}

if(operator[](left) >= operator[](end)){

T temp = operator[](left);

operator[](left) = operator[](end);

operator[](end) = temp;

}

else{

left++;

}

quickSort(start, left-1);

quickSort(left+1, end);

}

template <class T>

void C4<T>::stlSort(){

vector<T> temp;

for(int i=0;i<size;i++){

temp.push\_back(operator[](i));

}

sort(temp.begin(), temp.end());

for(int i=0;i<size;i++){

operator[](i) = temp[i];

}

}

#endif /\* CONNECT\_FOUR\_H \*/