**Project 2**

*Title:*

**Uno**

*Course:*

*Cis 17-C*

*Section:*

*40503*

*Due Date:*

*6/7/2020*

*Author:*

*Wesley Duong*

*Introduction:*

Title Uno:

This is a card strategy game, where the goal is to get rid of your card to win the game.

I spend one week and each day I do 6 hour per day. It took 864 line of code that to make the code work and run. My class for the Uno games are player, card\_elem, deck, and card. The code on github is at <https://github.com/WestAsian/Project_2_17C>.

*Approach to development:*

So the first thing I do is list the color, number, darw-2, skip, reverse, wild, and drawing-4-wild first what each of them do, then I would do the deck and how many card that it can hold and how many card should a each player hold from the start of the game. Then finally adding the rule of the game and how player are playing the game

*Game Rules:*

When running the program, it would ask how many players and it goes up from 2-5 players. Once the game start each player must match the number or the color except the wild card because they can change the color of the game, to decrease the number of card of each player is holding in order to win the game.

*Description of Code:*

So from the start it would ask as many players that want to play then, in the game when you choose to draw or to play a card, I would have where it also confirm that you want to play that move by press the number that indicted what player are you.

*Pseudo Code:*

Initialize

When start display “please enter number of player:”

Random pick a player

When pick confirm that player with their number

Then display the color, number of what is play

And also display the player hands of their number and card

When playing card

Confirm playing card

If play want to draw press -1

If no

Don’t play card and goes back to player hand

If yes

Draw the card next player turn

When reverse card play

Display yes or no to play the card

If no

Don’t play card and goes back to hand

If yes

Play the card next player turn

When wild card play ask color

then confirm

When wild-4 draw play ask color

then confirm and the next player draw 4

When skip

then confirm skip and the next player is skip

When play draw-2

then confirm draw-2 and next player draw 2

Else if play the card on hand

If their card

loop to the next player

else end game

display “\_\_\_\_\_ this player as won the game”

*Flowchart:*

*A close up of text on a white background

Description automatically generated*

*UML Code:*

*card.h:*

*#include <ostream>*

*enum COLOR { wild, red, green, blue, yellow};*

*/\*\**

*\* Represents a card in a deck*

*\*/*

*class card*

*{*

*public:*

*int number;*

*COLOR color;*

*bool operator==(card const & other) const;*

*bool operator!=(card const & other) const;*

*card();*

*card(int num, COLOR col);*

*};*

*std::ostream & operator<<(std::ostream & out, card const & temp\_card);*

*deck.h:*

*#include "card.h"*

*class deck : public card {*

*public:*

*deck();*

*deck(const deck & other);*

*const deck & operator= (const deck & other);*

*~deck();*

*void shuffle();*

*card draw();*

*void create();*

*int add\_card(card temp\_card);*

*void quick\_shuffle();*

*void print\_deck();*

*int get\_size();*

*private:*

*card \* ptr\_deck;*

*int size;*

*void clear();*

*void copy(const deck & other);*

*};*

*std::ostream & operator<<(std::ostream & out, deck const & temp\_deck);*

*player.h:*

*#include "card.h"*

*class player*

*{*

*public:*

*player();*

*player(const player & other);*

*const player & operator= (const player & other);*

*~player();*

*void hand\_add(card temp);*

*card hand\_remove(int pos);*

*void uno();*

*void print() const;*

*int get\_size() const;*

*card peek(int pos) const;*

*private:*

*class card\_elem*

*{*

*public:*

*card\_elem()*

*{*

*next = NULL;*

*}*

*card data;*

*card\_elem \* next;*

*};*

*card\_elem \* head;*

*int size;*

*void copy(const player & other);*

*void clear();*

*};*

*card.cpp:*

*#include "card.h"*

*card::card() : number(0), color(wild)*

*{*

*}*

*card::card(int num, COLOR col) : number(num), color(col)*

*{*

*}*

*bool card::operator==(card const & other) const*

*{*

*return number == other.number || color == other.color || color == wild || other.color == wild;*

*}*

*bool card::operator!=(card const & other) const*

*{*

*return !(\*this == other) ;*

*}*

*std::ostream & operator<<(std::ostream & out, card const & temp\_card)*

*{*

*out << "Number:";*

*switch (temp\_card.number)*

*{*

*case 10: out<< "DRAW-2" ; break;*

*case 11: out<< "SKIP"; break;*

*case 12: out<< "REVERSE"; break;*

*case 13: out<< "WILD" ; break;*

*case 14: out<< "DRAW-4-WILD" ; break;*

*default: out << (int) temp\_card.number; break;*

*}*

*out << " Color:" ;*

*switch (temp\_card.color)*

*{*

*case wild : out << "wild"; break;*

*case red : out << "red"; break;*

*case green : out << "green"; break;*

*case blue : out << "blue"; break;*

*case yellow : out << "yellow"; break;*

*default: out << "N/A"; break;*

*}*

*return out;*

*}*

*deck.cpp:*

*#include "deck.h"*

*#include "card.h"*

*#include <iostream>*

*#include <cstdlib>*

*using namespace std;*

*deck::deck()*

*{*

*ptr\_deck = new card[DECK\_SIZE];*

*size = 0;*

*}*

*void deck::create()*

*{*

*int num = 0 ;*

*for (int col = 1 ; col <= 4 ; col++)*

*{*

*ptr\_deck[size].number = num;*

*ptr\_deck[size].color = static\_cast<COLOR>(col);*

*size++;*

*}*

*for (num = 1; num <= 12; num++)*

*{*

*for (int x = 0 ; x < 2 ; x++ )*

*{*

*for (int col = 1 ; col <= 4 ; col++)*

*{*

*ptr\_deck[size].number = num;*

*ptr\_deck[size].color = static\_cast<COLOR>(col);*

*size++;*

*}*

*}*

*}*

*for (num = 13 ; num <= 14 ; num++)*

*{*

*for (int x = 0 ; x < 4 ; x++)*

*{*

*ptr\_deck[size].number = num;*

*ptr\_deck[size].color = wild;*

*size++;*

*}*

*}*

*}*

*deck::deck(const deck & other)*

*{*

*copy(other);*

*}*

*const deck & deck::operator= (const deck & other)*

*{*

*if (this != &other)*

*{*

*clear();*

*copy(other);*

*}*

*return \*this;*

*}*

*deck::~deck()*

*{*

*clear();*

*}*

*void deck::shuffle()*

*{*

*card \* temp\_deck = new card[size];*

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

*{*

*temp\_deck[i] = ptr\_deck[i];*

*}*

*int temp\_size = size;*

*int temp\_pos;*

*int pos;*

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

*{*

*srand(time(NULL));*

*pos = rand() % temp\_size;*

*ptr\_deck[i] = temp\_deck[pos];*

*temp\_size--;*

*for (temp\_pos = pos ; temp\_pos < temp\_size ; temp\_pos++)*

*{*

*temp\_deck[temp\_pos] = temp\_deck[temp\_pos+1];*

*}*

*}*

*delete [] temp\_deck;*

*}*

*card deck::draw()*

*{*

*if (size <= 0)*

*{*

*return card();*

*}*

*card temp\_card = ptr\_deck[size-1];*

*size--;*

*return temp\_card;*

*}*

*int deck::add\_card(card temp\_card)*

*{*

*if(size < DECK\_SIZE)*

*{*

*ptr\_deck[size] = temp\_card;*

*size++;*

*return 0;*

*}*

*else*

*return -1;*

*}*

*void deck::quick\_shuffle()*

*{*

*int pos;*

*int temp\_size = size-1;*

*card temp\_card;*

*while(temp\_size > 0 )*

*{*

*srand(time(NULL));*

*pos = rand() % temp\_size;*

*temp\_card = ptr\_deck[temp\_size];*

*ptr\_deck[temp\_size] = ptr\_deck[pos];*

*ptr\_deck[pos] = temp\_card;*

*temp\_size--;*

*}*

*}*

*void deck::print\_deck()*

*{*

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

*{*

*cout << i << ": " << ptr\_deck[i] << endl;*

*}*

*}*

*void deck::copy(const deck & other)*

*{*

*size = other.size;*

*ptr\_deck = new card[size];*

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

*{*

*ptr\_deck[i] = other.ptr\_deck[i];*

*}*

*}*

*void deck::clear()*

*{*

*delete []ptr\_deck;*

*ptr\_deck = NULL;*

*size = 0;*

*}*

*main.cpp:*

*#include "card.h"*

*#include "deck.h"*

*#include "player.h"*

*#include <iostream>*

*#include <fstream>*

*#include <string>*

*#include <cstdlib>*

*#include <limits>*

*using namespace std;*

*#define TEXTFILE "intro.txt"*

*#define PRINT\_ALL\_PLAYERS 0*

*#define TEMP\_DECK 1*

*#define TURN 2*

*#define TEST -1*

*void confirm\_turn (int x)*

*{*

*cout << "Confirm Player" << x << " by typing " << "'" << x << "'" << " and pressing enter" << ": " ;*

*int temp;*

*while( temp != x)*

*{*

*cin >> temp;*

*}*

*}*

*COLOR FromString (const string & str)*

*{*

*if ( str == "red")*

*return red;*

*else if (str == "green")*

*return green;*

*else if (str == "blue")*

*return blue;*

*else if (str == "yellow")*

*return yellow;*

*else*

*return wild;*

*}*

*int main()*

*{*

*system("clear");*

*int amount\_players;*

*int flag = 0;*

*while (flag == 0 )*

*{*

*cout << "Please enter amount of players: ";*

*cin >> amount\_players;*

*if (amount\_players >= 2 && amount\_players <= 5)*

*{*

*cout << amount\_players << " players entering game .... " << endl;*

*flag = 1;*

*break;*

*}*

*else*

*{*

*cout << "invalid amount of players" << endl;*

*}*

*}*

*deck main\_deck;*

*main\_deck.create();*

*main\_deck.quick\_shuffle();*

*player \* play\_array;*

*play\_array = new player[amount\_players];*

*for (int i =0 ; i < amount\_players ; i ++)*

*{*

*for (int k= 0 ; k< 7 ; k++)*

*{*

*card temp\_card;*

*temp\_card = main\_deck.draw();*

*play\_array[i].hand\_add(temp\_card);*

*}*

*}*

*deck temp\_deck;*

*card played\_card;*

*card temp\_card;*

*int card\_flag = 0 ;*

*while (card\_flag == 0 )*

*{*

*temp\_card = main\_deck.draw();*

*if (temp\_card.color != wild)*

*{*

*card\_flag = 1;*

*played\_card = temp\_card;*

*}*

*else*

*{*

*temp\_deck.add\_card(temp\_card);*

*}*

*}*

*#if TEST == PRINT\_ALL\_PLAYERS*

*/\*print out testing \*/*

*for (int i =0; i < amount\_players ; i++)*

*{*

*cout << "player: " << i << endl;*

*play\_array[i].print();*

*}*

*#endif*

*srand(time(NULL));*

*int turn = rand() % amount\_players;*

*cout << "PLAYER " << turn << " is randomly selected to play first" << endl;*

*confirm\_turn(turn);*

*bool force\_draw\_bool = false;*

*int turn\_flag = 1;*

*int win = 0 ;*

*while (win == 0 )*

*{*

*system("clear");*

*#if TEST == TEMP\_DECK*

*temp\_deck.print\_deck();*

*#endif*

*player \* curr\_player = &play\_array[turn%amount\_players];*

*cout << "PLAYER " << turn%amount\_players << endl;*

*if (force\_draw\_bool)*

*{*

*if (played\_card.number == 10)*

*{*

*cout << "Forced Draw-2" << endl;*

*card draw\_2;*

*for (int i = 0 ; i < 2; i ++)*

*{*

*draw\_2 = main\_deck.draw();*

*curr\_player->hand\_add(draw\_2);*

*}*

*}*

*if (played\_card.number == 14)*

*{*

*cout << "Forced Draw-4" << endl;*

*card draw\_4;*

*for (int i = 0 ; i < 4; i ++)*

*{*

*draw\_4 = main\_deck.draw();*

*curr\_player->hand\_add(draw\_4);*

*}*

*}*

*force\_draw\_bool = false;*

*}*

*cout << "Cards remaining for each player: " << endl;*

*for (int i= 0 ; i < amount\_players; i ++ )*

*{*

*cout << "PLAYER " << i << ": " << play\_array[i].get\_size() << " " ;*

*}*

*cout << endl;*

*cout << "Played Card: " << played\_card << endl;*

*cout << "PLAYER " << turn%amount\_players << endl;*

*curr\_player->print();*

*int check\_flag = 0 ;*

*int index;*

*int size = curr\_player->get\_size();*

*while (check\_flag == 0)*

*{*

*cout << "which card do you want to play? " << endl;*

*cout << "If you want to draw a card please enter '-1' " << endl;*

*cout << "Type the index of the card and press enter: ";*

*cin >> index;*

*if (index == -1)*

*{*

*card draw\_temp;*

*draw\_temp = main\_deck.draw();*

*cout << "DRAWN CARD: " << draw\_temp << endl;*

*if (draw\_temp == played\_card && draw\_temp.color != wild)*

*{*

*int play\_draw\_flag = 0 ;*

*while (play\_draw\_flag == 0 )*

*{*

*string temp\_play;*

*cout << "Do you want to play the drawn card [y/n] : " ;*

*cin >> temp\_play;*

*if (temp\_play == "y")*

*{*

*played\_card = draw\_temp;*

*temp\_deck.add\_card(draw\_temp);*

*if (played\_card.number >= 10 && played\_card.number <= 14)*

*{*

*force\_draw\_bool = true;*

*}*

*play\_draw\_flag = 1;*

*}*

*if (temp\_play == "n")*

*{*

*curr\_player->hand\_add(draw\_temp);*

*play\_draw\_flag = 1;*

*}*

*}*

*}*

*else*

*{*

*curr\_player->hand\_add(draw\_temp);*

*}*

*check\_flag = 1;*

*}*

*if (index >= 0 && index < size)*

*{*

*card temp = curr\_player->peek(index);*

*if (temp == played\_card)*

*{*

*curr\_player->hand\_remove(index);*

*temp\_deck.add\_card(temp);*

*played\_card = temp;*

*if(played\_card.color == wild)*

*{*

*int check\_color = 0;*

*COLOR temp\_color;*

*string str\_color;*

*while (check\_color == 0)*

*{*

*cout << "Please choose a color (red , green, blue, yellow) :";*

*cin >> str\_color;*

*temp\_color = FromString(str\_color);*

*if (temp\_color != wild)*

*{*

*played\_card.color = temp\_color;*

*check\_color = 1;*

*}*

*else*

*{*

*cout << "invalid color" << endl;*

*}*

*}*

*}*

*if (played\_card.number >= 10 && played\_card.number <= 14)*

*{*

*force\_draw\_bool = true;*

*}*

*check\_flag = 1;*

*}*

*else*

*{*

*cout << "card cannot be played " << endl;*

*}*

*}*

*else*

*{*

*cout<<"invalid index " << endl;*

*}*

*}*

*if ( curr\_player->get\_size() == 0 )*

*{*

*win =1;*

*cout << "PLAYER " << turn%amount\_players << " has won the game." << endl;*

*break;*

*}*

*if (played\_card.number == 11 && force\_draw\_bool == true )*

*{*

*if (turn\_flag == 1)*

*turn = turn+2;*

*else*

*turn = turn-2;*

*}*

*else if (played\_card.number == 12 && force\_draw\_bool == true)*

*{*

*if (amount\_players == 2)*

*{*

*turn = turn+2;*

*}*

*else*

*{*

*if (turn\_flag ==1)*

*{*

*turn\_flag = -1;*

*turn--;*

*}*

*else*

*{*

*turn\_flag = 1;*

*turn++;*

*}*

*}*

*}*

*else*

*{*

*if (turn\_flag == 1)*

*turn++;*

*else*

*turn--;*

*}*

*system("clear") ;*

*cout << "Cards remaining for each player: " << endl;*

*for (int i= 0 ; i < amount\_players; i ++ )*

*{*

*cout << "PLAYER " << i << ": " << play\_array[i].get\_size() << " " ;*

*}*

*cout << endl;*

*cout << "Played Card: " << played\_card << endl;*

*confirm\_turn(turn%amount\_players);*

*}*

*return 0;*

*}*

*player.cpp:*

*#include <cstdlib>*

*#include "card.h"*

*#include "player.h"*

*#include <iostream>*

*using namespace std;*

*player::player()*

*{*

*head = NULL;*

*size = 0;*

*}*

*player::player(const player & other)*

*{*

*copy(other);*

*}*

*const player & player::operator= (const player & other)*

*{*

*if ( this != &other)*

*{*

*clear();*

*copy(other);*

*}*

*return \*this;*

*}*

*player::~player()*

*{*

*clear();*

*}*

*void player::hand\_add(card temp\_card)*

*{*

*card\_elem \* temp\_ptr;*

*temp\_ptr = new card\_elem();*

*temp\_ptr->data = temp\_card;*

*temp\_ptr->next = head;*

*head = temp\_ptr;*

*size++;*

*}*

*card player::hand\_remove(int pos)*

*{*

*if (pos < 0 || pos >= size )*

*{*

*return card();*

*}*

*card\_elem \* prev\_ptr = head;*

*card\_elem \* target = prev\_ptr->next;*

*card temp\_card;*

*int temp\_pos = pos;*

*if (pos == 0 )*

*{*

*temp\_card = head->data;*

*head = head->next;*

*delete prev\_ptr;*

*size--;*

*return temp\_card;*

*}*

*while (temp\_pos > 1 )*

*{*

*prev\_ptr = prev\_ptr->next;*

*target = prev\_ptr->next;*

*temp\_pos--;*

*}*

*prev\_ptr->next = target->next;*

*temp\_card = target->data;*

*delete target;*

*size--;*

*return temp\_card;*

*}*

*void player::uno()*

*{*

*}*

*void player::print() const*

*{*

*int temp\_size = size;*

*int i = 0;*

*card\_elem \*temp\_ptr = head;*

*while (temp\_size > 0)*

*{*

*cout << i << ": " << temp\_ptr->data << endl;*

*temp\_ptr = temp\_ptr->next;*

*i++;*

*temp\_size--;*

*}*

*}*

*void player::copy(const player & other)*

*{*

*size = other.size;*

*if (size > 0 )*

*{*

*head = new card\_elem();*

*head->data = other.head->data;*

*}*

*else*

*{*

*head = NULL;*

*return;*

*}*

*card\_elem \* other\_ptr = other.head->next;*

*card\_elem \* temp\_ptr;*

*card\_elem \* prev\_ptr = head;*

*for (int i=1 ; i<size; i++)*

*{*

*temp\_ptr = new card\_elem();*

*prev\_ptr->next = temp\_ptr;*

*temp\_ptr->data = other\_ptr->data;*

*prev\_ptr = temp\_ptr;*

*temp\_ptr = NULL;*

*other\_ptr = other\_ptr->next;*

*}*

*}*

*void player::clear()*

*{*

*card\_elem \* temp\_ptr = head;*

*card\_elem \* next\_ptr;*

*while (size > 0)*

*{*

*next\_ptr = temp\_ptr->next;*

*delete temp\_ptr;*

*temp\_ptr = next\_ptr;*

*size--;*

*}*

*head = NULL;*

*}*

*int player::get\_size() const*

*{*

*return size;*

*}*

*card player::peek(int pos) const*

*{*

*int temp\_pos = pos;*

*card\_elem \* temp\_elem = head;*

*while (temp\_pos > 0 )*

*{*

*temp\_elem = temp\_elem->next;*

*temp\_pos--;*

*}*

*return temp\_elem->data;*

*}*