TABLE OF CONTENTS

Title of the project

UNO CARD GAME

Statement about the problem

Implementing multi-player UNO game using C++.

Reason/ Motivation to choose the topic

Well we all have played UNO at least once in our lives. But the idea to implement the game in the form of code seemed interesting. In the project, we tried to cover all the possible moves in the game so that the user gets the real feel to play UNO.

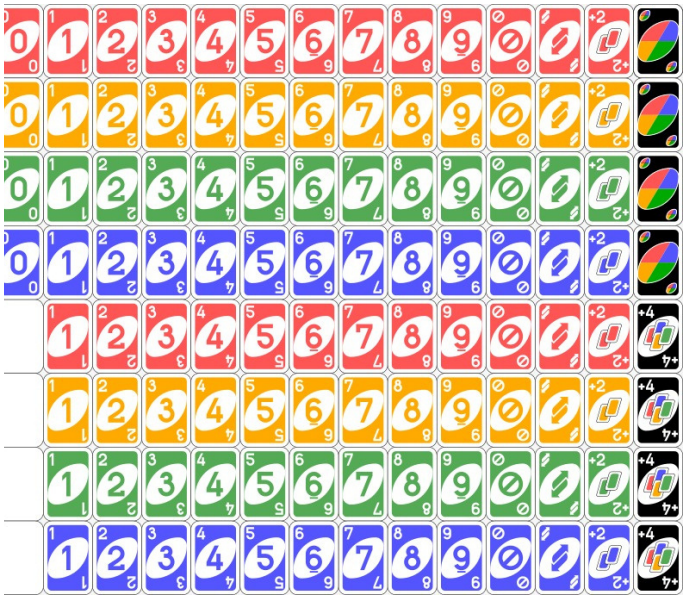
Objective and scope of the project

The project allows us to play UNO card game on our systems. Try to get rid of all the cards in your hand before your opponents. When it is your turn, try to match the card on the discard pile, either by number, color, or symbol.

Methodology (summary of the project)

Deck

For Uno to be played , firstly we needed the cards, but each card had its own identity , so we were supposed to come up with some encoding that could help us, the user and the code to understand which card we were dealing with. Upon searching on internet we came through this picture(below) of all cards. Through this we thought that the deck could be made as 2D array and each card then can be identified by their row and column.



For colours:

- if a card has row number equal to 0 or 4 then it is identified as Red.
- if a card has row number equal to 1 or 5 then it is identified as Yellow.
- if a card has row number equal to 2 or 6 then it is identified as Green.
- if a card has row number equal to 3 or 7 then it is identified as Blue.

For numbers:

- if card has column number below or equal to 9, then it is identified as its column number
- if card has column number equal to 10 it was identified as Skip card.
- if card has column number equal to 11 it was identified as Reverse card.
- if card has column number equal to 12 it was identified as Plus 2 card.

For Wild Cards:

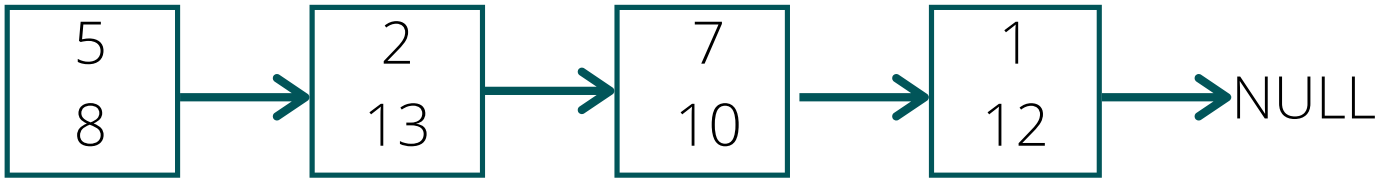
- if a card had column number equal to 13 it was considered as a wild card.
- if that card had row number from 0 to 3 it was a colour changer.
- if that card had row number from 4 to 7 it was a Plus 4.

This was all about the deck, now the issue was, hand of the player.

Methodology (summary of the project)

Player's Hand

For the Hand of the Player we decided to use Linked List. It is a simple and very convenient data structure which offers O(n) complexity for both insertion and deletion that in worst case.
Hence Linked List was a preferable choice over complicated tree structure.



Now our code was able to identify which card it was dealing with, but still the user will not be able to understand the card by its row and column.

For this problem we came up with the denotation that :

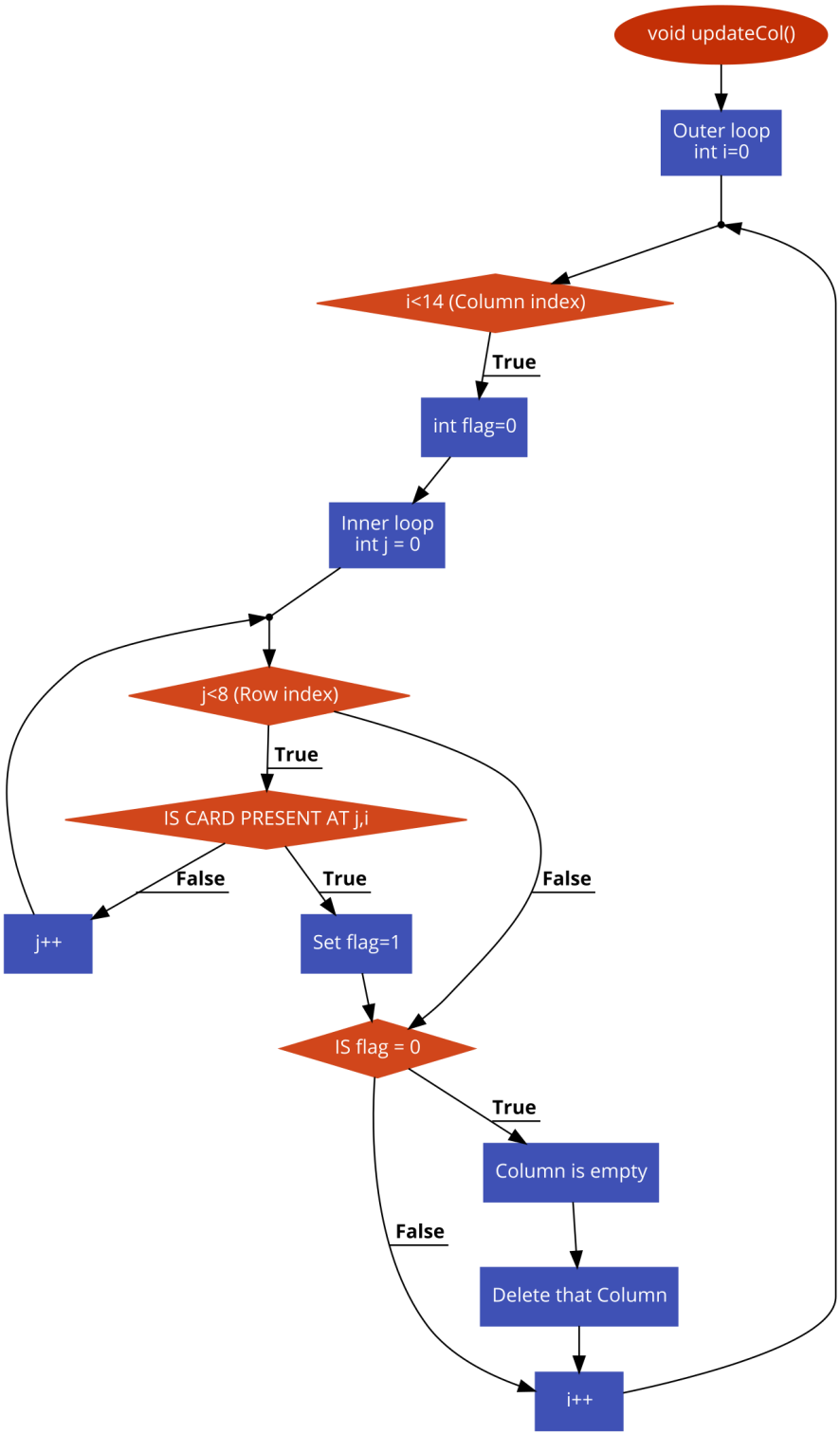
- each card would be represented as a string with two characters
- first character will tell the colour of the card , for e.g. B_ ,Y_ , G_ , R_
- second character will tell its number , for e.g. _2 , _4
- for skip we wrote _S , for reverse _R and for plus 2 _T
- for wild colour changer , the card was WC
- for wild plus 4 , the card was WF

This was the prior knowledge with which we started our work. Further details and complete methodology is explained in the comments. Below we have attached the flowcharts for some key functions that gave our application, its shape.

We tried our best to build the application and minimize errors, but still, an application is never really complete. The application is running on Code Blocks and all basic steps are conducted easily , but still there are chances that bugs may occur, if so we are very sorry for that. We hope that you can forgive us upon that and enjoy our application with your friends and family.

void updateRow()

After a certain amount of draws, there is a good chance that some rows have been emptied in the 2D Deck matrix. The purpose of this function is to delete such rows, significantly reducing the time taken in selecting random cards.



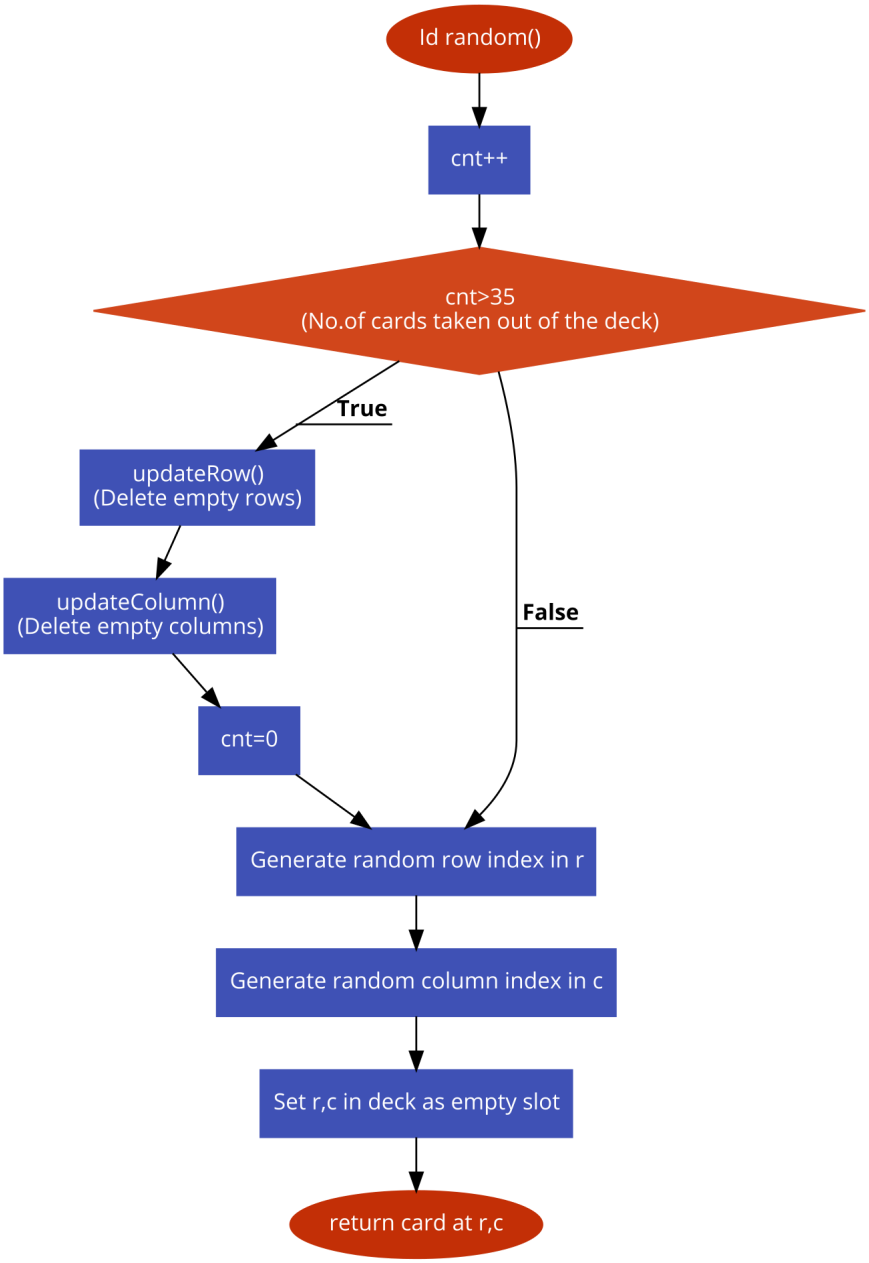
void updateCol()

Just like rows, some columns may be empty too. updateCol() deals with that.

Id random()

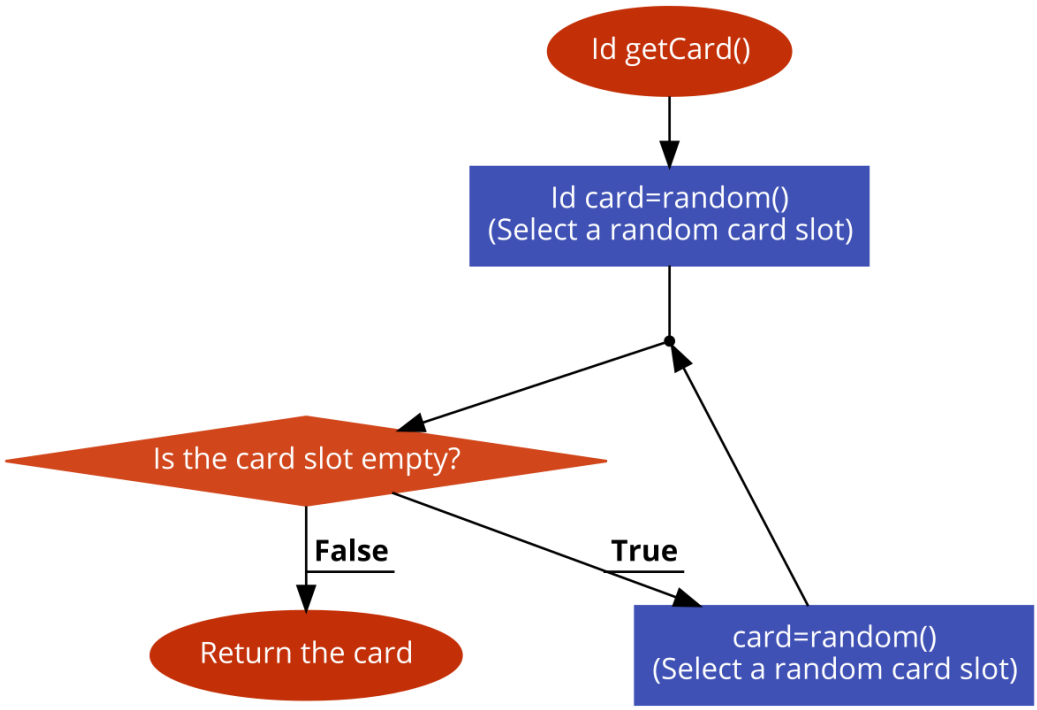
This function generates the indices of a card slot, with the help of the CPP in-built function rand(), thus returning a random card from the deck. It is used while dealing hands, drawing a card etc.

.



void getCard()

If random() returns a card that has already been drawn from the deck, getCard() calls it again until it returns an undrawn card, rectifying any possible errors.

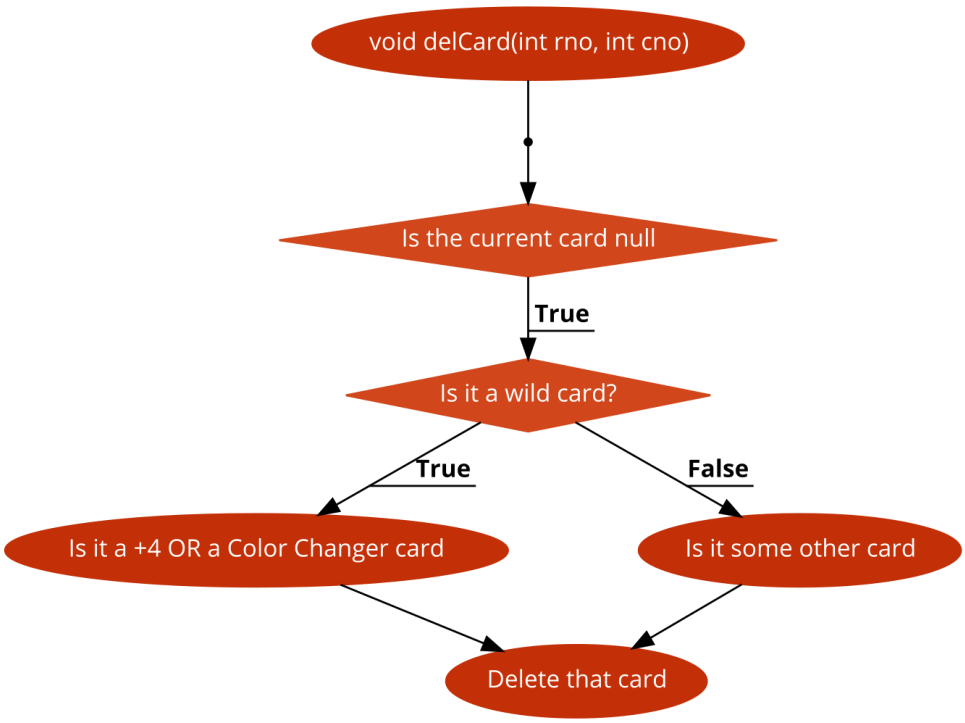


void addCard()

This function does the job of generating a players's hand by adding a random card called using getCard() in the end of linked list corresponding to that player.

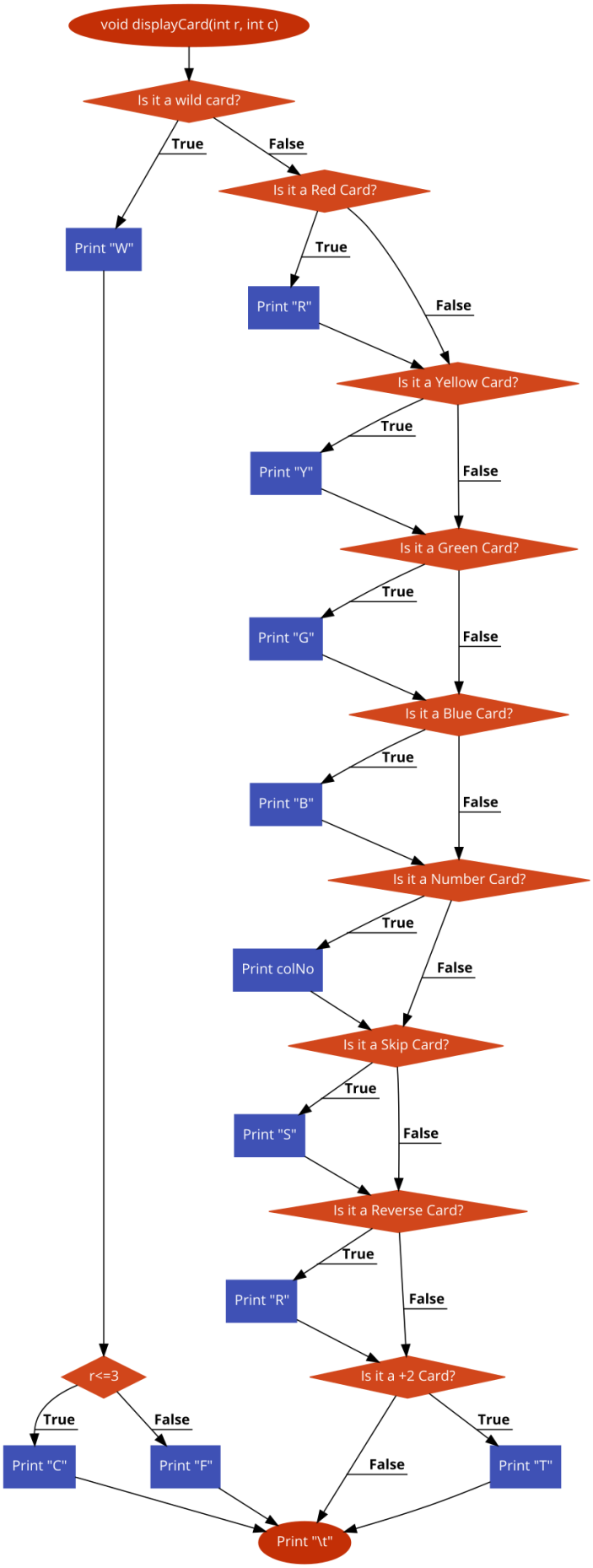
void delCard()

Whenever a card is played by a player, it has to be deleted from that player's hand. This operation is done by delCard().



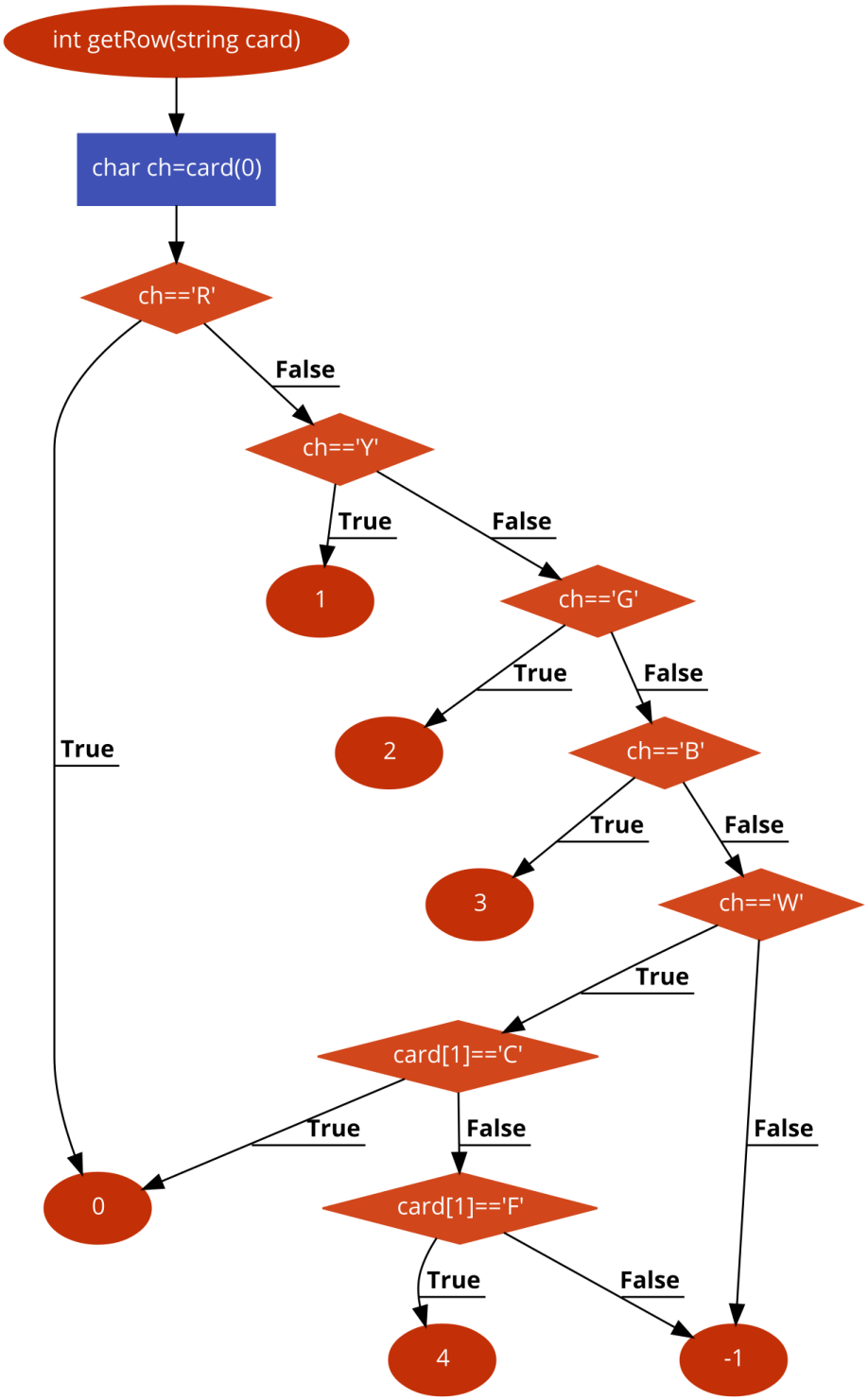
void displayCard()

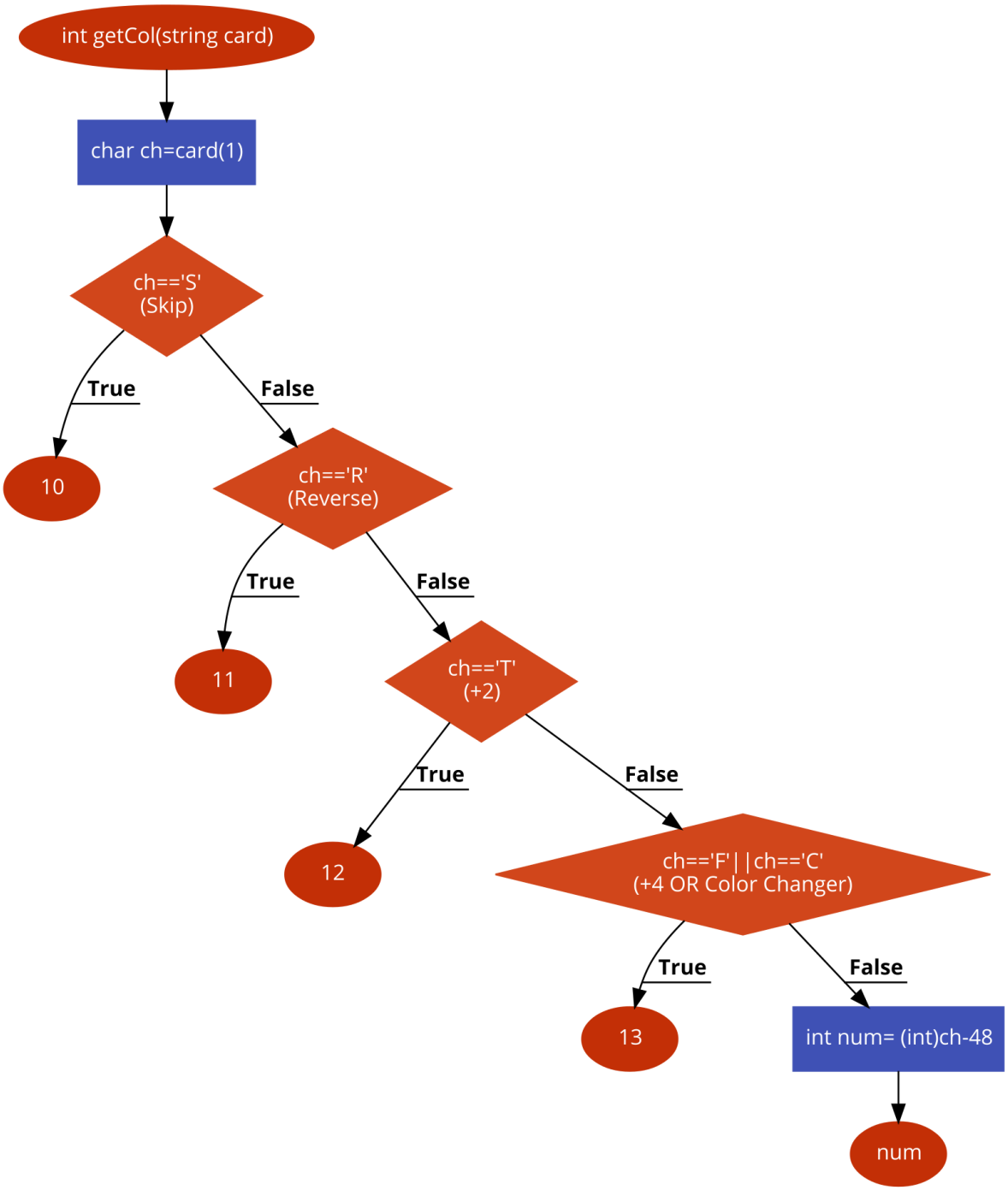
All the cards are stored using their row and column indices. To make it understandable to the player displayCard() is used. This function is called whenever a card is to be displayed in String form.


void displayCard(int r, int c)
Is it a wild card?
True
False
Print "W"
Is it a Red Card?
True
False
Print "R"
Is it a Yellow Card?
True
False
Print "Y"
Is it a Green Card?
True
False
Print "G"
Is it a Blue Card?
True
False
Print "B"
Is it a Number Card?
True
False
Print colNo
Is it a Skip Card?
True
False
Print "S"
Is it a Reverse Card?
True
False
Print "R"
Is it a +2 Card?
True
False
Print "T"
r<=3
True
False
Print "C"
Print "F"
Print "\t"

int getRow() and int getCol()

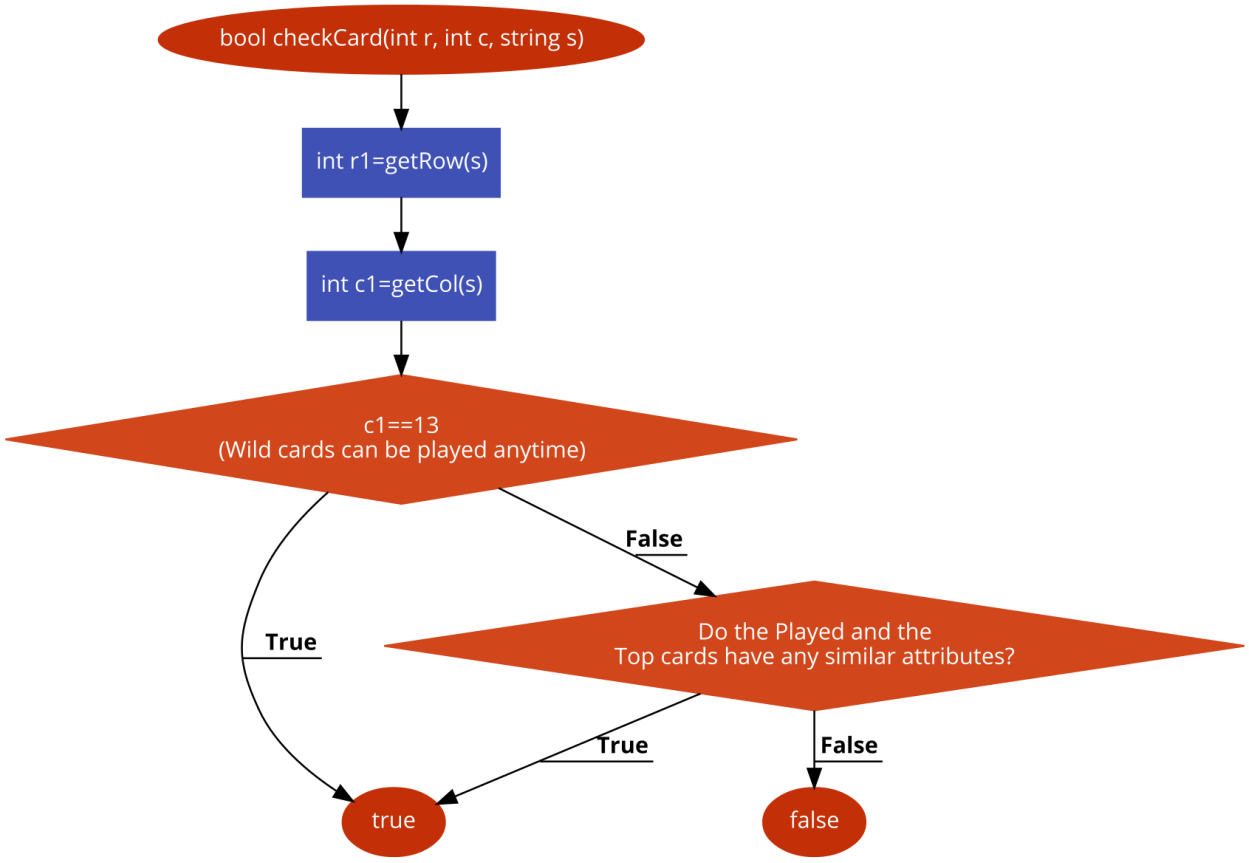
These functions have the opposite use of displayCard(). They convert user entered card choice in string form to corresponding row and column indices.




int getCol(string card)
char ch=card(1)
ch=='S'
(Skip)
True
False
10
ch=='R'
(Reverse)
True
False
11
ch=='T'
(+2)
True
False
12
ch=='F'||ch=='C'
(+4 OR Color Changer)
True
False
13
int num= (int)ch-48
num

bool checkCard()

For a card to be playable, it should have atleast one matching attribute with the last played card i.e. the top card(except in the case of a wild card, which has no restrictions) . bool checkCard() is used to verify the same.



6. Hardware & software requirements

Codeblocks or any other C++ compiler

7. Contribution that the project will be able to make

The game is very enjoyable if played with friends, and thus helps to reduce stress.