Project 1

<Bingo>

CIS5

Miguel Flores

1/28/2024

Intro

Title: Bingo

Bingo is played on a grid in which there is already a predetermined set of numbers that have been set within the grid, with the objective being to mark off the numbers as the game continues in hope to achieve a pattern such as diagonal/ straight line shape to win the game. The grid consists of a 5x5 grid and the bingo card will contain a new grid with numbers being arranged within the columns and rows of the grid, to be filled out by the passing of the game. The numbers are to be called out and if the number pulled matches one of the numbers within your given bingo card, then the player gets to mark off that number in hopes of getting more matching numbers to create the pattern to get the win. The game will continue on until a player wins and meets the required pattern on their card.

Expect a game in which will one way or another eventually fall into your favor if you have the right luck to become a winner and be crowned the bingo champion for that round, the game is interactive and will want you desiring the matching numbers to fill in your card, keeping you on the edge in hopes of not only getting a matching number, but being able to get multiple matching number of hopes of creating a pattern to win!

A player begins with a blank slate, just their given numbers in hopes of filling in matching numbers to be crowned the winner of the bingo match. If the number pulled does not attribute to be a matching number then the game will continue until you get a matching number, and the bingo chart is filled. Happy huntings!
and eggs remaining in their holes wins the game, and the game ends.

Summary/Versions

Project size: 600+ lines
?? The number of variables: about 30 The number of method: 17

V1.) I started by first getting a base code to later work off of in terms of reading off of the checkoff sheet and going down line by line in it to then more better refine what i am trying to create in the grand scheme of it, but i first started off with the skeletal version of the code, such as a rough rough first draft not really being to hard on myself in terms of the checkoff sheet just yet. I chose to include the libraries all having their own purpose for constructing the game such as <iostream>, <cstdlib>, and also <ctime>. I wanted to make sure I had the ability to use input/output operations in my program and have the ability to print results based on the game's outcome, i also needed to have random number generation and other utility function, to generate random numbers/results, and i also needed to be able to try to make sure that random numbers are generated to be different each time based on the numbers i have inputted in my code. I layed out the first version of the code and it was difficult for me to lay out all the calculation code, such as using the base code I set, working around a grid of a 5x5 game for the actual bingo game, so i set all the step by step process to the game, the started to fill in the line of codes, but i needed to work off step by step because it was overwhelming of thinking of the next step when trying to just start it off. When coding i worked on some essential parts of the code such as by separating in terms of, generating a maximum and minimum within the game, initializing an original 5x5 grid, simulating the generation of random numbers until a win condition is met, and then establishing what the win conditions are gonna be.

V2.) When beginning to write the code, I wanted to do the process in a somewhat ordered form to begin the process, so I started with creating the random number generator first, using the min and max parameters to represent the minimum and maximum values of the range of the generated number. I used parameters such as rand() to be able to generate the number set, rand to the max set rand, with the rest of the function will be able to generate the random number, within a specific set of parameters. In composing my main function, i used the srand once again for the random number generation, to ensure the numbers generated will be different and random each time the program gets ran, and also using const int, to be able to set the variables of the grid size, in which i want to do a 5x5 for my game construct, and numbers of the row to complete the shape up of the 5x5 grid, in pertaining to the rows/columns. During the same session, I also went ahead and went on coding the framework of the grid layout, so I put a bingo grid to layout the grid size, showing where the numbers will be placed. For each combination of the ‘i’ and the ‘j’ in the code, a unique square is to be randomly chosen within the bingo match, and the array element in which the ‘i’ and ‘j’ get assessed are within the loops. Int i=0; i <grdsize; ++i represents the outer loops that iterate the rows of the grid and int i=0; j <grdsize; ++j to set the constraints for the columns of the grid. The purpose of the code I wanted it to do was to ensure each cell within the grid and for the cells to be able to be manipulated and initialized during the game.

V3.) I went ahead and made a line which was intended to follow the ‘i’ and ‘j’ to address the cells within the units of the grid to generate a unique number for each cell within the grid. The line comprises of bingGrid[i][j]=genRandNum(j * numsPerRow + 1, (j+1) * numsPerRow), with its purpose to be assigned a random number to a specific cell position, having ability to maneuver within the column/lines of the code, and to also do things such as having constraints in terms of the low/high range of the random generated numbers. The main thing I was going for was to go for randomness in the generation of numbers and the pool cells the numbers can randomly land in. Finally to wrap up the session, I set the output of the game “thanks for playing bingo”.

V4.) The main focus of the session was to be able to compute a win condition to the game and be able to simulate the random generation of the numbers to get to that conclusion. The purpose was to draw random numbers until the win condition criteria was met and, using the (‘while (true)’) sequence to measure the repeated pulling of numbers until a wind condition is met or the player no longer wants to continue playing the bingo game. I wanted the loop’s function to establish some sort of progression within the game, to ensure numbers are being pulled to fill the bingo chart, and for them to be able to be displayed onto the user’s interface, just as the real life version is played with numbers being pulled until a player can achieve the bingo win conditions, such as a pattern. I wanted the line of code ‘int drawNum = genRandNum(1, grdSize * numPerRow) to generate the number between 1, and the max input, followed by a set specified range with the (min,max), hoping to generate a random number within the set parameters. I feel this was one of the most important steps within the creation of the game as not only does it contribute towards the game’s overall progression but it also is constantly pulling numbers until a win condition is met. I then went on to start the start of the win conditions and how they would look like, using ‘bool win = false’ to track whether or not the win condition is being met through the game’s rows and columns of the grid, and furthering it by adding ‘for (int i=0; i < grdSize; ++i)’ to actually go through the columns and rows of the grid, setting a loop, and finally having ‘bool rowWin = true, colWin = true;’ to show within the loop a winner if the requirements are met, with the two variables rowWin and colWin representing variables being tracked/checked to see if the win condition is met.

V5.) This session i wanted to continue on sorting out the win conditions, so i continued with the next line to create, using (int j=0; j<grdSize; ++j to show the analyzing of the

columns after the generation of the numbers with j being the loop control ranging through the grid, and the next following line (bingGrid[i][j] != drawnNum) to check if the number pulled is in the row (i) and the column (j) of the grid trying to make it so if the pulled number does not equal the number set within the grid box then it will not get represented within the grid box setting the rowWin to 'false', and vice versa (bingGrid[j][i] != drawnNum) having the 'drawnNum' being checked if present within the column 'i' and the row 'j' within the grid, with if the number doesn't match the pulled number then colWin is subject to being 'false'. The Desired purpose was for the line to be able to look through the cells within the grid to be checked and to subsequently being able to determine whether or not the pulled number is present within one of the cells within the columns and rows of the grid, and if the number is not present, it will match the conditions of a 'false', you will get the option then to continue pulling numbers as you got nothing to match. I then went on to integrate (rowWin I colWin) and then win=true, and break to show if either of the variables rowWin or colWin is matching the winning condition which then the matches down to win=true, showing the a win condition was met within the game, and after that is met i wanted to provide a restart of the code to repeat once more after the condition is met so i implemented a break, so the game can continue the process of pulling a win condition of a number until the winner is decided.

V6.) I wanted to continue adding on the win conditions, by introducing a final prompt in which displays 'bingo! You are the winner!', so i set the variable to 'win' and if the conditions being met, the variable is 'true', therefore determining if the win condition has been fulfilled/met within the bingo game, then having the message being displayed after the matter of fact, showing you, you have won the bingo game. Just to restart and start a new game I used a break to be able to end the game from looping and to end the game.

V7.) In this session the main goal was to finish off the conditions to the game, giving other messages to the player if the game still needed to continue, such as still having to keep pulling numbers until a win was met, and signifying a thank you message for playing the game. I showed the message 'no winner yet, continue drawing numbers' as an indication that you need to keep drawing numbers in order to see if you will win and showing the game will continue, and also the thank you for playing message to provide a concluding statement to the game played, and to get the game ready for another session if the player wishes to continue.

V8.) The main purpose of the session is to look at the code in which i first initialized and now crossing some of the bullet points off that are of concern off the cross reference list making sure i can incorporate the various different techniques within the guideline, and build off of what i already have, and remove what i don't need. I wanted to first address the things i thought held a lot of emphasis, so i went ahead and wanted to make sure i utilized variables in ways in which the game statistics and other data could be held within the game files itself, so i went ahead and defined variables such as the number of rounds that were played, the number of wins that the player has, and the total of numbers that have been drawn. The main purpose of the change was to utilize variables within the game that can track the active game stats. The next thing I wanted to change is ensure I was not using doubles in the code, so I wanted to ensure that my floats within the code are correct and used properly, so I just looked over everything and made sure the floating point variables were declared and accounted for. I then wanted to see if I could include an input/output system so it could read and save the game stats for the next time you play to see, almost similar to a leaderboard but not quite the same, just wanting it to serve like a history system. For the file input functionality I wanted it to be able to read the game stats from when the game starts and be able to save the game stats at the end of a game for the file output, all serving the purpose to store and retrieve the previous/present game history through the file named game_stats.txt within the game. I then moved on to making sure i could include as many libraries which pertain to the code, such as iomanip for formatting the output, string for the string operations, ctime for the time manipulation factor, and the other libraries to be included into the code.

V9.) For the session I wanted to look at what I had laid out so far in terms of the code, and be able to explain the purpose of each section of the code, make sure it is flowing with a consistent style and is able to be understood. I also wanted to double check and ensure there were no breaks within the loops, and if I see any to remove them and replace them with a more appropriate manner and flow. I added a std::string line of code in which I wanted to be able to display the player's name and prompt the player to start the game, implementing some strings into the code. I wanted to implement validating the user input after ensuring the player's name is not empty, and that a name is present to start the game session, and if the name is empty, the game will not be started until a name is provided and stated, but also having a way to include to validate user input. I then went ahead and tried to cover and reiterate everything in which I missed in doing the loops, operators, and checking the grand scheme of the overall project code as an entirety, and if it flows to work in one motion. I then went more intricate with the ways in which one ways such as putting how the player won, in the manner that they won.

V10.) To finalize and polish the code, i noticed i still needed some requirements filled out, so i went ahead to integrate some operators still and some loops, so i went ahead and implemented a conditional operator in the user input validation. I also used another loop to display the bingo grid, used other loops within the code such as a while loop in the drawing numbers sequence, and added a float to calculate the players score as the game progresses through the rounds.

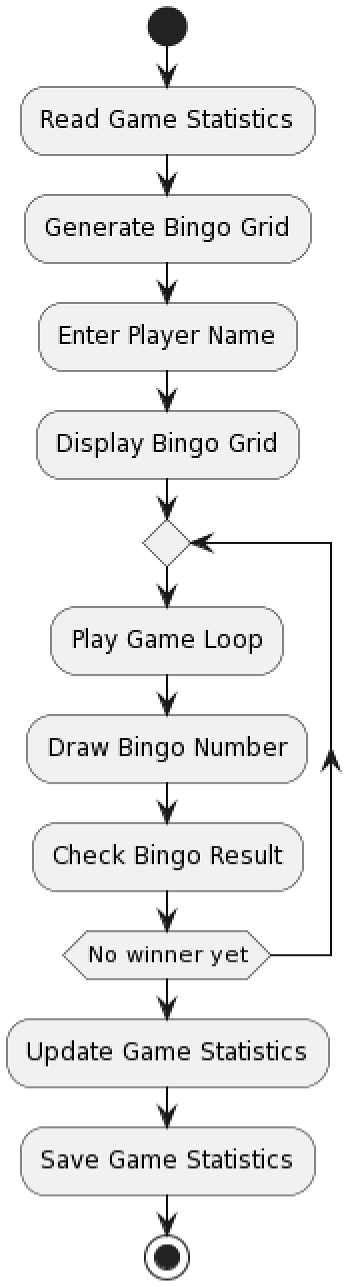
Description

The main point of this program is to allow players to enjoy a game of bingo, through the generation of random numbers in hope of achieving a pattern to win.

FlowChart/ Pseudo Code

```
Initialize game stats/ variables
Read game stats in start menu, if available
Generate and let the bingo grid initialize
Prompt the player for their name
Display grid/ start game
Draw numbers until a win condition is
    row/column
Save stats of the game
Say congrats to the winner, and end game
```

Bingo Game Flowchart



Program

```
//Bingo

#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <string>

// Function to generate a random number between min and max (inclusive)
int genRnd(int min, int max) {
   return rand() % (max - min + 1) + min;
}

int main() {
   srand(time(0)); // Prepare for generating random numbers
   const int grdSz = 5; // Number of rows and columns in the bingo grid
   const int numsPrR = 5; // Number of numbers per row/column in the grid

   // Variables to track game statistics
   int rndsPl = 0; // Number of rounds played
   int wins = 0; // Number of wins
   int numsDrwn = 0; // Total number of numbers drawn
   float plyrScr=0.0f;// the player's score

   // Read game statistics from a file
   std::ifstream inF("game_stats.txt");
   if (inF.is_open()) {
      inF >> rndsPl >> wins >> numsDrwn;
      inF.close();
   } else {
      std::cout << "Could not open file. Starting with default game statistics.\n";
   }

   // Initialize the bingo grid with random numbers
   int bngGd[grdSz][grdSz];
   for (int i = 0; i < grdSz; ++i) {
      for (int j = 0; j < grdSz; ++j) {

         // Generate a unique number for each cell in the grid
         bngGd[i][j] = genRnd(j * numsPrR + 1, (j + 1) * numsPrR);
      }
```

```
}

std::string plyrNm; //Player can input their name

std::cout << "Welcome to Bingo!\n";

// Validate user input for player's name
do {
    std::cout << "Please enter your name: ";
    std::getline(std::cin, plyrNm); // Get player's name
    if (plyrNm.empty()) {
        std::cout << "Invalid name. Please try again.\n";
    }
} while (plyrNm.empty());

std::cout << "Hello, " << plyrNm << "! Let's start the game!\n";

// Display the bingo grid
std::cout << "Bingo Grid:\n";
for (int i = 0; i < grdSz; ++i) {
    for (int j = 0; j < grdSz; ++j) {
        std::cout << std::setw(4) << bngGd[i][j];
    }
    std::cout << std::endl;
}

// Simulate drawing numbers until a win condition is met
while (true) {
    int drwnNum = genRnd(1, grdSz * numsPrR); // Generate a random number
    std::cout << "The drawn number is: " << drwnNum << "\n";
    ++numsDrwn; // Keep track of the total number of drawn numbers

    // Check if the drawn number matches any row or column in the grid
    bool rwWin = false;
    bool colWin = false;
    for (int i = 0; i < grdSz; ++i) {
        bool rwMtch = true;
        bool colMtch = true;
        for (int j = 0; j < grdSz; ++j) {
            if (bngGd[i][j] != drwnNum) // Check the row
                rwMtch = false;
            if (bngGd[j][i] != drwnNum) // Check the column
                colMtch = false;
        }
```

```
            if (rwMtch)
                rwWin = true;
            if (colMtch)
                colWin = true;
        }

        if (rwWin || colWin) {
            if (rwWin && colWin) {
                std::cout << "Bingo! " << plyrNm << " won by matching both a row and a column!\n";
            } else if (rwWin) {
                std::cout << "Bingo! " << plyrNm << " won by matching a row!\n";
            } else {
                std::cout << "Bingo! " << plyrNm << " won by matching a column!\n";
            }
            break; // Exit the loop if there's a win
        }

        std::cout << "No winner yet. Keep drawing numbers...\n";
    }

    // Save game statistics to a file
    std::ofstream outF("game_stats.txt");
    if (outF.is_open()) {
        outF << rndsPl << " " << wins << " " << numsDrwn;
        outF.close();
    } else {
        std::cout << "Could not save game statistics to file.\n";
    }

    std::cout << "Thanks for playing!\n";
    return 0;
}
```