Introduction

Game Title: Code Breaker

Being fascinated by the ability family members or strangers to crack codes so effortlessly to the point of admiration was a motivation to pursuit building this mind number game. anybody who strives to obtain and sharpen the skill of cracking codes can play and have fun with this game. In order to train such a skill a reconstruction of the game Mastermind, made by Invicta,

So, what is Code Breaker? It is a number logic game that arrange number sequences together to generate a code only known to the opponent or a computer system. In addition to the fun part, many benefits will come from playing this game. To start, this game again confirms the rule of practice makes perfect because the more you play this game, the better logic and thinking skills get. The emotional, social, and mental reward is very important in playing this game; it aids with bettering self-esteem, mathematical abilities, especially arithmetic skills like sequences.

At the competitive gamers level, strategy becomes very important element in this game. Every number will be considered crucial and well calculated. This strategy is a useful skill in life and work place.

These aspects will certainly translate into the computer code by giving this game the versatility and the ability to expand capability and options to users. This game can be portable everywhere in everyone’s pocket. With network technologies, this game can be at every corner and every environment of our life. Digital Code Breaker brings fun and skill to everyone who has a phone or a smart device.

This game can be played everywhere, on the go, or train commute, lunch break, or at the park while your kids a playing on the swings.

Game Rules

Step 1: One player or code maker (The computer in this case) secretly generates a random code of 6 digits combination of numbers from 1 through 9. The code is only known to the computer system. The code consists of any combination of six digits comprised of any combination of numbers; it can include repeated numbers, incrementing numbers, or numbers repeated numbers which are next to each other. For example, 111111, 112233.

Step 2: The player known as the Codebreaker, types a 6 digits number for the first time based on a guess within the boundaries of and criteria set in the beginning. The computer will prompt the player to type the code in the designed table.

Step 3: The computer system will then return a feedback of the code correct digits and misplaced digits in the designed spaces. Based on the information outputted by the system, the player then is prompted to enter another attempt where he can change the order of digits or change the code to his or her preference.

Step 4: This version of game limits the number of the code breaker attempts to certain number. If all attempts are used than the game is over and the player loses the game. In case he guesses the code correctly, a congratulation message will show and the player will be prompted to play again.

Summary

Project size: 172 lines

Description lines: 32 lines

Code lines: 165

Variables around roughly 13

Library: #include <iostream, cstdlib, iomanip, ctime>; .h file was attempted to

Improve performance, but was unsuccessful. Function prototype were used instead.

Function: around 6

This program includes many concepts learned in the lecture as it contains operators and functions that were covered in the previous lessons; however, this code was built with expansion and diversity in mind. This project will include room for adding new concepts and methods like more scoring options, arrays, and may be visual effects and even sounds.

This project is difficult and challenging yet informative and rewarding; it offered a new experience to expand knowledge and test ability for attempting new challenges. This code is only a starting point for getting to discovering coding and the limitless boundaries of computer code as it brings an empowering experience.

Description:

This game is going to consist in reading headings and instructions, as well as prompts to enter digits of a certain range to match the generated code. At this stage, no restrictions in attempts will be implemented, but an attempt counter will display at the end of the game to show how many attempts it took to crack the code. This game will be scoreless also, and will be subject to project 2.

Flow Chart:

Pseudo Code:

Code Title: CodeBreaker

/\*

\* File: main.cpp

\* Author: Kal Dridi

\* Created on July 19, 2017, 9:04 PM

\* Purpose: Template to be utilized in building

\* Code Breaker using CPP

\*/

//System Libraries

//Global Constants

//User Libraries

//Range of digits allowed in the code (only digits from 1 to 9 are allowed)

//Length of code

//limits the number of guesses; this version is more difficult than previous

//Functions Prototype

//Main code starts here

//Declare random variables and initialize

//Random Number Generator

//Declare Variables

//secret code and player's guess

//player input at the end of game

//initialize Game by Output Welcome message

//While loop for validation Condition of Game Over

//while loop for guessed code to test for validity of code

//Nested if to check player's code.

//for loop to check for ascending code digit

// checking for correct digit

// checking for misplaced digits

/\* Print player's guess and number of correct/misplaced digits \*/

/\* If they are, print a message and the guess counter, and flip the "guessed code" flag to end current round \*/

//Message Output that asks the user to Play again!!

//Data Input repeat game.

// end while

//Print Exit Message

// end main

//Function Definition and Declare internal variables

//Declare local variables and initialize variable for get\_player\_code function

//While loop for Testing valid code

//Nested if else if loop test of player input length

//Beginning of for loop to start counter for player input

//Nested if conditional statement

// end while

// end get\_player\_code()

//Function Definition for random number generator used by code randint()

//Random number generator

//Random number generator

Major Variables:

Cross reference for Project 1

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | Section | Topic | Where in code/Line # |
| Chapter 2 | 2 | The cout object |  |
|  | 3 | #include directive |  |
|  | 4 | Variables |  |
|  | 5 | Identifiers |  |
|  | 6 | Integer data type |  |
|  | 7 | The char data type |  |
|  | 8 | The C++ sting data |  |
|  | 9 | Floating point data |  |
|  | 10 | The bool data type |  |
|  | 11 | Determining size |  |
|  | 12 | Variable assignment |  |
|  | 13 | Scope |  |
|  | 14 | Arithmetic operators |  |
|  | 15 | Comments |  |
|  | 16 | Named constants |  |
|  | 17 | Programming style |  |
|  |  |  |  |
| Chapter 3 | 1 | The cin object |  |
|  | 2 | Mathematical expres |  |
|  | 3 | Mixing variables |  |
|  | 4 | Overflow/underflow |  |
|  | 5 | Type casting |  |
|  | 6 | Multiple assignment |  |
|  | 7 | Formatting output |  |
|  | 8 | Objects |  |
|  | 9 | Mathematical library |  |
|  | 10 | Debugging |  |
|  | 11 | Problem solving |  |
| Chapter 4 | 1 | Relational operators |  |
|  | 2 | If statement |  |
|  | 3 | Expanding if |  |
|  | 4 | The if/else |  |
|  | 5 | Nested if |  |
|  | 6 | If/else if statement |  |
|  | 7 | Flags |  |
|  | 8 | Logical operator |  |
|  | 9 | Checking numeric |  |
|  | 10 | Menue |  |
|  | 11 | Validating input |  |
|  | 12 | Compring char |  |
|  | 13 | conditional |  |
|  | 14f | Switch statement |  |
|  | 15 | Block variable |  |
|  |  |  |  |
| Chapter 5 | 1 | Increment/decrement |  |
|  | 2 | Introduction to loops |  |
|  | 3 | While loop |  |
|  | 4 | Counters |  |
|  | 5 | Do while loop |  |
|  | 6 | The for loop |  |
|  | 7 | Running totail |  |
|  | 8 | Sentinels |  |
|  | 9 | Deciding whick loop |  |
|  | 10 | Nested loop |  |
|  | 11 | Using files |  |
|  | 12 | Breaking |  |
|  |  |  |  |
| Chapter 6 | 1 | Modular program |  |
|  | 2 | Calling functions |  |
|  | 3 | Prototype |  |
|  | 4 | Sending data function |  |
|  | 5 | Passing data |  |
|  | 6 | Using function |  |
|  | 7 | Return statement |  |
|  | 8 | Returning value |  |
|  | 9 | Boolean value |  |
|  | 10 | Local/global variable |  |
|  | 11 | Static local |  |
|  | 12 | Default argument |  |
|  | 13 | Using reference |  |
|  | 14 | overloading function |  |
|  | 15 | Exit function |  |
|  | 16 | Stubs and drivers |  |

References:

1. Textbook Gaddis 8th edition chapter 1 through 6
2. Netbeans 8.2 version with Cpp compilers
3. GitHub.com

Program:

// Code Breaker

/\*

\* File: main.cpp

\* Author: Kal Dridi

\* Created on July 19, 2017, 9:04 PM

\* Purpose: Template to be utilized in building

\* Code Breaker using CPP

\*/

//System Libraries

#include <iostream>

#include <cstdlib>

#include <iomanip>

#include <ctime>

using namespace std;

//Global Constants

//User Libraries

//Range of digits allowed in the code(only digits from 1 to 9 are allowed)

const char CODE\_SPAN = 9;

//Length of code

const int CODE\_LENGTH = 6;

//limits the number of guesses; this version is more difficult than previous

const int MAX\_GUESSES = 10;

//Functions Prototype

int randint(int);

int randint(int, int);

string get\_player\_code();

string generate\_code();

int main() { //Main code starts here

//Declare random variables and initialize

//Random Number Generator

srand( time(NULL) );

//Declare Variables

//secret code and player's guess

string player\_guess, secret\_code, secret\_copy, player, player\_copy;

//player input at the end of game

char repeat\_game;

int num\_guesses;

bool game\_over = false, guessed\_code = false;

//initialize Game by Output Welcome message

cout << "WELCOME TO CODEBREAKER!\n\n";

cout << "Oh no! The computer has generated a secret code. Can you guess it?\n";

//While loop for validation Condition of Game Over

while ( !game\_over ) {

cout << '\n' << setw(40) << "YOUR GUESSES:" << setw(15) << "CORRECT:" << setw(15) << "MISPLACED:" << '\n';

cout << string(70, '=') << '\n';

secret\_code = generate\_code();

secret\_copy = secret\_code;

num\_guesses = 0;

//while loop for guessed code to test for validity of code

while (!guessed\_code) {

secret\_code = secret\_copy;

player = get\_player\_code();

player\_copy = player;

int correct = 0;

int misplaced = 0;

num\_guesses++;

//Nested if to check player's code.

if (player == secret\_code){

cout << "Congratulations, you guessed the code!\nIt only took you " << num\_guesses << " guess(es)\n\n";

break;

}

//for loop to check for ascending code digit

for (int x = 0; x < CODE\_LENGTH; x++){ // checking for correct digit

if (player[x] == secret\_code[x]){

correct++;

player[x] = 'c';

secret\_code[x] = 'd';

}

}

for (int x = 0; x < CODE\_LENGTH; x++){ // checking for misplaced digits

for(int i = 0; i < CODE\_LENGTH; i++){

if (player[x] == secret\_code[i]){

misplaced++;

player[x] = 'z';

secret\_code[i] = 'm';

}

}

}

if(num\_guesses < MAX\_GUESSES && player == secret\_code){

guessed\_code = true;

}

/\* Print player's guess and number of correct/misplaced digits \*/

cout << '\n' << setw(40) << player\_copy << setw(15) << correct << setw(15) << misplaced << '\n';

cout << string(70, '=') << '\n';

if ( !guessed\_code ) {

/\* If they are, print a message and the guess counter, and flip the "guessed code" flag to end current round \*/

cout << "Sorry, you've run out of guesses!\n";

cout << "The secret code was: " << secret\_code << "!\n\n";

}

else{

cout<<"Congrats! You've guessed the secret code in "<< num\_guesses << " guess(es)"<<"."<<'\n';

}

}

//Message Output that asks the user to Play again!!

cout << "Would you like to play again (Y/N)? ";

//Data Input repeat game.

cin >> repeat\_game;

repeat\_game = toupper(repeat\_game);

if ( repeat\_game == 'N' ) {

game\_over = true;

}

else {

guessed\_code = false;

}

} // end while

//Print Exit Message

cout << "Thanks for playing!\n";

return 0;

} // end main

//Function Definition and Declare internal variables

string get\_player\_code() {

//Declare local variables and initialize variable for get\_player\_code function

string player\_code = "0000";

string player\_input;

bool valid\_code = false;

//While loop for Testing valid code

while ( !valid\_code ) {

player\_code = "0000";

cout << "Enter Code: ";

cin >> player\_input;

//Nested if else if loop test of player input length

if (player\_input.size() < CODE\_LENGTH) {

cout << "ERROR: Code Too Short!\n\n";

}

else if (player\_input.size() > CODE\_LENGTH){

cout << "ERROR: Code Too Long!\n\n";

}

//Beginning of for loop to start counter for player input

for (int x = 0; x < player\_input.size(); x++){

if (player\_input[x] > (CODE\_SPAN + '0') ){

player\_code = "1";

}

}

//Nested if conditional statement

if (player\_code == "1"){

cout << "ERROR: Code contains digits out of range specified or are not integers!\n\n";

}

if (player\_input.size() == CODE\_LENGTH && player\_code != "1"){

valid\_code = true;

}

} // end while

player\_code = player\_input;

return player\_code;

} // end get\_player\_code()

// Function definition for generate\_code function

//Generate code

string generate\_code() {

string code = "";

for (int x = 0; x < CODE\_LENGTH; x++){

code.push\_back( randint(1, CODE\_SPAN + 1) + '0' );

}

return code;

} // end generate\_code()

//Function Definition for random number generator used by code randint()

//Random number generator

int randint(int max) {

return ( rand() % max );

}

//Random number generator

int randint(int min, int max) {

return ( randint( max - min ) + min );

}