


Connections (NYT) Clone



George Kong
CIS-25 Final Project



Overview

- Clone of NYT's popular "Connections" word game coded in C++
- Player is given 16 words and must group 4 words into 4 correct categories
- Different levels of difficulty for each category
- Total of 4 lives

Core features:

- File-Based Save System
- Functional Logic
- Object-Oriented Programming

Program Flow 1

- Starts with username input
- Choice between two loaded puzzles
- After user picks one, display 16 shuffled words

```
cout << "Enter your username: ";
getline(cin, username);

cout << "Hello "<<username<<"! Choose a puzzle to load : \n";
cout << "1. Puzzle 1\n";
cout << "2. Puzzle 2\n";
cout << "Enter choice (1 or 2): ";
cin >> choice;
cin.ignore(); // ignore newline character after integer input

if (choice == 1) {
    puzzleFile = "puzzle1.txt";
}
else if (choice == 2) {
    puzzleFile = "puzzle2.txt";
}
else {
    cout << "Invalid choice. Exiting.\n";
    return 0;
}
```

```
Enter your username: test_runner
Hello test_runner! Choose a puzzle to load:
1. Puzzle 1
2. Puzzle 2
Enter choice (1 or 2): 1
Welcome to the Connections Game!
```

```
Available Words:
Bred Last Bull Stand Hold Hawk Spouse Stay Born Lute Bear Education Doe Cache Dove Occupation
```

Program Flow 2

- User enters a 4-word category. If correct, the program will output:

```
Available Words:  
Bred Last Bull Stand Hold Hawk Spouse Stay Born Lute Bear Education Doe Cache Dove Occupation  
  
Enter 4 words separated by commas (or type 'exit' to quit): Hold, Last, Stand, Stay  
Correct! You found the group: PERSISTENCE
```

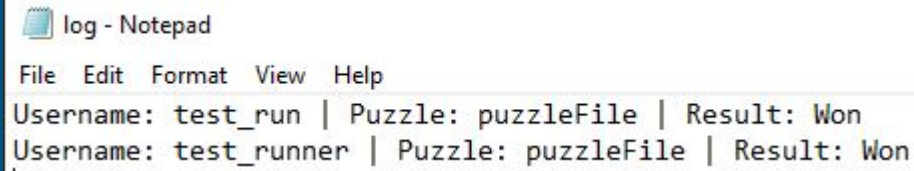
- However, if incorrect, the program will output and result in the player losing a life:

```
Available Words:  
Bred Bull Hawk Spouse Born Lute Bear Education Doe Cache Dove Occupation  
  
Enter 4 words separated by commas (or type 'exit' to quit): Bear, Bull, Hawk, Doe  
Incorrect group. Attempts left: 3
```

Program Flow 3

- Once all categories are found or all lives are lost, the game will end and the program will automatically update log.txt

```
// final result messages
if (solvedGroups.size() == 4) {
    cout << "Congratulations! You've solved all groups!\n";
    logResult(true, "puzzleFile");
}
else {
    cout << "Game Over! Better luck next time.\n";
    logResult(false, "puzzleFile");
}
```



log - Notepad

File Edit Format View Help

Username: test_run | Puzzle: puzzleFile | Result: Won
Username: test_runner | Puzzle: puzzleFile | Result: Won

Data Structures Used

```
std::map<std::string, std::vector<std::string>> groups;
```

- Stores group names and their words

```
std::vector<std::string> allWords;
```

- Holds all 16 words

```
std::set<std::string> solvedGroups;  
std::set<std::string> usedWords;
```

- Tracks solved groups and used words

```
std::string username;
```

- Stores user input, group names, and puzzle file names

Logic for checking correct groups

```
// check if the attempted group is correct
bool GameManager::isCorrectGroup(const vector<string>& attempt) {
    for (const auto& pair : groups) {
        const string& groupName = pair.first;
        const vector<string>& words = pair.second;
        set<string> attemptSet(attempt.begin(), attempt.end());
        set<string> correctSet(words.begin(), words.end());
        if (attemptSet == correctSet && solvedGroups.find(groupName) == solvedGroups.end()) {
            solvedGroups.insert(groupName);
            usedWords.insert(attempt.begin(), attempt.end());
            cout << "Correct! You found the group: " << groupName << "\n\n";
            return true;
        }
    }
    return false;
}
```

Logic for loading the .txt files and parsing groups

```
bool GameManager::loadPuzzle(const string& filename) {
    ifstream file(filename);
    if (!file) {
        cout << "Failed to open puzzle file.\n";
        return false;
    }

    string line;
    while (getline(file, line)) {
        if (line.rfind("GROUP:", 0) == 0) {
            string groupName = line.substr(6);
            getline(file, line);
            stringstream ss(line);
            string word;
            vector<string> words;

            // split comma-separated words
            while (getline(ss, word, ',')) {
                word.erase(remove_if(word.begin(), word.end(), ::isspace), word.end());
                words.push_back(word);
                allWords.push_back(word);
            }
            groups[groupName] = words;
        }
    }

    // shuffle words for randomness
    random_shuffle(allWords.begin(), allWords.end());
    return true;
}
```


Main Gameplay Loop

```
void GameManager::run(const string& puzzleFile) {
    cout << "Welcome to the Connections Game!\n";

    while (solvedGroups.size() < 4 && lives > 0) {
        printWords();

        cout << "Enter 4 words separated by commas (or type 'exit' to quit): ";
        string input, word;
        getline(cin, input);
        stringstream ss(input);
        vector<string> guess;

        // tokenize input into words
        while (getline(ss, word, ',')) {
            word.erase(remove_if(word.begin(), word.end(), ::isspace), word.end());
            if (!word.empty()) {
                guess.push_back(word);
            }
        }

        if (guess.size() != 4) {
            cout << "Please enter exactly 4 words.\n\n";
            continue;
        }

        // check for correct group
        if (!isCorrectGroup(guess)) {
            lives--;
            cout << "Incorrect group. Attempts left: " << lives << "\n\n";
        }
    }

    // final result messages
    if (solvedGroups.size() == 4) {
        cout << "Congratulations! You've solved all groups!\n";
        logResult(true, "puzzleFile");
    }
    else {
        cout << "Game Over! Better luck next time.\n";
        logResult(false, "puzzleFile");
    }
}
```

Features to add in the future

- Use AI to generate new word categories daily
- Replicate a similar GUI to the real game
- Leaderboard to track wins and loss records