C++ Challenge

Today you will implement together with your teammates a client program which can play Connect-Four against a server provided by us. Whenever you have got a question, just ask!

Exercise 1: First Steps (15 minutes)

Create a new project named “CppChallenge” in CLion. Copy the code template provided with this exercise sheet into your “CppChallenge.cpp”. Run the project. If no compiler error shows up, you are ready to start.

Exercise 2: Network Code (1 hour)

The provided code template is not complete. Read through the code and understand what is happening. You can ignore the first 60 lines.

a) Implement the Constructor in the *Connection* class.
b) Implement the Destructor in the *Connection* class.
c) Implement the *sendInt(<int>), sendString(<string>)* function in the *Connection* class.
d) Write a function that can print a description of a *GameInfo* object to the console window.
e) Choose a unique team-name, not longer than 15 characters.
f) Test your code by connecting to the server and sending your team-name. The server responds with sending a *GameInfo* object. You can receive it by the *Connection::receive* function. Print it to the console and let your program exit.

Exercise 3: Playing Connect-Four (30 minutes)

In this exercise you should implement the steps 1 to 4 of the protocol specified below. The goal is to play a single game against the server. After each game you played, your program should exit.

a) After receiving the first *GameInfo* object from the server and printing it to the console, read an integer from the console Window.
b) Send the integer to the server. This is the column (from 0 to 6) where you want to insert your next coin.
c) Again, the server responds by sending a *GameInfo* object. Continue at a) until the *Result* variable in the *GameInfo* object does not hold the value *GAME_RESULT::CONTINUE* anymore.
d) Output the winner, and check for the reason.
e) After the game is over let your program exit.

Remember: You only got a few (~10) seconds to answer to the server, otherwise the connection will timeout!

Exercise 4: AI (30 minutes)

Write an AI to play Connect-Four for you. Instead of reading an Integer from the console window, you should now write code playing Connect-Four by itself.

Hint: Start with a simple proof-of-work AI. Improve your AI after finishing exercise 5!
Your AI should be able to play with yellow and red coins and as player 1 and player 2.

Exercise 5: Implement the full protocol (30 minutes)

Implement the full protocol as specified below (step 5). After each game the server will either send you a new *GameInfo* object to let you play again or closes the connection. Play a new game whenever you can! You can detect a closed connection by testing the return value of *Connection::receive*. If the return value is *false*, the connection has been closed! The server will always play at least one game, and the server will never close a connection midgame.

Challenge!

When everybody finished coding, we will let you connect to the competitive server. (We will announce the port). This server schedules matches between all the teams, such that every team will play against every other team. After every game, the winner gains three points or, if it was a draw, both teams gain 1 point. If you loose connection during the challenge for some reason, wait at least 15 seconds and then reconnect.

Protocol Specification:

Ip-Address: 13.80.22.187
Port: 40596

1. Connect to the server via a TCP/IP socket.
2. Send a team-name not longer than 15 characters!
3. The Game server sends you a *GameInfo* object. You can get it by calling *Connect::receive*. Jump to step 5 if the *Result* variable is not *GAME_RESULT::CONTINUE*.
4. Send an integer indicating the column (from 0 to 6), where you want to set your next coin. Continue with step 3.
5. Try to receive another *GameInfo* object. If you succeed continue at step 4, otherwise let your program exit.