Kiki i Buba

OFF 2025

Kod zadania: **Kik**
Limit pamięci: **64 MiB**

Kiki i Buba są dobrymi przyjaciółmi i gdy czasem nudziło im się na lekcji, grali w zeszycie w kółko i krzyżyk. Jednak dosyć szybko zorientowali się, że gra zawsze kończy się remisem. Postanowili zmodyfikować zasady gry, by ta była ciekawsza. W nowej grze plansza nie jest ograniczona do kratki 3x3, ale można grać na całej (nieskończonej) kartce. Dodatkowo w nowej wersji trzeba połączyć aż 5 symboli z rzędu. Nowa gra okazała się niezłym wyzwaniem, dlatego aby poćwiczyć, Buba poprosił cię o napisanie bota, z którym będzie mógł się uczyć grać.

Komunikacja

Zaimplementuj program, który zagra z Bubą w kółko i krzyżyk. Do komunikacji skorzystaj z dostarczonej biblioteki: `#include "interface.h"`

Biblioteka udostępnia następujące funkcje:

- `void submitMove(int x, int y)` – Funkcja rysuje odpowiednio X lub O (w zależności od tego, którym graczem w danej grze jesteś) w kratce o koordynatach (x, y).
- `pair<char, vector<pair<pair<int, int>, char>>>` readInitialState() – Zawołaj tę funkcję raz, na początku wykonywania programu. Funkcja zwróci symbol gracza którym jesteś i listę dotychczasowych ruchów.
- `std::pair<int, int> readOpponentMove()` – Funkcja zwraca ruch przeciwnika. Ta funkcja musi zostać wywołana dokładnie raz po każdym ruchu.

Twój program nie może otwierać żadnych plików ani używać standardowego wejścia i wyjścia. Może korzystać ze standardowego wyjścia diagnostycznego (`stderr`), jednak pamiętaj, że zużywa to cenny czas.
Rozwiązanie będzie kompilowane wraz z biblioteką następującym poleceniem:

```
g++ -O3 -static -std=c++20 kib.cpp -o kib
```

Uwaga: Podane na górze ograniczenie pamięci dotyczy tylko Twojego rozwiązania, a zatem nie wlicza pamięci wykorzystywanej przez bibliotekę.

Przykłady

Poniżej możesz zobaczyć przykładową komunikację między Twoim botem a przeciwnikiem (Bubą).

Ruch	Wywołania funkcji	Opis
0	`readInitialState(Game) → {'X',{{0,0},'X'},{{1,0},'O'}}`	Bot wczytuje początkowy stan gry (X na pozycji $(0,0)$ i O na pozycji $(1,0)$)
1	`submitMove(0, 1)`	Bot stawia X na polu $(0,1)$.
2	`readOpponentMove() → (1, 1)`	Bot otrzymuje kolejny ruch Buby.
3	`submitMove(0, 2)`	Bot stawia X na polu $(0,2)$.
4	`readOpponentMove() → (1, 2)`	Buba gra dalej w tej samej kolumnie.
5	`submitMove(0, 3)`	Bot buduje linię w kolumnie 0.
6	`readOpponentMove() → (1, 3)`	Ruch przeciwnika.
7	`submitMove(0, 4)`	Bot stawia X – piąty symbol w pionie i wygrywa.

Po tej sekwencji Twój bot ma następujące ruchy: (0, 0), (0, 1), (0, 2), (0, 3), (0, 4) – tworzy więc pionową piątkę i wygrywa grę. Buba próbował przerywać linię, ale robił to równolegle: (1, 0), (1, 1), (1, 2), (1, 3) – co było nieskuteczne.

Ocenianie

Twój bot pod koniec obozu weźmie udział w turnieju, w którym rozegra parę gier, zaczynając od wylosowanej pozycji. Punktacja będzie odwrotnie proporcjonalna do miejsca zajętego w turnieju. Dodatkowo w trakcie trwania konkursu przewidujemy turnieje testowe, na których będzie można zmierzyć się z innymi zawodnikami. Na wykonanie jednego ruchu będziesz miał 0.25s. Jeśli masz pytania dotyczące działania biblioteki lub zasad gry, możesz zgłaszać je przez sio2.
Pomimo, że gra jest w teorii nieskończona to Kiki i Buba umówili się, że koordynaty i-tego ruchu (indeksując od 1) muszą być mniejsze lub równe $i * 100$. Dzięki takiej zasadzie nie muszą się martwić ruchami przeciwnika które są bardzo daleko od środka.

Pliki

W zakładce pliki możesz znaleźć:

- `interface.h` – interface którym możesz wysyłać i czytać ruchy.
- `default_bot.cpp` – przykładowa implementacja bota.
- `engine.h` – program zarządzający grą.
- `agent.py` – program do uruchamiania turniejów w podanym folderze, oraz umożliwiający zagranie z wybranym botem.

PS: Grę możesz znaleźć pod linkiem https://tic.netlify.app/