Zasady i zestaw zadań

Gliwice, 22-23 kwietnia 2014

Spis treści

1. Wstępniak

To już szósty finał maratonu programistycznego Deadline24 organizowany przez firmę Future Processing, tym razem także pod zaszczytnym patronatem rektorów, dziekanów uniwersytetów i uczelni technicznych. Po raz pierwszy natomiast znajdujemy się w Markowni Kopalni Ludwik, która wspiera to wydarzenie. Z pewnością unikalna sceneria tego miejsca będzie niezapomnianym przeżyciem dla wszystkich drużyn walczących o każdy punkt do ostatniej sekundy konkursu.

Podobnie jak w ubiegłych latach oddajemy w Wasze ręce trzy zadania, tradycyjnie osadzone w świecie intrygujących stworzeń – żukoskoczków. Ich ambicja w odkrywaniu tajemnic świata zaprowadzi ich na wodną planetę, gdzie będą poszukiwać cennego amuletu (zadanie **Morskie Wojny**), a także postawi ich wobec konieczności walki z mitycznymi potworami (zadanie **Eksperyment**). W wolnej chwili żukoskoczki będą oddawać się też przyjemności rozgrywania karcianych partyjek (zadanie **Gra w Bezżuka**). Będziecie mogli im w tym wszystkim pomóc. Z jakim efektem? Zobaczymy już za moment.

Mamy nadzieję, że rozgrywka będzie emocjonująca, a system konkursowy i wszyscy uczestnicy wytrzymają napięcie przez całe 24 godziny. Niech wygra najlepszy!

Zespół Deadline24

2. Komunikacja z serwerem

Uzyskiwanie aktualnych informacji o wirtualnym świecie oraz wydawanie rozkazów jest możliwe za pomocą protokołu TCP/IP. Drużyna łączy się jako klient do odpowiedniego serwera konkursowego. Adres IP oraz port, z którym należy się połączyć są podane w sekcjach *Serwery* specyfikacji poszczególnych zadań. Można nawiązać wiele połączeń jednocześnie, jednak sumaryczny transfer przypadający na każdy komputer jest ograniczony. Maksymalna liczba połączeń i maksymalny transfer podane są w *Ustaleniach Technicznych*. Komunikacja odbywa się w trybie tekstowym. Bezpośrednio po połączeniu należy się zalogować, następnie sesja przechodzi w tryb poleceń.

2.1. Logowanie

Bezpośrednio po nawiązaniu połączenia serwer wysyła prośbę o login zakończoną znakiem końca linii: `LOGIN`. Należy wysłać swój login, a następnie znak końca linii. Następnie serwer zapyta o hasło (`PASS`), na co należy analogicznie odpowiedzieć hasłem. Jeśli autoryzacja przebiegła pomyślnie, serwer odpowie ciągiem znaków: `OK` i przejdzie w stan oczekiwania na komendy. W przeciwnym wypadku dostaniemy odpowiedź `FAILED 1 bad login or password`, po czym nastąpi zamknięcie połączenia.

Poniżej znajduje się przykładowy zapis komunikacji w czasie logowania.

klient → serwer	serwer → klient
	`LOGIN`
`login1`	
	`PASS`
`secret`	
	`OK`

2.2. Komendy

Każde polecenie składa się z nazwy komendy, argumentów (liczba zależna od polecenia) oraz znaku końca linii. Parametry powinny być oddzielone co najmniej jednym białym znakiem.

Na każdą komendę serwer odpowiada jednym z poniższych ciągów:

- '`OK`' — w przypadku zaakceptowania komendy,
- '`FAILED` e msg' — w przypadku błędu; gdzie e to kod błędu, a msg — komunikat błędu.

Następnie, zależnie od komendy, serwer może opcjonalnie wysłać lub odebrać dodatkowe dane. Jeśli dodatkowe dane są wysyłane od klienta do serwera, to po odebraniu tych danych serwer ponownie odpowie w sposób opisany powyżej. Przykładowe zapisy komunikacji z serwerem oraz zestawienia możliwych błędów dla poszczególnych zadań znajdują się w opisie każdego z nich.

Ograniczenie liczby komend Na każdym serwerze każdego z zadań obowiązuje limit na maksymalną liczbę komend wydawanych w czasie jednej tury. Osiągnięcie limitu zostanie zasygnalizowane odpowiednim błędem: `FAILED 6 commands limit reached, forced waiting activated`. Po wystąpieniu błędu, serwer prześle dodatkowy komunikat: `WAITING x` — gdzie x ($x \in \mathbb{R}$) oznacza liczbę sekund do zakończenia oczekiwania, tj. do końca bieżącej tury.

2.3. Konwencje

Jeśli nie jest zaznaczone inaczej, to przyjmujemy, że:

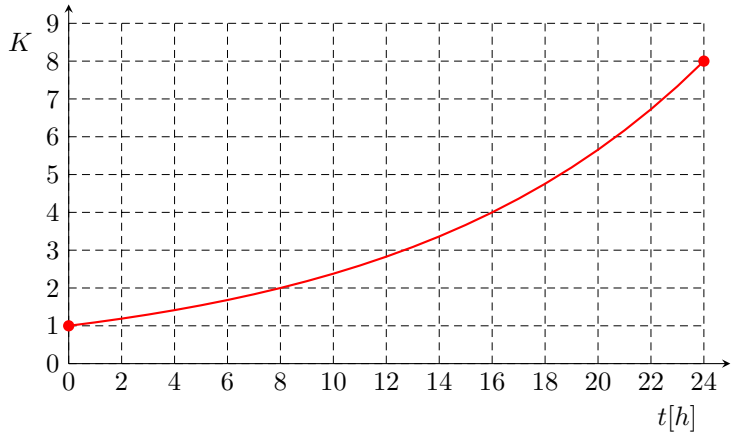
- Każda linia zakończona jest pojedynczym znakiem o kodzie ASCII 10 ('\n'). Znak powrotu karetki ('\r'; kod 13) towarzyszący mu na niektórych systemach operacyjnych będzie traktowany jako biały znak.
- W przypadku danych przesyłanych przez serwer, liczby oraz wyrazy oddzielone są pojedynczą spacją.

- W przypadku danych przesyłanych od klienta do serwera (np. parametry komendy) dozwolona jest dowolna (niezerowa) ilość białych znaków pomiędzy danymi, a także na końcu oraz na początku linii.
- Za białe znaki uznajemy: spację, powrót karetki ('\r') oraz tabulator ('\t').

3. Punktacja

3.1. Tempo rozgrywki

Współczynnik K Dla każdego serwera wynik wyznaczony dla drużyny jest dodatkowo mnożony przez K – współczynnik tempa rozgrywki. Wartość tego współczynnika wzrasta wykładniczo przez cały czas trwania konkursu od wartości 1 do wartości 8. Oznacza to, że w ostatnich godzinach konkursu można zdobyć kilkakrotnie więcej punktów niż na początku, a co za tym idzie warto ciągle udoskonalać swoje rozwiązania, by nie dać się wyprzedzić innym drużynom. Bieżąca wartość K jest dostępna w odpowiednim poleceniu każdego z zadań.



Rysunek 1: Zmiana współczynnika tempa rozgrywki K w czasie.

3.2. Punkty rankingowe

Punkty Na każdym serwerze drużyny zdobywają punkty w sposób zdefiniowany w specyfikacji zadania rozgrywanego na danym serwerze. Punkty zdobyte na poszczególnych serwerach są przeliczane na **punkty rankingowe**, które definiujemy jako iloczyn liczby 100 oraz stosunku wyniku drużyny (na danym serwerze) do średniej arytmetycznej trzech najlepszych wyników (na danym serwerze). Punkty przyznawane drużynie za konkretne zadanie to średnia punktów rankingowych z wszystkich serwerów danego zadania.

Ranking Ranking zawodów jest oparty o sumę punktów rankingowych wszystkich zadań.

4. Sytuacje awaryjne

W razie wystąpienia sytuacji, w której nie wszystkie drużyny będą w stanie brać udział w konkursie na ustalonych zasadach (np. brak prądu, awaria sieci lokalnej lub jej części, problemy z systemem konkursowym, itp.) oraz wina nie leży po stronie tych drużyn ani ich sprzętu, organizatorzy wstrzymają działanie systemu konkursowego, a o jego ponownym uruchomieniu uczestnicy zostaną poinformowani w lokalnym serwisie WWW konkursu. W tym czasie punkty nie będą naliczane. W takiej sytuacji mogą zostać zerwane wszystkie połączenia z serwerem konkursowym.

5. Gra w Bezżuka

5.1. Wprowadzenie

Oddziały sił zbrojnych żukoskoczków stacjonujące na planecie C–358 już od dość długiego czasu nie miały okazji sprawdzić się w boju. Przedłużający się pokój z mieszkańcami okolicznych planet ma, przynajmniej według Generała Żukka, negatywny wpływ na morale żołnierzy. By zapobiec całkowitemu odzwyczajeniu się armii od warunków bojowych, Generał nakazał dowództwu tych oddziałów organizację ćwiczeń.

Tak oto żukoskoczki z planety C–358 znalazły się na poligonie w Żuwsku Podlaskim, spędzając czas na nieustannym doskonaleniu swoich umiejętności bojowych i taktycznych. Nieliczne przerwy między manewrami wykorzystują na integrację z towarzyszami broni przy wodzie, konserwach i partyjce Bezżuka. Dołączycie się do gry?

5.2. Opis kart

Bezżuk to gra karciana tak stara i powszechnie znana, że w całym Universum nikt już nie pamięta gdzie i kiedy powstała. Żukoskoczkowe karty, co przeciwnicy teorii konwergencji mogą uznać za dość zaskakujące, są dokładnie takie same jak nasze. Talia składa się więc z 4 kolorów po 13 kart w każdym z nich. Tymi kolorami są pik, kier, karo i trefl (symbolizowane przez litery, odpowiednio: S, H, D, C). Rodzaje kart w jednym kolorze to, w kolejności od najniższej do najwyższej: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A. Opis pojedynczej karty składa się z rodzaju karty i jej koloru, podanych **bez** spacji. Przykładowo dwójka kier to 2H, dama pik to QS, a dziesiątka trefl to 10C. Zbiór wszystkich możliwych 52 opisów kart będziemy dalej oznaczali przez *Cards*.

5.3. Zasady gry

W grze uczestniczy trzech graczy ponumerowanych liczbami 1, 2 i 3. Potrzebna jest pojedyncza talia złożona z 52 kart, czyli bez jokerów. Gra składa się z 6 rozdań. W czasie każdego z nich dokładnie jeden spośród graczy jest *obierającym*. Obierającymi są, w kolejnych rozdaniach, gracze 1, 2, 3, 1, 2, 3. Podczas jednego rozdania obierający otrzymuje 20 kart, pozostali gracze po 16. Obierający z pierwszych 8 kart wybiera rodzaj gry (*Bez atu* lub *Bez lew*), który będzie grany w tym rozdaniu. Podczas gry każdy gracz jest 2 razy obierającym i musi obrać oba rodzaje gier (jeśli więc za pierwszym razem wybrał *Bez atu*, za drugim razem będzie musiał „wybrać" *Bez lew* i na odwrót). Po dokonaniu wyboru przez obierającego, ogląda on wszystkie swoje 20 kart i wyrzuca 4 z nich (wyrzucone karty nie są znane pozostałym graczom). Każdy ma więc po 16 kart i zaczyna się rozgrywka pojedynczego rozdania.

Rozgrywka pojedynczego rozdania składa się z 3×16 tur. W czasie jednej tury każdy z graczy kolejno, w kolejności zgodnej z numeracją (gracz nr 2 zaraz po graczu nr 1, nr 3 po nr 2, nr 1 po nr 3), wyrzuca na środek dokładnie jedną ze swoich kart. Gracz, który wyrzuci kartę największą w ważnym kolorze (*ważny* kolor to kolor tej karty, którą wyrzucił gracz rozpoczynający daną turę), zabiera te trzy karty jako pojedynczą *lewę*. Pierwszą turę zaczyna obierający, a każdą kolejną ten z graczy, który wziął ostatnią lewę.

5.3.1. Zasady ogólne

Zasada koloru: jeżeli gracz, którego teraz kolej na rzucenie karty, posiada co najmniej jedną kartę w ważnym kolorze, to musi wyrzucić jedną z nich. W przeciwnym razie gracz ma prawo wyrzucić dowolną ze swoich kart (następstwem czego aktualna lewa na pewno nie padnie jego łupem).

Zasada przebicia: jeżeli gracz, którego teraz kolej na rzucenie karty, posiada co najmniej jedną kartę w ważnym kolorze większą niż aktualnie największa na stole karta będąca w ważnym kolorze, to musi wyrzucić jedną z nich.

5.3.2. Zasady gry *Bez atu*

Obowiązuje zarówno zasada koloru, jak i zasada przebicia. Każdy z graczy otrzymuje na koniec rozgrywki tym więcej punktów, im więcej zbierze lew.

5.3.3. Zasady gry *Bez lew*

Obowiązuje zasada koloru, ale nie obowiązuje zasada przebicia. Każdy z graczy otrzymuje na koniec rozgrywki tym więcej punktów, im mniej zbierze lew.

5.4. Mechanizm gry

Co pewien czas każdy z serwerów przydziela uczestników do stołów oraz w sposób losowy przydziela im numery od 1 do 3. Następnie odbywa się 6 rozgrywek, tak jak opisano wyżej. Pierwsza tura każdej rozgrywki trwa 1 sekundę i podczas niej obierający wybiera rodzaj gry. Druga tura trwa 2 sekundy, podczas niej obierający wyrzuca 4 ze swoich kart. Każda z kolejnych 48 tur trwa 1 sekundę. W czasie jednej takiej tury gracz, którego przyszła kolej, rzuca na stół dokładnie jedną ze swoich kart.

5.5. Punktacja

Gracz dostaje 2 punkty za każdą wziętą lewę podczas gry w *Bez atu* oraz 1 punkt za każdą lewę, której nie wziął podczas gry w *Bez lew*. Aby dostać punkty za daną lewę, gracz musi samodzielnie wyrzucić w swojej kolejce prawidłową kartę – w przeciwnym razie za daną lewę dostanie 0 punktów.

Dodatkowo za zdobycie największej liczby lew w grze *Bez atu* gracz otrzymuje 10 punktów, a za zdobycie najmniejszej liczby lew w grze *Bez lew* 16 punktów. W przypadku remisu żaden z graczy nie dostaje tej premii. Premia nie jest przydzielana również wówczas, gdy dany gracz choć dwa razy nie wyrzucił na czas prawidłowej karty.

5.6. Komendy

Ogólne założenia protokołu komunikacji (nawiązywanie połączenia, logowanie, wysyłanie komend, format odpowiedzi) opisane są w rozdziale *Komunikacja z serwerem*. Poniżej znajduje się lista komend dostępnych dla zadania Gra w Bezżuka.

GET_STATE Zwraca identyfikator aktualnego stanu gry.

Parametry: brak

Dane (od serwera):

Serwer zwraca pojedyncze słowo:

- w ($w \in$ {BREAK, GAME_SELECTION, KITTY_SELECTION, TRICK}) — identyfikator obecnego stanu gry

Różne słowa oznaczają kolejno:

- BREAK — trwa przerwa w grach (tj. pomiędzy 6 rozgrywkami)
- GAME_SELECTION — obierający jest w trakcie wyboru gry
- KITTY_SELECTION — obierający jest w trakcie wyrzucania 4 ze swoich kart; w tej turze każdy z graczy może już zobaczyć wszystkie swoje karty
- TRICK — trwa rozgrywka pojedynczego rozdania

GET_MY_ID Zwraca numer przydzielony Wam przez serwer.

Parametry: brak

Dane (od serwera):

Serwer zwraca pojedynczą liczbę:

- ID ($ID \in \{1, 2, 3\}$) — numer przydzielony przez serwer obowiązujący w ciągu gry

GET_MY_CARDS Zwraca trzymane aktualnie w ręku karty.

Parametry: brak

Dane (od serwera):

Serwer zwraca dwie linie, które składają się z:

- L ($L \in \mathbb{N}$, $0 \leqslant L \leqslant 20$) — liczba trzymanych kart
- $c_1, c_2, ..., c_L$ ($c_1, c_2, ..., c_L \in Cards$) — wszystkie trzymane karty w przypadkowej kolejności

SELECT_GAME Tym poleceniem obierający wybiera rodzaj aktualnej rozgrywki.

Parametry:

Należy przekazać dokładnie jedną literę:

- l ($l \in$ {A, L}) — litera symbolizująca rodzaj rozgrywki; litera A oznacza grę w *Bez atu*, a L grę w *Bez lew*

W przypadku niepodjęcia na czas decyzji o rodzaju rozgrywki automatycznie ogłaszana jest gra w *Bez atu*.

SELECT_KITTY Tym poleceniem obierający wybiera karty do wyrzucenia.

Parametry:

Należy przekazać w jednej linii oddzielone spacją 4 opisy kart:

- c_1, c_2, c_3, c_4 ($c_1, c_2, c_3, c_4 \in Cards$) — karty przeznaczone do wyrzucenia; nie biorą one udziału w dalszej rozgrywce

W przypadku niepodjęcia na czas decyzji o kartach do wyrzucenia wyrzucane są automatycznie 4 losowe karty.

CURRENT_GAME Zwraca rodzaj aktualnej rozgrywki.

Parametry: brak

Dane (od serwera):

Serwer zwraca dokładnie jedną literę:

- l ($l \in \{\text{A, L}\}$) — litera symbolizująca rodzaj aktualnej rozgrywki; litera A oznacza grę w *Bez atu*, a L grę w *Bez lew*

SELECTOR_ID Zwraca numer obierającego w tej rozgrywce.

Parametry: brak

Dane (od serwera):

Serwer zwraca pojedynczą liczbę:

- ID ($ID \in \{1, 2, 3\}$) — numer obierającego w tej rozgrywce

TURN_ID Zwraca numer gracza, który w tej turze powinien wyrzucić kartę.

Parametry: brak

Dane (od serwera):

Serwer zwraca pojedynczą liczbę:

- ID ($ID \in \{1, 2, 3\}$) — numer gracza wyrzucającego kartę w tej turze

ON_TABLE Zwraca karty obecne w tym momencie na stole.

Parametry: brak

Dane (od serwera):

Serwer zwraca dwie linie:

- L ($L \in \mathbb{N}$, $0 \leqslant L \leqslant 2$) — liczba kart leżących na stole
- $c_1, c_2, ..., c_L$ ($c_1, c_2, ..., c_L \in Cards$) — karty leżące na stole, podane w kolejności ich rzucania

PLAY Tym poleceniem gracz rzuca na stół kartę.

Parametry:

Należy przekazać opis swojej jednej karty:

- c ($c \in Cards$) — karta przeznaczone do zagrania; wyrzucenie tej karty musi być zgodne z zasadami danej rozgrywki

W przypadku niepodjęcia na czas decyzji o rzuceniu karty rzucana jest najwyższa posiadana karta w ważnym kolorze, a gdy takiej nie ma, wyrzucana jest losowa karta.

LAST_TRICK Zwraca informacje o ostatniej lewej.

Parametry: brak

Dane (od serwera):

Serwer zwraca dwie linie:

- c_1, c_2, c_3 ($c_1, c_2, c_3 \in Cards$) — karty rzucone na stół podczas rozgrywania ostatnio zakończonej lewej (w kolejności rzucania ich na stół)
- ID_1, ID_{win} ($ID_1, ID_{win} \in \{1, 2, 3\}$) — numer gracza który wyrzucił kartę jako pierwszy oraz numer tego gracza, który wziął tę lewę

WAIT Czeka do momentu rozpoczęcia kolejnej tury.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią postaci:

- WAITING S

gdzie S ($S \in \mathbb{R}, S \geqslant 0$) oznacza liczbę sekund do zakończenia oczekiwania. Po upływie tego czasu wysyła dodatkowo linię:

- OK

5.7. Błędy

Zgodnie z opisem w rozdziale *Komunikacja z serwerem*, w wypadku podania nieprawidłowego polecenia, serwer odpowiada komunikatem:

- '`FAILED` *e msg*',

gdzie *e* to kod błędu, a *msg* — komunikat błędu. W tabeli znajduje się zestawienie błędów, które mogą wystąpić przy wydawaniu poleceń w zadaniu Gra w Bezżuka.

kod błędu	komunikat błędu
1	bad login or password
2	unknown command
3	bad format
4	too many arguments
5	internal error, sorry...
6	commands limit reached, forced waiting activated
101	no game currently being played
102	not time to select the game
103	you are not the game selector
104	not time to select the kitty
105	no such card in your hand
106	game not selected yet
107	game not begun yet
108	not your turn to play
109	rules violation
110	no trick played in the current game
111	game already selected
112	kitty already selected
113	you have already played in this turn
114	you are not member of any game

5.8. Serwery

Rozgrywki będą odbywać się na kilku serwerach, różniących się od siebie wartością parametru t, oznaczającego po ilu godzinach od rozpoczęcia konkursu na danym serwerze rozpocznie się gra.

nazwa	adres:port	parametry
Cards1	universum.dl24:20000	t = 1
Cards2	universum.dl24:20001	t = 6
Cards3	universum.dl24:20002	t = 12

5.9. Przykład

Poniżej znajduje się przykładowy zapis komunikacji z serwerem.

klient → serwer	serwer → klient
	LOGIN
login1	
	PASS
secret	
	OK
GET_STATE	
	OK BREAK
WAIT	
	OK WAITING 0.28000 OK
GET_STATE	
	OK GAME_SELECTION
GET_MY_ID	
	OK 1
SELECTOR_ID	
	OK 1
GET_MY_CARDS	
	OK 8 7D JC JH 4C 2H 2S 3S 10H
SELECT_KITTY JC JH 10H 7D	
	FAILED 104 not time to select the kitty
SELECT_GAME L	
	OK
SELECT_KITTY JC JH 10H 7D	
	FAILED 104 not time to select the kitty
WAIT	
	OK WAITING 0.15000 OK
GET_MY_CARDS	
	OK 20 AS AC 2D KD 2S 3S QH 5H JC 10H 7D 9C 4D 2H JH 4C 8S 3C AH 9D
SELECT_KITTY AC AH AS KD	
	OK
CURRENT_GAME	
	OK L

klient → serwer	serwer → klient
`WAIT`	
	`OK`
	`WAITING 0.42000`
	`OK`
`GET_STATE`	
	`OK`
	`TRICK`
`TURN_ID`	
	`OK`
	`1`
`ON_TABLE`	
	`OK`
	`0`
`PLAY 3C`	
	`OK`
`WAIT`	
	`OK`
	`WAITING 0.58000`
	`OK`
`ON_TABLE`	
	`OK`
	`1`
	`3C`
`WAIT`	
	`OK`
	`WAITING 0.95000`
	`OK`
`ON_TABLE`	
	`OK`
	`3C 4C`
`WAIT`	
	`OK`
	`WAITING 0.75000`
	`OK`
`LAST_TRICK`	
	`OK`
	`3C 4C 2C`
	`1 2`
`PLAY QH`	
	`FAILED 108 not your turn to play`
`TURN_ID`	
	`OK`
	`2`
`WAIT`	
	`OK`
	`WAITING 0.88000`
	`OK`
`TURN_ID`	
	`OK`
	`3`

klient → serwer	serwer → klient
`WAIT`	
	`OK`
	`WAITING 0.92000`
	`OK`
`ON_TABLE`	
	`OK`
	`2`
	`4S KS`
`PLAY QH`	
	`FAILED 109 rules violation`
`PLAY 8S`	
	`OK`
`WAIT`	
	`OK`
	`WAITING 0.68000`
	`OK`
`LAST_TRICK`	
	`OK`
	`4S KS 8S`
	`2 3`

6. Morskie Wojny

6.1. Wprowadzenie

Ostatnie odkrycia archeologiczne przyczyniły się do wzrostu zainteresowania jedną spośród wodnych planet. Według pewnego starego zwoju o sprawdzonej autentyczności, na drobnych wysepkach tej planety ukryto części pradawnego amuletu, który nadaje niezwykłą moc swojemu posiadaczowi. Pomimo niejasnych wskazówek dotyczących możliwości wykorzystania amuletu, wśród żukoskoczków szybko pojawiły się grupy łowców skarbów.

Niezależne drużyny śmiałków wyruszyły w poszukiwaniu cennych artefaktów, by na własnej skórze przekonać się o ich sile. Dowodzisz jedną z takich grup i chcesz jak najlepiej wykorzystać swoją flotę zebraną na planecie, by nie dać się uprzedzić innym. Trzeba się jednak spieszyć także z innego powodu. Zgodnie z treścią zwoju, artefakty składające się na amulet stracą swoje właściwości przed najbliższą pełnią tamtejszego księżyca...

6.2. Model rozgrywki

Rozgrywka odbywa się w turach równej długości. W trakcie trwania każdej tury drużyny komunikują się z serwerem i wydają polecenia swoim jednostkom.

Zakończenie rozgrywki Od początku rozgrywki znany jest czas po jakim artefakty tracą swoją niezwykłą moc. Późniejsza rywalizacja traci sens. Rozgrywka może się jednak zakończyć wcześniej, jeśli jedna z drużyn wejdzie w posiadanie określonej liczby artefaktów i zdoła je połączyć. Po tym zdarzeniu do końca rozgrywki zostanie nie więcej niż 15 tur.

6.3. Świat

Rozgrywka toczy się na powierzchni pewnego obszaru, na którym wyróżnić możemy ląd oraz wodę. Obszar jest reprezentowany przez regularną siatkę składającą się z $N \times N$ kwadratowych elementów, które nazywamy **polami**. Pola rozmieszczone są w N rzędach po N pól. Każde pole jest identyfikowane przez parę liczb (P_x, P_y), które przyjmują wartości całkowite z przedziału od 1 do N. Każde z pól może być wyłącznie albo wodą, albo wyspą (lądem).

6.3.1. Wyspy

Z akwenu na planecie wynurzają się nieliczne wyspy. Na nich z kolei mogą znajdować się obiekty ważne pod względem strategicznym: twierdze oraz porty. Każde wpłynięcie statkiem na suchy ląd kończy się jego nieodwracalnym uszkodzeniem – drużyna traci tę jednostkę pływającą.

6.3.2. Twierdze

Na początku gry w każdej twierdzy możne znajdować się pojedynczy artefakt. Pozyskanie artefaktu jest możliwe tylko po zdobyciu twierdzy. Każda twierdza ma określoną – bieżącą oraz maksymalną (100) – obronność. Statki mogą zaatakować twierdzę, co powoduje spadek jej bieżącej obronności. Jeśli w danej rundzie twierdza nie została zaatakowana przez żadną z drużyn, jej obronność rośnie o 1 aż do osiągnięcia maksymalnej wartości (100). Zderzenie statku z twierdzą może zmniejszyć jej obronność, ale dla statku zawsze kończy się tak samo jak wpłynięcie na ląd – zniszczeniem jednostki.

Zdobywanie twierdzy Na początku rozgrywki żadna z twierdz nie posiada właściciela. Każda twierdza może jednak zostać zdobyta – przypisana na własność jednej z drużyn. Zdobycie twierdzy polega na obniżeniu jej obronności do zera. Przez cały okres atakowania konkretnego budynku gromadzona jest jego historia ataków (wyłącznie z ostrzałów), która decyduje o późniejszym przyznaniu prawa własności. Twierdzę zdobywa ta drużyna, która w momencie swojego ataku i po spadku obronności twierdzy do 0 zaatakowała twierdzę największą liczbę razy. Jeśli liczba ataków jest równa dla więcej niż jednej drużyny, zdobywanie twierdzy trwa aż do uzyskania przewagi jednej z tych drużyn.

Twierdza po zdobyciu i zakończeniu ataków jest automatycznie odbudowywana (jej obronność rośnie o 1 co turę). Jeśli ataki na twierdzę ustaną, a jej obronność powróci do stanu maksymalnego, historia ataków zostaje wyczyszczona.

Przenoszenie i łączenie artefaktów znajdujących się na polu z twierdzą jest możliwe tylko dla drużyny, która jest aktualnie właścicielem tej budowli. Możliwość umieszczania artefaktu w twierdzy nie zależy zaś od jej właściciela.

6.4. Flota

Każda drużyna rozporządza pewną liczbą jednostek pływających różnych typów. Typy statków różnią się od siebie własnościami: wytrzymałością, szybkostrzelnością, zasięgiem i manewrowością. Statek zostaje bezpowrotnie zniszczony, kiedy jego bieżąca wytrzymałość spadnie do zera. Każdy statek zajmuje jedno pole obszaru i nie może go dzielić z innym statkiem.

Ewentualne zderzenie jednostek na polu powoduje odjęcie od jednostki z największą aktualnie wytrzymałością sumy wytrzymałości pozostałych jednostek. Jeśli wynik jest dodatni – przetrwa tylko ta jednostka. W przeciwnym wypadku wszystkie statki na tym polu zostają zniszczone.

Dodatkowo cała flota może być wyposażona w systemy antykolizyjne. Nie można wtedy przemieścić się na pole, na którym znajduje się (lub znajdzie się w następnej turze) inna jednostka lub wyspa. W przypadku wykrycia potencjalnej kolizji, konkretne pojedyncze polecenie zmiany pozycji zostanie usunięte z kolejki poleceń statku bez jego wykonania.

Typy statków Drużyny dysponują trzema typami statków. Wszystkie statki mają to samo pole widzenia – 4 pola od swojego aktualnego położenia (widoczny jest wycinek mapy o rozmiarze 9×9 – statek znajduje się pośrodku). Różnice między statkami różnych typów znajdują się w poniższej tabeli.

Typ	Wytrzymałość	Bezwładność	Szybkość strzelania	Zasięg
Krążownik	15	6	2 / 1 turę	3 – 7
Niszczyciel	4	2	4 / 1 turę	1 – 3
Zwiadowca	3	1	1 / 1 turę	1 – 2

Bezwładność oznacza, że wydanie polecenia zmiany pozycji dla danego statku będzie miało efekt dopiero po upływie wskazanej bezwładnością liczbie tur. Zasięg i szybkość strzelania odnoszą się do możliwości ofensywnych jednostki – kiedy może ona zaatakować wroga w zależności od jego odległości oraz ile razy może to nastąpić w czasie jednej tury.

Porty Każda drużyna posiada na początku rozgrywki swój punkt bazowy – port. W jego sąsiedztwie (8 pól) każdy statek tej drużyny może co turę odzyskać 1 punkt wytrzymałości, także kiedy jednostka jest w ruchu. Port ma określoną obronność (40) i może zostać bezpowrotnie zniszczony. W żaden sposób nie można zwiększyć obronności portu, ani przejąć go na własność od innej drużyny. Wpłynięcie statkiem w pole, na którym znajduje się port, powoduje zniszczenie statku oraz odjęcie od obronności portu wytrzymałości statku (czyli także ewentualne zniszczenie portu).

Zdobywanie nowych statków Za celne trafienia w statki należące do floty przeciwnika drużyna otrzymuje materiały na budowę kolejnych statków. Po każdych 15 trafieniach istnieje możliwość zwodowania kolejnej jednostki. Budowa statków odbywa się w porcie, toteż ich wodowanie następuje w jego pobliżu. Jeśli port zostanie zniszczony, uniemożliwia to tworzenie nowych jednostek.

6.5. Artefakty

Obiektem pożądania Waszej grupy śmiałków są artefakty występujące w dwóch rodzajach: *GETATABLE_ARTIFACT* oraz *SECURED_ARTIFACT*. Wszystkie operacje na artefaktach typu *GETATABLE_ARTIFACT* można wykonywać od razu, bez dodatkowego oczekiwania. Drugi typ artefaktu jest chroniony przed niepowołanym użyciem i w związku z tym, aby taki przedmiot zabrać lub połączyć z innym, należy podać specjalny, **jednorazowy** klucz.

Wasza drużyna składa się, rzecz jasna, z poszukiwaczy skarbów pierwszej kategorii, toteż jesteście w stanie poradzić sobie ze złamaniem takiego zabezpieczenia. Zajmuje to jednak sporo czasu (50 tur). Proces łamania zabezpieczeń nie może zostać przerwany. Jeśli w jego trakcie statek się poruszy lub zostanie zniszczony, proces należy rozpocząć na nowo.

Artefakty znajdujące się w jakiejkolwiek twierdzy widoczne są dla drużyny tylko jeśli w ich bezpośrednim sąsiedztwie (8 pól) będzie statek należący do tej drużyny. Poza twierdzą każdy artefakt widoczny jest w całym polu widzenia jednostek.

6.5.1. Przenoszenie i łączenie artefaktów

Jeden statek może przenosić w danej chwili tylko jeden artefakt. Jeśli podczas przenoszenia statek zostanie zniszczony, artefakt będzie unosił się na wodzie (jest niezatapialny) w polu w jakim znajdowała się jednostka pływająca przed zniszczeniem. Jeśli artefakt zostanie umieszczony w twierdzy, która stanowi własność innej drużyny, wtedy on sam też zmienia właściciela.

Tworzenie amuletu mocy (łączenie artefaktów) jest możliwe wyłącznie w twierdzy, po zebraniu w niej odpowiedniej liczby cennych obiektów. W każdym łączeniu muszą brać udział oba rodzaje artefaktów. Raz połączone artefakty na stałe podnoszą dorobek drużyny, a co za tym idzie nie biorą udziału w dalszej rozgrywce – nie można ich ponownie przenieść, połączyć z innymi lub przejąć na własność.

6.6. Klucze do artefaktów

Operacje łączenia i przenoszenia artefaktów typu *SECURED_ARTIFACT* wymagają posiadania klucza, by taka operacja wykonała się natychmiast. Zdobycie klucza jest możliwe tylko po wykazaniu się odpowiednią wiedzą. Podczas prac poszukiwawczych na planecie pojawiają się różnorakie ważne dla żukoskoczków zagadki. Ich rozwiązanie może przyczynić się do pozyskania stosownych kluczy.

Wszystkie klucze zdobyte przez drużynę są do jej dyspozycji aż do momentu ich użycia, tzn. niewykorzystane klucze zostają na koncie drużyny w kolejnych rozgrywkach.

6.7. Początkowy stan rozgrywki

Każda drużyna na początku rozgrywki dysponuje tą samą liczbą statków poszczególnych typów. Statki każdej drużyny zgromadzone są w pobliżu jednego z portów (dla każdej drużyny innego).

6.8. Cel rozgrywki i rywalizacja

Głównym celem drużyny jest pozyskanie amuletu złożonego z określonej (lub większej) liczby części. Możliwe jest połączenie mniejszej liczby artefaktów, jednak moc takiego złączenia nie będzie wystarczająca do zakończenia rozgrywki.

Punktacja Za każdy posiadany przez drużynę artefakt drużyna otrzymuje 20 punktów, niezależnie od tego czy obiekt ten jest niezależnym przedmiotem czy też częścią amuletu.

Dodatkowo za każdy posiadany amulet drużyna otrzyma odpowiednio:

Liczba złączonych części	Liczba punktów
2	100
3	250
4	500
5	1200
6 lub więcej	2500

Do punktacji wliczany jest także aktywny udział w walce o artefakty. Za każde zaliczone trafienie w przeciwnika (statek, port lub twierdza) drużyna otrzyma 1 punkt. Ponadto drużyna (lub drużyny) z największą liczbą trafień w rozgrywce otrzymują specjalny bonus w postaci powiększenia sumarycznego dorobku punktowego o 20% (w zaokrągleniu do całości).

Punkty liczone są na bieżąco, a zatem z uwagi na możliwość utraty artefaktów oraz specjalnego bonusu dopuszczalne jest zmniejszanie się dorobku punktowego drużyny w czasie pojedynczej rozgrywki.

6.9. Komendy

Ogólne założenia protokołu komunikacji (nawiązywanie połączenia, logowanie, wysyłanie komend, format odpowiedzi) opisane są w rozdziale *Komunikacja z serwerem*. Poniżej znajduje się lista komend dostępnych dla zadania Morskie Wojny.

DESCRIBE_WORLD Zwraca parametry rozgrywki i wartość współczynnika skalującego wynik.

Parametry: brak

Dane (od serwera):

W pojedynczej linii serwer zwróci cztery wartości:

- N ($N \in \mathbb{N}$, $100 \leqslant N \leqslant 300$) — długość boku planszy
- A ($A \in \mathbb{N}$, $2 \leqslant A \leqslant 6$) — minimalna liczba artefaktów, które tworzą amulet mocy pozwalający zakończyć bieżącą rozgrywkę
- T ($T \in \mathbb{N}$, $1 \leqslant T \leqslant 5$) — czas trwania pojedynczej tury w sekundach
- K ($K \in \mathbb{R}$, $1 \leqslant K \leqslant 8$) — wartość współczynnika skalującego wynik

TIME_TO_FULL_MOON Zwraca liczbę tur pozostałych do pełni księżyca oraz informację o tym, czy w obszarze rozgrywki udało się utworzyć amulet mocy. Informacje dotyczą stanu z początku aktualnej tury.

Parametry: brak

Dane (od serwera):

Serwer zwraca w pierwszej linii oddzielone spacjami wartości:

- *AmuletStatus* (*AmuletStatus* $\in \{$*FORMED*, *PARTS*$\}$) — jeden z dwóch ciągów znaków: *PARTS*, jeśli żadna drużyna nie zebrała jeszcze w jednym miejscu i nie połączyła ze sobą odpowiedniej liczby artefaktów kończących rozgrywkę lub *FORMED*, jeśli ktoś tego dokonał
- M ($M \in \mathbb{N}$) — największa liczba artefaktów w posiadaniu jednej drużyny
- L ($L \in \mathbb{N}$) — liczba tur pozostałych do zakończenia rozgrywki
- P ($P \in \mathbb{N}$) — liczba twierdz, w których ktoś aktualnie próbuje dokonać złączenia artefaktów

Kolejne P linii opisuje pozycje twierdz, w których dokonywana jest próba złączenia artefaktów:

- P_X ($P_X \in \mathbb{N}$, $1 \leqslant P_X \leqslant N$) — współrzędna X pola, w którym znajduje się twierdza
- P_Y ($P_Y \in \mathbb{N}$, $1 \leqslant P_Y \leqslant N$) — współrzędna Y pola, w którym znajduje się twierdza

LIST_SHIPS Zwraca listę statków podlegających dyspozycji drużyny.

Parametry: brak

Dane (od serwera):

Pierwsza linia:

- B ($B \in \mathbb{N}$) — liczba statków

Każda kolejna linia opisuje jeden statek za pomocą oddzielonych spacjami wartości oznaczających kolejno:

- *ID* (*ID* $\in \mathbb{N}$) — identyfikator statku
- P_X ($P_X \in \mathbb{N}$, $1 \leqslant P_X \leqslant N$) — współrzędna X pola, w którym znajduje się statek
- P_Y ($P_Y \in \mathbb{N}$, $1 \leqslant P_Y \leqslant N$) — współrzędna Y pola, w którym znajduje się statek
- *Type* (*Type* $\in \{$*CRUISER*, *DESTROYER*, *PATROL*$\}$) — typ statku (kolejno: krążownik, niszczyciel, zwiadowca)
- W ($W \in \mathbb{N}$) — aktualna wytrzymałość statku
- G ($G \in \mathbb{N} \cup \{NA\}$) — liczba tur do zakończenia pozyskiwania lub łączenia artefaktów lub *NA* jeśli statek nie wykonuje żadnej operacji na artefakcie

MOVE Wydaje jednostce polecenie przejścia do sąsiedniego pola, wskazywanego przy pomocy względnych współrzędnych. Przynajmniej jedna ze względnych współrzędnych D_X, D_Y musi być niezerowa. Wykonanie operacji jest opóźnione o tyle tur ile wynosi wartość bezwładności statku. Wykonanie może zostać anulowane (automatycznie) przez moduł antykolizyjny, jeśli jest on aktywny. Nie ma innej możliwości wymuszenia anulowania już raz wydanego rozkazu przemieszczenia. W danej rundzie można wydać tylko jedno poprawne polecenie MOVE dla danego statku.

Parametry:

- *ID* ($ID \in \mathbb{N}$) — numer ID statku
- D_X ($D_X \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi X
- D_Y ($D_Y \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi Y

SHOOT Wydaje jednostce polecenie ataku wybranego pola wskazywanego przy pomocy względnych współrzędnych. Pole musi znajdować się w zasięgu broni jednostki. Operacja wykonywana jest natychmiast.

Parametry:

- *ID* ($ID \in \mathbb{N}$) — numer ID statku
- D_X ($D_X \in \{-7, -6, ... 6, 7\}$) — różnica wzdłuż osi X
- D_Y ($D_Y \in \{-7, -6, ... 6, 7\}$) — różnica wzdłuż osi Y

GET Wydaje jednostce polecenie załadowania na pokład artefaktu znajdującego się w odległości co najwyżej jeden od statku. Operacja wykonywana jest natychmiast jeśli artefakt to umożliwia (jest typu GETATABLE_ARTIFACT) lub drużyna wykorzysta do tego klucz w oddzielnym poleceniu. W przeciwnym wypadku polecenie trwa 50 tur. Operacja zostaje anulowana jeśli statek się poruszy lub zostanie zniszczony. Jeśli na danym polu jest więcej niż jeden artefakt, w pierwszej kolejności załadowane będą artefakty typu GETATABLE_ARTIFACT.

Parametry:

- *ID* ($ID \in \mathbb{N}$) — numer ID statku
- D_X ($D_X \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi X
- D_Y ($D_Y \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi Y

PUT Wydaje jednostce polecenie rozładowania posiadanego artefaktu do pola odległego o co najwyżej jeden. Operacja wykonywana jest natychmiast. Rozładunek na ląd jest możliwy tylko wtedy, gdy na wyspie znajduje się twierdza.

Parametry:

- *ID* ($ID \in \mathbb{N}$) — numer ID statku
- D_X ($D_X \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi X
- D_Y ($D_Y \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi Y

JOIN Wydaje jednostce polecenie połączenia dwóch lub większej liczby artefaktów zgromadzonych w sąsiadującej ze statkiem twierdzy. Operacja wykonywana jest natychmiast jeśli drużyna wykorzysta do tego klucz w oddzielnym poleceniu. W przeciwnym wypadku polecenie trwa 50 tur. Operacja zostaje anulowana jeśli statek się poruszy lub zostanie zniszczony.

Parametry:

- *ID* ($ID \in \mathbb{N}$) — numer ID statku, który ma dokonywać połączenia
- D_X ($D_X \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi X wskazujące na pole, w którym znajduje się twierdza
- D_Y ($D_Y \in \{-1, 0, 1\}$) — przesunięcie wzdłuż osi Y wskazujące na pole, w którym znajduje się twierdza

LAUNCH_SHIP Woduje nowy statek. Jednostka pojawi się na losowym wolnym polu w sąsiedztwie portu drużyny.

Parametry:

- *ShipType* ($ShipType \in \{CRUISER, DESTROYER, PATROL\}$) — odpowiednio: krążownik, niszczyciel, zwiadowca

Dane (od serwera):

- *ID* ($ID \in \mathbb{N}$) — identyfikator nowego statku
- P_X ($P_X \in \mathbb{N}, 1 \leqslant P_X \leqslant N$) — współrzędna X pola, w którym znajduje się statek
- P_Y ($P_Y \in \mathbb{N}, 1 \leqslant P_Y \leqslant N$) — współrzędna Y pola, w którym znajduje się statek

LIST_LANDS_NEARBY Zwraca listę punktów naziemnych będących w polu widzenia drużyny. Informacje dotyczą stanu z początku aktualnej tury.

Parametry: brak

Dane (od serwera):

Pierwsza linia:

- B ($B \in \mathbb{N}$) — liczba punktów

Każda kolejna linia opisuje jeden punkt naziemny za pomocą oddzielonych spacjami wartości określających kolejno:

- P_X ($P_X \in \mathbb{N}, 1 \leqslant P_X \leqslant N$) — współrzędna X pola
- P_Y ($P_Y \in \mathbb{N}, 1 \leqslant P_Y \leqslant N$) — współrzędna Y pola
- *Type* ($Type \in \{ISLAND, TOWER, PORT\}$) — typ pola z lądem
- *Owner* ($Owner \in \{ME, ENEMY, NOBODY\}$) — kto jest właścicielem portu lub twierdzy
- W ($W \in \mathbb{N} \cup \{INF\}$) — wytrzymałość obiektu; dla *ISLAND* zawsze *INF*

LIST_PRIMARY_TARGETS Zwraca listę widzianych wrogów i artefaktów, które nie są jeszcze w posiadaniu drużyny. Pole widzenia dotyczy sumarycznego obszaru wszystkich statków należących do drużyny. Informacje dotyczą stanu z początku aktualnej tury.

Parametry: brak

Dane (od serwera):

Pierwsza linia:

- B ($B \in \mathbb{N}$) — liczba obiektów

Każda kolejna linia opisuje jeden obiekt za pomocą oddzielonych spacjami wartości określających kolejno:

- P_X ($P_X \in \mathbb{N}, 1 \leqslant P_X \leqslant N$) — współrzędna X pola
- P_Y ($P_Y \in \mathbb{N}, 1 \leqslant P_Y \leqslant N$) — współrzędna Y pola
- *Type* ($Type \in \{GETATABLE\_ARTIFACT, SECURED\_ARTIFACT, CRUISER, DESTROYER, PATROL\}$) — typ obiektu
- W ($W \in \mathbb{N} \cup \{INF\}$) — wytrzymałość obiektu; dla artefaktów zawsze *INF*.

LIST_OWNED_ITEMS Zwraca listę posiadanych przez drużynę obiektów. Dotyczy twierdz, artefaktów oraz portu.

Parametry: brak

Dane (od serwera):

Pierwsza linia:

- B ($B \in \mathbb{N}$) — liczba obiektów

Każda kolejna linia opisuje jeden obiekt za pomocą oddzielonych spacjami wartości, które oznaczają kolejno:

- P_X ($P_X \in \mathbb{N}$, $1 \leqslant P_X \leqslant N$) — współrzędna X pola
- P_Y ($P_Y \in \mathbb{N}$, $1 \leqslant P_Y \leqslant N$) — współrzędna Y pola
- $Type$ ($Type \in \{GETATABLE\_ARTIFACT, SECURED\_ARTIFACT, TOWER, PORT\}$) — typ obiektu
- W ($W \in \mathbb{N} \cup \{INF\}$) — aktualna wytrzymałość obiektu

RIDDLE_INFO Zwraca informacje dotyczące jednej z zagadek. Zagadki dostarczane są oddzielnie jako zabezpieczone hasłem archiwa.

Parametry: brak

Dane (od serwera):

Jedyna linia zawiera:

- $RiddleNo$ ($RiddleNo \in \mathbb{N} \cup \{NIL\}$) — numer zagadki lub *NIL* jeśli informacja jest niedostępna
- $Keys$ ($Keys \in \mathbb{N} \cup \{NIL\}$) — liczba kluczy do zdobycia poprzez rozwiązanie zagadki lub *NIL* jeśli informacja jest niedostępna
- $Pass$ ($Pass \in PasswordString \cup \{NIL\}$) — hasło otwierające archiwum z zagadką lub *NIL* jeśli informacja jest niedostępna; *PasswordString* oznacza ciąg znaków, który może zawierać małe [a-z] i duże [A-Z] litery oraz cyfry [0-9].

ANSWER Udziela odpowiedzi na wybraną zagadkę i pozwala zdobyć klucze do artefaktów (w przypadku poprawnej odpowiedzi). Polecenia można użyć tylko raz na 300 tur.

Parametry:

- $RiddleNo$ ($RiddleNo \in \mathbb{N}$) — numer zagadki
- Ans ($Ans \in String$) — odpowiedź na zagadkę; *String* oznacza ciąg znaków zgodny z formatem podanym w treści zagadki

USE_KEY Wykorzystuje jeden z wcześniej zdobytych przez drużynę kluczy do przyspieszenia operacji na artefakcie.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator statku, który próbuje przenieść lub połączyć artefakty

MY_STAT Zwraca informacje dotyczące osiągnięć drużyny.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią postaci:

- $Ships_r$ ($Ships_r \in \mathbb{N}$) — liczba statków gotowych do zwodowania
- $Shoots$ ($Shoots \in \mathbb{N}$) — liczba zaliczonych trafień w przeciwnika
- $Bonus$ ($Bonus \in \{0\%\} \cup \{20\%\}$) — uzyskany bonus za największą liczbę trafień
- $KeyCnt$ ($KeyCnt \in \mathbb{N}$) — liczba posiadanych kluczy do artefaktów
- $ArtCnt$ ($ArtCnt \in \mathbb{N}$) — liczba posiadanych artefaktów
- $Amulet_m$ ($Amulet_m \in \mathbb{N}$) — największy rozmiar posiadanego amuletu
- $Points$ ($Points \in \mathbb{R}$) — liczba zdobytych w tej rozgrywce punktów (bez mnożenia przez wartość K)

WAIT Czeka do momentu rozpoczęcia kolejnej tury.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią postaci:

- WAITING S

gdzie S ($S \in \mathbb{R}, S \geqslant 0$) oznacza liczbę sekund do zakończenia oczekiwania. Po upływie tego czasu wysyła dodatkowo linię:

- OK

6.10. Błędy

Zgodnie z opisem w rozdziale *Komunikacja z serwerem*, w wypadku podania nieprawidłowego polecenia, serwer odpowiada komunikatem:

- '`FAILED` *e msg*',

gdzie *e* to kod błędu, a *msg* — komunikat błędu. W tabeli znajduje się zestawienie błędów, które mogą wystąpić przy wydawaniu poleceń w zadaniu Morskie Wojny.

kod błędu	komunikat błędu
1	bad login or password
2	unknown command
3	bad format
4	too many arguments
5	internal error, sorry...
6	commands limit reached, forced waiting activated
100	incorrect identifier
101	incorrect destination field type
102	incorrect destination coordinates
103	incorrect ship type
104	ship already loaded
105	field contains no artifact
106	field is not your possession
107	no ships to launch
108	no free docks available
109	no more keys available
110	no more shoots available
111	incorrect answer
112	usage limit reached for this command
113	command already executed
114	no operation to unlock
115	ship contains no artifact
116	not enough artifacts
117	riddle already answered

6.11. Serwery

Rozgrywki będą odbywać się na serwerach, różniących się między sobą parametrami.

nazwa	adres:port	aktywny moduł antykolizyjny
Ships1	universum.dl24:20003	tak
Ships2	universum.dl24:20004	nie

6.12. Przykład

Poniżej znajduje się przykładowy zapis komunikacji z serwerem.

klient → serwer	serwer → klient
	`LOGIN`
`login1`	
	`PASS`
`secret`	
	`OK`
`DESCRIBE_WORLD`	
	`OK`
	`100 2 1 7.9`
`TIME_TO_FULL_MOON`	
	`OK`
	`PARTS 1 100 0`
`LIST_SHIPS`	
	`OK`
	`2`
	`1 10 20 DESTROYER 4 NA`
	`2 90 91 PATROL 3 NA`
`LIST_LANDS_NEARBY`	
	`OK`
	`3`
	`10 22 PORT ENEMY 1`
	`89 91 TOWER NOBODY 1`
	`50 50 PORT ME 10`
`LIST_PRIMARY_TARGETS`	
	`OK`
	`2`
	`11 20 CRUISER 3`
	`89 91 SECURED_ARTIFACT INF`
`SHOOT 1 0 2`	
	`OK`
`SHOOT 1 1 0`	
	`OK`
`SHOOT 1 1 0`	
	`OK`
`SHOOT 1 1 0`	
	`OK`
`SHOOT 2 -1 0`	
	`OK`
`GET 2 -1 0`	
	`OK`
`WAIT`	
	`OK`
	`WAITING 0.13001`
	`OK`
`LIST_PRIMARY_TARGETS`	
	`OK`
	`0`

```
LIST_LANDS_NEARBY
                        OK
                        3
                        10 22 ISLAND NOBODY INF
                        89 91 TOWER ME 5
LIST_SHIPS
                        OK
                        2
                        1 10 20 DESTROYER 4 NA
                        2 90 91 PATROL 3 50
RIDDLE_INFO
                        OK
                        16 1 CBA
ANSWER 16 ThatWasHard
                        OK
MY_STAT
                        1 15 0% 1 0 0 98.000000
USE_KEY 2
                        OK
LIST_OWNED_ITEMS
                        OK
                        3
                        89 91 TOWER 5
                        90 91 SECURED_ARTIFACT INF
                        50 50 PORT 10
LAUNCH_SHIP CRUISER
                        OK
                        3 20 20
WAIT
                        OK
                        WAITING 0.65000
                        OK
LIST_SHIPS
                        OK
                        3
                        1 10 20 DESTROYER 4 NA
                        2 90 91 PATROL 3 NA
                        3 51 51 CRUISER 15 NA
LIST_PRIMARY_TARGETS
                        OK
                        1
                        57 55 GETATABLE_ARTIFACT INF
LIST_LANDS_NEARBY
                        OK
                        3
                        10 22 ISLAND NOBODY INF
                        89 91 TOWER ME 10
                        50 50 PORT ME 10
                        57 55 TOWER ENEMY 100
MOVE 1 1 0
                        OK
MOVE 2 0 -1
                        OK
SHOOT 3 6 4
                        OK
```

7. Eksperyment

7.1. Wprowadzenie

Był piękny, choć całkiem normalny, wiosenny poranek. Jak co dzień na niebie można było dostrzec nieliczne obłoki, a z oddali dało się słyszeć śpiew ptaków, siedzących na gałęziach coraz śmielej zakwitających drzew. Było, jak na tę porę roku, zupełnie zwyczajnie.

Profesor Beetlestein, wyraźnie zamyślony, rozsiadł się wygodnie w swoim ulubionym przenośnym, składanym fotelu. W jego oczach nie można było już dostrzec tego samego błysku, który towarzyszył mu przez jego pierwsze, najbardziej owocne lata kariery naukowca. Wtedy jego publikacje pojawiały się we wszystkich poważnych czasopismach naukowych, a jego badania doprowadziły do wynalezienia próżni w spreju, bez której dziś żukoskoczki nie wyobrażają sobie życia. Od tamtej pory minęło jednak wiele wiosennych (i nie tylko) poranków, a profesorowi nigdy więcej nie udało się nawiązać do tamtych sukcesów. Nazwisko Beetlestein całkowicie przestało być używane w bibliografiach, a w kuluarach coraz częściej spekulowano o przejściu sędziwego profesora na emeryturę. Najgorsze było jednak to, że co bardziej złośliwi zaczęli coraz głośniej podawać w wątpliwość jego wcześniejsze osiągnięcia, sugerując przywłaszczenie sobie przez niego wyników jego byłych, choć jeszcze nieznanych z nazwiska, zdolnych doktorantów.

To wszystko miało się jednak za chwilę zmienić. Właśnie dziś zakończyły się przygotowania do eksperymentu, któremu profesor poświęcił kilka ostatnich lat swojego życia. Beetlestein za nic miał głosy kolegów po fachu, którzy wypominali mu sprzeczność jego teorii z drugim prawem Żukova. „Banda ignorantów" – pomyślał, a na jego twarzy zagościł lekki uśmiech. Wiedział, że aby zamknąć usta krytykom potrzebował czegoś naprawdę spektakularnego, dlatego za cel swego eksperymentu obrał Ubluru, świętą górę żukoskoczków. Teleportacja całej góry z niedostępnych terenów Heksagonii wprost do ośrodka badawczego w samym środku stolicy z pewnością nie umknie uwadze żadnego z nich. Przy okazji, mając swobodny dostęp do góry i najnowocześniejszych technologii znajdujących się w ośrodku, można będzie zweryfikować prawdziwość legend o pełnym drogocennych skarbów ukrytym labiryncie, do którego wejścia dotychczas nie odnaleziono.

Właśnie z tego powodu teraz, u samego podnóża Ubluru, znajdowała się niebagatelnych rozmiarów maszyna, obok której w swym fotelu siedział gotowy na wszystko Beetlestein. Pewny wyniku eksperymentu, w pełni delektując się chwilą, powoli wcisnął specjalnie na tę okazję przygotowany czerwony guzik z napisem „Zwycięstwo".

Niemal natychmiast, z tajemniczego urządzenia wystrzeliła niebieska struga światła, która skierowała się na sam środek góry. Ziemia zatrzęsła się, a całe Ubluru okryło się gęstą chmurą pyłu, który opadł dopiero po niespełna minucie, ukazując oczom zdumionego naukowca – tam, gdzie jeszcze przed chwilą było samo serce góry – słusznej wielkości budowlę w kształcie piramidy, z doskonale widocznym, misternie zdobionym portalem.

Wiedziony instynktem Beetlestein, zupełnie zapominając o teleportacji, eksperymencie i Ubluru (gdziekolwiek ona teraz była), ruszył biegiem w stronę wejścia. Po chwili zachwytu nad zdobiącymi portal grafikami pchnął ogromne wrota, które bez żadnego oporu, niemal bezszelestnie otworzyły się, ukazując ogromną powierzchnię wnętrza budowli.

— Zróbcie coś z tym przeklętym obrazem, poruczniku! — polecił zniecierpliwiony Generał Żukk siedzącemu przy terminalu młodemu żukoskoczkowi.

— Robię, co mogę, Generale — odparł porucznik, gorączkowo próbując ustabilizować komunikację z minikamerą podrzuconą tydzień wcześniej Beetlesteinowi przez wojskowy wywiad. Tak potężny teleporter miałby ogromny potencjał militarny, nic więc dziwnego, że wojsko śledziło uważnie rozwój badań profesora. Teraz jednak liczyła się tylko tajemnicza piramida i to, co kryło się w jej środku.

Zgromadzeni w sali oficerowie w milczeniu wpatrywali się w monitor ze stabilnym już obrazem, na którym profesor Beetlestein przemierza ciemne korytarze, zafascynowany swym niezwykłym odkryciem. Nagle ściany labiryntu zaczęły się gwałtownie poruszać. Profesor przystanął.

— Jak ja stąd teraz wyjdę? — pomyślał, obracając się i w ostatniej chwili świadomości dostrzegając otwierające się tuż przed nim ogromne szczęki.

W sali projekcyjnej zapadła niezręczna cisza, którą przerwał dopiero stanowczy głos Żukka:

— Pułkowniku Kretschmer, wyślijcie tam natychmiast cały batalion! Chcę wiedzieć co tam jest, dlaczego tam jest, i jak możemy to wykorzystać do naszych celów!

7.2. Zadanie

Dany jest labirynt składający się z jednakowych rozmiarów pomieszczeń. W części z pomieszczeń sufit uległ zawaleniu, co skutecznie uniemożliwia przebywanie w nich. Plan labiryntu przedstawia prostokąt o wymiarach $W \times H$. Każde pomieszczenie ma co najwyżej 4 sąsiednie pomieszczenia. Niektóre z pomieszczeń mają połączenie ze światem zewnętrznym i są przez to traktowane jako wyjścia z labiryntu.

Każda z drużyn dowodzi grupą eksplorujących labirynt postaci, które dowolnie przemieszczają się między dostępnymi pomieszczeniami. W jednym pomieszczeniu może jednocześnie przebywać dowolna liczba postaci. Po upływie znanego od samego początku czasu cały labirynt ulega zawaleniu, grzebiąc wszystkie przebywające w nim postacie poza tymi, które znajdują się w pomieszczeniach – wyjściach.

W niektórych pomieszczeniach znajdują się wartościowe skarby. W labiryncie nikomu się na nic nie przydadzą – co innego w kwaterze głównej Generała Żukka. Skarby mogą być transportowane przez postacie i wynoszone z labiryntu przez wyjścia. Celem rozgrywki jest wyniesienie jak największej liczby skarbów.

7.3. Opis labiryntu

Oddziały przemierzające labirynt są zaopatrzone w nowoczesne urządzenia lokalizacyjne, dzięki którym dokładnie znają aktualny kształt labiryntu. W każdym z pomieszczeń może znajdować się:

- skarb o określonej wartości,
- potwór, czyli strzegąca skarbu bestia, która rzuca się wściekle na każdego kto znajdzie się w pobliżu,
- zbrojownia, umożliwiająca zakup broni do walki z potworami.

Na początku rozgrywki każda z postaci znajduje się w jednym z pomieszczeń będących wyjściami z labiryntu. Wyjścia różnią się od innych pomieszczeń tym, że:

- przebywanie w nich chroni postać przed śmiercią w chwili całkowitego zawalenia się labiryntu,
- odłożenie w nich skarbu powoduje jego wydobycie na zewnątrz i automatyczne umieszczenie jego wartości na koncie danej drużyny,
- nigdy nie przebywają w nich potwory,
- nie ma tam zbrojowni.

Przedmioty

Każdą postać B charakteryzują dwie wartości: udźwig C_B oraz udźwig maksymalny C_B^{max}, określone wzorami:

$$C_B = 10 \cdot (1.1)^n$$
$$C_B^{max} = \frac{3}{2} C_B$$

gdzie n to liczba pokonanych przez B potworów. B może przenosić ze sobą przedmioty, do których zaliczają się bronie oraz skarby, o łącznym ciężarze nie większym niż C_B^{max}. Jeśli jednak przenoszone przedmioty ważą więcej niż C_B, B będzie się poruszać dwa razy wolniej niż zwykle (będzie mogła wykonać tylko jeden ruch na dwie tury).

Zbrojownie

W dostępnym pomieszczeniu, które nie jest wyjściem, może być zlokalizowana zbrojownia. W zbrojowni nie może znajdować się ani skarb, ani żaden potwór. Można tam wymienić część posiadanego przez żukoskoczka skarbu na broń. Wszystkie dostępne w danym momencie zbrojownie zawierają dokładnie ten sam zestaw broni, dostępnej za te same ceny. Broni każdego z rodzajów jest zawsze wystarczająco dużo, żeby wyposażyć wszystkie żukoskoczki przebywające w labiryncie.

Jedna sztuka broni nabyta w zbrojowni wystarcza do pokonania jedynie dwóch potworów. Po tym czasie ulega ona zniszczeniu (znika z ekwipunku postaci). Posiadaną broń można wyrzucić tylko w zbrojowni.

Potwory

W dostępnym pomieszczeniu, nie będącym ani wyjściem ani zbrojownią, może znajdować się co najwyżej jeden potwór. Potwory nie zmieniają swojego położenia, ale mogą się pojawiać nagle tam, gdzie aktualnie nie przebywa żadna postać. Każdy potwór M posiada siłę ataku A_M podaną wzorem:

$$A_M = \min\left(\lceil V_M \rceil, 100\right),$$

gdzie V_M oznacza wartość pilnowanego przez niego skarbu.

Walka Wchodząc do pomieszczenia z potworem, postać musi stoczyć z nim walkę, która odbywa się zawsze tuż po zakończeniu tury, w której wydano rozkaz wejścia do pomieszczenia. W walce tej uczestniczą wszystkie znajdujące się tam wtedy postacie. Walka polega na porównaniu siły ataku potwora oraz sumy sił ataków obecnych tam postaci i kończy się zwycięstwem postaci, jeśli ich sumaryczna siła ataku jest większa od siły ataku potwora. W tym przypadku potwór ginie, a jego skarb rozdzielany jest według zasad opisanych poniżej. W przeciwnym razie walka kończy się zwycięstwem potwora – wszystkie postacie uczestniczące w walce giną, a posiadane przez nich skarby zostają przejęte przez potwora (więc także, być może, zwiększa się jego siła ataku).

Siła ataku postaci posiadającej broń jest równa sile ataku posiadanej przez nią broni. Jeśli żadna broń nie jest w jej posiadaniu, siła ataku jest równa 1. Postać w danej chwili może posiadac tylko jedną broń.

Podział łupów Skarb pilnowany przez pokonanego potwora rozdzielany jest według następujących zasad:

1. Skarb należny każdej postaci P wyliczany jest według wzoru:

$$U_P = S \cdot \frac{A_P}{\sum_{p \in P_{all}} A_p},$$

gdzie S oznacza wartość skarbu pozostałego do rozdzielenia, A_p to siła ataku postaci P, a P_{all} to zbiór wszystkich postaci branych pod uwagę przy podziale skarbu. Jeśli w wyniku walki z potworem broń postaci uległa zniszczeniu, to przy podziale łupów liczy się tak, jakby ta postać nie miała żadnej broni (ma wartość ataku równą 1).

2. Jeśli dana postać nie może przyjąć całego należnego jej skarbu z powodu dopuszczalnego udźwigu, następuje próba przydzielenia tego skarbu do pozostałych obecnych w pomieszczeniu członków danej drużyny. Zostają oni po kolei obdarowywani do swojego maksymalnego udźwigu, dopóki cały nadmiar skarbu nie zostanie rozdzielony lub wszystkie postacie z drużyny, biorące udział w podziale, nie będą już posiadały wolnego udźwigu.
3. Jeśli po poprzednim kroku pozostanie jakiś skarb, to jest on rozdzielony zaczynając od punktu 1 z uwzględnieniem tylko tych postaci, które posiadają jeszcze wolny udźwig.
4. Jeśli nie ma już postaci posiadających wolny udźwig, a skarb dalej nie został rozdzielony w całości, pozostała jego część pozostaje w pomieszczeniu.

Skarby

Wartość skarbu jest równa jego ciężarowi. Gdy skarb nie jest pilnowany, można podjąć próbę podniesienia go w całości bądź tylko w pewnej jego części. W razie potrzeby część skarbu moża zostawić (wyrzucić) w pomieszczeniu, w którym aktualnie znajduje się postać.

Jeżeli dwie lub więcej postaci próbuje unieść skarb w tym samym pomieszczeniu, to zostanie on między nie rozdzielony według tych samych zasad, które obowiązują przy podziale skarbu strzeżonego przez potwora. Także jeśli jedna z postaci zechce podnieść tylko część obecnego skarbu algorytm jest taki sam, z tym że udźwig maksymalny każdej postaci zostaje, na czas podziału, zmniejszony do zadeklarowanej przez nią wartości.

7.3.1. Zmiany w labiryncie

Labirynt może zmieniać się w trakcie trwania rozgrywki: pomieszczenia wcześniej niedostępne mogą stać się dostępne i odwrotnie, a w dostępnych pomieszczeniach mogą pojawić się nowe skarby i potwory. Zmiany w dostępności dokonują się tylko w wybranych turach. Czas do następnej takiej zmiany jest znany.

7.4. Punktacja

Po zakończeniu rozgrywki każda drużyna otrzymuje punkty w wysokości:

$$P = L \cdot G,$$

gdzie L to procent postaci danej drużyny, które przeżyły po ostatecznym zawaleniu się labiryntu, a G to suma wartości skarbów zebranych przez drużynę.

7.5. Komendy

Ogólne założenia protokołu komunikacji (nawiązywanie połączenia, logowanie, wysyłanie komend, format odpowiedzi) opisane są w rozdziale *Komunikacja z serwerem*. Poniżej znajduje się lista komend dostępnych dla zadania Eksperyment.

DESCRIBE_WORLD Zwraca parametry rozgrywki.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią zawierającą dwie wartości oddzielone spacją:

- T ($T \in \mathbb{N},\ 1 \leqslant T \leqslant 3$) — czas trwania pojedynczej tury, w sekundach
- K ($K \in \mathbb{R},\ 1 \leqslant K \leqslant 8$) — wartość współczynnika skalującego wynik

TIME_TO_CHANGE Zwraca liczbę tur pozostałych do następnej zmiany struktury labiryntu oraz liczbę tur pozostałych do jego całkowitego zawalenia się.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią składającą się z dwóch oddzielonych spacją wartości:

- T_M ($T_M \in \mathbb{N},\ 0 \leqslant T_M \leqslant 3600$) — liczba tur pozostałych do następnej zmiany struktury labiryntu
- T_C ($T_C \in \mathbb{N},\ 0 \leqslant T_C \leqslant 3600$) — liczba tur pozostałych do całkowitego zawalenia się labiryntu

MAZE Zwraca opis labiryntu. Polecenie można wydać tylko raz na 15 tur i dodatkowo po zmianie struktury labiryntu.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera dwie oddzielone spacją liczby:

- W ($W \in \mathbb{N},\ 10 \leqslant W \leqslant 200$) — szerokość prostokąta, przedstawiającego plan labiryntu
- H ($H \in \mathbb{N},\ 10 \leqslant H \leqslant 200$) — wysokość prostokąta, przedstawiającego plan labiryntu

Każda z następnych H linii zawiera W znaków, opisujących poszczególne pomieszczenia labiryntu w taki sposób, że i-ty znak w j-tej linii opisuje pomieszczenie o współrzędnych (i, j). Każdy taki znak może być jednym z poniższych:

- `#` — pomieszczenie niedostępne
- `.` — pomieszczenie dostępne (można się po nim poruszać)
- `A` — zbrojownia
- `E` — wyjście z labiryntu
- `T` — skarb

EQUIPMENT Zwraca listę broni dostępnych w zbrojowniach.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera jedną liczbę:

- N ($N \in \mathbb{N}$) — liczba typów broni dostępnych aktualnie w każdej zbrojowni

Każda z następnych N linii opisuje kolejne typy broni. Opis jednego typu broni składa się z czterech oddzielonych spacją liczb:

- I ($I \in \mathbb{N},\ 1 \leqslant I \leqslant N$) — numer
- P ($P \in \mathbb{R},\ 1 \leqslant P \leqslant 30$) — cena

- A ($A \in \mathbb{N},\ 1 \leqslant A \leqslant 100$) — siła ataku
- W ($W \in \mathbb{R},\ 1 \leqslant W \leqslant 30$) — ciężar

TREASURES Zwraca listę skarbów znajdujących się w labiryncie.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera jedną liczbę:

- N ($N \in \mathbb{N}$) — liczba skarbów znajdujących się w labiryncie

Każda z następnych N linii opisuje kolejne skarby. Opis jednego skarbu składa się z trzech oddzielonych spacją liczb:

- X ($X \in \mathbb{N},\ 1 \leqslant X \leqslant W$) — pierwsza współrzędna pomieszczenia, w którym znajduje się skarb
- Y ($Y \in \mathbb{N},\ 1 \leqslant Y \leqslant H$) — druga współrzędna pomieszczenia, w którym znajduje się skarb
- V ($V \in \mathbb{R},\ V > 0$) — wartość skarbu

MONSTERS Zwraca listę potworów, informacje o ich sile ataku i posiadanych skarbach.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera jedną liczbę:

- N ($N \in \mathbb{N}$) — liczba potworów znajdujących się w labiryncie

Każda z następnych N linii opisuje kolejne potwory. Opis jednego potwora składa się z czterech oddzielonych spacją liczb:

- X ($X \in \mathbb{N},\ 1 \leqslant X \leqslant W$) — pierwsza współrzędna pomieszczenia, w którym znajduje się potwór
- Y ($Y \in \mathbb{N},\ 1 \leqslant Y \leqslant H$) — druga współrzędna pomieszczenia, w którym znajduje się potwór
- A ($A \in \mathbb{N},\ 1 \leqslant A \leqslant 100$) — siła ataku potwora
- V ($V \in \mathbb{R},\ V \geqslant 1$) — wartość skarbu strzeżonego przez potwora

ALL_EXPLORERS Zwraca informacje o postaciach znajdujących się w labiryncie.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera jedną liczbę:

- N ($N \in \mathbb{N}$) — liczba wszystkich postaci znajdujących się w labiryncie

Każda z następnych N linii opisuje kolejne postacie. Opis jednej postaci składa się z trzech oddzielonych spacją liczb:

- X ($X \in \mathbb{N},\ 1 \leqslant X \leqslant W$) — pierwsza współrzędna pomieszczenia, w którym znajduje się postać
- Y ($Y \in \mathbb{N},\ 1 \leqslant Y \leqslant H$) — druga współrzędna pomieszczenia, w którym znajduje się postać
- A ($A \in \mathbb{N},\ 1 \leqslant A \leqslant 100$) — siła ataku postaci

MY_EXPLORERS Zwraca informacje o postaciach danej drużyny.

Parametry: brak

Dane (od serwera):

Pierwsza linia odpowiedzi zawiera jedną liczbę:

- N ($N \in \mathbb{N}$) — liczba postaci należących do drużyny znajdujących się w labiryncie

Każda z następnych N linii opisuje kolejne postacie. Opis jednej postaci składa się z:

- ID ($ID \in \mathbb{N}$) — identyfikator
- X ($X \in \mathbb{N},\ 1 \leqslant X \leqslant W$) — pierwsza współrzędna pomieszczenia, w którym znajduje się postać
- Y ($Y \in \mathbb{N},\ 1 \leqslant Y \leqslant H$) — druga współrzędna pomieszczenia, w którym znajduje się postać
- A ($A \in \mathbb{N},\ 1 \leqslant A \leqslant 100$) — siła ataku postaci
- W ($W \in \mathbb{R},\ 0 \leqslant W \leqslant 10$) — ciężar broni posiadanej przez postać lub 0 jeśli postać nie posiada broni
- U ($U \in \{0, 1, 2\}$) — liczba potworów, które można jeszcze pokonać przy użyciu posiadanej broni lub 0 jeśli postać nie posiada broni
- V ($V \in \mathbb{R},\ V \geqslant 0$) — wartość skarbu będącego w posiadaniu postaci
- C ($C \in \mathbb{R},\ C \geqslant 10$) — udźwig postaci
- B ($B \in \{0, 1, 2\}$) — liczba tur (łącznie z obecną), przez które postać będzie zajęta lub 0 jeśli postaci nie przydzielono jeszcze żadnych rozkazów

TAKE_TREASURE Podejmuje próbę podniesienia skarbu znajdującego się w danym pomieszczeniu. Polecenie wykonywane jest tuż po zakończeniu tury, w której zostało wydane.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator postaci
- V ($V \in \mathbb{R},\ V > 0$) — wartość skarbu jaką postać ma podnieść

DROP_TREASURE Odrzuca posiadany skarb. Polecenie wykonywane jest natychmiast.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator postaci
- V ($V \in \mathbb{R},\ V > 0$) — wartość skarbu jaką postać ma odrzucić

BUY_WEAPON Dokonuje zakupu w zbrojowni. Polecenie wykonywane jest natychmiast.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator postaci
- I ($I \in \mathbb{N},\ 1 \leqslant I \leqslant N$) — numer broni do kupienia

LEAVE_WEAPON Zwraca posiadaną broń w zbrojowni. Polecenie wykonywane jest natychmiast.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator postaci

MOVE Przemieszcza postać na sąsiadujące pole. Polecenie wykonywane jest tuż po zakończeniu tury, w której zostało wydane. Dokładnie jedna ze współrzędnych (D_X, D_Y) musi być równa 0.

Parametry:

- ID ($ID \in \mathbb{N}$) — identyfikator postaci
- D_X ($D_X \in \{-1, 0, 1\}$) — przesunięcie wzdłuż pierwszej współrzędnej
- D_Y ($D_Y \in \{-1, 0, 1\}$) — przesunięcie wzdłuż drugiej współrzędnej

WAIT Czeka do momentu rozpoczęcia kolejnej tury.

Parametry: brak

Dane (od serwera):

Serwer odpowiada jedną linią postaci:

- WAITING S

gdzie S ($S \in \mathbb{R}, S \geqslant$) oznacza liczbę sekund do zakończenia oczekiwania. Po upływie tego czasu wysyła dodatkowo linię:

- OK

7.6. Błędy

Zgodnie z opisem w rozdziale *Komunikacja z serwerem*, w wypadku podania nieprawidłowego polecenia, serwer odpowiada komunikatem:

- '`FAILED` *e msg*',

gdzie *e* to kod błędu, a *msg* — komunikat błędu. W tabeli znajduje się zestawienie błędów, które mogą wystąpić przy wydawaniu poleceń w zadaniu Eksperyment.

kod błędu	komunikat błędu
1	bad login or password
2	unknown command
3	bad format
4	too many arguments
5	internal error, sorry...
6	commands limit reached, forced waiting activated
101	incorrect explorer identifier
102	explorer is busy
103	invalid treasure amount
104	cannot take treasure in current location
105	no treasure to drop
106	cannot drop treasure in current location
107	not in armory
108	invalid weapon identifier
109	already has weapon
110	not enough capacity to buy selected weapon
111	no weapon to leave
112	destination is not a neighbor
113	destination is collapsed
114	not enough treasure to buy weapon
115	command not yet available

7.7. Serwery

Rozgrywki będą odbywać się na kilku serwerach, różniących się między sobą własnościami spotykanych w labiryncie obiektów.

nazwa	adres:port
Maze1	universum.dl24:20005
Maze2	universum.dl24:20006
Maze3	universum.dl24:20007

7.8. Przykład

Poniżej znajduje się przykładowy zapis komunikacji z serwerem.

klient → serwer	serwer → klient
	`LOGIN`
`login1`	
	`PASS`
`secret`	
	`OK`
`DESCRIBE_WORLD`	
	`OK`
	`1 1.0`
`TIME_TO_CHANGE`	
	`OK`
	`10 1200`
`MAZE`	
	`OK`
	`10 10`
	`##########`
	`#....#...#`
	`#.#..#.T.#`
	`#.#..#...#`
	`#.#.....A#`
	`#.########`
	`#........#`
	`########.#`
	`#.E......#`
	`##########`
`TREASURES`	
	`OK`
	`1`
	`8 3 1`
`MONSTERS`	
	`OK`
	`1`
	`8 2 9 9`
`ALL_EXPLORERS`	
	`OK`
	`2`
	`9 5 1`
	`8 3 10`
`MY_EXPLORERS`	
	`OK`
	`2`
	`1 9 5 1 0 0 1 10 0`
	`2 8 3 10 10 1 0 10 0`
`TAKE_TREASURE 2 1`	
	`OK`
`EQUIPMENT`	
	`OK`
	`1`
	`1 1 10 1`

klient → serwer	serwer → klient
`BUY_WEAPON 1 1`	
	`OK`
`WAIT`	
	`OK`
	`WAITING 0.79000`
	`OK`
`TREASURES`	
	`OK`
	`0`
`MY_EXPLORERS`	
	`OK`
	`2`
	`1 9 5 10 1 2 0 10 0`
	`2 8 3 10 1 1 1 10 0`
`MOVE 2 0 -1`	
	`OK`
`WAIT`	
	`OK`
	`WAITING 0.53000`
	`OK`
`MONSTERS`	
	`OK`
	`0`
`MY_EXPLORERS`	
	`OK`
	`2`
	`1 9 5 10 1 2 0 10 0`
	`2 8 2 1 0 0 10 10 0`