

Wstęp do programowania – laboratorium, potok imperatywny (Inf 21/22.Z

[Kokpit](#)[Moje kursy](#)[WPI.Lab.Inf.21/22.Z](#)[Laboratorium 8 \(22 i 24 listopada 2021\)](#)[Poziomka \(treść, rozwiązanie i ocena jakości\)](#)

Poziomka (treść, rozwiązanie i ocena jakości)

Wprowadzenie

Poziomka to dwuosobowa gra kombinatoryczna rozgrywana na prostokątnej planszy podzielonej na wiersze i kolumny.

Gra jest parametryzowana trzema dodatnimi liczbami całkowitymi:

- **POLA** to liczba pól planszy zajmowanych przez gracza w jednym ruchu,
- **WIERSZE** to liczba wierszy planszy,
- **KOLUMNY** to liczba kolumn planszy.

Dla wartości **POLA** równej 2 gra znana jest pod nazwą [Domineering](#), a dla **POLA** równego 3 to Triomineering.

Gracze, nazywani dalej *Lewym* i *Prawym*, siedzą przy sąsiednich bokach planszy, nie naprzeciwko siebie. Stałe **WIERSZE** i **KOLUMNY** określają rozmiar planszy z punktu widzenia Prawego. Kolumny nazywamy wielkimi literami, od 'A', a wiersze nazywamy małymi literami, od 'a'.

Gracze wykonują ruchy na przemian, zaczynając od Lewego. Lewy może przekazać prawo wykonania pierwszego ruchu Prawemu.

Ruch gracza polega na umieszczeniu na planszy bloku zajmującego spójny zestaw wolnych pól. Blok ma szerokość **POLA**, wysokość 1 i jest poziomy z punktu widzenia gracza, który go umieścił, a pionowy dla przeciwnika. Grając na kartce papieru, gracze rysują poziomą dla siebie kreskę przez określoną liczbę wolnych pól.

Gracz, który nie może wykonać ruchu, gdy jest jego kolej, lub uzna, że nie chce kontynuować rozgrywki, poddaje się. Przeciwnik gracza, który się poddał, wygrywa nawet, jeżeli sam nie może już wykonać ruchu.

Ocena planszy

Zwyczaję gry poznajemy po jej zakończeniu, ale czasem, podczas trwania rozgrywki, chcemy określić aktualny wynik. Nazywamy go *oceną planszy*.

Przyjmujemy, że ocena planszy z punktu widzenia danego gracza jest różnicą między liczbą bloków, które w danej chwili na raz mógłby umieścić na planszy a liczbą bloków, które na raz mógłby umieścić na planszy jego przeciwnik.

Na planszy do gry z wartością **POLA** równą 3, **WIERSZE** równą 7 i **KOLUMNY** równą 11:

	A	B	C	D	E	F	G	H	I	J	K	
a	.	.	.	#	.	.	#	
b	.	.	.	#	.	.	#	
c	#	#	#	.	#	#	#	#	.	.	.	
d	#	
e	.	.	#	.	.	#	.	.	#	#	#	
f	.	.	#	.	.	.	#	#	#	.	.	
g	.	.	#	
-												+

gdzie pola zajęte oznaczono znakiem '#', a pola puste oznaczono znakiem '.', gracz Lewy, czyli patrzący na planszę z lewej strony, może umieścić 6 bloków a Prawy, patrzący na planszę od dołu, 9. Dla Prawego oceną tej planszy jest więc $9 - 6 = 3$.

Polecenie

Napisz program grający, jako Prawy, w Poziomkę z parametrami **POLA**, **WIERSZE**, **KOLUMNY**.

Wykonanie programu zaczyna się od opcjonalnego przekazania prawa pierwszego ruchu Prawemu przez Lewego. Po tym program czyta zapis kolejnych ruchów Lewego i odpowiada na nie za Prawego, pisząc jego ruchy. Prawy poddaje się, gdy nie może wykonać żadnego ruchu. Program kończy pracę, gdy jeden z graczy podda się.

Spśród wszystkich możliwości program wybiera ten ruch Prawego, po którym ocena planszy będzie dla niego maksymalna. Gdyby takich ruchów było wiele, wybiera jeden z nich. O wyborze decyduje, w sposób opisany poniżej, wartość stałej **WYBOR**.

Wybór ruchu przez Prawego zaczynamy od wyznaczenia ciągu wszystkich ruchów maksymalizujących ocenę planszy, uporządkowanych w kolejności rosnących nazw wiersza a w ramach wiersza, w kolejności rosnących nazw kolumn. Zakładając, że długością tego ciągu jest **n**, numer wybranego ruchu, liczony od **0**, będzie wartością wyrażenia **WYBOR % n**.

Wartości **POLA**, **WIERSZE**, **KOLUMNY** i **WYBOR** są określone za pomocą stałych symbolicznych zdefiniowanych opcją **-D** kompilatora.

W kodzie programu są podane wartości domyślne tych stałych:

- **POLA** ma wartość **5**,
- **WIERSZE** ma wartość **26**,
- **KOLUMNY** ma wartość **26**,
- **WYBOR** ma wartość **1000**.

Postać danych

Na początku danych programu opcjonalnie może być wiersz zawierający tylko znak **'-'**. Informuje on o przekazaniu prawa wykonania pierwszego ruchu przez Lewego Prawemu.

Kolejne wiersze danych to zapis ruchów Lewego, po jednym w wierszu. Ruch Lewego reprezentują wielka litera i mała litera wskazujące, odpowiednio, kolumnę, w której ma być blok oraz wiersz, w którym ma być pierwsze pole bloku.

Wiersz danych zawierający tylko znak **'.'** sygnalizuje poddanie się Lewego.

Postać wyniku

Program pisze ruchy Prawego, po jednym w wierszu. Gdy Prawy podda się, program pisze wiersz zawierający tylko znak **'.'**.

Ruch Prawego reprezentują mała litera i wielka litera wskazujące, odpowiednio, wiersz, w którym ma być blok oraz kolumnę, w której ma być pierwsze pole bloku.

W tekście wynikowym programu nie ma spacji ani żadnych innych znaków, które nie zostały wymienione powyżej.

Diagram zamieszczony w opisie oceny planszy przedstawia stan planszy po wczytaniu ruchów Lewego:

```
Fc
Ce
Da
Ga
```

i wypisaniu ruchów Prawego:

```
cH
eI
cA
fG
```

Przykłady

Poniższe przykłady są wynikiem działania programu skompilowanego poleceniem:

```
gcc @opcje -DPOLA=4 -DWIERSZE=10 -DKOLUMNY=7 -DWYBOR=234 zadanie2.c -o zadanie2
```

Do treści zadania dołączone są pliki **.in** z przykładowymi danymi i pliki **.out** z wynikami wzorcowymi.

- Dla danych [przyklad1.in](#) poprawny wynik to [przyklad1.out](#).
- Dla danych [przyklad2.in](#) poprawny wynik to [przyklad2.out](#).
- Dla danych [przyklad3.in](#) poprawny wynik to [przyklad3.out](#).

Walidacja i testy

- Rozwiązania podlegają walidacji, wstępnie badającej zgodność ze specyfikacją.

Walidacja sprawdza działanie programu na przykładach dołączonych do treści zadania.

Pomyślne przejście walidacji jest warunkiem dopuszczenia programu do testów poprawności. Program, który walidacji nie przejdzie, dostaje zerową ocenę poprawności.

- Walidacja i testy są prowadzone na komputerze `students`.
- Programy są kompilowane poleceniem:

```
gcc @opcje ... nazwa.c -o nazwa
```

gdzie we fragmencie wy kropkowanym mogą być opcje `-D` definiujące stałe `POLA`, `WIERSTY`, `KOLUMNY` i `WYBOR`, `nazwa.c` to nazwa pliku z kodem źródłowym, a plik `opcje` ma zawartość:

```
-std=c17
-pedantic
-Wall
-Wextra
-Wformat-security
-Wduplicated-cond
-Wfloat-equal
-Wshadow
-Wconversion
-Wjump-misses-init
-Wlogical-not-parentheses
-Wnull-dereference
-Wvla
-Werror
-fstack-protector-strong
-fsanitize=undefined
-fno-sanitize-recover
-g
-fno-omit-frame-pointer
-O1
```

Opcje `-std=c17`, `-pedantic` wskazują, że kompilator ma dbać o zgodność kodu z aktualnym standardem języka C.

Dzięki opcjom `-Wall`, `-Wextra` kompilator zgłosi zauważone usterki.

Opcje `-Wformat-security`, `-Wduplicated-cond`, `-Wfloat-equal`, `-Wshadow`, `-Wconversion`, `-Wjump-misses-init`, `-Wlogical-not-parentheses`, `-Wnull-dereference` umożliwiają wykrywanie dodatkowych usterek.

Opcja `-Wvla` sprawia, że użycie tablic zmiennej długości jest uznawane za usterkę.

Opcja `-Werror` wskazuje, że kompilator ma uznać usterki za błędy.

Dzięki opcji `-fstack-protector-strong`, podczas wykonania programu zostaną wykryte niektóre błędne odwołania do pamięci na stosie.

Opcje `-fsanitize=undefined`, `-fno-sanitize-recover` umożliwiają wykrywanie operacji, które mają efekt nieokreślony.

Opcje `-g`, `-fno-omit-frame-pointer` poprawiają jakość komunikatów o błędach wykonania.

Opcja `-O1` włącza optymalizacje, co zwiększa prawdopodobieństwo ujawnienia się błędów.

Wymagane są wszystkie wymienione opcje kompilatora. Nie będą do nich dodawane żadne inne.

Zwracamy uwagę, że poszczególne wersje kompilatora `gcc` mogą się różnić sposobem obsługi tych samych opcji. Przed wysłaniem rozwiązania warto więc skompilować je i przetestować na `students` w sposób opisany powyżej.

- Podczas walidacji i testów, program `nazwa` jest uruchamiany pod kontrolą programu Valgrind poleceniem:

```
valgrind --leak-check=full -q --error-exitcode=1 ./nazwa
```

Jeśli Valgrind wykryje błąd, to nawet, gdyby wynik był prawidłowy, uznajemy, że program testu nie przeszedł.

Opcja `-q` powoduje, że jedyne informacje, wypisywanymi przez program Valgrind, są komunikaty o błędach.

Opcja `--leak-check=full` wskazuje Valgrindowi, że powinien, między innymi, szukać wycieków pamięci.

Opcja `--error-exitcode=1` określa kod wyjścia programu w przypadku, gdy Valgrind wykryje błąd.

- Przyjmujemy, że niezerowy wynik funkcji `main()` informuje o błędzie wykonania programu.

- Poprawność wyniku sprawdzamy, przekierowując na wejście programu zawartość pliku `.in` i porównując rezultat, za pomocą programu `diff`, z plikiem `.out`, np.:

```
< przyklad.in ./nazwa | diff - przyklad.out
```

Ocena poprawności wyniku jest binarna. Wynik uznajemy za poprawny, jeżeli program `diff` nie wskaże żadnej różnicy względem wyniku wzorcowego.

Uwagi i wskazówki

- Jako rozwiązanie należy wysłać plik tekstowy `.c` z kodem źródłowym w języku C.
- Wolno założyć, że dane są poprawne.
- Wolno założyć, że każdy wiersz danych, w tym ostatni, kończy się reprezentacją końca wiersza `'\n'`.


Należy zadbać, by warunek ten spełniał także wynik programu.

- Wolno założyć, że wartości stałych `POLA`, `WIERSZE`, `KOLUMNY`, jeśli zostały określone podczas kompilacji, są liczbami całkowitymi od `1` do `26`.
- Wolno założyć, że wartość stałej `WYBOR`, jeśli została określona podczas kompilacji, jest nieujemną liczbą całkowitą mieszczącą się w zakresie typu `int`.

Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Ocenione
Termin oddania	wtorek, 7 grudnia 2021, 20:00
Pozostały czas	Zadanie zostało złożone 3 dni 3 godzin przed terminem
Ostatnio modyfikowane	sobota, 4 grudnia 2021, 16:07

Przesyłane pliki

 [poziomka.c](#) 4 grudnia 2021, 16:07

Komentarz do przesłanego zadania

 [Komentarze \(0\)](#)

Informacja zwrotna