Wstęp do programowania, potok imperatywny (Info, I rok) 16/17, laboratorium

NAWIGACJA

Moja strona domowa
- Strona główna

Strony
Bieżący przedmiot
WPI.LAB.INFO.I.16/17
Uczestnicy
Odznaki
Główne składowe
Zadanie 0 (treningowe)
Zadanie 1
Zadanie 1: Poziomka
Zadanie 1: poprawność
Zadanie 2
Zadanie 3
Zadanie 4 (poprawkowe)
Moje kursy

ADMINISTRACJA

Administracja

Zadanie 1: Poziomka

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. Zakładamy, że `WIERSZE` i `KOLUMNY` określają rozmiar planszy z punktu widzenia Prawego. Kolumny będziemy nazywali wielkimi literami od `'A'` natomiast wiersze małymi literami od `'a'`.

Gracze wykonują ruchy naprzemian, zaczynając od Lewego. Przyjmujemy, że Lewy może przekazać prawo wykonania pierwszego ruchu Prawemu.

Gracz w jednym ruchu zajmuje spójny blok wolnych pól szerokości `POLA` i wysokości 1. Dalej będziemy to nazywali umieszczeniem bloku na planszy. Gracz w swoim ruchu umieszcza na planszy blok poziomy z jego punktu widzenia a pionowy dla jego przeciwnika. Grając na kartce papieru, gracze rysują poziomą dla siebie kreskę przez określoną liczbę pól.

Gracz nie mogący wykonać ruchu, gdy jest jego kolej, przegrywa a jego przeciwnik wygrywa grę.

Ocena planszy

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

Przyjmiemy, 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 mógłby umieścić jego przeciwnik.

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

```
 ABCDEFGHIJK|
a   #  #    |
b   #  #    |
c#### ##### |
d     #     |
e  #  #  ###|
f  #   ###  |
g  #        |
------------+
```

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

Polecenie

Napisz program grający w Poziomkę jako gracz Prawy.

Wykonanie programu zaczyna się od opcjonalnego przekazania prawa pierwszego ruchu Prawemu przez Lewego. Po tym program wczytuje z wejścia zapis kolejnych ruchów gracza Lewego oraz wypisuje na wyjście ruchy gracza Prawego. Kończy pracę, gdy nie może wykonać ruchu, czyli przegrał, lub gdy Lewy podda się kończąc podawanie swoich ruchów.

Spośród wszystkich możliwych ruchów program wybiera ten, po którym ocena planszy będzie dla niego maksymalna. Gdyby takich ruchów było wiele, wybiera losowo jeden z nich.

Parametry gry

Liczby `POLA`, `WIERSZE` i `KOLUMNY`, jak również opisana poniżej liczba `LOSOWE` parametryzująca generator liczb pseudolosowych, będą zdefiniowane za pomocą stałych symbolicznych.

Wartości stałych symbolicznych można określić podczas kompilacji programu przez `gcc` za pomocą opcji `-D`, np.

```
gcc -std=c89 -pedantic -Wall -Wextra -Werror -DPOLA=3 -DWIERSZE
=7 -DKOLUMNY=11 -DLOSOWE=123 zadanie1.c -o zadanie1
```

Na wypadek, gdyby nie zdefiniowano ich podczas kompilacji, dla stałych `POLA`, `WIERSZE` i `KOLUMNY` określamy też wartości domyślne, odpowiednio 5, 26 i 26. Można to zrobić, umieszczając w programie dyrektywy preprocesora

```
#ifndef POLA
# define POLA 5
#endif

#ifndef WIERSZE
# define WIERSZE 26
#endif

#ifndef KOLUMNY
# define KOLUMNY 26
#endif
```

Losowy wybór ruchu

Do losowego wyboru ruchu spośród najlepszych użyjemy standardowego generatora liczb pseudolosowych. Inicjujemy go stałą `LOSOWE` lub, jeśli nie zdefiniowano jej podczas kompilacji, funkcją `time()`. W tym celu na początku treści funkcji `main()` należy umieścić instrukcje

```
#ifdef LOSOWE
    srand(LOSOWE);
#else
    srand((unsigned) time(NULL));
#endif
```

Wybór ruchu przez gracza Prawego zaczynamy od wyznaczenia ciągu wszystkich ruchów maksymalizujących ocenę planszy, uporządkowanych w kolejności rosnących numerów wiersza a w ramach wiersza, w kolejności rosnących numerów kolumn. Zakładając, że długością tego ciągu jest `n`, numer wybranego ruchu, liczony od 0, będzie wartością wyrażenia `randof(n)`, gdzie funkcja `randof()` jest zdefiniowana następująco

```
int randof(int n) {
    assert(n > 0);
    return rand() / (RAND_MAX + 1.0) * n;
}
```

Postać danych

Jeżeli pierwszy wiersz na wejściu programu jest pusty, interpretujemy to jako przekazanie prawa wykonania pierwszego ruchu Prawemu przez Lewego.

W kolejnych wierszach wejścia jest zapis ruchów Lewego, po jednym w wierszu. Zapisem jego ruchu jest wielka litera i mała litera wskazujące, odpowiednio kolumnę, w której ma być blok oraz wiersz, w którym ma być pierwsze pole bloku.

Postać wyniku

Wynikiem programu jest ciąg wierszy z zapisem ruchów gracza Prawego, po jednym w wierszu. Zapisem jego ruchu jest 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 powinno być spacji ani żadnych innych znaków, które nie zostały wymienione powyżej. Każdy wypisywany wiersz, także ostatni, ma być zakończony reprezentacją końca wiersza `'\n'`.

Stan planszy po wczytaniu ruchów Lewego

```
Fc
Ce
Da
Ga
```

i wypisaniu ruchów Prawego

```
cH
eI
cA
fG
```

przedstawia diagram planszy zamieszczony powyżej, w opisie oceny planszy.

Przykłady

Poniższe przykłady są wynikiem działania na `students` programu skompilowanego poleceniem

```
gcc -std=c89 -pedantic -Wall -Wextra -Werror -DPOLA=3 -DWIERSZE
=7 -DKOLUMNY=8 -DLOSOWE=123 zadanie1.c -o zadanie1
```

Dla danych:

```
Cd
Ea
Ae
Bc
He
Aa
```

wynikiem programu powinno być:

```
bB
fE
cF
eD
bF
aB
```

Dla danych:

```
Fa
Ce
Gc
Eb
He
Ae
Ha
```

wynikiem programu powinno być:

```
bB
fE
cA
eD
aC
dA
gD
```

Dla danych:

```
Ed
Ha
Cc
Bc
```

wynikiem programu powinno być:

```
bB
eF
bE
fA
```

Uwagi i wskazówki

- Wolno założyć, że dane są poprawne.

  W przypadku programu pobierającego dane bezpośrednio od użytkownika założenie to jest nierealistyczne. Przyjmujemy je po to, by uprościć zadanie. Jeśli ktoś chce, może zaimplementować kontrolę poprawności danych. Jest ona jednak nieobowiązkowa i nie będzie miała wpływu ocenę rozwiązania.

- Wolno założyć, że wartości stałych `POLA`, `WIERSZE` i `KOLUMNY`, jeśli zostaną określone podczas kompilacji, będą liczbami całkowitymi od 1 do 26.

- Wolno założyć, że wartość stałej `LOSOWE`, jeśli zostanie określona podczas kompilacji, będzie dozwolonym argumentem funkcji `srand()`.

- Dane można wczytywać albo funkcją `scanf()` albo za pomocą funkcji `getchar()`, która wczytuje jeden znak. Jej wynikiem, typu `int`, jest wartość stałej `EOF` jeśli dane się skończyły lub pierwszy znak z wejścia

w przeciwnym przypadku. Do kompletu jest też funkcja `putchar()` wypisująca na wyjście znak przekazany jej jako argument.

- Program do testów będzie kompilowany poleceniem:

  ```
  gcc -std=c89 -pedantic -Wall -Wextra -Werror ... nazwa.c -o
  nazwa
  ```

  gdzie we fragmencie wykropkowanym mogą być opcje `-D` definiujące stałe `POLA`, `WIERSZE`, `KOLUMNY` i `LOSOWE`.

  Wszystkie wymienione opcje kompilatora są obowiązkowe i nie wolno dodawać do nich żadnych innych.

- Przyjmujemy, że wynik funkcji `main()` inny niż 0 informuje o błędzie wykonania programu.

- Do treści zadania dołączone są pliki z rozszerzeniem `.in` zawierające przykładowe dane i odpowiadające im pliki z rozszerzeniem `.out` z wynikami wzorcowymi.

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

  ```
  < przyklad1.in ./zadanie1 | diff - przyklad1.out
  ```

  Wynik uznamy za poprawny tylko, jeśli jest identyczny z wynikiem wzorcowym.

- Jeśli rozwiązanie zadania skompilujemy z domyślnymi wartościami `POLA`, `WIERSZE` i `KOLUMNY` do pliku `zadanie1` i wykonamy na `students`, jako jedno polecenie,

  ```
  f=rozgrywka.txt ; \
  mkfifo fifo ; \
  ( echo ; cat fifo ) | \
      stdbuf -o0 -- tr a-zA-V A-Z$(echo {v..a} | tr -d ' ') |
  \
      stdbuf -o0 -- ./zadanie1 | \
      stdbuf -o0 -- tr a-zA-V $(echo {Z..A} | tr -d ' ')a-v |
  \
      tee -a $f | \
      stdbuf -o0 -- ./zadanie1 | \
      tee -a $f \
  > fifo ; \
  rm fifo
  ```

  to powstanie plik `rozgrywka.txt` z zapisem gry w Poziomkę, w której za obu graczy ruchy będzie wykonywał komputer.

Odpowiedzi na pytania do treści

- Każdy wybór ruchu spośród `n` możliwych wymaga obliczenia `randof(n)`. Nawet gdy `n` jest równe 1 i z góry wiemy, jaki będzie

wynik, powinniśmy `randomf()` wywołać.

- Algorytmy generatora liczb pseudolosowych na poszczególnych platformach różnią się. Nie ma nawet gwarancji, że te same wartości będą generowane w różnych wersjach Linuxa.

  Wyniki rozwiązania wzorcowego podane w przykładach pochodzą ze `students`. W innych środowiskach, w szczególności pod Windows, ten sam program mógłby losować inne ruchy.

przyklad1.in
przyklad1.out
przyklad2.in
przyklad2.out
przyklad3.in
przyklad3.out

Status przesłanego zadania

Numer próby	To jest próba nr 1.
Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Ocenione
Termin oddania	środa, 30 listopad 2016, 10:00
Pozostały czas	Zadanie zostało złożone 11 godz. 47 min. przed terminem
Ostatnio modyfikowane	wtorek, 29 listopad 2016, 22:12
Przesyłane pliki	PIOTR_SZUBERSKI_ZADANIE1.c
Komentarz do przesłanego zadania	Komentarze (0)

Informacja zwrotna

Ocena	10,00 / 10,00
Ocenione dnia	środa, 7 grudzień 2016, 21:51
Ocenione przez	Artur Zaroda

Komentarz zwrotny	Ocena jakości: 10/10 pkt Znak '#' rozpoczynający dyrektywę preprocesora powinien być w pierwszej kolumnie tekstu. Bezparametrową funkcję ...