

# Wstęp do programowania, potok imperatywny (Info, I rok) 18/19, laboratorium

Kokpit ► Moje kursy ► WPI.LAB.INFO.I.18/19 ► Laboratorium 6 ► Palindrom

## Palindrom

### Wprowadzenie

Gra *Palindrom* jest rozgrywana za pomocą pionów na 64-polowej planszy podzielonej na kolumny, od **a** do **h**, oraz wiersze.

W grze bierze udział dwóch graczy, nazywanych Pierwszym i Drugim. Każdy gracz ma swój rodzaj piona.

Na początku gry plansza jest pusta. Zaczyna gracz Pierwszy.

Gracze, na przemian, wykonują ruch, kładąc swój pion na polu planszy. Pion jest umieszczany w wybranej przez gracza kolumnie, w jej pierwszym wolnym polu, licząc od dołu planszy.

Gracz wygrywa, jeśli swoim ruchem zbudował palindrom z pięciu pionów na kolejnych polach tego samego wiersza, kolumny lub przekątnej.

### Polecenie

Napisz program umożliwiający grę w Palindrom dwóm graczom.

Program, w pętli:

1. pisze diagram aktualnego stanu planszy i wskazuje, który gracz ma wykonać ruch;
2. próbuje wczytać polecenie gracza;
3. jeśli się nie udało, bo dane się skończyły, kończy pracę;
4. jeśli wczytał polecenie błędne, ignoruje je;
5. jeśli wczytał polecenie wykonania ruchu, wykonuje je;
6. jeśli wczytał polecenie przerywania gry, kończy pracę.

Pętla kończy się, gdy:

- skończą się dane, lub
- program dostanie polecenie przerywania gry, lub

- jeden z graczy wygra.

Jeśli jeden z graczy wygra, program, na zakończenie pracy, pisze diagram końcowego stanu planszy i informuje, kto wygrał.

## Postać danych

Na wejściu programu są polecenia graczy. Każde polecenie zajmuje jeden wiersz.

Jedynym znakiem w wierszu polecenia wykonania ruchu jest mała litera od `a` do `h`, będąca nazwą kolumny, w której gracz chce umieścić pion.

Polecenie jest poprawne, jeśli we wskazanej kolumnie jest wolne pole.

Jedynym znakiem w wierszu polecenia przzerwania gry jest `=`.

Wiersz, który nie jest poprawnym poleceniem wykonania ruchu lub przzerwania gry, jest poleceniem błędnym.

Wolno założyć, że każde polecenie użytkownika, także ostatnie, jest w wierszu poprawnie zakończonym reprezentacją końca wiersza `\n`.

## Postać wyniku

Na wyjściu programu jest ciąg diagramów planszy. Po każdym diagramie jest wiersz z informacją, kto ma wykonać ruch lub kto wygrał.

Diagram opisuje pole planszy za pomocą znaku

- `-` gdy pole jest puste;
- `1` gdy na polu jest pion gracza Pierwszego;
- `2` gdy na polu jest pion gracza Drugiego.

Przed każdym znakiem opisującym pole jest spacja.

Opisy pól są pogrupowane w wiersze, a w wierszu uporządkowane w kolejności kolumn od `a` do `h`.

Po ostatnim wierszu pól diagramu jest wiersz z nazwami kolumn. Każda z nich jest poprzedzona spacją.

Informacja o tym, kto ma wykonać ruch, ma postać wiersza z nazwą gracza, `1` lub `2`, po której jest dwukropek.

Informacja o tym, kto wygrał, ma postać wiersza z nazwą gracza, `1` lub `2`, po której jest kropka.

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

Każdy wypisywany wiersz, także ostatni, jest zakończony końcem wiersza `\n`.

## Przykłady

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 przykład3.in poprawny wynik to przykład3.out.

## Walidacja i testy

- Rozwiązania zostaną poddane walidacji, wstępnie sprawdzającej zgodność ze specyfikacją. Pomyślne przejście walidacji jest warunkiem dopuszczenia programu do testów poprawności.
- Walidacja i testy zostaną przeprowadzone na komputerze `students`.
- Programy będą kompilowane poleceniem:

```
gcc -std=c11 -pedantic -Wall -Wextra -Werror -fstack-protector-strong -g nazwa.c  
-o nazwa
```

gdzie `nazwa.c` to nazwa pliku z kodem źródłowym.

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

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

```
valgrind -q ./nazwa
```

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

- Przyjmujemy, że wynik funkcji `main()` inny niż `0` 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. Uznajemy go za poprawny, jeżeli program `diff` nie wypisze żadnej różnicy między wynikiem programu a wynikiem wzorcowym.

## Wskazówki

- Pod Linuxem, pracując z programem interakcyjnie na konsoli, koniec danych sygnalizujemy, naciskając klawisze `Ctrl - D`.
  - W przygotowaniu danych testowych może pomóc polecenie `tee`. Przesyła ono dane z wejścia na wyjście, jednocześnie zapisując ich kopię w pliku, którego nazwa jest argumentem polecenia.
- Wykonanie


```
tee test.in | ./nazwa
```

uruchomi program `nazwa` w trybie interakcyjnym, tworząc kopię danych testowych w pliku `test.in`. Dzięki temu test na tych samych danych będzie można powtórzyć, wykonując polecenie

```
< test.in ./nazwa > test.out
```

- 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.

## Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Nie ocenione
Termin oddania	Tuesday, 20 November 2018, 18:00 PM
Pozostały czas	Zadanie zostało złożone 5 godz. 5 min. przed terminem
Ostatnio modyfikowane	Tuesday, 20 November 2018, 12:54 PM
Przesyłane pliki	 pal.c
Komentarz do przesłanego zadania	► Komentarze (0)

◀ obrot.c

Przejdź do...

przyklad1.in ►

Jesteś zalogowany(a) jako Gevorg Chobanyan (Wyloguj)

WPI.LAB.INFO.I.18/19

Podsumowanie zasad przechowywania danych

Pobierz aplikację mobilną

Moodle, wersja 3.5.7+ (Build: 20190823) | moodle@mimuw.edu.pl