Warsztat programisty – Zadanie projektowe. Etap 3 – prezentacja ustawienia figur

Łukasz Kuszner, Radosław Ziemann

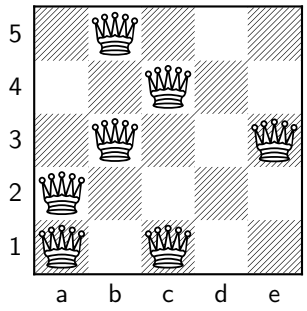
18 grudnia 2024

1 Zadanie

Ten etap polega głównie na prezentacji danych wczytanych w poprzednim etapie realizacji projektu z wykorzystaniem pakietów `chessboard` i `xskak` z systemu LaTeX.

1.1 Elementy obowiązkowe (20 pkt.)

- Spodziewane efekty z etapu 1: konfiguracja przy pomocy `cmake`, historia zmian w repozytorium `git`, uruchomienie skryptem w Bashu.
- Spodziewane efekty z etapu 2, w tym obsługa ewentualnych błędów.
- Kod programu w C znajduje się w przynajmniej dwóch różnych plikach.
- Każda konfiguracja odpowiada rozdziałowi (`\section`) w wygenerowanym dokumencie PDF.
- Każdy rozdział zawiera: szachownicę zgodnie z zaimplementowanymi opcjami (patrz niżej) oraz tabelę. Tabela zawiera: rozmiar szachownicy, łączną liczbę figur na szachownicy, oraz dla każdego wiersza i każdej kolumny szachownicy liczbę figur w danym wierszu i danej kolumnie. Przykładowy fragment tabeli opisujący liczbę figur w poszczególnych wierszach i kolumnach szachownicy może wyglądać tak:



linia	l.f.	kolumna	l.f.
5	1	a	2
4	1	b	2
3	2	c	2
2	1	d	0
1	2	e	1

Ale na pewno może wyglądać ładniej. Ułożenie danych w tabeli i szachownicy w obrębie rozdziału jest elementem wykonania projektu i zależy od wykonawcy.

- Z poziomu wywołania skryptu dostępna jest opcja `-C`, która sprawia, że w całym dokumencie wyświetlane są czarne figury zamiast białych (bez opcji).

1.2 Elementy nieobowiązkowe (maks. 20 pkt)

Poniższe elementy nie są obowiązkowe, to znaczy można zaliczyć projekt bez ich wykonania. Za każdy podpunkt prawidłowo zaimplementowany można otrzymać dodatkowe punkty, łącznie nie więcej niż 20 pkt.

- (+2 pkt) Projekt zawiera plik `README` z opisem programu, sposobem jego użycia i dostępnych opcji.
- (+2 pkt) Projekt korzysta ze stworzonej biblioteki innymi słowy część utworzonego kodu to biblioteka (ang. library) skonfigurowana w systemie `cmake`.
- (+2 pkt) Każda prezentowana szachownica opatrzona jest podpisem/tytułem (ang. caption). Tytuł powinien się różnić dla każdej konfiguracji (np. „Konfiguracja numer $< NR >$” lub według własnej inwencji).
- (+2 pkt) Dokument zawiera rozdział z podsumowaniem, w którym zaprezentowane są statystyki dotyczące konfiguracji w dokumencie dla każdego rozmiaru szachownicy, w tym: ich liczba, najmniejsza i największa liczba ustawionych figur i coś jeszcze według własnej inwencji.
- (+4 pkt) W tabeli dostępna jest informacja o liczbie pól, które nie są atakowane przez żadnego hetmana, a na diagramie przedstawiającym

szachownicę pola te są wyróżnione w wybrany sposób. Na przykład w czerwonej ramce albo wyróżnione kolorem.

- (+4 pkt.) Dostępne są dwa sposoby prezentacji danych w tabeli a wybór pomiędzy nimi odbywa się poprzez dodatkową opcję `-L` podawaną przy wywołaniu skryptu.

- (+4 pkt.) Z poziomu wywołania skryptu dostępna jest dodatkowa opcja `-T`, która powoduje, że w każdym rozdziale dokumentu oprócz tabeli generowany jest jeszcze opis słowny konfiguracji. Opis powinien się różnić w zależności od parametrów konfiguracji i nie zawierać liczb. To znaczy zamiast wygenerować tekst (przykład): „...6 figur ...” otrzymamy „...sześć figur...”. Wskazówka: Aby uniknąć problemów z kodowaniem znaków można zapisywać polskie znaki diakrytyczne w postaci zakodowanej (zobacz: kodowanie polskich znaków w LaTeXu). Na przykład zamiast „ę” możemy wypisać „`\k{e}`”. Wygenerowany tekst powinien być poprawny językowo i opisywać wszystkie elementy umieszczone w tabeli.

1.3 Elementy ponadobowiązkowe (bez limitu punktów)

Za realizację każdego elementu można uzyskać dodatkowe punkty, co pozwala uzyskać ponad 100% punktów przewidzianych za ten etap projektu.

- (+2 pkt) Dostępna jest opcja `-S`, której zastosowanie sprawia, że konfiguracje są prezentowane w dokumencie w takiej kolejności, że najpierw występują te z mniejszymi szachownicami a później te z większymi. W przypadku, gdy szachownice są takie same najpierw występują te z mniejszą liczbą figur. W przypadku takiego samego rozmiaru szachownicy i tej samej liczby figur kolejność jest zgodna z podaną w danych wejściowych.

  Wskazówka: należy wczytać i zapamiętać wszystkie konfiguracje. Informacje o konfiguracjach można dodatkowo zapamiętać w tablicy struktur, umieszczając tam dane potrzebne do sortowania: rozmiar szachownicy, liczba figur i numer kolejny na wejściu. Następnie wywołać funkcję `qsort` dla tej dodatkowej tablicy struktur.

- (+2 pkt) Dokument zawiera stronę tytułową, na której jest grafika generowana przynajmniej częściowo w trakcie działania programu. Kolejne wywołania programu powinny tworzyć różne grafiki.

Wskazówka: można do tego celu użyć wybranego pakietu LaTeXa lub programu do tworzenia grafiki, np `ImageMagick`.

opcja	wartość	status	skrócony opis
`-C`	0 pkt	obowiązkowa	wybór czarnych figur na szachownicy
`-L`	4 pkt	nie obowiązkowa	dodatkowe sposób prezentacji danych (ang. table layout)
`-T`	4 pkt	nie obowiązkowa	generowanie tekstu na podstawie danych w tabeli
`-S`	2 pkt	ponad obowiązkowa	sortowanie konfiguracji

Tabela 1: Podsumowanie opcji

skrócony opis	wartość	status
README	2 pkt	nie obowiązkowa
wykorzystanie biblioteki	2 pkt	nie obowiązkowa
Podpisy pod diagramami	2 pkt	nie obowiązkowa
Dane o nieatakowanych polach	4 pkt	nie obowiązkowa
podsumowanie	2 pkt	nie obowiązkowa
strona tytułowa	2 pkt	ponad obowiązkowa

Tabela 2: Podsumowanie funkcjonalności

2 Wskazówki i zalecenia implementacyjne

- Po zaprojektowaniu wyglądu tabeli warto najpierw ręcznie wpisać kod LaTeXa, który taką tabelę generuje dla wybranego szczególnego przypadku a dopiero w drugiej kolejności pisać program, który to robi ogólnie dla zadanych parametrów.
- Do obsługi opcji można wykorzystać `getopts` po stronie Basha oraz `getopt` po stronie programu w `C`.
- Warto podzielić zadanie na możliwie małe elementy.

- Warto zastanowić się czy nie dałoby się już napisanego i działającego kodu przepisać lepiej (jeśli się nie uda zawsze można wrócić do wersji poprzedniej – dlatego właśnie korzystamy z gita).
- Proszę pamiętać, że głównym celem wykonania projektu jest nauka wykorzystania poznanych narzędzi z warsztatu programisty.
- Praca nad projektem zajmie więcej niż 1 dzień roboczy – warto zaplanować sobie czas na jego realizację.
- **Uwaga:** Proszę uważnie testować. W momencie prezentacji rozwiązania elementy deklarowane jako wykonane powinny działać prawidłowo.