

# Indywidualny projekt programistyczny (Info, I rok) 17/18

Kokpit ► Kursy ► Archiwum ► Rok 2017/2018 ► Informatyka ►  
Informatyczne studia I stopnia ► I rok ► IPP.INFO.I.17/18 ► Temat 8 ►  
Zadanie telefony, część 2

## Zadanie telefony, część 2

Jako drugą część dużego zadania należy zaimplementować program, który, korzystając z modułu zaimplementowanego w części pierwszej, udostępnia operacje na numerach telefonów przez interfejs tekstowy. Ponadto należy zaimplementować skrypt w bashu.

## Interfejs tekstowy

Program czyta dane ze standardowego wejścia, wyniki wypisuje na standardowe wyjście, a informacje o błędach na standardowe wyjście diagnostyczne.

## Poprawne dane wejściowe

Dane wejściowe wyraża się w pewnym języku programowania. W języku tym są trzy rodzaje leksemów:

- `numer` – niepusty ciąg cyfr 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- `identyfikator` – niepusty ciąg składający się z małych lub dużych liter alfabetu angielskiego i cyfr dziesiętnych, zaczynający się od litery – jest nazwą bazy przekierowań;
- `operator`.

W języku tym są cztery operatory:

- `NEW`
- `DEL`
- `>`
- `?`

Słowa `NEW` i `DEL` są zastrzeżone – nie może być takich identyfikatorów.

Język udostępnia następujące operacje tworzenia, przełączania i usuwania bazy przekierowań:

- `NEW identyfikator` – jeśli baza przekierowań o podanej nazwie nie istnieje, to tworzy nową bazę o tej nazwie i ustawia ją jako aktualną, a jeśli baza o takiej nazwie już istnieje, to tylko ustawia ją jako aktualną;
- `DEL identyfikator` – usuwa bazę przekierowań o podanej nazwie;

Język udostępnia następujące operacje, dotyczące aktualnej bazy przekierowań, wykonywane na numerach:

- `numer > numer` – dodaje przekierowanie numerów;

- `numer ?` – wypisuje przekierowanie z podanego numeru;
- `? numer` – wypisuje przekierowania na podany numer;
- `DEL numer` – usuwa wszystkie przekierowania, których numer jest prefiksem.

Między leksemami może nie być odstępów albo może być dowolna liczba białych znaków (spacja, tabulator, znak nowej linii, znak powrotu karetki). Między leksemami musi być co najmniej jeden biały znak, jeśli jego brak powodowałby błędną interpretację.

W języku mogą pojawić się komentarze. Komentarz rozpoczyna i kończy się sekwencją `$$`.

Program powinien obsługiwać co najmniej 100 różnych baz przekierowań.

## Obsługa błędów składniowych

Powinna zostać wypisana jedna linia z komunikatem:

```
ERROR n
```

gdzie `n` to numer pierwszego znaku, który nie daje się zinterpretować jako poprawne wejście. Numer znaku jest to kolejny numer bajtu wczytany przez program, licząc od jedynek.

Jeśli dane wejściowe kończą się niespodziewanie, powinna zostać wypisana jedna linia z komunikatem:

```
ERROR EOF
```

## Obsługa błędów wykonania

Powinna zostać wypisana jedna linia z komunikatem:

```
ERROR operator n
```

gdzie `operator` to nazwa operatora, którego wykonanie spowodowało błąd, a `n` numer pierwszego znaku tego operatora. Numer znaku jest to kolejny numer bajtu wczytany przez program, licząc od jedynek.

Przy poprawnym wykonaniu operator `?` powinien wypisać co najmniej jeden numer. Brak numeru do wypisania należy traktować jako błąd.

Na początku nie ma ustawionej aktualnej bazy danych i wszelkie operacje na numerach należy traktować jako błędne.

## Zakończenia działania programu

Poprawne zakończenie programu po przetworzeniu wszystkich danych wejściowych powinno być sygnalizowane kodem wyjścia (ang. *exit code*) 0. Zakończenie programu z błędem powinno być sygnalizowane kodem wyjścia 1. W powyższym opisie stwierdzenie „powinna zostać wypisana jedna linia z komunikatem” oznacza, że po wypisaniu tej linii program kończy działanie z błędem. Niezależnie od tego, czy program zakończył się poprawnie czy z błędem, powinien zwolnić zaalokowaną pamięć.

## Skrypt

Wyniki działania operatora `?` jest po prostu wynikiem działania funkcji `phfwdGet` lub `phfwdReverse`. Niestety funkcja `phfwdReverse` nie wyznacza przeciwobrazu funkcji `phfwdGet`. Jeśli `phfwdGet(x) = y`, to `x` należy do wyniku `phfwdReverse(y)`. Implikacja w drugą stronę nie zachodzi. Należy napisać skrypt, który dla podanego numeru `y` wyznaczy wszystkie takie numery `x`, że `phfwdGet(x) = y`. Skrypt przyjmuje trzy parametry:

- pierwszy wskazuje (ścieżka i nazwa) na plik wykonywalny programu, który jest wyspecyfikowany w poprzednim punkcie tej części zadania;
- drugi wskazuje (ścieżka i nazwa) na plik z operacjami przekierowania numerów, zawiera ciąg użyć operatora `>`;
- trzeci to numer `y`.

## Dostarczamy

Rozwiązanie drugiej części zadania powinno korzystać z własnego rozwiązania części pierwszej.

## Wymagamy

Jako rozwiązanie części 2 zadania wymagamy:

- umieszczenia kodu źródłowego implementacji w katalogu `src`,
- uzupełnienia dokumentacji w formacie `doxygen` tak, aby była przydatna dla programistów rozwijających program,
- dostosowania pliku konfiguracyjnego dla programu `cmake`,
- stworzenia skryptu o nazwie `phone_forward.sh` i umieszczenia go w głównym katalogu rozwiązania.

Gotowe rozwiązanie powinno się kompilować w dwóch wersjach: release i debug, jak to opisano w pierwszej części zadania.

## Oddawanie rozwiązania

Rozwiązanie należy oddawać, podobnie jak części 1, przez repozytorium git. W repozytorium mają się znaleźć wszystkie pliki niezbędne do zbudowania pliku wykonywalnego i dokumentacji oraz skrypt. *W repozytorium nie wolno umieszczać plików binarnych ani tymczasowych.* W Moodle jako rozwiązanie należy umieścić tekst zawierający identyfikator finalnej wersji rozwiązania, na przykład:

```
Finalna wersja mojego rozwiązania części 2 zadania telefony znajduje się
w repozytorium w wersji 518507a7e9ea50e099b33cb6ca3d3141bc1d6638.
```

Rozwiązanie należy zatwierdzić (`git commit`) i wysłać do repozytorium (`git push`) najpóźniej do godz. 12.00, 2 czerwca 2018 r.


## Punktacja

Za w pełni poprawne rozwiązanie zadania implementujące wszystkie wymagane funkcjonalności można zdobyć maksymalnie 20 punktów. Od tej oceny będą odejmowane punkty za niższe uchybienia:

- Za problemy ze skompilowaniem rozwiązania można stracić wszystkie punkty.
- Za każdy test, którego implementacja nie przejdzie, traci się 1 punkt.
- Za złe działanie skryptu można stracić do 5 punktów.
- Za problemy z zarządzaniem pamięcią można stracić do 6 punktów.

- Za niezgodną ze specyfikacją strukturę plików w rozwiązaniu, niezgodne ze specyfikacją nazwy plików w rozwiązaniu lub umieszczenie w repozytorium niepotrzebnych albo tymczasowych plików można stracić do 4 punktów.
- Za złą jakość kodu źródłowego lub błędy w stylu kodowania można stracić do 4 punktów.
- Za braki w dokumentacji można stracić do 2 punktów.
- Za ostrzeżenia wypisywane przez kompilator można stracić do 2 punktów.

**Rozwiązania należy implementować samodzielnie pod rygorem niezaliczenia przedmiotu.**

 telefony\_testy\_2.zip

◀ Doxygen

Przejdź do...



Testy jednostkowe ▶

## ADMINISTRACJA



Administracja kursem

Jesteś zalogowany(a) jako Franciszek Budrowski (Wyloguj)

IPP.INFO.I.17/18

Podsumowanie zasad przechowywania danych

Pobierz aplikację mobilną

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