

Indywidualny projekt programistyczny (Info, I rok) 18/19

Kokpit ► Moje kursy ► IPP.INFO.I.18/19 ► Laboratorium 5 ► Duże zadanie, część 1

Duże zadanie, część 1

Zadanie drogi, część 1

Tegoroczne duże zadanie polega na zaimplementowaniu obsługi map dróg krajowych. Na potrzeby tego zadania przyjmujemy następujące definicje.

Mapa dróg jest zbiorem miast (ang. city) połączonych odcinkami dróg (ang. road).

Miasto reprezentowane jest przez jego nazwę, która jest niepustym napisem w stylu C niezawierającym kodów od 0 do 31 ani średnika i zakończonym zerem.

Każdy odcinek drogi łączy dwa różne miasta. Między parą miast może być co najwyżej jeden odcinek drogi. Odcinek drogi ma dwa atrybuty: długość, która jest dodatnią liczbą całkowitą i rok, który jest rokiem budowy lub rokiem ostatniego remontu i jest liczbą całkowitą. Wartości dodatnie reprezentują lata n.e., a wartości ujemne to lata p.n.e. Uwaga: nie było roku 0.

Droga jest to ciąg odcinków drogowych łączących dwa różne miasta, bez przerw, samoprzecięć i pętli.

Droga krajowa (ang. route) jest to droga. Jeden odcinek drogi może należeć do wielu dróg krajowych. Droga krajowa identyfikowana jest przez jej numer, który jest liczbą całkowitą z przedziału od 1 do 999. Długość drogi krajowej to suma długości jej odcinków.

Jako pierwszą część zadania należy zaimplementować moduł operacji na mapach drogowych. Opis interfejsu modułu znajduje się w pliku `src/map.h` w formacie komentarzy dla programu `doxygen`. Przykład użycia znajduje się w pliku `src/map_main.c`.

Dostarczamy

W repozytorium `https://git.mimuw.edu.pl/IPP-login.git` (gdzie login to identyfikator używany do logowania się w laboratorium komputerowym) znajduje się szablon implementacji rozwiązania tego zadania. Znajdują się tam następujące pliki:

- `src/map.h` – deklaracja interfejsu modułu wraz z jego dokumentacją w formacie `doxygen`,
- `src/map_main.c` – przykład użycia modułu,
- `CMakeLists.txt` – plik konfiguracyjny programu `cmake`,
- `Doxyfile.in` – plik konfiguracyjny programu `doxygen`,
- `MainPage.dox` – strona główna dokumentacji w formacie `doxygen`.

Zastrzegamy sobie możliwość nanoszenia poprawek do tego szablonu. Będziemy je umieszczać w gałęzi `template/part1`. Lista poprawek:

- na razie nie ma żadnych poprawek.

Wymagamy

Jako rozwiązanie części 1 zadania wymagamy:

- uzupełnienia implementacji w pliku `src/map.h`,
- stworzenia pliku `src/map.c` z implementacją modułu,
- uzupełnienia dokumentacji w formacie `doxygen` tak, aby była przydatna dla programistów rozwijających moduł.

Powinna być możliwość skompilowania rozwiązania w dwóch wersjach: release i debug. Wersję release kompiluje się za pomocą sekwencji poleceń:

```
mkdir release
cd release
cmake ..
make
make doc
```

Wersję debug kompiluje się za pomocą sekwencji poleceń:

```
mkdir debug
cd debug
cmake -D CMAKE_BUILD_TYPE=Debug ..
make
make doc
```

W wyniku kompilacji odpowiednio w katalogu `release` lub `debug` powinien powstać plik wykonywalny `map` oraz dokumentacja. W poleceniu `cmake` powinno być również możliwe jawne określenie wariantu release budowania pliku wynikowego:

```
cmake -D CMAKE_BUILD_TYPE=Release ..
```

Zawartość dostarczonych przez nas plików można modyfikować, o ile nie zmienia to interfejsu modułu i zachowuje wymagania podane w treści zadania, przy czym nie wolno usuwać opcji kompilacji: `-std=c11 -Wall -Wextra`. Zmiany mogą dotyczyć np. stylu, dokumentacji, deklaracji typedef, włączania plików nagłówkowych, implementacji funkcji jako `static inline`. Inne pliki źródłowe będące częścią rozwiązania można umieścić w katalogu `src`. Funkcja `main` programu musi znajdować się w pliku `src/map_main.c`, ale zawartość tego pliku nie będzie oceniana w tej części zadania.

Oddawanie rozwiązania

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

```
518507a7e9ea50e099b33cb6ca3d3141bc1d6638
```

Rozwiązanie należy zatwierdzić (`git commit`) i wysłać do repozytorium (`git push`) przed terminem podanym w Moodle.

Indywidualny projekt programistyczny (Info, I rok) 18/19

Kokpit ► Moje kursy ► IPP.INFO.I.18/19 ► Laboratorium 7 ► Duże zadanie, część 2

Duże zadanie, część 2

Zadanie drogi, część 2

Jako drugą część dużego zadania należy zaimplementować program, który, korzystając z modułu zaimplementowanego w części pierwszej, udostępnia operacje na mapie dróg. Program obsługuje tylko jedną mapę dróg. 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.

Dane wejściowe

Program akceptuje następujące polecenia.

```
numer drogi krajowej;nazwa miasta;długość odcinka drogi;rok budowy lub ostatniego rer
```

Format polecenia jest taki sam, jak w wyniku funkcji `getRouteDescription`. To polecenie tworzy drogę krajową o podanym numerze i przebiegu. Jeśli jakieś miasto lub odcinek drogi nie istnieje, to go tworzy. Jeśli odcinek drogi już istnieje, ale ma wcześniejszy rok budowy lub ostatniego remontu, to modyfikuje ten atrybut odcinka drogi. Za błąd uznajemy, jeśli odcinek drogi już istnieje, ale ma inną długość albo późniejszy rok budowy lub ostatniego remontu. To polecenie niczego nie wypisuje na standardowe wyjście.

```
addRoad;city1;city2;length;builtYear
```

Wywołuje na mapie dróg funkcję `addRoad` z podanymi parametrami. Niczego nie wypisuje na standardowe wyjście.

```
repairRoad;city1;city2;repairYear
```

Wywołuje na mapie dróg funkcję `repairRoad` z podanymi parametrami. Niczego nie wypisuje na standardowe wyjście.

```
getRouteDescription;routeId
```

Wywołuje na mapie dróg funkcję `getRouteDescription` z podanym parametrem. Jeśli wynik działania tej funkcji jest inny niż NULL, to wypisuje na standardowe wyjście jedną linię z wynikiem działania tej funkcji.

Każde polecenie znajduje się w osobnej linii. Puste linie i linie zaczynające się znakiem `#` należy ignorować. W poleceniach nazwa miasta jest niepustym napisem niezawierającym kodów od 0 do 31 ani średnika, liczby są zapisywane przy podstawie 10. Spacje w nazwie miasta są istotne.

Obsługa błędów

Jeśli polecenie jest niepoprawne składniowo lub jego wykonanie zakończyło się błędem, czyli odpowiednia funkcja zakończyła się wynikiem `false` lub `NULL`, to wypisuje na standardowe wyjście diagnostyczne jednoliniowy komunikat

```
ERROR n
```

gdzie `n` jest numerem linii w danych wejściowych zawierającym to polecenie. Linie numerujemy od jedynki i uwzględniamy ignorowane linie.

Zakończenie działania programu

Program po przetworzeniu wszystkich danych wejściowych powinien zakończyć się kodem wyjścia 0 (ang. *exit code*) i powinien zwolnić całą zaalokowaną pamięć.

Skrypt

Należy napisać skrypt, którego pierwszy parametr wskazuje (nazwa poprzedzona opcjonalnie ścieżką) na plik z wynikami działania funkcji `getRouteDescription`, każdy wynik w osobnej linii. Kolejne parametry (przynajmniej jeden) to numery dróg krajowych.

Skrypt dla każdego podanego w argumentach numeru drogi krajowej po kolei szuka w podanym pliku informacji o tej drodze krajowej, wylicza jej długość i wypisuje jedną linię w formacie:

```
numer drogi krajowej;długość
```

Jeśli w pliku nie ma informacji o żądanej drodze krajowej, to skrypt nic dla niej nie wypisuje.

Zakładamy, że zawartość pliku wskazanego pierwszym parametrem skryptu jest poprawna. W szczególności dla danego numeru drogi krajowej jest w nim co najwyżej jedna o niej informacja. Natomiast skrypt powinien jak najdokładniej sprawdzać poprawność parametrów i jeśli jest ich za mało lub któryś z nich jest niepoprawny, to powinien zakończyć się kodem wyjścia 1 (ang. *exit code*). Jeśli parametry są poprawne, skrypt powinien zakończyć się kodem wyjścia 0.

Dostarczamy

Nie dostarczamy żadnego kodu źródłowego. Rozwiązanie drugiej części zadania powinno korzystać z własnego, ewentualnie samodzielnie zmodyfikowanego, 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 `map.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.

Indywidualny projekt programistyczny (Info, I rok) 18/19

Kokpit ► Moje kursy ► IPP.INFO.I.18/19 ► Laboratorium 8 ► Duże zadanie, część 3

Duże zadanie, część 3

Zadanie drogi, część 3

W trzeciej części zadania oczekujemy poprawienia ewentualnych błędów z poprzednich części oraz zmodyfikowania programu. Skrypt z części drugiej zadania nie podlega poprawie, ale ma znajdować się w rozwiązaniu. Obowiązują ustalenia z treści poprzednich części zadania i z forum dyskusyjnego dla studentów.

Modyfikacja modułu operacji na mapie dróg

Moduł `map` należy uzupełnić o następującą funkcję.

```
bool removeRoute(Map *map, unsigned routeId);
```

Usuwa z mapy dróg drogę krajową o podanym numerze, jeśli taka istnieje, dając wynik `true`, a w przeciwnym przypadku, tzn. gdy podana droga krajowa nie istnieje lub podany numer jest niepoprawny, niczego nie zmienia w mapie dróg, dając wynik `false`. Nie usuwa odcinków dróg ani miast.

Modyfikacja interfejsu tekstowego

Należy dodać obsługę następujących, brakujących poleceń.

```
newRoute;routeId;city1;city2
```

Wywołuje na mapie dróg funkcję `newRoute` z podanymi parametrami. Niczego nie wypisuje na standardowe wyjście.

```
extendRoute;routeId;city
```

Wywołuje na mapie dróg funkcję `extendRoute` z podanymi parametrami. Niczego nie wypisuje na standardowe wyjście.

```
removeRoad;city1;city2
```

Wywołuje na mapie dróg funkcję `removeRoad` z podanymi parametrami. Niczego nie wypisuje na standardowe wyjście.

```
removeRoute;routeId
```

Wywołuje na mapie dróg funkcję `removeRoute` z podanym parametrem. Niczego nie wypisuje na standardowe wyjście.

Dokumentacja

Cały kod należy udokumentować w formacie `doxygen`.

Dostarczamy

Rozwiązanie części 3 zadania powinno korzystać z własnego rozwiązania poprzednich jego części.

Wymagamy

Jako rozwiązanie części 3 zadania wymagamy:

- zachowania lub poprawienia struktury plików z poprzednich części,
- zmodyfikowania plików `src/map.h`, `src/map.c` i ewentualnie innych plików źródłowych z poprzednich części rozwiązania,
- zmodyfikowania implementacji interfejsu tekstowego,
- uzupełnienia pliku konfiguracyjnego dla programu `cmake`,
- uzupełnienia dokumentacji w formacie `doxygen` tak, aby była przydatna dla programistów rozwijających program,

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

UWAGA 1: funkcja `main` programu musi znajdować się w pliku `src/map_main.c`.

UWAGA 2: w wyniku kompilacji powinien powstać plik wykonywalny `map`.

Oddawanie rozwiązania

Rozwiązanie należy oddawać, podobnie jak części 1 i 2, przez repozytorium git. W repozytorium mają się znaleźć wszystkie pliki niezbędne do zbudowania plików wykonywalnych i dokumentacji, i tylko te pliki. *W repozytorium nie wolno umieszczać plików binarnych ani tymczasowych.* W Moodle jako rozwiązanie należy umieścić tekst zawierający identyfikator commitu finalnej wersji rozwiązania, na przykład:

```
518507a7e9ea50e099b33cb6ca3d3141bc1d6638
```

Rozwiązanie należy zatwierdzić (`git commit`) i wysłać do repozytorium (`git push`) przed terminem podanym w Moodle.


Punktacja

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

- Za problemy ze skompilowaniem rozwiązania można stracić wszystkie punkty.
- Za każdy test, którego program nie przejdzie, traci się do 1 punktu.
- 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, brzydki styl kodowania można stracić do 4 punktów.
- Za ostrzeżenia wypisywane przez kompilator można stracić do 2 punktów.

- Za braki w dokumentacji można stracić do 2 punktów.

Rozwiązania należy implementować samodzielnie pod rygorem niezaliczenia przedmiotu.

 map_tests_3.zip

Status przesłanego zadania

Status przesłanego zadania	Nie próbowano
Stan oceniania	Nie ocenione
Termin oddania	czwartek, 13 czerwiec 2019, 20:00
Pozostały czas	Opóźnienie w przesłaniu: 49 dni 16 godz.
Ostatnio modyfikowane	-
Komentarz do przesłanego zadania	► Komentarze (0)

◀ cmocka_example.zip

Przejdź do...

Zadanie poprawkowe małe ►

Jesteś zalogowany(a) jako Gevorg Chobanyan (Wyloguj)

IPP.INFO.I.18/19

Moodle, wersja 3.5.1+ (Build: 20180824) | moodle@mimuw.edu.pl

```
bool extendRoute ( Map *      map,
                  unsigned    routeld,
                  const char * city
                )
```

Wydłuża drogę krajową do podanego miasta.

Dodaje do drogi krajowej nowe odcinki dróg do podanego miasta w taki sposób, aby nowy fragment drogi krajowej był najkrótszy. Jeśli jest więcej niż jeden sposób takiego wydłużenia, to dla każdego wariantu wyznacza wśród dodawanych odcinków dróg ten, który był najdawniej wybudowany lub remontowany i wybiera wariant z odcinkiem, który jest najmłodszy.

Parametry

- [in, out] **map** – wskaźnik na strukturę przechowującą mapę dróg;
- [in] **routeld** – numer drogi krajowej;
- [in] **city** – wskaźnik na napis reprezentujący nazwę miasta.

Zwraca

Wartość `true`, jeśli droga krajowa została wydłużona. Wartość `false`, jeśli wystąpił błąd: któryś z parametrów ma niepoprawną nazwę, nie istnieje droga krajowa o podanym numerze, nie ma miasta o podanej nazwie, przez podane miasto już przechodzi droga krajowa o podanym numerze, podana droga krajowa kończy się w podanym mieście, nie można jednoznacznie wyznaczyć nowego fragmentu drogi krajowej lub nie udało się zaalokować pamięci.

```
char const* getRouteDescription ( Map *      map,
                                  unsigned    routeld
                                )
```

Udostępnia informacje o drodze krajowej.

Zwraca wskaźnik na napis, który zawiera informacje o drodze krajowej. Alokuje pamięć na ten napis. Zwraca pusty napis, jeśli nie istnieje droga krajowa o podanym numerze. Zaalokowaną pamięć trzeba zwolnić za pomocą funkcji `free`. Informacje wypisywane są w formacie: numer drogi krajowej;nazwa miasta;długość odcinka drogi;rok budowy lub ostatniego remontu;nazwa miasta;długość odcinka drogi;rok budowy lub ostatniego remontu;nazwa miasta;...;nazwa miasta. Kolejność miast na liście jest taka, aby miasta `city1` i `city2`, podane w wywołaniu funkcji `newRoute`, które utworzyło tę drogę krajową, zostały wypisane w tej kolejności.

Parametry

- [in, out] **map** – wskaźnik na strukturę przechowującą mapę dróg;
- [in] **routeld** – numer drogi krajowej.

Zwraca

Wskaźnik na napis lub `NULL`, gdy nie udało się zaalokować pamięci.


```
bool newRoute ( Map *      map,
                unsigned    routeld,
                const char * city1,
                const char * city2
              )
```

Łączy dwa różne miasta drogą krajową.

Tworzy drogę krajową pomiędzy dwoma miastami i nadaje jej podany numer. Wśród istniejących odcinków dróg wyszukuje najkrótszą drogę. Jeśli jest więcej niż jeden sposób takiego wyboru, to dla każdego wariantu wyznacza wśród wybranych w nim odcinków dróg ten, który był najdawniej wybudowany lub remontowany i wybiera wariant z odcinkiem, który jest najmłodszy.

Parametry

- [in, out] **map** – wskaźnik na strukturę przechowującą mapę dróg;
- [in] **routeld** – numer drogi krajowej;
- [in] **city1** – wskaźnik na napis reprezentujący nazwę miasta;
- [in] **city2** – wskaźnik na napis reprezentujący nazwę miasta.

Zwraca

Wartość `true`, jeśli droga krajowa została utworzona. Wartość `false`, jeśli wystąpił błąd: któryś z parametrów ma niepoprawną wartość, istnieje już droga krajowa o podanym numerze, któreś z podanych miast nie istnieje, obie podane nazwy miast są identyczne, nie można jednoznacznie wyznaczyć drogi krajowej między podanymi miastami lub nie udało się zaalokować pamięci.

```
bool removeRoad ( Map *      map,
                  const char * city1,
                  const char * city2
                )
```

Usuwa odcinek drogi między dwoma różnymi miastami.

Usuwa odcinek drogi między dwoma miastami. Jeśli usunięcie tego odcinka drogi powoduje przerwanie ciągu jakiejś drogi krajowej, to uzupełnia ją istniejącymi odcinkami dróg w taki sposób, aby była najkrótsza. Jeśli jest więcej niż jeden sposób takiego uzupełnienia, to dla każdego wariantu wyznacza wśród dodawanych odcinków drogi ten, który był najdawniej wybudowany lub remontowany i wybiera wariant z odcinkiem, który jest najmłodszy.

Parametry

[in, out] **map** – wskaźnik na strukturę przechowującą mapę dróg;

[in] **city1** – wskaźnik na napis reprezentujący nazwę miasta;

[in] **city2** – wskaźnik na napis reprezentujący nazwę miasta.

Zwraca

Wartość `true`, jeśli odcinek drogi został usunięty. Wartość `false`, jeśli z powodu błędu nie można usunąć tego odcinka drogi: któryś z parametrów ma niepoprawną wartość, nie ma któregoś z podanych miast, nie istnieje droga między podanymi miastami, nie da się jednoznacznie uzupełnić przerwanego ciągu drogi krajowej lub nie udało się zaalokować pamięci.

```
bool repairRoad ( Map *      map,
                  const char * city1,
                  const char * city2,
                  int         repairYear
                )
```

Modyfikuje rok ostatniego remontu odcinka drogi.

Dla odcinka drogi między dwoma miastami zmienia rok jego ostatniego remontu lub ustawia ten rok, jeśli odcinek nie był jeszcze remontowany.

Parametry

- [in, out] **map** – wskaźnik na strukturę przechowującą mapę dróg;
- [in] **city1** – wskaźnik na napis reprezentujący nazwę miasta;
- [in] **city2** – wskaźnik na napis reprezentujący nazwę miasta;
- [in] **repairYear** – rok ostatniego remontu odcinka drogi.

Zwraca

Wartość `true`, jeśli modyfikacja się powiodła. Wartość `false`, jeśli wystąpił błąd: któryś z parametrów ma niepoprawną wartość, któreś z podanych miast nie istnieje, nie ma odcinka drogi między podanymi miastami, podany rok jest wcześniejszy niż zapisany dla tego odcinka drogi rok budowy lub ostatniego remontu.