Zadanie 1

Otwarto: czwartek, 30 marca 2023, 00:00
Wymagane do: wtorek, 25 kwietnia 2023, 23:59

Zbiory równoważnych ciągów

Zadanie polega na zaimplementowaniu w języku C dynamicznie ładowanej biblioteki obsługującej zbiory ciągów z relacją równoważności. Elementami zbiorów są niepuste ciągi, których elementami są liczby `0`, `1` i `2`. W implementacji ciąg reprezentujemy jako napis. Na przykład ciąg `{0, 1, 2}` reprezentujemy jako napis `"012"`. Klasom abstrakcji można nadawać nazwy.

Interfejs biblioteki

Funkcje oraz nazwę typu, które ma udostępniać biblioteka, są zadeklarowane w załączonym do treści zadania pliku `seq.h`. Znajdują się tam poniżej opisane deklaracje. Poprawna reprezentacja ciągu jest niepustym napisem składającym się ze znaków `0`, `1` lub `2` i jest zakończona terminalnym zerem. Poprawna nazwa klasy abstrakcji jest niepustym napisem zakończonym terminalnym zerem. Dodatkowe szczegóły działania biblioteki, w szczególności informacje, co jest niepoprawnym parametrem, należy wywnioskować z załączonego do treści zadania pliku `seq_example.c`, który jest integralną częścią specyfikacji. Używane poniżej określenie, że zbiór ciągów się nie zmienił, oznacza, że nie zmienił się obserwowalny stan zbioru ciągów.

```
typedef struct seq seq_t;
```

Jest to nazwa typu strukturalnego reprezentującego zbiór ciągów z relacją równoważności. Typ ten trzeba zdefiniować (zaimplementować) w ramach tego zadania.

```
seq_t * seq_new(void);
```

Funkcja `seq_new` tworzy nowy pusty zbiór ciągów.

Wynik funkcji:

- wskaźnik na strukturę reprezentującą zbiór ciągów lub
- `NULL` – jeśli wystąpił błąd alokowania pamięci; funkcja ustawia wtedy `errno` na `ENOMEM`.

```
void seq_delete(seq_t *p);
```

Funkcja `seq_delete` usuwa zbiór ciągów i zwalnia całą używaną przez niego pamięć. Nic nie robi, jeśli zostanie wywołana ze wskaźnikiem `NULL`. Po wykonaniu tej funkcji przekazany jej wskaźnik staje się nieważny.

Parametr funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów.

```
int seq_add(seq_t *p, char const *s);
```

Funkcja `seq_add` dodaje do zbioru ciągów podany ciąg i wszystkie niepuste podciągi będące jego prefiksem.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s` – wskaźnik na napis reprezentujący niepusty ciąg.

Wynik funkcji:

- `1` – jeśli co najmniej jeden nowy ciąg został dodany do zbioru;
- `0` – jeśli zbiór ciągów się nie zmienił;

?

- `-1` – jeśli któryś z parametrów jest niepoprawny lub wystąpił błąd alokowania pamięci; funkcja ustawia wtedy `errno` odpowiednio na `EINVAL` lub `ENOMEM`.

```
int seq_remove(seq_t *p, char const *s);
```

Funkcja `seq_remove` usuwa ze zbioru ciągów podany ciąg i wszystkie ciągi, których jest on prefiksem.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s` – wskaźnik na napis reprezentujący niepusty ciąg.

Wynik funkcji:

- `1` – jeśli co najmniej jeden ciąg został usunięty ze zbioru;
- `0` – jeśli zbiór ciągów się nie zmienił;
- `-1` – jeśli któryś z parametrów jest niepoprawny; funkcja ustawia wtedy `errno` na `EINVAL`.

```
int seq_valid(seq_t *p, char const *s);
```

Funkcja `seq_valid` sprawdza, czy podany ciąg należy do zbioru ciągów.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s` – wskaźnik na napis reprezentujący niepusty ciąg.

Wynik funkcji:

- `1` – jeśli ciąg należy do zbioru ciągów;
- `0` – jeśli ciąg nie należy do zbioru ciągów;
- `-1` – jeśli któryś z parametrów jest niepoprawny; funkcja ustawia wtedy `errno` na `EINVAL`.

```
int seq_set_name(seq_t *p, char const *s, char const *n);
```

Funkcja `seq_set_name` ustawia lub zmienia nazwę klasy abstrakcji, do której należy podany ciąg. Podaną nazwę należy skopiować, gdyż napis wskazywany przez wskaźnik `n` może przestać istnieć po zakończeniu tej funkcji.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s` – wskaźnik na napis reprezentujący niepusty ciąg;
- `n` – wskaźnik na napis z nową niepustą nazwą.

Wynik funkcji:

- `1` – jeśli nazwa klasy abstrakcji została przypisana lub zmieniona;
- `0` – jeśli ciąg nie należy do zbioru ciągów lub nazwa klasy abstrakcji nie została zmieniona;
- `-1` – jeśli któryś z parametrów jest niepoprawny lub wystąpił błąd alokowania pamięci; funkcja ustawia wtedy `errno` odpowiednio na `EINVAL` lub `ENOMEM`.

```
char const * seq_get_name(seq_t *p, char const *s);
```

Funkcja `seq_get_name` daje wskaźnik na napis zawierający nazwę klasy abstrakcji, do której należy podany ciąg. Nie wolno modyfikować pamięci wskazywanej przez ten wskaźnik. Wskaźnik ten może zostać unieważniony po jakiejkolwiek zmianie w strukturze zbioru ciągów.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s` – wskaźnik na napis reprezentujący niepusty ciąg.

Wynik funkcji:

- wskaźnik na napis zawierający nazwę lub
- `NULL`– jeśli ciąg nie należy do zbioru ciągów lub klasa abstrakcji zawierająca ten ciąg nie ma przypisanej nazwy; funkcja ustawia wtedy `errno` na `0`.
- `NULL`– jeśli któryś z parametrów jest niepoprawny; funkcja ustawia wtedy `errno` na `EINVAL`.

```
int seq_equiv(seq_t *p, char const *s1, char const *s2);
```

Funkcja `seq_equiv` łączy w jedną klasę abstrakcji klasy abstrakcji reprezentowane przez podane ciągi. Jeśli obie klasy abstrakcji nie mają przypisanej nazwy, to nowa klasa abstrakcji też nie ma przypisanej nazwy. Jeśli dokładnie jedna z klas abstrakcji ma przypisaną nazwę, to nowa klasa abstrakcji dostaje tę nazwę. Jeśli obie klasy abstrakcji mają przypisane różne nazwy, to nazwa nowej klasy abstrakcji powstaje przez połączenie tych nazw. Jeśli obie klasy abstrakcji mają przypisane taką same nazwę, to ta nazwa pozostaje nazwą nowej klasy abstrakcji.

Parametry funkcji:

- `p` – wskaźnik na strukturę reprezentującą zbiór ciągów;
- `s1` – wskaźnik na napis reprezentujący niepusty ciąg;
- `s2` – wskaźnik na napis reprezentujący niepusty ciąg.

Wynik funkcji:

- `1` – jeśli powstała nowa klasa abstrakcji;
- `0` – jeśli nie powstała nowa klasa abstrakcji, bo podane ciągi należą już do tej samej klasy abstrakcji lub któryś z nich nie należy do zbioru ciągów;
- `-1` – jeśli któryś z parametrów jest niepoprawny lub wystąpił błąd alokowania pamięci; funkcja ustawia wtedy `errno` odpowiednio na `EINVAL` lub `ENOMEM`.

Uwaga: napisy reprezentujące ciągi mogą przestać istnieć po zakończeniu funkcji.

Wymagania

Jako rozwiązanie zadania należy wstawić w Moodle archiwum zawierające plik `seq.c` oraz opcjonalnie inne pliki `*.h` i `*.c` z implementacją biblioteki, oraz plik `makefile` lub `Makefile`. Archiwum nie powinno zawierać innych plików ani podkatalogów, w szczególności nie powinno zawierać plików binarnych. Archiwum powinno być skompresowane programem `zip`, `7z` lub `rar`, lub parą programów `tar` i `gzip`. Po rozpakowaniu archiwum wszystkie pliki powinny się znaleźć w bieżącym katalogu.

Dostarczony w rozwiązaniu plik `makefile` lub `Makefile` powinien zawierać cel `libseq.so`, tak aby po wywołaniu polecenia `make libseq.so` zostało uruchomione kompilowanie biblioteki i aby w bieżącym katalogu powstał plik `libseq.so`. Polecenie to powinno również kompilować i dołączać do biblioteki załączony do treści zadania plik `memory_tests.c`. Wywołanie `make clean` powinno usuwać wszystkie pliki utworzone podczas kompilowania. Plik `makefile` lub `Makefile` może zawierać też inne cele, na przykład cel kompilujący i linkujący z biblioteką przykład jej użycia zawarty w załączonym do treści zadania pliku `seq_example.c` czy też cel uruchamiający testy.

Do kompilowania należy użyć `gcc`. Biblioteka powinna się kompilować w laboratorium komputerowym pod Linuksem. Pliki z implementacją biblioteki należy kompilować z opcjami:

```
-Wall -Wextra -Wno-implicit-fallthrough -std=gnu17 -fPIC -O2
```

Pliki z implementacją biblioteki należy linkować z opcjami:

```
-shared -Wl,--wrap=malloc -Wl,--wrap=calloc -Wl,--wrap=realloc -Wl,--wrap=reallocarray -Wl,--wrap=free -Wl,--wrap=strdup -Wl,--wrap=strndup
```

Opcje `-Wl,--wrap=` sprawiają, że wywołania funkcji `malloc`, `calloc` itd. są przechwytywane odpowiednio przez funkcje `__wrap_malloc`, `__wrap_calloc` itd. Funkcje przechwytujące są zaimplementowane w załączonym do treści zadania pliku `memory_tests.c`.

Implementacja biblioteki nie może gubić pamięci ani pozostawiać struktury danych w niespójnym stanie, również wtedy gdy wystąpił błąd alokowania pamięci. Poprawność implementacji będzie sprawdzana za pomocą programu `valgrind`. Implementacja nie może zawierać sztucznych ograniczeń na rozmiar przechowywanych danych – jedynymi ograniczeniami są rozmiar pamięci dostępnej w komputerze i rozmiar słowa maszynowego użytego komputera.

Ocena

Za w pełni poprawne rozwiązanie zadania implementujące wszystkie wymagania można zdobyć 20 punktów, z tego 14 punktów zostanie wystawionych na podstawie testów automatycznych, a 6 punktów to ocena jakości kodu. Za problemy ze skompilowaniem rozwiązania lub niespełnienie wymogów formalnych można stracić wszystkie punkty. Za ostrzeżenia wypisywane przez kompilator może być odjęte do 2 punktów.

Rozwiązania należy implementować samodzielnie pod rygorem niezaliczenia przedmiotu. Zarówno korzystanie z cudzego kodu, jak i prywatne lub publiczne udostępnianie własnego kodu jest zabronione.

Załączniki

Załącznikami do treści zadania są następujące pliki: