Wstęp do programowania w języku C

Lista zadań nr 6

Na zajęcia 27 listopada 2017

UWAGA! Należy regularnie poświęcać czas na naukę funkcji biblioteki standardowej języka C!

Idea rozwiązania powinna być elegancka i prosta do zrozumienia, a kod czytelny, tj.:

- sformatowany[1] zgodnie z wybraną konwencją,
- zmienne i procedury powinny być nazwane zgodnie z ich przeznaczeniem,
- należy unikać powtarzającego się kodu poprzez zamykanie go w procedury,
- złożone zadania trzeba podzielić na podprocedury.

Dodatkowo swoje programy należy kompilować z flagami «`-g -O -std=c11 -Wall -Wextra`», traktując ostrzeżenia kompilatora jako potencjalne błędy! Programy powinny być wolne od błędów związanych z niewłaściwym użyciem pamięci. W tym celu należy użyć narzędzia instrumentacji kodu AddressSanitizer[2] przez dodanie flagi «`-fsanitize=address`» do opcji kompilatora.

Zadanie 1 (10*). Kopiec binarny[3] to tablicowa struktura danych reprezentująca drzewo binarne.

Niezmiennikiem (ang. *invariant*) kopca jest warunek narzucony na każdy węzeł drzewa i mówiący, że jego dzieci są od niego mniejsze względem ustalonej relacji porządku. Warunkiem wstępnym (ang. *precondition*) operacji wstawiania elementu x do kopca K jest zdanie „K jest prawidłowym kopcem o długości n i x nie należy do K”, a warunkiem końcowym (ang. *postcondition*) jest zdanie „K jest prawidłowym kopcem o długości $n+1$ i x należy do K”.

Zaimplementuj kopiec binarny o poniższym interfejsie, który należy umieścić w pliku «`binheap.h`»:

```
#pragma once

#define BH_CMP(a, b) ((a) - (b)) /* relacja porządku */
#define BH_TYPE      int         /* typ elementów kopca */

typedef struct binheap {
  unsigned size;      /* liczba wykorzystanych elementów */
  unsigned capacity;  /* długość tablicy data */
  BH_TYPE *data;      /* wskaźnik na tablicę z elementami kopca */
} binheap_t;

binheap_t *binheap_new(unsigned capacity);
void binheap_delete(binheap_t *heap);
bool binheap_valid(binheap_t *heap);
bool binheap_find(binheap_t *heap, BH_TYPE elem);
bool binheap_add(binheap_t *heap, BH_TYPE elem);
bool binhead_remove(binheap_t *heap, BH_TYPE elem);
```

W pliku «`main.c`» przetestuj poprawność implementacji kopca używając funkcji «`assert`» do wymuszenia warunków końcowych oraz niezmienników każdej operacji. Sprawdź co najmniej poprawność dodawania i usuwania elementów kopca, w którym jest około 10,000 losowych elementów.

[1] `https://clang.llvm.org/docs/ClangFormatStyleOptions.html\#configurable-format-style-options`
[2] `https://github.com/google/sanitizers/wiki/AddressSanitizer`
[3] `https://pl.wikipedia.org/wiki/Kopiec_binarny`

Miejsce na kopiec trzeba przydzielić dynamicznie procedurą «malloc». Gdy w kopcu wyczerpie się miejsce należy zmienić jego pojemność z użyciem «realloc».

Zadanie 2 (10). Napisz program, który czyta ze standardowego wejścia tekst w języku angielskim[4] i konstruuje z nich słownik wszystkich wyrazów wraz z częstotliwość ich występowania. Twój program ma dawać wynik naśladujący wywołanie następującego zestawu poleceń powłoki uniksa:

```
egrep -e '[A-Za-z]+' -o plik.txt | tr '[A-Z]' '[a-z]' | \
  sort | uniq -c | sort -n
```

Kolejne polecenia w powyższym jednolinijkowcu wykonują: ekstrakcję słów, czyli ciągów małych i dużych liter; translację dużych liter na małe litery; leksykograficzne posortowanie wszystkich wyrazów; zliczenie powtarzających się wyrazów, usunięcie duplikatów i dodanie kolumny z liczbą powtórzeń; posortowanie numeryczne po pierwszej kolumnie.

Zestaw plików tekstowych do testów można pobrać z katalogu dokumentacji[5] jądra systemu *Linux*. Twój program nie może z góry założyć jaka jest długość pliku wejściowego.

Podpowiedź: Jeśli nie skorzystasz z procedur biblioteki standardowej «stdlib.h» i «string.h» to wykonanie tego zadania będzie bardzo ciężkie. Kluczowym jest znalezienie właściwych narzędzi do rozwiązania problemu!

Zadanie 3 (10). Napisać program, który rozwiązuje zadanie oznaczone jako *Lista 6 zadanie 3* w systemie Moodle. Rozwiązanie tego zadania będzie sprawdzane automatycznie z użyciem sprawdzarki.

[4] dla uproszczenia – nie chcemy rozpatrywać polskich znaków diakrytycznych

[5] https://www.kernel.org/doc/Documentation/