Wstęp do programowania w języku C

Lista zadań nr 5

Na zajęcia 20 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 -Werror`», traktując ostrzeżenia kompilatora jako potencjalne błędy!

Zadanie 1 (10*). Zaprogramuj pulę obiektów, czyli menadżer pamięci przechowujący wyłącznie elementy tego samego typu. Statycznie przydziel na stercie miejsce na n elementów typu `T` i wektor n bitów, dla zadanego n podzielnego przez 8. Należy udostępnić funkcje «`T* T_alloc(void)`» oraz «`void T_free(T*)`. Jeśli i-ty bit w wektorze alokacji jest zapalony to znaczy, że i-ty element został przydzielony, w przeciwnym wypadku jest wolny. Jeśli skończyły się wolne elementy, to należy zwrócić wartość `NULL`. Błędy, np. podwójne zwolnienie pamięci, należy wyłapywać funkcją `assert`.

UWAGA! Rozwiązanie należy solidnie przetestować w pliku «`main.c`» – to też jest punktowane!

Pulę obiektów należy zawrzeć w pliku «`pool.h`» tak, by nadawała się do użycia dla dowolnego typu `T` i liczby n. Poniżej podano przykład użycia i wzorcowy plik implementacji do uzupełnienia. Podwójny znak kratki ## to operator preprocesora języka C, który skleja dwa tokeny w jeden.

```c
/* plik: point.c */

typedef struct point {
  float x, y, z;
} point_t;

#define POOL_TYPE point_t
#define POOL_SIZE 4096
#define POOL_NAME point
#include "pool.h"

/* Tu widoczne funkcje:
 *
 * point_t* point_alloc(void);
 * void point_free(point_t *);
 */
```

```c
/* plik: pool.h */

#define CONCAT2(x,y) x ## y
#define CONCAT(x,y) CONCAT2(x,y)

#define BVEC CONCAT(POOL_NAME, _bvec)
#define ELEM CONCAT(POOL_NAME, _elem)

static uint8_t BVEC[(POOL_SIZE)/8];
static POOL_TYPE ELEM[(POOL_SIZE)];

POOL_TYPE* CONCAT(POOL_NAME, _alloc)(void) { }
void CONCAT(POOL_NAME, _free)(POOL_TYPE* item) { }

#undef ELEM
#undef BVEC
#undef POOL_TYPE
#undef POOL_SIZE
#undef POOL_NAME
```

[1] https://clang.llvm.org/docs/ClangFormatStyleOptions.html\#configurable-format-style-options

Zadanie 2 (10). Zaprogramuj strukturę danych kolejki przechowującej maksymalnie n elementów typu `T`. Zaimplementuj ją jako bufor cykliczny w statycznie przydzielonej tablicy. Rozwiązanie implementujące poniższy interfejs należy dostarczyć w pliku «`ringbuf.h`» używając podobnej techniki jak w poprzednim zadaniu.

```
/* definicja struktury kolejki */
static struct {
  unsigned head;  /* indeks czoła kolejki (stąd usuwamy elementy) */
  unsigned tail;  /* indeks ogona kolejki (tu dostawiamy elementy) */
  unsigned count; /* liczba elementów w kolejce */
  T item[N];      /* elementy */
} T_queue;

/* przenosi element pod wskaźnikiem ‘from‘ do kolejki
 * i zwraca TRUE; jeśli kolejka pełna zwraca FALSE */
bool T_push(T* from);

/* przenosi element z kolejki do elementu pod wskaźnikiem ‘to‘
 * i zwraca TRUE; jeśli kolejka pusta zwraca FALSE */
bool T_pop(T* to);

/* j.w. ale nie usuwa elementu z kolejki */
bool T_front(T *to);
```

UWAGA! Rozwiązanie należy solidnie przetestować w pliku «`main.c`» – to też jest punktowane!

Tym razem do implementacji testów należy użyć minimalistycznej biblioteki MinUnit[2] składającej się z jednego pliku nagłówkowego «`minunit.h`», który należy skopiować do własnego projektu.

Zadanie 3 (10). Napisać program, który rozwiązuje zadanie oznaczone jako *Lista 5 zadanie 3* w systemie Moodle. Rozwiązanie tego zadania będzie sprawdzane automatycznie z użyciem sprawdzarki.

[2] `https://github.com/siu/minunit`