Wstęp do programowania w języku C

Grupa MSz we wtorek

Lista 8 na zajęcia 4.12.2018

---

1. *(15 punktów w trakcie pierwszej pracowni, 10 punktów później)*

W tym zadaniu chodzi o efektywną implementację rozszerzalnej kolejki używającej tablicy.

```
struct ArrayQueue {
  int *t;
  size_t size; // Liczba elementów w kolejsce
  size_t first; // Indeks pierwszego elementu w kolejce
  size_t capacity; // Wielkość tablicy
};
```

Tablica jest wykorzystywana cyklicznie:
i-ty element kolejki to zawsze `t[(first+i)%capacity]`.

Interfejs jest następujący (to jest to co trzeba zaimplementować):

```
struct ArrayQueue make_queue(size_t initial_capacity);
int pop_first(struct ArrayQueue *q);
void push_last(struct ArrayQueue *q, int value);
```

Funkcja `push_last` wstawia nowy element na koniec kolejki, a funkcja `pop_first` usuwa i zwraca pierwszy aktualny element (zakładamy, że nie będzie wywoływana dla pustej kolejki).

W momencie gdy przy wywołaniu `push_last` rozmiar tablicy jest zbyt mały by pomieścić dodatkowy element, należy ją rozszerzyć (`realloc`), tak by nowa pojemność była 2 razy większa niż poprzednia. Wtedy należy też poprzesuwać elementy i opcjonalnie zmienić indeks `first`, tak by funkcje `push_last` i `pop_first` działały

poprawnie. Jeśli nie rozszerzamy tablicy, `push_last` i `pop_first` powinny działać w czasie stałym.
Test:

```
struct ArrayQueue q = make_queue(1);
for (int i = 0; i < 4; i++) push_last(&q, i);
printf("%lu %lu %lu,", q.size, q.first, q.capacity);
for (int i = 0; i < 7; i++) {
  printf(" %i", pop_first(&q));
  push_last(&q, i);
}
push_last(&q, 0);
printf(", %i, %lu %lu %lu\n", pop_first(&q), q.size, q.first, q.capacity);
```

Wynik:

```
4 0 4, 0 1 2 3 0 1 2, 3, 5 3 8
```

Wartości `first` mogą się różnić od tych w przykładzie, gdyż zależą od wybranej strategii przesuwania elementów po powiększeniu tablicy.