Wstęp do programowania w języku C (2017/2018)

Grupa MSZ

Lista 7 na zajęcia 28.11.2017

Zadanie 1 (10 pkt. na pracowni, 5 pkt. 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);
void push_last(struct ArrayQueue *q, int value);
int pop_first(struct ArrayQueue *q);
```

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.

Zadanie 2 (10 pkt.).

Na wejściu dane są liczby naturalne M i N oraz sekwencja N liczb całkowitych. Napisz program, który znajduje takie wstawienie operatorów „+”, „−”, „*” i „/” między liczby w sekwencji, które daje wyrażenie o wyniku M.

Przyjmujemy arytmetykę całkowitoliczbową, czyli dzielenie odrzuca część ułamkową. Obowiązuje tradycyjna kolejność działań: mnożenie i dzielenie przed dodawaniem i odejmowaniem. Poszukujemy tylko wyrażeń w których częściowe wyniki nie przekraczają zakresu typu INT – jeśli zakres jest przekroczony w którymkolwiek momencie to należy uznać takie wyrażenie za nieakceptowalne. Jeśli istnieje więcej niż jedno rozwiązanie to wystarczy znaleźć dowolne z nich.
Przykład 1:

```
6 3 2 2 2
```

Odpowiedź:

```
2 * 2 + 2
```

lub

```
2 + 2 * 2
```

Przykład 2:

```
2147483647 4 2147483647 1 1 2147483647
```

Możliwa odpowiedź:

```
2147483647 * 1 + 1 / 2147483647
```

Zadanie 3 (10 pkt.). Opisane w SKOS.