



Осмо вежбање

Вежба 1

Имплементирати кружни бафер у изведби са два показивача (не индекса) и без додатне променљиве за памћење величине (дакле, са једним празним местом при максималној попуњености). Величина бафера, као и тип података које садржи, треба да буду подесиве помоћу претпроцесорских директива. Подесити затим тако да тип података буде интеџер, а величина 10. Бафер имплементирати у посебној датотеци, као посебан модул. Сам бафер треба да буде садржан у структури која се зове `CircularBuffer`.

Функције при раду са бафером треба да буду следеће:

1. `CircularInit` – Иницијализација бафера. Мора бити позвана пре прве употребе бафера.
2. `CircularIsFull` – Проверава да ли је бафер скроз пун. Враћа „тачно“ ако јесте, а „нетачно“ ако није.
3. `CircularIsEmpty` – Исто као пређашња функција, само обрнуто. Враћа „тачно“ ако је бафер празан, а „нетачно“ ако није.
4. `CircularPut` – Додаје елемент у кружни бафер. Не проверава попуњеност!
5. `CircularGet` – Узима елемент из бафера. Не проверава да ли је бафер већ празан!
6. `CircularEmptyBuff` – Празни бафер комплетно. Као ресетовање.
7. `CircularDump` – Исписује садржај бафера на стандардни излаз.

Свака од наведених функција прима показивач на структуру `CircularBuffer` као први параметар. Другим речима, бафер се прослеђује „по референци“ као први параметар.

Направите `main.c`, укључите модул са имплементацијом кружног бафера и играјте се са њим. Направите више од једног кружног бафера. Проверите све функције и ситуације које могу да се десе.



Додатни (напредни) задатак 1: Подесити величину бафера и тип његовог поља тако да се добије дупли бафер. Проверити да ли ради. Шта је мана оваквог приступа и шта треба променити да би се мана отклонила?

Додатни (напредни) задатак 2: Како је тренутно наведено у задатку, величина кружног бафера је фиксна за све бафере које направимо. Међутим, могуће је овакву имплементацију кружног бафера променити тако да омогући променљиву величину бафера (која би се задавала у CircularInit функцији). Шта се све мора изменити? Да ли то има неке последице на спрегу CircularBuffer модула?

Вежба 2

Имплементирати двоструко повезану листу. Елемент листе треба да буде ова структура:

```
struct employee
{
    char name[NAME_SIZE];
    char group[GROUP_NAME_SIZE];
    float experience;
    struct employee* prev;
    struct employee* next;
};
```

Листа треба да буде имплементирана као засебан модул и треба да буде представљена следећом структуром:

```
struct EmployeeList {
    struct employee* head; // Pointer to first element in list
    struct employee* tail; // Pointer to last element in list
}
```



Следеће функције треба да буду имплементирани:

1. `EmployeeListCreate` – Прави празну листу `EmployeeList`.
2. `EmployeeListDestroy` – Ослобађа меморију заузету од стране елемената листе.
3. `EmployeeListInsert` – Убацује елемент на одређено место у листи (одређено адресом елемента иза којег желимо да убацимо нови).
4. `EmployeeListDelete` – Брише елемент са одређеног места у листи (одређеног адресом елемента који желимо да обришемо).
5. `EmployeeListEmpty` – Враћа информацију да ли је листа празна.
6. `EmployeeListDump` – Исписује све елементе листе на стандардни излаз.

У функцији `main` направити неку измишљену листу. Омогућити кориснику да преко тастатуре може да унесе редни број елемента који жели да обрише или редни број елемента који жели да дода.