NAPREDNO PROGRAMIRANJE

Vježba 9. – Strukture podataka

Priprema za vježbu:

Ponoviti strukture podataka kao što su međuspremnik (eng. buffer), dvostruki međuspremnik, povezane liste, redovi, stog (eng. stack).

Rad u laboratoriju:

1. Zadatak:

Kreirajte biblioteku *circularBuffer* koja se sastoji od circularBuffer.h i circularBuffer.c datoteka. U biblioteci je potrebno implementirati kružni buffer u izvedbi sa dva pokazivača (ne indeksa) i bez dodatne varijable za pamćenje veličine (dakle, sa jednim praznim mjestom pri maksimalnoj popunjenosti). Veličina buffera treba biti podesiva pomoću pretprocesorske direktive. Podesiti zatim tako da tip podataka bude integer, a veličina 10. U datoteci circularBuffer.h deklarirajte sljedeće tipove podataka te funkcije:

```
#define BUFFER_SIZE 11
#define BUFFER_GET_ERROR -1111111111
typedef int buffer_type;

typedef struct buffer_struct {
     buffer_type buffer[BUFFER_SIZE];
     buffer_type *front;
     buffer_type *rear;
} buffer_struct;

void Init(buffer_struct* buf);

int isEmpty(buffer_struct* buf);

int isFull(buffer_struct* buf);

int put(buffer_struct* buf, buffer_type x);

buffer_type get(buffer_struct* buf);

void empty(buffer_struct* buf);

void dump(buffer_struct* buf);
```

1. Kopirajte datoteku mainCircular.c u projekt.
2. U datoteci circularBuffer.c definirajte funkcije deklarirane u datoteci zaglavlja zaglavlju.

3. Kompajlirajte program i analizirajte kod.

_____
_____
_____
_____
_____

2. Zadatak

1. Kreirajte biblioteku *DoubleLinkedList* koja se sastoji od DoubleLinkedList.h i DoubleLinkedList.c datoteka. U biblioteci je potrebno implementirati dvostruko povezanu listu
2. U datoteci DoubleLinkedList.h deklarirajte sljedeće tipove podataka te funkcije:

```
#define NAME_SIZE 20
#define GROUP_NAME_SIZE 20

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

typedef struct EmployeeList {
     employee* head;  // Pointer to first element in list
     employee* tail;  // Pointer to last element in list
     int       size;   // Number of elements in list
} EmployeeList;

// Creates empty EmployeeList
int EmployeeListCreate(EmployeeList* list);

// Frees all memory allocated by list
void EmployeeListDestroy(EmployeeList* list);

//Function inserts employee element to specific position
(index) in the list
```

```
int EmployeeListInsert(EmployeeList* list, employee* element,
int index);

//Function deletes employee element from specific
position(index) in the list
int EmployeeListDelete(EmployeeList* list, int index);

// Function returns employee element at specific position
(index) in the list
employee * EmployeeListGetElement (EmployeeList* list, int
index);

//Function returns number of elements in the list
int EmployeeListSize (EmployeeList* list);

// Prints all list elements on standard out
void EmployeeListDump  (EmployeeList* list);
```

3. Kopirajte datoteku mainDoubleLinkedList.c u projekt.
4. U datoteci DoubleLinkedList.c definirajte funkcije deklarirane u datoteci zaglavlja zaglavlju.
5. Kompajlirajte program i analizirajte kod.

_____
_____
_____
_____
_____