Programowanie Strukturalne

19-12-2022

Niech dana będzie struktura:

```
typedef struct node
{
    struct node *prev, *next;
    int data;
} node;

typedef struct list
{
    node * first, * last;
    int size;
} list;
```

gdzie `prev`, `next` wskazuja na poprzedni i następny element w liście dwukierunkowej, `list` jest kontenerem listy dwukierunkowej. Napisz poniższe funkcje, tak aby zachowywały poprawność danych w kontenerze `list`, które:

- `void init(list *L)` inicjalizuje `L`, tak aby `size` $= 0$, `next` $= NULL$, `prev` $= NULL$,
- `node * create_node(int data)` tworzy nowy element gdzie `data=d`,
- `void add2begin(list *L,node *N)`, `void add2end(list *L,node *N)` dodają element na początek, koniec listy odpowiednio,
- `node * remove_begin(list *L,node *N)`, `node * remove_end(list *L,node *N)` *odpinają* pierwszy, ostatni element od listy (o ile istnieją) i zwraca wskaźnik do odpiętego elementu lub $NULL$ w.p.p.,
- `void print(list L)`, `void print_rev(list L)`, które drukują na ekran listę od początku do końca i od końca do początku odpowiednio,
- `node * find(list L,int d)` która zwraca wskaźnik do pierwszego elementu listy, dla którego `data=d` lub $NULL$ w.p.p.,
- `void remove(list * L,node *N)` która odpina element `N` od listy,
- `void delete_data(list * L,int d)` która kasuje wszystkie elementy z listy `L`, gdzie których `data=d`,
- `void addsort(list *L,node *N)`, która dodaje element `N` do listy tak aby zachowany był niemalejący porządek wartości `data` w kolejnych elementach listy `L`. Zakładamy, że lista `L` była posortowana,
- `void sort(list *L)` która sortuje listę wykorzystując sortowanie bąbelkowe (porównujący tylko sąsiednie elementy i, $i+1$),
- `void sort2(list *L)` która sortuje listę wykorzystując sortowanie przez wybór (porównujący elementy i, j).