

Računske vježbe 9

Programiranje I

Elementi liste su sortirani cijeli brojevi u rastućem poretku. Napisati sljedeće funkcije:

- a) `insert_node` koja vrši umetanje elementa u listu tako da se ne naruši poredak,
- b) `reverse` koja okreće elemente liste,
- c) `remove_node` koja vrši brisanje prvog elementa čija je vrijednost jednaka proslijeđenom argumentu,
- d) `free_list` koja oslobađa memoriju zauzetu za listu.

Glavni program kreira listu i poziva funkcije napisane pod a, b, c i d.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node
5 {
6     int number;
7     struct node* next;
8 };
9
10 void print_list(struct node*);
11 struct node* insert_node(struct node*, int);
12 struct node* reverse(struct node*);
13 struct node* remove_node(struct node*, int);
14 void free_list(struct node*);
15
16 int main()
17 {
18     struct node* current, * previous, * head, * result;
19     int i, n, insertNumber;
20     puts("Unesite broj elemenata liste");
21     scanf("%d", &n);
22     puts("Unesite elemente liste");
23     for (i = 0; i < n; i++)
24     {
25         current = (struct node*)malloc(sizeof(struct node));
26         scanf("%d", &(current->number));
27         current->next = NULL;
28         if (i == 0) head = current;
29         else previous->next = current;
30         previous = current;
31     }
32     printf("Unesite trazeni broj:");
33     scanf("%d", &insertNumber);
34     printf("Vrsimo umetanje broja: %d\n", insertNumber);
35     head = insert_node(head, insertNumber);
36     print_list(head);
```

```

37 puts("Okrecemo listu");
38 head = reverse(head);
39 print_list(head);
40 printf("Vrsimo brisanje broja: %d\n", insertNumber);
41 head = remove_node(head, insertNumber);
42 print_list(head);
43 puts("Oslobadjamo memoriju");
44 free_list(head);
45 head = NULL; // ukazujemo da glava ni na sta ne pokazuje
46 }
47
48 void print_list(struct node* head)
49 {
50     printf("Elementi liste su: ");
51     while (head != NULL)
52     {
53         printf("%d ", head->number);
54         head = head->next;
55     }
56     printf("\n");
57 }
58
59 struct node* insert_node(struct node* head, int number)
60 {
61     struct node* newNode, * start, * previous;
62     //u promjenljivoj start pamtimo glavu liste
63     start = head;
64     newNode = (struct node*)malloc(sizeof(struct node));
65     newNode->number = number;
66     //dio programa gdje se vrsi provjera da li broj treba umetnuti na pocetku liste
67     if (number <= head->number)
68     {
69         newNode->next = head;
70         return newNode;
71     }
72     //dio programa gdje se vrsi provjera da li broj treba umetnuti unutar liste
73     previous = head;
74     head = head->next;
75     while (head != NULL)
76     {
77         if (number <= head->number)
78         {
79             previous->next = newNode;
80             newNode->next = head;
81             return start;
82         }
83         else
84         {
85             previous = head;
86             head = head->next;
87         }
88     }
89     //ako program stigne do ove linije onda broj treba umetnuti na kraju liste
90     previous->next = newNode;
91     newNode->next = NULL;
92     return start;
93 }
94
95

```

```

96 struct node* reverse(struct node* head)
97 {
98     struct node* previous = NULL;
99     struct node* current = head;
100    struct node* next = NULL;
101    while (current != NULL) {
102        // Cuvamo next
103        next = current->next;
104        // Okrecemo next za tekuci cvor
105        current->next = previous;
106        // Idemo naprijed
107        previous = current;
108        current = next;
109    }
110    return previous;
111 }
112
113 struct node* remove_node(struct node* head, int value)
114 {
115     struct node* temp = head, * previous;
116     // dio programa gdje se vrši provjera da li broj treba obrisati na pocetku liste
117     if (temp != NULL && temp->number == value) {
118         head = temp->next; // glava se mijenja
119         free(temp); // brisemo staru glavu
120         return head;
121     }
122     // Vrsimo pretragu po vrijednosti, vodimo racuna o
123     // prethodnom cvoru jer moramo promijeniti 'previous->next'
124     while (temp != NULL && temp->number != value) {
125         previous = temp;
126         temp = temp->next;
127     }
128     // ukoliko vrijednost nije pronadjena u listi
129     if (temp == NULL)
130         return head;
131     // preskacemo element koji treba obrisati
132     previous->next = temp->next;
133
134     free(temp); //oslobadjamo memoriju
135
136     return head;
137 }
138
139 void free_list(struct node* head)
140 {
141     struct node* temp;
142
143     while (head != NULL)
144     {
145         temp = head;
146         head = head->next;
147         free(temp);
148     }
149 }
150 }

```