

"C programmers never die. They are just cast into void."

- Alan Perlis

CSE102 Computer Programming with C

2019-2020 Spring Semester

Linked Lists

© 2015-2020 Yakup Genç

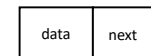
May 2020

CSE102 Lecture 11

1

1

```
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;
```



May 2020

CSE102 Lecture 11

2

2

```
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;
```

```
121 {
122     node * l = NULL;
123     l = (node *)malloc(sizeof(node));
124     l->data = 10;
125     l->next = NULL;
126 }
```

l

May 2020

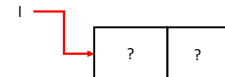
CSE102 Lecture 11

3

3

```
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;
```

```
121 {
122     node * l = NULL;
123     l = (node *)malloc(sizeof(node));
124     l->data = 10;
125     l->next = NULL;
126 }
```



May 2020

CSE102 Lecture 11

4

4

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

121 {
122     node * l = NULL;
123     l = (node *)malloc(sizeof(node));
124     l->data = 10;
125     l->next = NULL;
126 }

```

May 2020 CSE102 Lecture 11 5

5

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

121 {
122     node * l = NULL;
123     l = (node *)malloc(sizeof(node));
124     l->data = 10;
125     l->next = NULL;
126 }

```

May 2020 CSE102 Lecture 11 6

6

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

node * c = l;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

May 2020 CSE102 Lecture 11 7

7

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

node * c = l;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

May 2020 CSE102 Lecture 11 8

8

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

node * c = 1;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

Diagram illustrating the state of a linked list after the first node is created. The list has two nodes. The first node contains the value 10 and its next pointer is 0xFF00. The second node contains unknown values (represented by ?).

May 2020 CSE102 Lecture 11 9

9

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

node * c = 1;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

Diagram illustrating the state of a linked list after the second node is created. The list has two nodes. The first node contains the value 10 and its next pointer is 0xFF00. The second node contains the value 14 and its next pointer is unknown (represented by ?).

May 2020 CSE102 Lecture 11 10

10

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

node * c = 1;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

Diagram illustrating the state of a linked list after the second node is created. The list has two nodes. The first node contains the value 10 and its next pointer is 0xFF00. The second node contains the value 14 and its next pointer is unknown (represented by ?).

May 2020 CSE102 Lecture 11 11

11

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

node * c = 1;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```

Diagram illustrating the state of a linked list after the second node is created. The list has two nodes. The first node contains the value 10 and its next pointer is 0xFF00. The second node contains the value 14 and its next pointer is 0x0000, which points to a NULL box.

May 2020 CSE102 Lecture 11 12

12

```

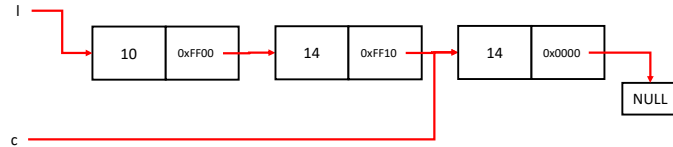
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

node * c = l;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```



May 2020

CSE102 Lecture 11

13

13

```

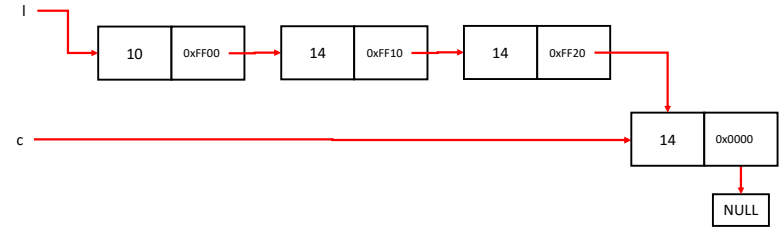
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

node * c = l;
c->next = (node *)malloc(sizeof(node));
c->data = 14;
c = c->next;
c->next = NULL;

```



May 2020

CSE102 Lecture 11

14

14

```

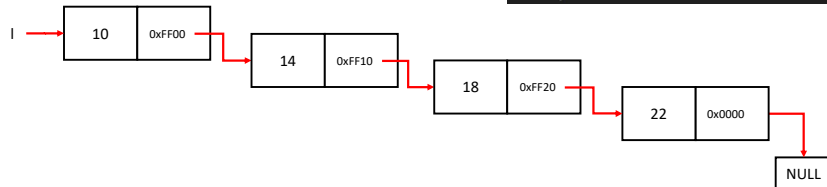
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

11 void ll_print(node * l) {
12     while (l!=NULL) {
13         printf("%d\n", l->data);
14         l = l->next;
15     }
16 }

```



May 2020

CSE102 Lecture 11

15

15

```

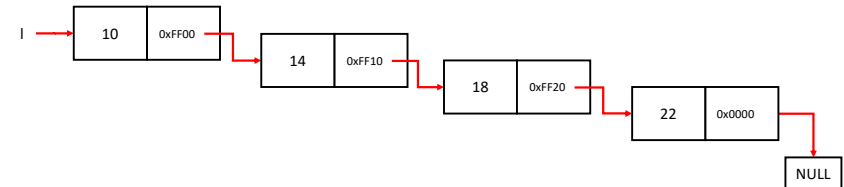
5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

18 void ll_print_r(node * l) {
19     if (l!=NULL) {
20         printf("%d\n", l->data);
21         ll_print_r(l->next);
22     }
23 }

```



May 2020

CSE102 Lecture 11

16

16

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

26 int ll_get_nth(node * l, int n) {
27     int i;
28     for (i=0; i<n; i++) {
29         l = l->next;
30     }
31     return l->data;
32 }

```

May 2020 CSE102 Lecture 11 17

17

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

86 void ll_insert_end(node * l, int k) {
87     if (l==NULL) {
88         /* TO DO SOMETHING HERE */
89     }
90     while (l->next!=NULL) l = l->next;
91     l->next = (node *) malloc(sizeof(node));
92     l->next->data = k;
93     l->next->next = NULL;
94 }

```

May 2020 CSE102 Lecture 11 18

18

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

Search?

May 2020 CSE102 Lecture 11 19

19

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

37 node * ll_remove(node * l, int k) {
38     node * cp, * bp;
39     cp = bp = l;
40     while (cp!=NULL && cp->data!=k) {
41         bp = cp;
42         cp = cp->next;
43     }
44     if (cp!=NULL) {
45         if (cp==bp) l = cp->next;
46         else bp->next = cp->next;
47         free(cp);
48     }
49     return l;
50 }

```

May 2020 CSE102 Lecture 11 20

20

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

52 node * ll_sorted_insert(node * l, int k) {
53     node * ce, * ne, * n;
54     if (l==NULL || l->data>k) {
55         n = (node *) malloc(sizeof(node));
56         n->data = k;
57         n->next = l;
58         l = n;
59     }
60     else {
61         ce = l;
62         ne = ce->next;
63         while (ce!=NULL && ne!=NULL && !(ce->data <= k && ne->data <= k)) {
64             ce = ne;
65             ne = ce->next;
66         }
67         n = (node *) malloc(sizeof(node));
68         n->data = k;
69         n->next = ne;
70         ce->next = n;
71     }
72     return l;
73 }

```

```

graph LR
    l --> n1[10 | 0xFF00]
    n1 --> n2[14 | 0xFF10]
    n2 --> n3[18 | 0xFF20]
    n3 --> n4[22 | 0x0000]
    n4 --> NULL

```

May 2020 CSE102 Lecture 11 21

21

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

```

75 node * ll_sort(node * l) {
76     node * s1 = NULL;
77     node * t;
78     while (l!=NULL) {
79         s1 = ll_sorted_insert(s1, l->data);
80         t = l;
81         l = l->next;
82         free(t);
83     }
84 }

```

```

graph LR
    l --> n1[10 | 0xFF00]
    n1 --> n2[14 | 0xFF10]
    n2 --> n3[18 | 0xFF20]
    n3 --> n4[22 | 0x0000]
    n4 --> NULL
    s1 --> n1

```

May 2020 CSE102 Lecture 11 22

22

```

5  typedef struct node {
6      int data;
7      struct node * next;
8  } node;

```

In Place Sorting?

```

graph LR
    l --> n1[10 | 0xFF00]
    n1 --> n2[14 | 0xFF10]
    n2 --> n3[18 | 0xFF20]
    n3 --> n4[22 | 0x0000]
    n4 --> NULL

```

May 2020 CSE102 Lecture 11 23

23