一维数组程序的高阶分析————双链表结点的插入和删除

2022 年 8 月 16 日

1. 阅读程序，回答相应问题。

```
#include <stdio.h>
#include <limits.h>

#define NODE_COUNT 10

int main()
{
    double data[NODE_COUNT];
    int previous[NODE_COUNT];
    int next[NODE_COUNT];

    int link_node_start = -1, link_node_end = -1;
    int free_node_start = 0, free_node_end = NODE_COUNT - 1;


    for(int i = 1; i < NODE_COUNT - 1; i++)
    {
        previous[i] = i - 1;
        next[i] = i + 1;
    }
    previous[0] = -1;
    previous[NODE_COUNT - 1] = NODE_COUNT - 2;
    next[0] = 1;
    next[NODE_COUNT - 1] = -1;

```

```
    printf("Initially ...\n");
    printf("prev:\t");
    for(int i = 0; i < NODE_COUNT; i++)
    {
        printf("%d\t", previous[i]);
    }
    printf("\n");
    printf("data fields contain garbage...\n");
    printf("next:\t");
    for(int i = 0; i < NODE_COUNT; i++)
    {
        printf("%d\t", next[i]);
    }
    printf("\n\n");

    double a[] = {1.25, -2.5, -3.375, -4.0, -5.25};
    int len = sizeof(a) / sizeof(double);

    for(int i = 0; i < len; i++)
    {
        if(free_node_start == -1)
        {
            printf("No free nodes can be allocated.\n");
            return 1;
        }

        int new_node = free_node_start;
        free_node_start = next[free_node_start];

        data[new_node] = a[i];

        if(link_node_start == -1)
        {
            link_node_start = new_node;
            link_node_end = new_node;
            previous[new_node] = -1;
            next[new_node] = -1;
            continue;
        }
```

```
65
66          previous[link_node_start] = new_node;
67          next[new_node] = link_node_start;
68          link_node_start = new_node;
69          previous[new_node] = -1;
70      }
71
72      printf("after inserting a list of nodes...\n");
73
74      printf("prev:\t");
75      for(int i = 0; i < NODE_COUNT; i++)
76      {
77          printf("%d\t", previous[i]);
78      }
79      printf("\n");
80
81      printf("data:\t");
82      for(int i = 0; i < NODE_COUNT; i++)
83      {
84          printf("%.3lf\t", data[i]);
85      }
86      printf("\n");
87
88      printf("next:\t");
89      for(int i = 0; i < NODE_COUNT; i++)
90      {
91          printf("%d\t", next[i]);
92      }
93      printf("\n");
94
95      printf("Actual nodes in the order of a list:\n");
96      for(int i = link_node_start; i != next[link_node_end]; i = next[i])
97      {
98          printf("%.3lf\t", data[i]);
99      }
100
101     printf("\n");
102
103     printf("Actual nodes in the reversed order of a list:\n");
```

```
104     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
105     {
106         printf("%.3lf\t", data[i]);
107     }
108 
109     printf("\n\n");
110 
111     for(int j = 0; j < 3; j++)
112     {
113         if(link_node_start == -1)
114         {
115             link_node_end = -1;
116             break;
117         }
118         int eliminated_node = link_node_end;
119         link_node_end = previous[link_node_end];
120 
121         next[eliminated_node] = free_node_start;
122         previous[free_node_start] = eliminated_node;
123         free_node_start = eliminated_node;
124     }
125 
126     printf("after eliminating a list of nodes...\n");
127 
128     printf("prev:\t");
129     for(int i = 0; i < NODE_COUNT; i++)
130     {
131         printf("%d\t", previous[i]);
132     }
133     printf("\n");
134 
135     printf("data:\t");
136     for(int i = 0; i < NODE_COUNT; i++)
137     {
138         printf("%.3lf\t", data[i]);
139     }
140     printf("\n");
141 
142     printf("next:\t");
```

```
143     for(int i = 0; i < NODE_COUNT; i++)
144     {
145         printf("%d\t", next[i]);
146     }
147     printf("\n");
148 
149     printf("Actual nodes in the order of a list:\n");
150     for(int i = link_node_start; i != next[link_node_end]; i = next[i])
151     {
152         printf("%.3lf\t", data[i]);
153     }
154 
155     printf("\n");
156     printf("Actual nodes in the reversed order of a list:\n");
157     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
158     {
159         printf("%.3lf\t", data[i]);
160     }
161 
162     printf("\n\n");
163 
164     double b[] = {-3.25, -16.5, 7.75, 8.625};
165     len = sizeof(b) / sizeof(double);
166 
167     for(int i = 0; i < len; i++)
168     {
169         if(free_node_start == -1)
170         {
171             printf("No free nodes can be allocated.\n");
172             return 1;
173         }
174 
175         int new_node = free_node_start;
176         free_node_start = next[free_node_start];
177         previous[free_node_start] = -1;
178 
179         data[new_node] = b[i];
180 
181         if(link_node_start == -1)
```

```
182         {
183             link_node_start = new_node;
184             link_node_end = new_node;
185             previous[new_node] = -1;
186             next[new_node] = -1;
187             continue;
188         }
189 
190         previous[link_node_start] = new_node;
191         next[new_node] = link_node_start;
192         link_node_start = new_node;
193         previous[new_node] = -1;
194     }
195 
196     printf("after inserting another list of nodes...\n");
197 
198     printf("prev:\t");
199     for(int i = 0; i < NODE_COUNT; i++)
200     {
201         printf("%d\t", previous[i]);
202     }
203     printf("\n");
204 
205     printf("data:\t");
206     for(int i = 0; i < NODE_COUNT; i++)
207     {
208         printf("%.3lf\t", data[i]);
209     }
210     printf("\n");
211 
212     printf("next:\t");
213     for(int i = 0; i < NODE_COUNT; i++)
214     {
215         printf("%d\t", next[i]);
216     }
217     printf("\n");
218 
219 
220     printf("Actual nodes in the order of a list:\n");
```

```
221     for(int i = link_node_start; i != next[link_node_end]; i = next[i])
222     {
223         printf("%.3lf\t", data[i]);
224     }
225     printf("\n");
226 
227     printf("Actual nodes in the reversed order of a list:\n");
228     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
229     {
230         printf("%.3lf\t", data[i]);
231     }
232     printf("\n\n");
233 
234     return 0;
235 }
```

(a) 分析程序，按如下要求写出程序输出的部分结果。

i. 写出程序第 95 行到第 107 行的输出。

ii. 写出程序第 149 行到第 160 行的输出。

iii. 写出程序第 220 行到第 231 行的输出。

(b) 考虑程序中的数组 `data`。

i. 按顺序写出它的各个单元先后被赋值的操作。

ii. 它的各个单元先后被赋值的次数分别是多少?

(c) 考虑程序中的变量 `free_node_start`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(d) 考虑程序中的变量 `link_node_start`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(e) 考虑 `previous[1]` 和 `next[1]`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(f) 考虑 `previous[2]` 和 `next[2]`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(g) 考虑 `free_node_start` 和 `link_node_start`。

i. 它们各自的取值何时为`-1`，何时不为`-1`? 请简要说明。

ii. 当上述二者各自的取值不为-1 时，

A. next[free_node_start] 和 next[link_node_start] 先后的取值分别是什么？如有需要，请简要说明理由。

B. 当上述二者各自的取值不为-1 时，分别考虑 next[next[free_node_start]] 和 next[next[link_node_start]]，它们先后的取值分别是什么？如有需要，请简要说明理由。

(h) 第 59 行被执行的次数是多少？请简要说明理由。

(i) 第 66 行被执行的次数是多少？请简要说明理由。

(j) 第 118 行被执行的次数是多少？请简要说明理由。

(k) 第 183 行被执行的次数是多少？请简要说明理由。

2. 阅读程序，回答相应问题。

```c
#include <stdio.h>
#include <limits.h>

#define NODE_COUNT 10

int main()
{
    double data[NODE_COUNT];
    int previous[NODE_COUNT];
    int next[NODE_COUNT];

    int link_node_start = -1, link_node_end = -1;
    int free_node_start = 0, free_node_end = NODE_COUNT - 1;


    for(int i = 1; i < NODE_COUNT - 1; i++)
    {
        previous[i] = i - 1;
        next[i] = i + 1;
    }
    previous[0] = -1;
    previous[NODE_COUNT - 1] = NODE_COUNT - 2;
    next[0] = 1;
    next[NODE_COUNT - 1] = -1;
```

```
25
26      printf("Initially ...\n");
27      printf("prev:\t");
28      for(int i = 0; i < NODE_COUNT; i++)
29      {
30          printf("%d\t", previous[i]);
31      }
32      printf("\n");
33      printf("data fields contain garbage...\n");
34      printf("next:\t");
35      for(int i = 0; i < NODE_COUNT; i++)
36      {
37          printf("%d\t", next[i]);
38      }
39      printf("\n\n");
40
41      double a[] = {1.25, -2.5, -3.375, -4.0, -5.25};
42      int len = sizeof(a) / sizeof(double);
43
44      for(int i = 0; i < len; i++)
45      {
46          if(free_node_start == -1)
47          {
48              printf("No free nodes can be allocated.\n");
49              return 1;
50          }
51
52          int new_node = free_node_start;
53          free_node_start = next[free_node_start];
54
55          data[new_node] = a[i];
56
57          if(link_node_start == -1)
58          {
59              link_node_start = new_node;
60              link_node_end = new_node;
61              previous[new_node] = -1;
62              next[new_node] = -1;
63              continue;
```

```
64          }
65
66          previous[link_node_start] = new_node;
67          next[new_node] = link_node_start;
68          link_node_start = new_node;
69          previous[new_node] = -1;
70      }
71
72      printf("after inserting a list of nodes...\n");
73
74      printf("prev:\t");
75      for(int i = 0; i < NODE_COUNT; i++)
76      {
77          printf("%d\t", previous[i]);
78      }
79      printf("\n");
80
81      printf("data:\t");
82      for(int i = 0; i < NODE_COUNT; i++)
83      {
84          printf("%.3lf\t", data[i]);
85      }
86      printf("\n");
87
88      printf("next:\t");
89      for(int i = 0; i < NODE_COUNT; i++)
90      {
91          printf("%d\t", next[i]);
92      }
93      printf("\n");
94
95      printf("Actual nodes in the order of a list:\n");
96      for(int i = link_node_start; i != next[link_node_end]; i = next[i])
97      {
98          printf("%.3lf\t", data[i]);
99      }
100
101     printf("\n");
102
```

```
103     printf("Actual nodes in the reversed order of a list:\n");
104     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
105     {
106         printf("%.3lf\t", data[i]);
107     }
108 
109     printf("\n\n");
110 
111     for(int j = 0; j < 3; j++)
112     {
113         if(link_node_start == -1)
114         {
115             link_node_end = -1;
116             break;
117         }
118         int eliminated_node = link_node_start;
119         link_node_start = next[link_node_start];
120 
121         next[eliminated_node] = free_node_start;
122         previous[free_node_start] = eliminated_node;
123         free_node_start = eliminated_node;
124     }
125 
126     printf("after eliminating a list of nodes...\n");
127 
128     printf("prev:\t");
129     for(int i = 0; i < NODE_COUNT; i++)
130     {
131         printf("%d\t", previous[i]);
132     }
133     printf("\n");
134 
135     printf("data:\t");
136     for(int i = 0; i < NODE_COUNT; i++)
137     {
138         printf("%.3lf\t", data[i]);
139     }
140     printf("\n");
141 
```

```
142     printf("next:\t");
143     for(int i = 0; i < NODE_COUNT; i++)
144     {
145         printf("%d\t", next[i]);
146     }
147     printf("\n");
148 
149     printf("Actual nodes in the order of a list:\n");
150     for(int i = link_node_start; i != next[link_node_end]; i = next[i])
151     {
152         printf("%.3lf\t", data[i]);
153     }
154 
155     printf("\n");
156     printf("Actual nodes in the reversed order of a list:\n");
157     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
158     {
159         printf("%.3lf\t", data[i]);
160     }
161 
162     printf("\n\n");
163 
164     double b[] = {-3.25, -16.5, 7.75, 8.625};
165     len = sizeof(b) / sizeof(double);
166 
167     for(int i = 0; i < len; i++)
168     {
169         if(free_node_start == -1)
170         {
171             printf("No free nodes can be allocated.\n");
172             return 1;
173         }
174 
175         int new_node = free_node_start;
176         free_node_start = next[free_node_start];
177         previous[free_node_start] = -1;
178 
179         data[new_node] = b[i];
180 
```

```
181         if(link_node_start == -1)
182         {
183             link_node_start = new_node;
184             link_node_end = new_node;
185             previous[new_node] = -1;
186             next[new_node] = -1;
187             continue;
188         }
189 
190         previous[link_node_start] = new_node;
191         next[new_node] = link_node_start;
192         link_node_start = new_node;
193         previous[new_node] = -1;
194     }
195 
196     printf("after inserting another list of nodes...\n");
197 
198     printf("prev:\t");
199     for(int i = 0; i < NODE_COUNT; i++)
200     {
201         printf("%d\t", previous[i]);
202     }
203     printf("\n");
204 
205     printf("data:\t");
206     for(int i = 0; i < NODE_COUNT; i++)
207     {
208         printf("%.3lf\t", data[i]);
209     }
210     printf("\n");
211 
212     printf("next:\t");
213     for(int i = 0; i < NODE_COUNT; i++)
214     {
215         printf("%d\t", next[i]);
216     }
217     printf("\n");
218 
219 
```

```
220     printf("Actual nodes in the order of a list:\n");
221     for(int i = link_node_start; i != next[link_node_end]; i = next[i])
222     {
223         printf("%.3lf\t", data[i]);
224     }
225     printf("\n");
226 
227     printf("Actual nodes in the reversed order of a list:\n");
228     for(int i = link_node_end; i != previous[link_node_start]; i = previous[i])
229     {
230         printf("%.3lf\t", data[i]);
231     }
232     printf("\n\n");
233 
234     return 0;
235 }
```

(a) 分析程序，按如下要求写出程序输出的部分结果。

i. 写出程序第 95 行到第 107 行的输出。

ii. 写出程序第 149 行到第 160 行的输出。

iii. 写出程序第 220 行到第 231 行的输出。

(b) 考虑程序中的数组 `data`。

i. 按顺序写出它的各个单元先后被赋值的操作。

ii. 它的各个单元先后被赋值的次数分别是多少?

(c) 考虑程序中的变量 `free_node_start`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(d) 考虑程序中的变量 `link_node_start`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(e) 考虑 `previous[1]` 和 `next[1]`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(f) 考虑 `previous[2]` 和 `next[2]`，它先后的取值分别是什么? 如有需要，请简要说明理由。

(g) 考虑 `free_node_start` 和 `link_node_start`。

i. 它们各自的取值何时为`-1`，何时不为`-1`? 请简要说明。

ii. 当上述二者各自的取值不为-1 时，

A. `next[free_node_start]` 和 `next[link_node_start]` 先后的取值分别是什么？如有需要，请简要说明理由。

B. 当上述二者各自的取值不为-1 时，分别考虑 `next[next[free_node_start]]` 和 `next[next[link_node_start]]`，它们先后的取值分别是什么？如有需要，请简要说明理由。

(h) 第 59 行被执行的次数是多少？请简要说明理由。

(i) 第 66 行被执行的次数是多少？请简要说明理由。

(j) 第 118 行被执行的次数是多少？请简要说明理由。

(k) 第 183 行被执行的次数是多少？请简要说明理由。