《数据结构》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机科学与技术（国际联合培养） | 年级 | 19级 | 班级 | 1901 |
| 学生姓名 | 姬彬荃 | 学号 | 201911020125 | 指导教师 | 魏晓超 |
| 实验题目 | 实验二：实现单链表各种基本运算的算法 | | | 提交时间 | 2020.09.23 |

一、实验目的和要求

（1）熟悉C语言的上机环境，进一步掌握C语言的结构特点。

（2）掌握线性表的链式存储结构的定义及基本运算

二、实验环境

Visual Studio 2019

三、实验内容及实施）

**实验1：单链表的建立及运算**

实验要求：

1. 初始化单链表；
2. 一次采用尾插法插入a、b、c、d、e元素；
3. 输出单链表；
4. 输出单链表长度；
5. 判断单链表是否为空；
6. 输出单链表的第3个元素；
7. 输出元素d的位置；
8. 在第4个元素位置上插入f元素；
9. 输出插入后的单链表；
10. 删除单链表中的第2个元素；
11. 输出删除后的单链表；
12. （选做）单链表“原地”逆转，要求算法空间复杂度为O（1）。

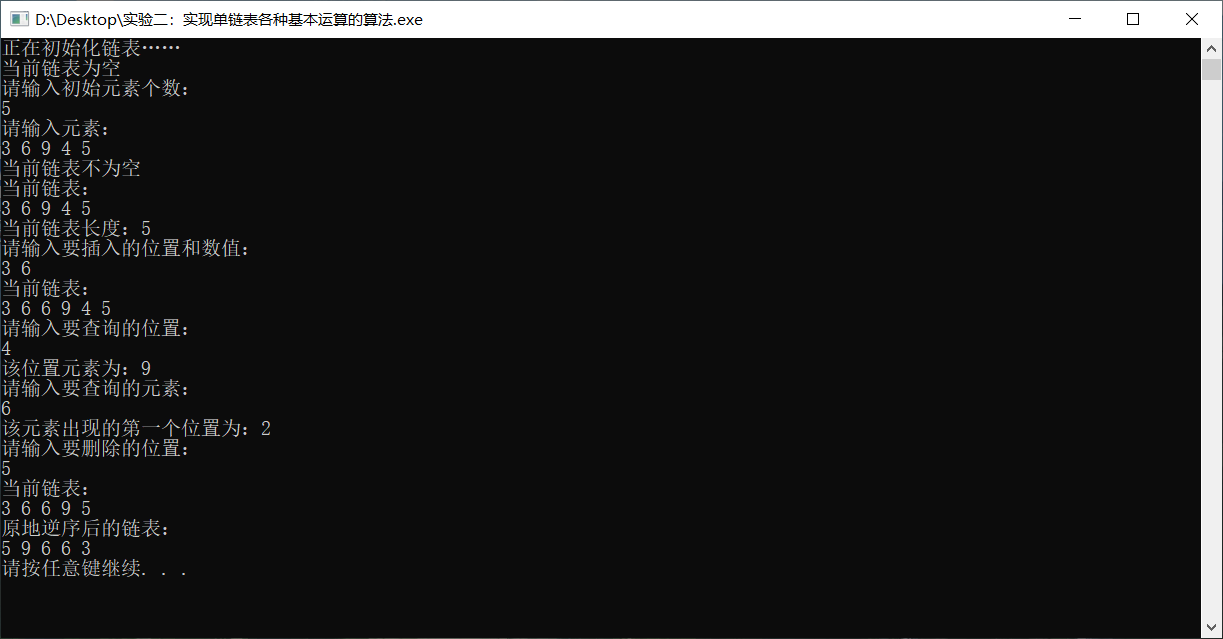
**【源程序】**

****

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. \*\* 功能 : 实验二：实现单链表各种基本运算的算法
3. \*\* 作者 : 2019Jibinquan
4. \*\* 创建 : 2020-9-23 / 15:57
5. \*\* 联系 : 431263064@qq.com
6. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
8. #include <iostream>
9. #include <cstdio>
10. #include <cmath>
11. #include <algorithm>
12. #include <cstring>
13. #include <queue>
14. #include <stack>
15. #include <vector>
16. #include <map>
17. #include <set>
18. #define ios ios::sync\_with\_stdio(false);cin.tie(0);cout.tie(0);
19. #define debug(a) cout << #a << " " << a << endl
20. **using** **namespace** std;
21. **typedef** **long** **long** ll;
22. **const** **double** pi = acos(-1);
23. **const** **double** eps = 1e-8;
24. **const** **int** inf = 0x3f3f3f3f;
25. **const** **int** maxn = 100007;//1e5+7
26. **const** ll mod = 1000000007;//1e9+7
28. **typedef** **struct** LNode{
29. **int** data;
30. LNode\* next;
31. }LNode,\*Linklist;

34. Linklist initList() //初始化链表
35. {
36. Linklist L = **new** LNode;
37. L->next = NULL;
38. **return** L;
39. }
41. **void** insert\_back(Linklist &L,**int** e) //尾插法插入元素
42. {
43. LNode \*p = L;
44. **while** (p->next != NULL) {
45. p = p->next;
46. }
47. LNode\* s = **new** LNode;
48. s->data = e;
49. s->next = NULL;
50. p->next = s;
51. }
53. **void** Creatlist(Linklist& L) //创建一个链表 ，可一次插入多组数据
54. {
55. LNode\* p = L;
56. **while** (p->next != NULL) {
57. p = p->next;
58. }
59. **int** n,e;
60. LNode\* s = **new** LNode;
61. LNode\* pre = **new** LNode;
62. pre = p;
63. cout << "请输入初始元素个数：" << endl;
64. cin >> n;
65. cout << "请输入元素：" << endl;
66. **for** (**int** i = 1; i <= n; i++) {
67. cin >> e;
68. s = **new** LNode;
69. s->data = e;
70. s->next = NULL;
71. pre->next = s;
72. pre = pre->next;
73. }
74. }
76. **void** printList(Linklist L) //打印链表
77. {
78. LNode\* p = L->next;
79. **while** (p->next != NULL) {
80. cout << p->data << ' ';
81. p = p->next;
82. }
83. cout << p->data << endl;
84. }
86. **int** getlength(Linklist L) //得到链表长度
87. {
88. **int** len = 0;
89. LNode\* p = L;
90. **while** (p->next != NULL) {
91. p = p->next;
92. len++;
93. }
94. **return** len;
95. }
97. **bool** isempty(Linklist L) //链表判空
98. {
99. LNode\* p = L;
100. **if** (p->next == NULL) {
101. **return** **true**;
102. }
103. **else** **return** **false**;
104. }
106. **int** find(Linklist L, **int** pos)//根据下标查找某一元素
107. {
108. LNode\* p = L->next;
109. **for** (**int** i = 1; i < pos; i++) {
110. p = p->next;
111. }
112. **return** p->data;
113. }
115. **int** findpos(Linklist L, **int** e)//寻找某一元素第一次出现的位置
116. {
117. **int** pos = 1;
118. LNode\* p = L->next;
119. **while** (p->next != NULL) {
120. **if** (p->data == e) {
121. **return** pos;
122. }
123. p = p->next;
124. pos++;
125. }
126. **return** -1;
127. }
129. **void** insertelm(Linklist& L, **int** pos, **int** e)//在指定位置插入元素
130. {
131. LNode\* p = L;
132. **for** (**int** i = 1; i < pos; i++){
133. p = p->next;
134. }
135. LNode\* s = **new** LNode;
136. s->data = e;
137. s->next = p->next;
138. p->next = s;
139. }
141. **void** deleteelm(Linklist& L, **int** pos)//删除指定位置的元素
142. {
143. LNode\* p = L;
144. **for** (**int** i = 1; i < pos; i++) {
145. p = p->next;
146. }
147. LNode\* s = p->next;
148. p->next = p->next->next;
149. free(s);
150. }
152. **void** reverseList(Linklist& L)//反转链表
153. {
154. LNode\* p = L->next->next;
155. LNode\* pre = L->next;
156. pre->next = NULL;
157. LNode\* tp = **new** LNode;
158. **while** (p->next != NULL) {
159. tp = p->next;
160. p->next = pre;
161. pre = p;
162. p = tp;
163. }
164. p->next = pre;
165. L->next = p;
166. }
168. /\*----------------------------------\*
169. Main Function
170. \*-----------------------------------\*/
172. **int** main()
173. {
174. **int** pos, e;
175. Linklist L = initList();
176. cout << "正在初始化链表……" << endl;
177. **if** (isempty(L)) {
178. cout << "当前链表为空" << endl;
179. }
180. **else** {
181. cout << "当前链表不为空" << endl;
182. }
183. Creatlist(L);
184. **if** (isempty(L)) {
185. cout << "当前链表为空" << endl;
186. }
187. **else** {
188. cout << "当前链表不为空" << endl;
189. }
190. cout << "当前链表：" << endl;
191. printList(L);
192. cout <<"当前链表长度："<< getlength(L) << endl;
193. cout << "请输入要插入的位置和数值：" << endl;
194. cin >> pos >> e;
195. insertelm(L, pos, e);
196. cout << "当前链表：" << endl;
197. printList(L);
198. cout << "请输入要查询的位置：" << endl;
199. cin >> pos;
200. cout << "该位置元素为：" << find(L, pos) << endl;
201. cout << "请输入要查询的元素：" << endl;
202. cin >> e;
203. pos = findpos(L, e);
204. **if** (pos == -1) {
205. cout << "未找到该元素！" << endl;
206. }
207. **else** {
208. cout << "该元素出现的第一个位置为：" << pos << endl;
209. }
210. cout << "请输入要删除的位置：" << endl;
211. cin >> pos;
212. deleteelm(L, pos);
213. cout << "当前链表：" << endl;
214. printList(L);
215. cout << "原地逆序后的链表：" << endl;
216. reverseList(L);
217. printList(L);
218. system("pause");
220. **return** 0;
221. }

**四、实验结果 (程序的执行结果)**



**五、实验讨论（可选）**

指针还是比较难理解的，错了几次才发现LNode\* == Linklist,且除了new的时候外指针类型的一定要带\*，发现“.”和“->”好像存在区别，使用“.”会出现编译不过的问题。