计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：杨佳森 学号：2112080106 专业：\_数据科学与大数据技术\_年级：\_2021\_

课程： 数据结构 主讲教师：\_\_刘成\_\_ 辅导教师：\_\_\_\_\_无\_\_

实验时间：2023年 02月 24日 下午15时至\_17时，实验地点： 计算机大楼213\_

实验题目： 链表的基本操作

实验目的： 通过该实验，深入理解链表的逻辑结构、物理结构等概念，掌握链表基本操作的编程实现，熟练掌握C语言中指针的操作。和实验3对比，掌握线性结构两种不同存储方式的区别。

实验环境（硬件和软件） PC和codeblocks

实验内容：

编程实现链表下教材第二章定义的线性表的基本操作，最好用菜单形式对应各个操作，使其编程一个完整的小软件。

实验步骤：

单链表的存储结构：

typedef struct Node {

    int data;

    struct Node \*next;

} Node,\*LinkList;

1. 初始化或重置链表：

void InitList(LinkList &L) {

    L = new Node;

    L->next = NULL;

cout << "初始化成功！" << endl;

}

1. 销毁链表：

void  ListLength(LinkList L) {

    LinkList p = L->next; *//p指向第一个结点*

    int count = 0;

    while(p) {

        ++count;

        p = p->next;

    }

    cout << "链表中数据元素个数为：" << p << endl;;

 }

1. 链表中数据元素个数：

void  ListLength(LinkList L) {

    LinkList p = L->next; *//p指向第一个结点*

    int count = 0;

    while(p) {

        ++count;

        p = p->next;

    }

    cout << "链表中数据元素个数为：" << p << endl;;

 }

1. 所指位序的元素值：

void GetElem(LinkList L,int i,int &e) {

    LinkList p = L->next;

    int j = 1;

    while(p && j < i) {

        p = p->next;

        ++j;

    }

    if(!p || j > i) cout << "ERROR!" << endl;

  else {

   e = p->data;

   cout << "第" << i << "的元素值为：" << e << endl;

  }

 }

1. 链表已存在元素的位序：

void LocateElem(LinkList L,int e) {

    LinkList p = L->next;

    int j = 1;

    while(p && p->data != e) {

   p = p->next;

   j++;

  }

    cout << "该元素的位序为：" << j << endl;

 }

1. 请输入元素，求直接前驱：

void ListPrior(LinkList L,int e) {

  LinkList q = L->next;*//第一个结点*

  if(q->data == e) {

    cout << "无前驱！" << endl;

  }

  else {

    LinkList p;

    while(q && q->data != e) {

     p = q;

     q = q->next;

    }

   cout << "该元素前驱为：" << p->data << endl;

  }

 }

1. 请输入元素，求直接后继：

void ListNext(LinkList L,int e) {

  LinkList p = L->next;

  while(p && p->data != e) p = p->next;

  if(p->next) cout << "该元素后继为：" << p->next->data << endl;

  else cout << "无后继！" << endl;

 }

1. 在第i个位置直接插入元素：

void ListInsert(LinkList &L,int i,int e) {

    LinkList p = L;

    int j = 0;

    while(p && (j < i - 1)) {

        p = p->next;

        ++j;

    }

    if(!p || j > i - 1) {

   cout << "ERROR!" << endl;

  }

  else {

   LinkList s = new Node;

   s->data = e;

   s->next = p->next;

   p->next = s;

   cout << "插入成功！" << endl;

  }

 }

1. 删除第i个元素：

void ListDelete(LinkList &L,int i) {

    LinkList p = L;

    int j = 0;

    while(p->next && j < i-1) {

        p = p->next;

        ++j;

    }

    if(!(p->next) || j > i-1) {

   cout << "ERROR!" << endl;

  }

  else {

   LinkList r = p->next;

   p->next = r->next;

   delete r;

  }

 }

1. 输出所输入的链表元素：

void DisplayList(LinkList L) {

    LinkList p = L->next;

    while(p) {

        cout << p->data << "  ";

        p = p->next;

    }

    cout << endl;

}

1. 初始化并输入链表元素：

void CreateList\_H(LinkList &L,int n) {

    L = new Node;

    L->next = NULL;

    for(int i = 0; i < n; i++) {

        LinkList p = new Node;

        cin >> p->data;

        p->next = L->next;

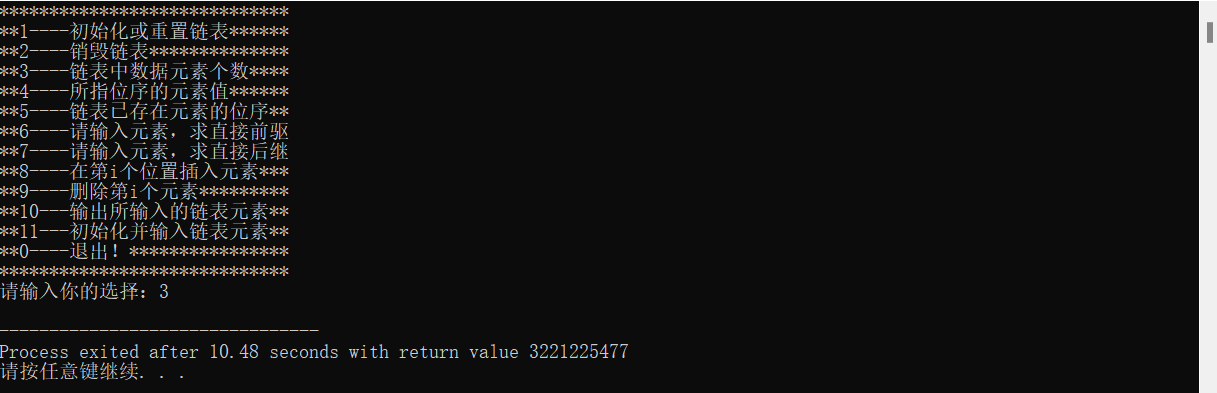
        L->next = p;

    }

}

实验数据记录：

1. 没有进行初始化前进行其他操作，程序是否能控制住

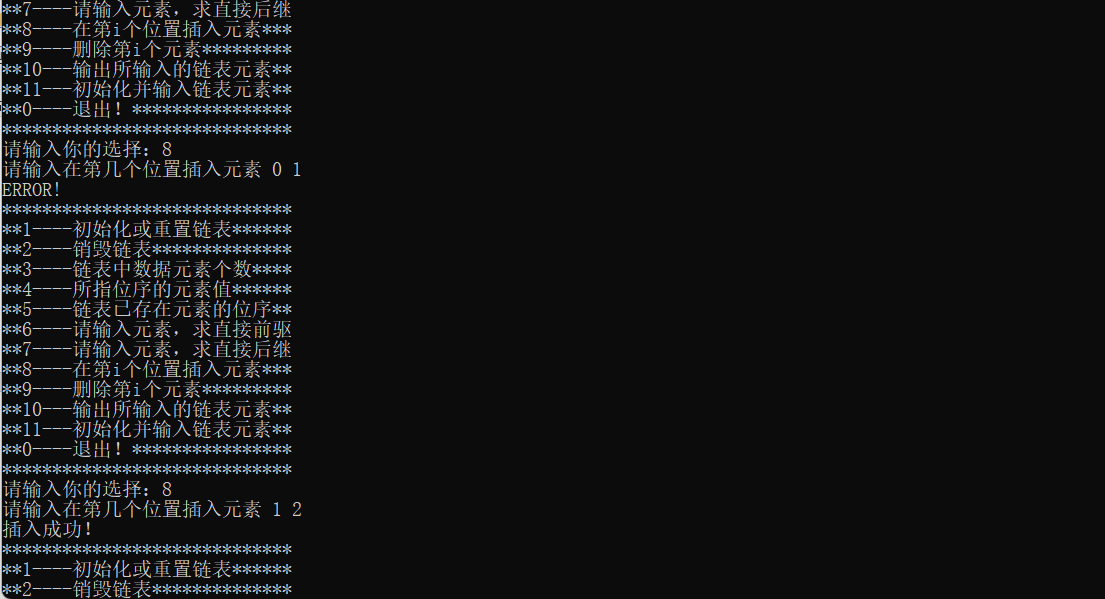


所以没有进行初始化前进行其他操作程序不能控制住。

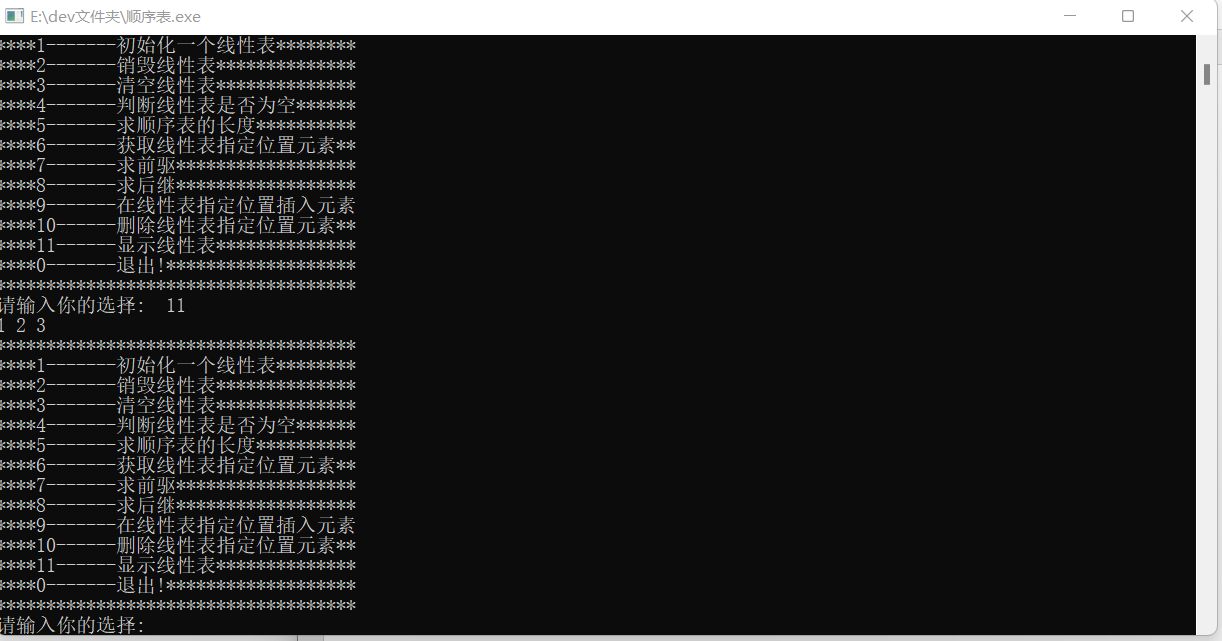
1. 初始化一个链表



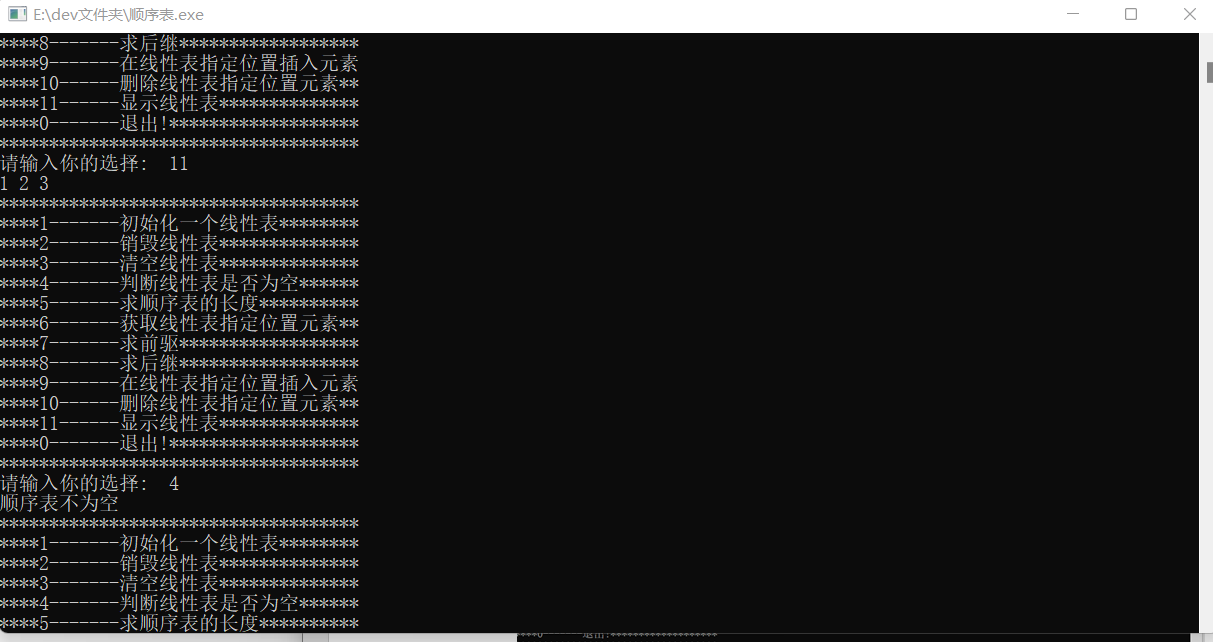
3、插入数据(位置， 数据)，要测插入位置不合法的情况（0,1）、（2,1），正确插入4个数据（1，2）、（1，1）、（3,3）



1. 显示顺序表中的数据，屏幕输出1， 2， 3



1. 判空，屏幕输出顺序表非空



6、顺序表长度，屏幕输出3



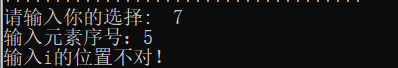
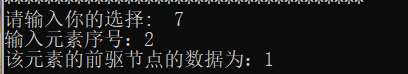
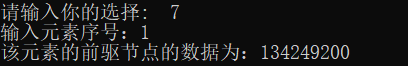
7、获取指定位置元素，要测指定位置在【1，3】范围之外的情况和之内的情况



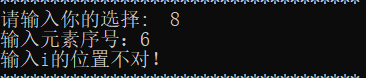
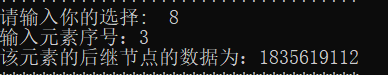
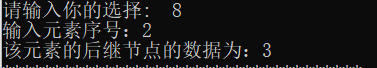
8、定位，输入：4， 输出：不存在，输入2，输出位置为2



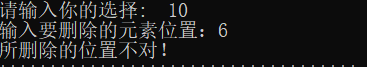
9、求直接前驱，要测求第一个元素的前驱、不存在顺序表中的元素的直接前驱，其他元素的直接前驱



10、求直接后继，要测最后一个元素的后继、不存在顺序表中的元素的直接后继，其他元素的直接后继



1. 删除，要测位置在【1，3】范围之外的情况和之内的情况



1. 清空操作后再测长度



1. 销毁链表



问题讨论：

1、没有进行初始化之前进行其他操作，程序会崩溃，应该在循环前加入判断语句。

2、第一个节点没有前驱节点，最后一个节点没有后继节点。

全部代码：

1. #include <bits/stdc++.h>
2. using namespace std;
3. *//单链表的存储结构*
4. typedef struct Node {
5. int data;
6. struct Node \*next;
7. } Node,\*LinkList;
8. *//1、初始化或重置链表*
9. void InitList(LinkList &L) {
10. L = new Node;
11. L->next = NULL;
12. cout << "初始化成功！" << endl;
13. }
14. *//2、销毁链表*
15. void DestroyList(LinkList &L) {
16. LinkList p;
17. while(L) {
18. p = L;
19. L = L->next;
20. delete p;
21. }
22. cout << "销毁成功！" << endl;
23. }
25. *//3、链表中数据元素个数*
26. void  ListLength(LinkList L) {
27. LinkList p = L->next; *//p指向第一个结点*
28. int count = 0;
29. while(p) {
30. ++count;
31. p = p->next;
32. }
33. cout << "链表中数据元素个数为：" << p << endl;;
34. }
36. *//4、所指位序的元素值*
37. void GetElem(LinkList L,int i,int &e) {
38. LinkList p = L->next;
39. int j = 1;
40. while(p && j < i) {
41. p = p->next;
42. ++j;
43. }
44. if(!p || j > i) cout << "ERROR!" << endl;
45. else {
46. e = p->data;
47. cout << "第" << i << "的元素值为：" << e << endl;
48. }
49. }
50. *//5、链表已存在元素的位序*
51. void LocateElem(LinkList L,int e) {
52. LinkList p = L->next;
53. int j = 1;
54. while(p && p->data != e) {
55. p = p->next;
56. j++;
57. }
58. cout << "该元素的位序为：" << j << endl;
59. }
61. *//6、请输入元素，求直接前驱*
62. void ListPrior(LinkList L,int e) {
63. LinkList q = L->next;*//第一个结点*
64. if(q->data == e) {
65. cout << "无前驱！" << endl;
66. }
67. else {
68. LinkList p;
69. while(q && q->data != e) {
70. p = q;
71. q = q->next;
72. }
73. cout << "该元素前驱为：" << p->data << endl;
74. }
75. }
77. *//7、请输入元素，求直接后继*
78. void ListNext(LinkList L,int e) {
79. LinkList p = L->next;
80. while(p && p->data != e) p = p->next;
81. if(p->next) cout << "该元素后继为：" << p->next->data << endl;
82. else cout << "无后继！" << endl;
84. }
86. *//8、在第i个位置插入元素*
87. void ListInsert(LinkList &L,int i,int e) {
88. LinkList p = L;
89. int j = 0;
90. while(p && (j < i - 1)) {
91. p = p->next;
92. ++j;
93. }
94. if(!p || j > i - 1) {
95. cout << "ERROR!" << endl;
96. }
97. else {
98. LinkList s = new Node;
99. s->data = e;
100. s->next = p->next;
101. p->next = s;
102. cout << "插入成功！" << endl;
103. }
104. }
105. *//9、删除第i个元素*
106. void ListDelete(LinkList &L,int i) {
107. LinkList p = L;
108. int j = 0;
109. while(p->next && j < i-1) {
110. p = p->next;
111. ++j;
112. }
113. if(!(p->next) || j > i-1) {
114. cout << "ERROR!" << endl;
115. }
116. else {
117. LinkList r = p->next;
118. p->next = r->next;
119. delete r;
120. }
121. }
122. *//10、输出所输入的链表元素*
123. void DisplayList(LinkList L) {
124. LinkList p = L->next;
125. while(p) {
126. cout << p->data << "  ";
127. p = p->next;
128. }
129. cout << endl;
130. }
131. *//11、初始化并输入链表元素*
132. void CreateList\_H(LinkList &L,int n) {
133. L = new Node;
134. L->next = NULL;
135. for(int i = 0; i < n; i++) {
136. LinkList p = new Node;
137. cin >> p->data;
138. p->next = L->next;
139. L->next = p;
140. }
141. }
142. void show() {
143. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
144. cout<<"\*\*1----初始化或重置链表\*\*\*\*\*\*"<<endl;
145. cout<<"\*\*2----销毁链表\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
146. cout<<"\*\*3----链表中数据元素个数\*\*\*\*"<<endl;
147. cout<<"\*\*4----所指位序的元素值\*\*\*\*\*\*"<<endl;
148. cout<<"\*\*5----链表已存在元素的位序\*\*"<<endl;
149. cout<<"\*\*6----请输入元素，求直接前驱"<<endl;
150. cout<<"\*\*7----请输入元素，求直接后继"<<endl;
151. cout<<"\*\*8----在第i个位置插入元素\*\*\*"<<endl;
152. cout<<"\*\*9----删除第i个元素\*\*\*\*\*\*\*\*\*"<<endl;
153. cout<<"\*\*10---输出所输入的链表元素\*\*"<<endl;
154. cout<<"\*\*11---初始化并输入链表元素\*\*"<<endl;
155. cout<<"\*\*0----退出！\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
156. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
157. }
158. int main(){
159. LinkList L;
160. int num;
161. while(1){
162. show();
163. cout << "请输入你的选择：";
164. cin >> num;
165. switch (num) {
166. case 1:
167. InitList(L);
168. break;
169. case 2:
170. DestroyList(L);
171. break;
172. case 3:
173. ListLength(L);
174. break;
175. case 4:
176. int i,e;
177. cout << "请输入位序：";
178. cin >> i;
179. GetElem(L,i,e);
180. break;
181. case 5:
182. int e1;
183. cout << "请输入该元素：";
184. cin >> e1;
185. LocateElem(L,e1);
186. break;
187. case 6:
188. int e2;
189. cout << "请输入元素：";
190. cin >> e2;
191. ListPrior(L,e2);
192. break;
193. case 7:
194. int e3;
195. cout << "请输入元素：";
196. cin >> e3;
197. ListNext(L,e3);
198. break;
199. case 8:
200. int i4,e4;
201. cout << "请输入在第几个位置插入元素 ";
202. cin >> i4 >> e4;
203. ListInsert(L,i4,e4);
204. break;
205. case 9:
206. int i5;
207. cout << "请输入删除第几个元素 ";
208. cin >> i5;
209. ListDelete(L,i5);
210. break;
211. case 10:
212. DisplayList(L);
213. break;
214. case 11:
215. int n;
216. cout << "请输入要输入的元素个数：";
217. cin >> n;
218. cout << "输入的元素为：";
219. CreateList\_H(L,n);
220. break;
221. case 12:
222. exit(0);
223. break;
224. default:
225. cout << "非法输入！请重新输入！" << endl;
226. break;
227. }
228. }
229. }