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21日 下午17时至18时，实验地点：计算机大楼213

实验题目： 顺序表的基本操作

实验目的：通过该实验，深入理解顺序表的逻辑结构、物理结构等概念，掌握顺序表基本操作的编程实现，注意顺序表插入、删除等操作过程中数据元素的移动现象，培养学生编写程序时，要考虑程序的强壮性，熟练掌握通过函数参数返回函数结果的办法。

实验环境（硬件和软件） PC和codeblocks

实验内容：编程实现顺序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：

1. 初始化一个线性表：

*//1.初始化一个线性表*

void InitList\_Sq(SqList &L){

  L.elem = new int[MAXSIZE];

  if(!L.elem) exit(OVERFLOW);

  L.length = 0;

  cout << "初始化成功！" << endl;

}

2、销毁线性表

*//2.销毁线性表*

void DestroyList(SqList &L){

    if(L.elem) delete[] L.elem;

    L.length = 0;

    L.elem = NULL;

    cout << "成功销毁线性表！" << endl;

}

3、清空线性表

*//3.清空线性表*

void ClearList(SqList &L){

    L.length = 0;

}

4、判断线性表是否为空

*//4.判断线性表是否为空*

bool IsEmpty(SqList L){

    if(L.length == 0) return true;

    else return false;

}

5、求线性表长度

*//5.求顺序表的长度*

int GetLength(SqList L){

    return L.length;

}

6、获取线性表指定位置元素

void GetElem(SqList L,int i,int &e){

    if(i < 1 || i > L.length) cout << "输入i错误！" << endl;

else{

e = L.elem[i-1];

   cout << "该元素为：" << e << endl;

  }

}

7、求前驱

*//7.求前驱*

void PriorElem(SqList L,int i,int &pe){

  if(i <= 1 || i > L.length) cout << "输入i的位置不对！" << endl;

  else{

   pe = L.elem[i-2];

   cout << "该元素的前驱节点的数据为：" << pe << endl;

  }

}

8、求后继

*//8.求后继*

void NextElem(SqList L,int i,int &ne){

  if(i < 1 || i >= L.length) cout << "输入i的位置不对！" << endl;

  else{

   ne = L.elem[i];

   cout << "该元素的后继节点的数据为：" << ne << endl;

  }

}

9、在线性表指定位置插入元素

*//9.在线性表指定位置插入元素*

void ListInsert\_Sq(SqList &L,int i,int e){

    if(i < 1 || i > L.length + 1) cout << "输入位置不合法！" << endl;

    else{

   if(L.length == MAXSIZE) cout << "空间已满！" << endl;

   else{

    for(int j = L.length - 1; j >= i - 1; j--)

     L.elem[j+1] = L.elem[j];

    L.elem[i-1] = e;

    ++L.length;

   }

  }

}

10、删除线性表指定位置元素

*//10.删除线性表指定位置元素*

void ListDelete(SqList &L,int i){

     if(i < 1 || i > L.length) cout << "所删除的位置不对！" << endl;

     else{

   for(int j = i; j < L.length; j++)

       L.elem[j-1] = L.elem[j];

   --L.length;

  }

}

11、显示线性表

*//11.显示线性表*

void TraverseList(SqList L){

    for(int i = 0; i < L.length; i++)

        cout << L.elem[i] << " ";

    cout << endl;

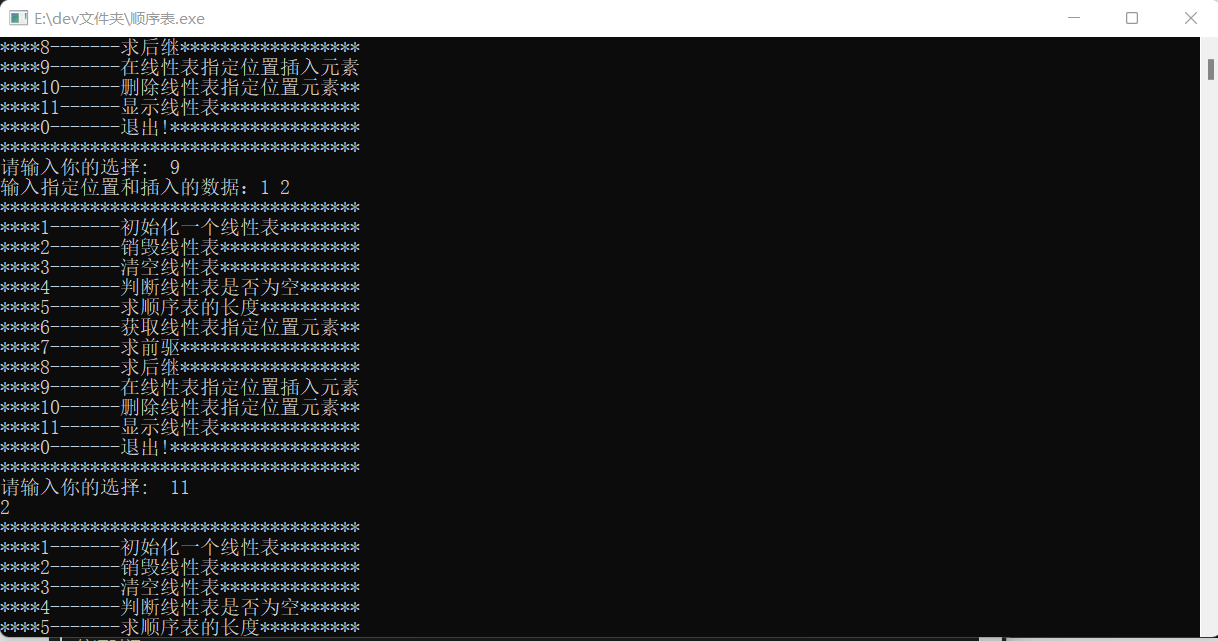
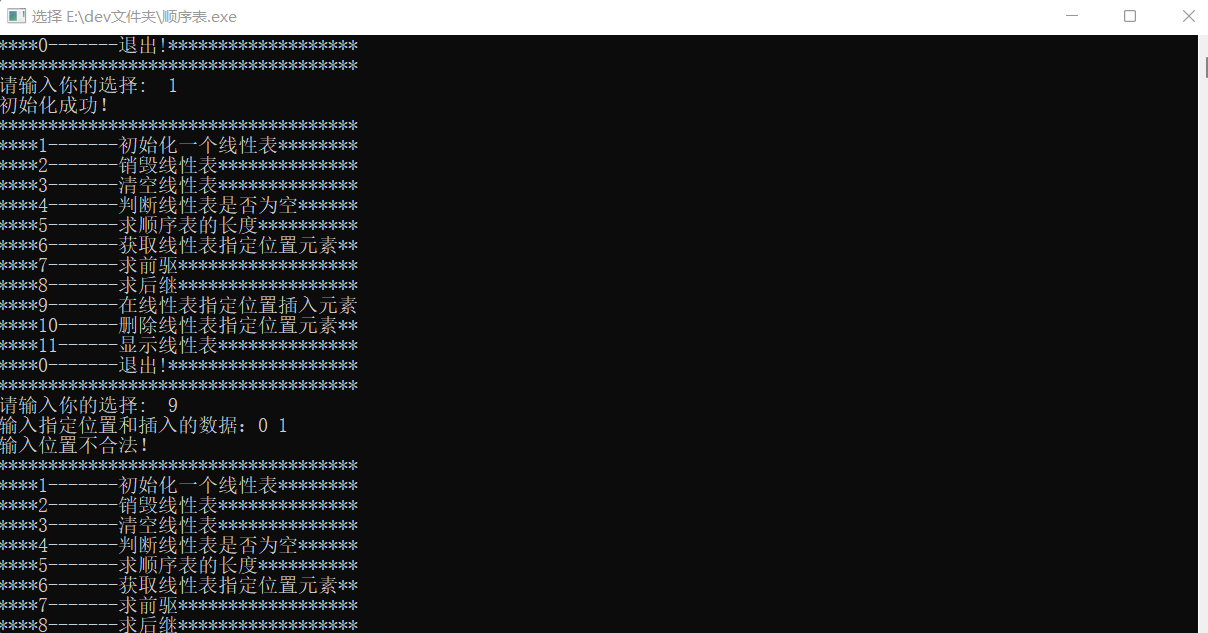
}

实验数据记录：

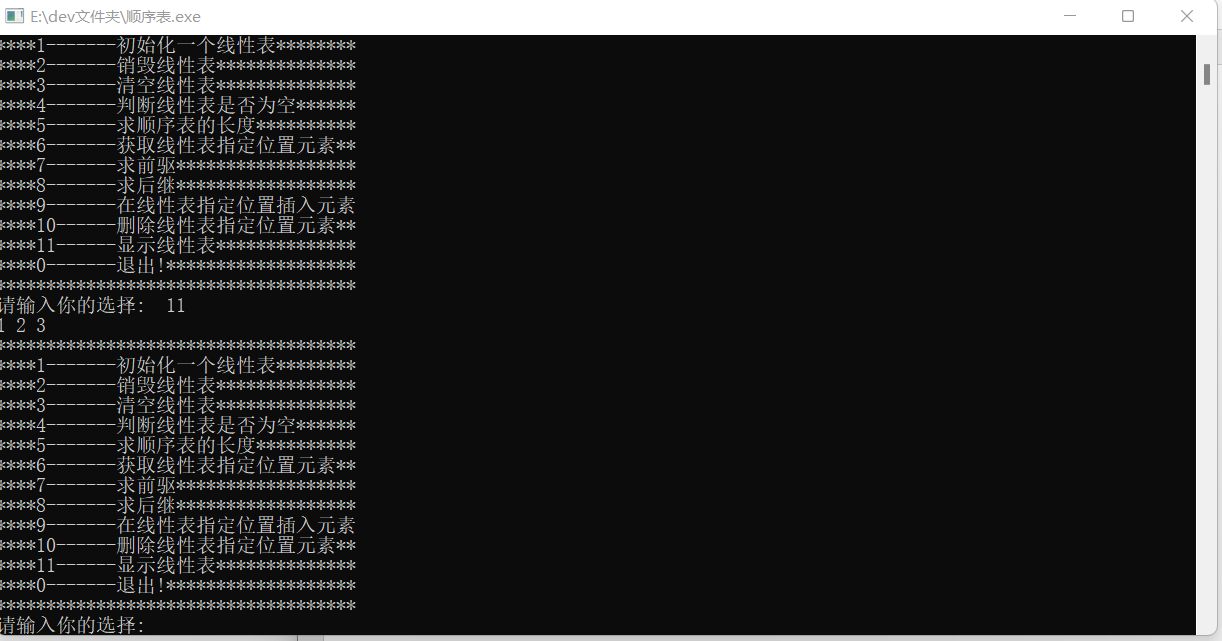
1. 没有初始化前进行其他操作

输入：5 输出：顺序表的长度为：4245856

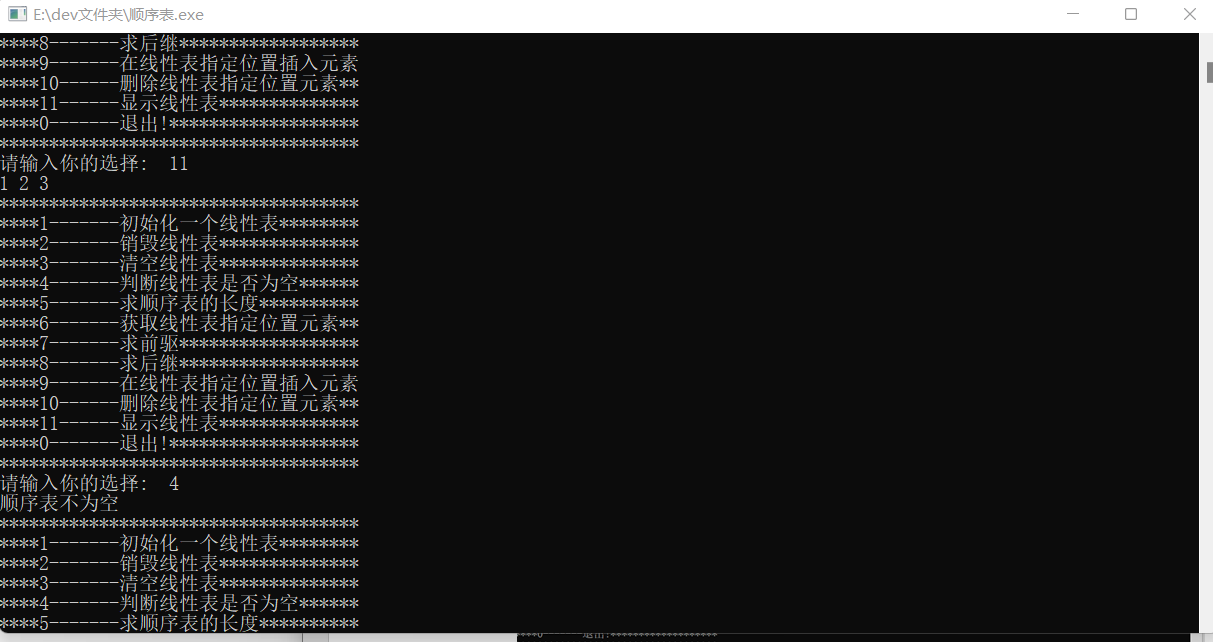
2、插入数据(位置， 数据)，要测插入位置不合法的情况（0,1）、（2,1），正确插入4个数据（1，2）、（1，1）、（3,3）



3、显示顺序表中的数据，屏幕输出1， 2， 3



4、判空，屏幕输出顺序表非空



5、顺序表长度，屏幕输出3



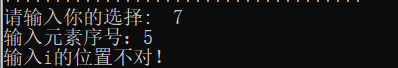
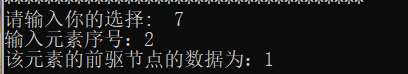
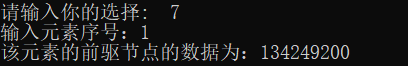
6、获取指定位置元素，要测指定位置在【1，3】范围之外的情况和之内的情况



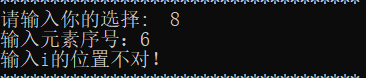
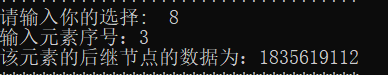
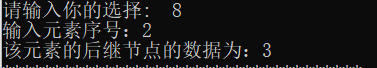
7、定位，输入：4， 输出：不存在，输入2，输出位置为2



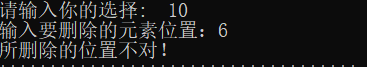
8、求直接前驱，要测求第一个元素的前驱、不存在顺序表中的元素的直接前驱，其他元素的直接前驱



9、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



10、删除，要测位置在【1，3】范围之外的情况和之内的情况



11、清空操作后再测长度

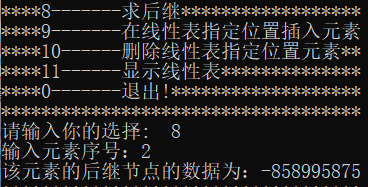


12、销毁顺序表



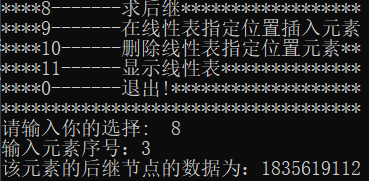
问题讨论：

1. 没有初始化前进行其他操作，程序会崩溃



 为什么要初始化？  
  首先elem是一个未初始化的指针，在使用未初始化的指针时，编译器是会出现读取错误的。而length是因为需要从0累计算的，这些方便处理后期使用的数据。

2、求顺序表最后一个元素的后继时，或者第一个元素的前驱时，应注意判断条件，因为在顺序表里除第一个元素只有一个后继和最后一个元素只有前驱外，其他的都有一个前驱和一个后继。



全部代码：

1. #include <bits/stdc++.h>
2. using namespace std;
3. #define MAXSIZE 100
4. typedef struct{
5. int \*elem;
6. int length;
7. }SqList;
8. *//1.初始化一个线性表*
9. void InitList\_Sq(SqList &L){
10. L.elem = new int[MAXSIZE];
11. if(!L.elem) exit(OVERFLOW);
12. L.length = 0;
13. cout << "初始化成功！" << endl;
14. }
15. *//2.销毁线性表*
16. void DestroyList(SqList &L){
17. if(L.elem) delete[] L.elem;
18. L.length = 0;
19. L.elem = NULL;
20. cout << "成功销毁线性表！" << endl;
21. }
22. *//3.清空线性表*
23. void ClearList(SqList &L){
24. L.length = 0;
25. }
26. *//5.求顺序表的长度*
27. int GetLength(SqList L){
28. return L.length;
29. }
30. *//4.判断线性表是否为空*
31. bool IsEmpty(SqList L){
32. if(L.length == 0) return true;
33. else return false;
34. }
35. *//6.获取线性表指定位置元素*
36. void GetElem(SqList L,int i,int &e){
37. if(i < 1 || i > L.length) cout << "输入i错误！" << endl;
38. else{
39. e = L.elem[i-1];
40. cout << "该元素为：" << e << endl;
41. }
42. }
43. *//7.求前驱*
44. void PriorElem(SqList L,int i,int &pe){
45. if(i <= 1 || i > L.length) cout << "输入i的位置不对！" << endl;
46. else{
47. pe = L.elem[i-2];
48. cout << "该元素的前驱节点的数据为：" << pe << endl;
49. }
50. }
51. *//8.求后继*
52. void NextElem(SqList L,int i,int &ne){
53. if(i < 1 || i >= L.length) cout << "输入i的位置不对！" << endl;
54. else{
55. ne = L.elem[i];
56. cout << "该元素的后继节点的数据为：" << ne << endl;
57. }
58. }
59. *//9.在线性表指定位置插入元素*
60. void ListInsert\_Sq(SqList &L,int i,int e){
61. if(i < 1 || i > L.length + 1) cout << "输入位置不合法！" << endl;
62. else{
63. if(L.length == MAXSIZE) cout << "空间已满！" << endl;
64. else{
65. for(int j = L.length - 1; j >= i - 1; j--)
66. L.elem[j+1] = L.elem[j];
67. L.elem[i-1] = e;
68. ++L.length;
69. }
71. }
72. }
73. *//10.删除线性表指定位置元素*
74. void ListDelete(SqList &L,int i){
75. if(i < 1 || i > L.length) cout << "所删除的位置不对！" << endl;
76. else{
77. for(int j = i; j < L.length; j++)
78. L.elem[j-1] = L.elem[j];
79. --L.length;
80. }
81. }
82. *//11.显示线性表*
83. void TraverseList(SqList L){
84. for(int i = 0; i < L.length; i++)
85. cout << L.elem[i] << " ";
86. cout << endl;
87. }
88. void show(){
89. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
90. cout << "\*\*\*\*1-------初始化一个线性表\*\*\*\*\*\*\*\*" << endl;
91. cout << "\*\*\*\*2-------销毁线性表\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
92. cout << "\*\*\*\*3-------清空线性表\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
93. cout << "\*\*\*\*4-------判断线性表是否为空\*\*\*\*\*\*" << endl;
94. cout << "\*\*\*\*5-------求顺序表的长度\*\*\*\*\*\*\*\*\*\*" << endl;
95. cout << "\*\*\*\*6-------获取线性表指定位置元素\*\*" << endl;
96. cout << "\*\*\*\*7-------求前驱\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
97. cout << "\*\*\*\*8-------求后继\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
98. cout << "\*\*\*\*9-------在线性表指定位置插入元素" << endl;
99. cout << "\*\*\*\*10------删除线性表指定位置元素\*\*" << endl;
100. cout << "\*\*\*\*11------显示线性表\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
101. cout << "\*\*\*\*0-------退出!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
102. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
103. }
104. int main()
105. {
106. bool b = false;
107. int n;
108. SqList L;
109. while(1)
110. {
111. show();
112. cout << "请输入你的选择:  ";
113. cin >> n;
114. switch(n)
115. {
116. case 1:
117. InitList\_Sq(L);
118. b = true;
119. break;
120. case 2:
121. if(b) DestroyList(L);
122. else cout << "请先对顺序表初始化！" << endl;
123. break;
124. case 3:
125. if(b) ClearList(L);
126. else cout << "请先对顺序表初始化！" << endl;
127. break;
128. case 4:
129. if(b){
130. if(IsEmpty(L)) cout << "顺序表为空" << endl;
131. else cout << "顺序表不为空" << endl;
132. }
133. else cout << "请先对顺序表初始化！" << endl;
134. break;
135. case 5:
136. if(b) cout << "顺序表的长度为：" << GetLength(L) << endl;
137. else cout << "请先对顺序表初始化！" << endl;
138. break;
139. case 6:
140. if(b){
141. int i1,e1;
142. cout << "输入元素位置：";
143. cin >> i1;
144. GetElem(L,i1,e1);
145. }
146. else cout << "请先对顺序表初始化！" << endl;
147. break;
148. case 7:
149. if(b){
150. int i2,e2;
151. cout << "输入元素序号：";
152. cin >> i2;
153. PriorElem(L,i2,e2);
154. }
155. else cout << "请先对顺序表初始化！" << endl;
156. break;
157. case 8:
158. if(b){
159. int i3,e3;
160. cout << "输入元素序号：";
161. cin >> i3;
162. NextElem(L,i3,e3);
163. }
164. else cout << "请先对顺序表初始化！" << endl;
165. break;
166. case 9:
167. if(b){
168. int i4,e4;
169. cout << "输入指定位置和插入的数据：";
170. cin >> i4 >> e4;
171. ListInsert\_Sq(L,i4,e4);
172. }
173. else cout << "请先对顺序表初始化！" << endl;
174. break;
175. case 10:
176. if(b){
177. int i5;
178. cout << "输入要删除的元素位置：";
179. cin >> i5;
180. ListDelete(L,i5);
181. }
182. else cout << "请先对顺序表初始化！" << endl;
183. break;
184. case 11:
185. if(b){
186. TraverseList(L);
187. }
188. else cout << "请先对顺序表初始化！" << endl;
189. break;
190. case 0:
191. exit(0);
192. default:
193. cout<<"非法输入，请重新输入!"<<endl;
194. break;
195. }
196. }
197. return 0;
198. }