**作业 HW1 实验报告**

姓名：何正潇 学号：1950095日期：2021年10月13日

1. **涉及数据结构和相关背景**

**本单元的数据结构主要涉及的都是线性结构，主要题目涉及的是顺序表和链表数据结构的考察。且可以使用静态和动态的内存申请进行编写程序。**

**2. 实验内容**

**2.1 最大子段和（题目名字）**

**2.1.1 问题描述**

# 简短地描述题目

**2.1.2 基本要求**

# 简短地描述题目要求

**2.1.3 数据结构设计**

# 图、描述均可

**2.1.4功能说明（函数、类）**

/\*

\* int add(int a, int b)

\*@para a 求和的第一个数

\*@para b 求和的第二个数

\*/

int add(int a, int b) // 求a+b的结果

class A {

/\*

\* int add(int a, int b)

\*@para a 求和的第一个数

\*@para b 求和的第二个数

\*/

int add(int a, int b) // 求a+b的结果

…

}

# 总之清晰描述类、函数的功能，参数、输出含义即可

**2.1.5 调试分析（遇到的问题和解决方法）**

# 简要描述调试过程

# 可以使用图片辅助说明

**2.1.6 总结和体会**

# 可以说说做此题的收获、分析这道题目的难点和易错点（数据边界、对算法效率的要求等）

**1-2 题目二有序表的合并问题**

**1.2.1 问题描述**

**已知线性表La和Lb的元素按值非递减有序排列，现将La和Lb归并成一个新的线性有序表Lc，且Lc中的数据元素仍然按照值非递减有序排列。输入La和Lb值的时候可以非有序，输出合并后的结果。**

**1.2.2 基本要求**

对La和Lb进行排序，然后采用归并排序算法输出Lc。

**1.2.3 数据结构设计**

1. **typedef** **struct** numbers
2. {
3. Elemtype**\*** elem**;**
4. **int** length**;**
5. **int** listsize**;**
6. } SQlist**;**

进行结构体定义，elem用来申请保存顺序表的空间，length用来记录顺序表的长度，listsize用来记录现在顺序表申请空间的大小。

**1.2.4功能说明（函数、类）**

1. **#include** <iostream>
2. **#include**<string.h>
3. using namespace std**;**
4. **#define** LIST\_INIT\_SIZE 10
5. **#define** LISTINCREMENT 10
6. **typedef** **int** Elemtype**;**
7. **typedef** **int** Status**;**
8. **typedef** **struct** numbers
9. {
10. Elemtype**\*** elem**;**
11. **int** length**;**
12. **int** listsize**;**
13. } SQlist**;**
14. //顺序表初始化程序
15. Status Initlist\_sq**(**SQlist**&** L**)**
16. {
17. L**.**elem **=** **(**Elemtype**\*)**malloc**(**LIST\_INIT\_SIZE **\*** **sizeof(**Elemtype**));**
18. **if** **(!**L**.**elem**)**
19. **return** **-**1**;**
20. L**.**length **=** 0**;**
21. L**.**listsize **=** LIST\_INIT\_SIZE**;**
22. **int** i **=** 1**;**
23. **while** **(**1**)**
24. {
25. **int** temp**;**
26. cin **>>** temp**;**
27. **if** **(**temp **==** 0**)**
28. **break;**
29. **if** **((**L**.**length+1**)** **>=** L**.**listsize**)**
30. {
31. Elemtype**\*** newbase**;**
32. newbase **=** **(**Elemtype**\*)**realloc**(**L**.**elem**,** **(**LISTINCREMENT**+**L**.**listsize**)** **\*** **sizeof(**Elemtype**));**
33. **if** **(!**newbase**)**
34. **return** **-**1**;**
35. L**.**elem **=** newbase**;**
36. L**.**listsize **=** L**.**listsize **+** LISTINCREMENT**;**
37. }
38. L**.**elem**[**i**++]** **=** temp**;**
39. L**.**length**++;**
40. }
41. **return** 0**;**
42. }
43. Status Initlist\_sq\_**(**SQlist**&** L**)**
44. {
45. L**.**elem **=** **(**Elemtype**\*)**malloc**(**LIST\_INIT\_SIZE **\*** **sizeof(**Elemtype**));**
46. **if** **(!**L**.**elem**)**
47. **return** **-**1**;**
48. L**.**length **=** 0**;**
49. L**.**listsize **=** LIST\_INIT\_SIZE**;**
50. **if** **((**L**.**length **+** 1**)** **>=** L**.**listsize**)**
51. {
52. Elemtype**\*** newbase**;**
53. newbase **=** **(**Elemtype**\*)**realloc**(**L**.**elem**,** **(**LISTINCREMENT **+** L**.**listsize**)** **\*** **sizeof(**Elemtype**));**
54. **if** **(!**newbase**)**
55. **return** **-**1**;**
56. L**.**elem **=** newbase**;**
57. L**.**listsize **=** L**.**listsize **+** LISTINCREMENT**;**
58. }
59. **return** 0**;**
60. }
61. //顺序表插入程序
62. Status ListInsert**(**SQlist**&** L**,** **int** i**,** Elemtype e**)**
63. {
64. **if** **(**i**<**1 **||** i**>(**L**.**length+1**)** **)**
65. **return** **-**1**;**
66. **if** **((**L**.**length **+** 1**)** **>=** L**.**listsize**)** {
67. Elemtype**\*** newbase**;**
68. newbase **=** **(**Elemtype**\*)**realloc**(**L**.**elem**,** **(**LISTINCREMENT **+** L**.**listsize**)** **\*** **sizeof(**Elemtype**));**
69. **if** **(!**newbase**)**
70. **return** **-**1**;**
71. L**.**elem **=** newbase**;**
72. L**.**listsize **=** L**.**listsize **+** LISTINCREMENT**;**
73. }
74. Elemtype**\*** q **=** **&(**L**.**elem**[**i **]);**
75. **for** **(**Elemtype**\*** p **=** **&(**L**.**elem**[**L**.**length **]);** p **>=** q**;** p**--)** {
76. memcpy**(**p **+** 1**,** p**,** **sizeof(**Elemtype**));**
77. }
78. memcpy**(**q**,** **&**e**,** **sizeof(**Elemtype**));**
79. L**.**length**++;**
80. **return** 0**;**
81. }
82. //顺序表La和Lb的归并排序程序
83. **void** Mergelist**(**SQlist La**,** SQlist Lb**,** SQlist**&** Lc**)**
84. {
85. **int** i**=**1**,** j **=** 1**;**
86. **int** k **=** 0**;**
87. **int** La\_len **=** La**.**length**,** Lb\_len **=** Lb**.**length**;**
88. **while** **((**i **<=** La\_len**)** **&&** **(**j **<=** Lb\_len**))**
89. {
90. **int** ai **=** La**.**elem**[**i **],** bj **=** Lb**.**elem**[**j **];**
91. **if** **(**ai **<=** bj**)**
92. {
93. ListInsert**(**Lc**,** **++**k**,** ai**);**
94. **++**i**;**
95. }
96. **else**
97. {
98. ListInsert**(**Lc**,** **++**k**,** bj**);**
99. **++**j**;**
100. }
101. }
102. **while** **(**i **<=** La\_len**)**
103. {
104. **int** ai **=** La**.**elem**[**i **];**
105. i**++;**
106. ListInsert**(**Lc**,** **++**k**,** ai**);**
107. }
108. **while** **(**j **<=** Lb\_len**)**
109. {
110. **int** bj **=** Lb**.**elem**[**j**];**
111. j**++;**
112. ListInsert**(**Lc**,** **++**k**,** bj**);**
113. }
114. }
115. //顺序表内部的冒泡排序程序
116. **void** Line\_up**(**SQlist**&** La**)**
117. {
118. **for** **(int** i **=** 0**;** i **<** La**.**length-1**;** i**++)**
119. {
120. **for** **(int** j **=** 1**;** j **<** La**.**length**-**i**;** j**++)**
121. {
122. **if** **(**La**.**elem**[**j**]** **>=** La**.**elem**[**j **+** 1**])**
123. {
124. **int** t **=** La**.**elem**[**j**];**
125. La**.**elem**[**j**]** **=** La**.**elem**[**j **+** 1**];**
126. La**.**elem**[**j **+** 1**]** **=** t**;**
127. }
128. }
129. }
130. }
131. **int** main**()**
132. {
133. SQlist a**,** b**,**c**;**
134. Initlist\_sq**(**a**),** Initlist\_sq**(**b**);**
135. Line\_up**(**a**),** Line\_up**(**b**);**
136. Initlist\_sq\_**(**c**);**
137. Mergelist**(**a**,** b**,** c**);**
138. **for** **(int** i **=** 1**;** i **<** c**.**length+1**;** i**++)**
139. cout **<<** c**.**elem**[**i**]<<**" "**;**
140. **return** 0**;**
141. }

**1.2.5 调试分析（遇到的问题和解决方法）**

**主要的难点在于线性表的排序和归并算法设计，我在这里使用的是冒泡排序方法和比较经典的归并排序算法。**

**1.2.6 总结和体会**

**通过这道题主要是体验归并排序的算法以及线性表的基本操作实践，难度总体来说不大。**

**1.4 题目四一元多项式的相加和相乘**

**1.4.1问题描述**

**输入两个线性表，进行线性表中保存的一元多项式的相加和相乘，同时按照指数从小到大的顺序进行输出**

**1.4.2基本要求**

**输入一元多项式长度，输入线性表La的系数以及指数，输入第二个一元多项式的长度，输入线性表Lb的系数以及指数。根据输入的命令，进行一元多项式的乘法和加法运算。**

**1.4.3**

**数据结构设计**

1. **typedef** **struct** poly {
2. **int** p**;***//系数*
3. **int** coe**;***//指数*
4. }poly**;**
6. **struct** list {
7. **struct** poly**\*** Elem**;**
8. **int** length**;**
9. }**;**

**设定数据结构poly记录一元多项式的次数和系数，同时结构体list保存线性表的首地址，同时也记录线性表的长度。以便后面的乘法和加法运算。**

**1.4.4功能说明（函数、类）**

1. */\*1950095 大数据 何正潇\*/*
2. **#define** \_CRT\_SECURE\_NO\_WARNINGS
3. **#include** <iostream>
4. **#include** <cstring>
5. using namespace std**;**
6. */\**
7. *注释:此处用顺序表较为简单，因此没有使用链表，基本使用的就是归并排序的思想，与教科书例题代码相似。*
8. *\*/*
9. **typedef** **struct** poly {
10. **int** p**;***//系数*
11. **int** coe**;***//指数*
12. }poly**;**
14. **struct** list {
15. **struct** poly**\*** Elem**;**
16. **int** length**;**
17. }**;**
18. */\*冒泡排序\*/*
19. **void** sort**(**list**&** L**)** {
20. poly k**;**
21. **for** **(int** i **=** 1**;** i **<=** L**.**length **-** 1**;** i**++)**
22. **for** **(int** j **=** 1**;** j **<=** L**.**length **-** i**;** j**++)**
23. **if** **(**L**.**Elem**[**j**].**coe **>** L**.**Elem**[**j **+** 1**].**coe**)**
24. {
25. k **=** L**.**Elem**[**j**];**
26. L**.**Elem**[**j**]** **=** L**.**Elem**[**j **+** 1**];**
27. L**.**Elem**[**j **+** 1**]** **=** k**;**
28. }
29. }
31. */\*输出多项式\*/*
32. **void** print**(**list**&** L**)** {
33. **for** **(int** i **=** 1**;** i **<=** L**.**length**;** i**++)** *//输出*
34. **if** **(**L**.**Elem**[**i**].**p **!=** 0**)** *//系数不为0输出*
35. cout **<<** L**.**Elem**[**i**].**p **<<** " " **<<** L**.**Elem**[**i**].**coe **<<** " "**;**
36. cout **<<** endl**;**
37. }
38. */\*归并排序\*/*
39. **void** add**(**list**&** L1**,** list**&** L2**,** list**&** L3**)** {
40. sort**(**L1**);**
41. sort**(**L2**);**
42. **int** i **=** 1**,** j **=** 1**,** k **=** 1**;**
43. **while** **(**i **<=** L1**.**length **&&** j **<=** L2**.**length**)** {
44. **if** **(**L1**.**Elem**[**i**].**coe **<** L2**.**Elem**[**j**].**coe**)**
45. L3**.**Elem**[**k**++]** **=** L1**.**Elem**[**i**++];**
46. **else** **if** **(**L1**.**Elem**[**i**].**coe **>** L2**.**Elem**[**j**].**coe**)**
47. L3**.**Elem**[**k**++]** **=** L2**.**Elem**[**j**++];**
48. **else** {
49. L3**.**Elem**[**k**].**coe **=** L1**.**Elem**[**i**].**coe**;**
50. L3**.**Elem**[**k**++].**p **=** L1**.**Elem**[**i**++].**p **+** L2**.**Elem**[**j**++].**p**;**
51. }
52. }
53. **while** **(**i **<=** L1**.**length**)**
54. L3**.**Elem**[**k**++]** **=** L1**.**Elem**[**i**++];**
55. **while** **(**j **<=** L2**.**length**)**
56. L3**.**Elem**[**k**++]** **=** L2**.**Elem**[**j**++];**
57. L3**.**length **=** k **-** 1**;**
58. }
59. */\*项数Delete\*/*
60. **void** Delete**(**list**&** L**,** **int** position**)**
61. {
62. **for** **(int** i **=** position**;** i **<=** L**.**length**;** i**++)**
63. L**.**Elem**[**i**]** **=** L**.**Elem**[**i **+** 1**];**
64. L**.**length**--;**
65. }
66. */\*乘法\*/*
67. **void** multiply**(**list**&** L1**,** list**&** L2**,** list**&** L3**)** {
68. **int** i**,** j**,** k **=** 1**;**
69. **for** **(**i **=** 1**;** i **<=** L1**.**length**;** i**++)**
70. **for** **(**j **=** 1**;** j **<=** L2**.**length**;** j**++)**
71. {
72. L3**.**Elem**[**k**].**p **=** L1**.**Elem**[**i**].**p **\*** L2**.**Elem**[**j**].**p**;**
73. L3**.**Elem**[**k**].**coe **=** L1**.**Elem**[**i**].**coe **+** L2**.**Elem**[**j**].**coe**;**
74. k**++;**
75. }
76. L3**.**length **=** L1**.**length **\*** L2**.**length**;**
77. sort**(**L3**);**
78. **for** **(int** i **=** 1**;** i **<** L3**.**length**;** i**++)** {
79. **if** **(**L3**.**Elem**[**i**].**coe **==** L3**.**Elem**[**i **+** 1**].**coe**)**
80. {
81. L3**.**Elem**[**i**].**p **+=** L3**.**Elem**[**i **+** 1**].**p**;**
82. Delete**(**L3**,** i **+** 1**);**
83. i**--;***//此处需要注意*
84. }
85. }
86. }
88. **int** main**()**
89. {
90. list L1**,** L2**,** L3**;**
91. L1**.**Elem **=** **(**poly**\*)**malloc**(**10000 **\*** **sizeof(**poly**));**
92. L2**.**Elem **=** **(**poly**\*)**malloc**(**10000 **\*** **sizeof(**poly**));**
93. L3**.**Elem **=** **(**poly**\*)**malloc**(**10000 **\*** **sizeof(**poly**));**
94. L3**.**length **=** 0**;**
95. cin **>>** L1**.**length**;**
96. **for** **(int** i **=** 1**;** i **<=** L1**.**length**;** i**++)**
97. cin **>>** L1**.**Elem**[**i**].**p **>>** L1**.**Elem**[**i**].**coe**;***//输入*
98. cin **>>** L2**.**length**;**
99. **for** **(int** i **=** 1**;** i **<=** L2**.**length**;** i**++)**
100. cin **>>** L2**.**Elem**[**i**].**p **>>** L2**.**Elem**[**i**].**coe**;***//输入*
101. **int** order**;**
102. cin **>>** order**;//输入命令**
103. **if** **(**order **==** 0**)** {
104. add**(**L1**,** L2**,** L3**);**
105. print**(**L3**);**
106. }
107. **else** **if** **(**order **==** 1**)** {
108. multiply**(**L1**,** L2**,** L3**);**
109. print**(**L3**);**
110. }
111. **else** **if** **(**order **==** 2**)** {
112. add**(**L1**,** L2**,** L3**);**
113. print**(**L3**);**
114. multiply**(**L1**,** L2**,** L3**);**
115. print**(**L3**);**
116. }
117. free**(**L1**.**Elem**);**
118. free**(**L2**.**Elem**);**
119. free**(**L3**.**Elem**);**
120. **return** 0**;**
121. }

**1.4.5 调试分析（遇到的问题和解决方法）**

**本题难度相对来说较低，但是综合性较高，其中包含了归并排序，冒泡排序，链表的合成等多个知识点。**

**1.4.6 总结和体会**

**通过这道题主要是体验归并排序的算法以及线性表的基本操作实践，我本来打算在这里使用链表进行操作，但是发现实际上在本题中链表操作并不简单，于是改用顺序表，调试一次以后通过本次程序。**

**1-5 级数相加**

**1.5.1问题描述**

若干行，在每一行中给出整数N和A的值，（1<=N<=150，0<=A<=15），计算得到级数运算的结果。

**1.5.2基本要求**

**由于该题中涉及大数的乘法和加法，所以需要用数组保存每一位的数字。才能保证大数字的计算值正确。**

**1.5.3数据结构设计**

**int** answer**[**400**]**

**该题主要考察的是简单顺序表的应用，所以不需要复杂的数据结构。**

**1.5.4功能说明（函数、类）**

1. */\*1950095 大数据 何正潇\*/*
2. **#include** <iostream>
3. **#include** <cstring>
4. using namespace std**;**
6. */\*函数功能：实现高精度乘法\*/*
7. **void** mul**(int** num1**[**400**],** **int** answer**[**400**],** **int** num2**,** **int&** length**,int&** length\_**)**
8. {
9. **int** i **=** 0**;**
10. **int** residue **=** 0**;**
11. **for** **(**i **=** 0**;** i **<** length**;** i**++)**
12. {
13. **int** temp **=** num1**[**i**];**
14. answer**[**i**]** **=** **(**residue **+** num2 **\*** **(**num1**[**i**]))** **%** 10**;**
15. residue **=** **(**residue **+** num2 **\*** **(**temp**))** **/** 10**;**
17. }
18. **while** **(**residue**)**
19. {
21. answer**[**i**]** **=** residue **%** 10 **;**
22. residue **=** residue **/** 10**;**
23. length**++;**
24. i**++;**
25. }
26. length\_ **=** length**;**
28. }
29. */\*函数功能：实现高精度加法\*/*
30. **void** add**(int** num1**[**400**],** **int** num2**[**400**],** **int** answer**[**400**],** **int&** length**,** **int&** length\_**)**
31. {
32. **int** residue **=** 0**;**
33. **int** i **=** 0**;**
34. **for** **(**i **=** 0**;** i **<** length\_**;** i**++)**
35. {
36. **int** temp **=** num2**[**i**];**
37. answer**[**i**]** **=** **(**residue **+** num2**[**i**]** **+** num1**[**i**])** **%** 10**;**
38. residue **=** **(**residue **+** num1**[**i**]** **+** temp**)** **/** 10**;**
39. }
40. **while** **(**residue**)**
41. {
43. answer**[**i**]** **=** residue **%** 10**;**
44. residue **=** residue **/** 10**;**
45. length\_**++;**
46. i**++;**
47. }
49. }
50. **int** main**()**
51. {
52. **int** N**,** A**;**
53. cin **>>** N **>>** A**;**
54. **int** length **=** 0**;**
55. **int** length\_ **=** 0**;**
56. **if** **(**A **<** 10 **&&** A **>=** 0**)**
57. length\_ **=** 1**;**
58. **else**
59. length\_ **=** 2**;**
60. length **=** length\_**;**
61. **int** num1**[**400**];**
62. **int** num2**[**400**];//没有必要，可以忽略**
63. **int** answer**[**400**];**
64. memset**(**answer**,** 0**,** 1600**);**
65. memset**(**num1**,** 0**,** 1600**);**
66. num1**[**0**]** **=** A **%** 10**,** num1**[**1**]** **=** A **/** 10 **%** 10**,** num1**[**2**]** **=** A **/** 10 **/** 10 **%** 10**;**
67. **for** **(int** i **=** 1**;** i **<=** N**;** i**++)**
68. {
69. **if** **(**A **<** 10 **&&** A **>=** 0**)**
70. length **=** 1**;**
71. **else**
72. length **=** 2**;**
73. **if** **(**i **==** 1**)**
74. {
75. **if** **(**length\_ **==** 1**)**
76. answer**[**0**]** **=** A**;**
77. **else**
78. {
79. answer**[**0**]** **=** A **%** 10**;**
80. answer**[**1**]** **=** A **/** 10 **%** 10**;**
81. }
82. }
83. **else**
84. {
85. **for** **(int** j **=** 1**;** j **<** i**;** j**++)**
86. {
87. mul**(**num1**,** num1**,** A**,** length**,**length\_**);**
88. }
89. mul**(**num1**,** num1**,** i**,** length**,** length\_**);**
90. add**(**num1**,** answer**,** answer**,** length**,**length\_**);**
91. **for** **(int** i **=** 3**;** i **<** length\_**;** i**++)**
92. num1**[**i**]** **=** 0**;**
93. num1**[**0**]** **=** A **%** 10**,** num1**[**1**]** **=** A **/** 10 **%** 10**,** num1**[**2**]** **=** A **/** 10 **/** 10 **%** 10**;**
95. }
97. }
98. **for** **(int** i **=** length\_ **-** 1**;** i **>=** 0**;** i**--)**
99. cout **<<** answer**[**i**];**
100. */\*    for (int i = 1; i <= N; i++)*
101. *{*
102. *for (int j = 1; j <= i; j++)*
103. *{*
104. *mul(num1,answer,)*
105. *}*
106. *}\*/*
107. **return** 0**;**
108. }

**1.5.5 调试分析（遇到的问题和解决方法）**

**本题总体来说数据结构的考察难度不高，但是其中也容易遇到问题，在几个函数中需要设定临时变量temp记录数值，否则会导致问题。程序的核心在于每一位的运算逻辑。**

**1.5.6 总结和体会**

**本题运用了顺序表进行大数字的乘法和加法，总体趣味性较高，也为超过变量保存范围的数字运算方法提供了一种新的思路。**