

电子信息与通信学院

实 验 报 告

|  |  |
| --- | --- |
| 实验名称 | 课程综合练习 |
| 课程名称 | 计算机基础  与程序设计(C) |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 邹世杰 | 学号 | U202413721 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2024.12 | 地点 | 华中科技大学 |

|  |  |  |  |
| --- | --- | --- | --- |
| 成绩 |  | 教师 | 刘威 |

# 实验目的

完成大数计算系列代码（日历系列、大数计算系列，选择其中之一）。

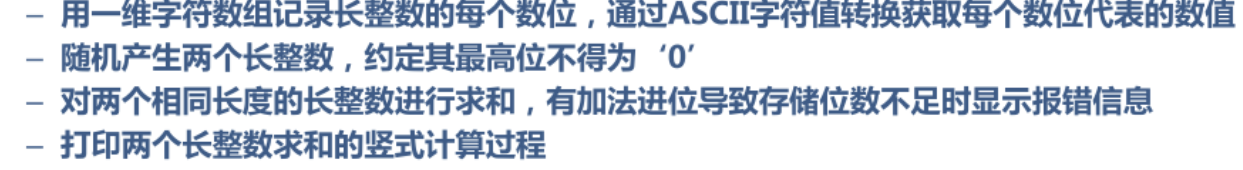
# 实验环境

操作系统：Windows 10

编程工具：CodeBlocks 16.01

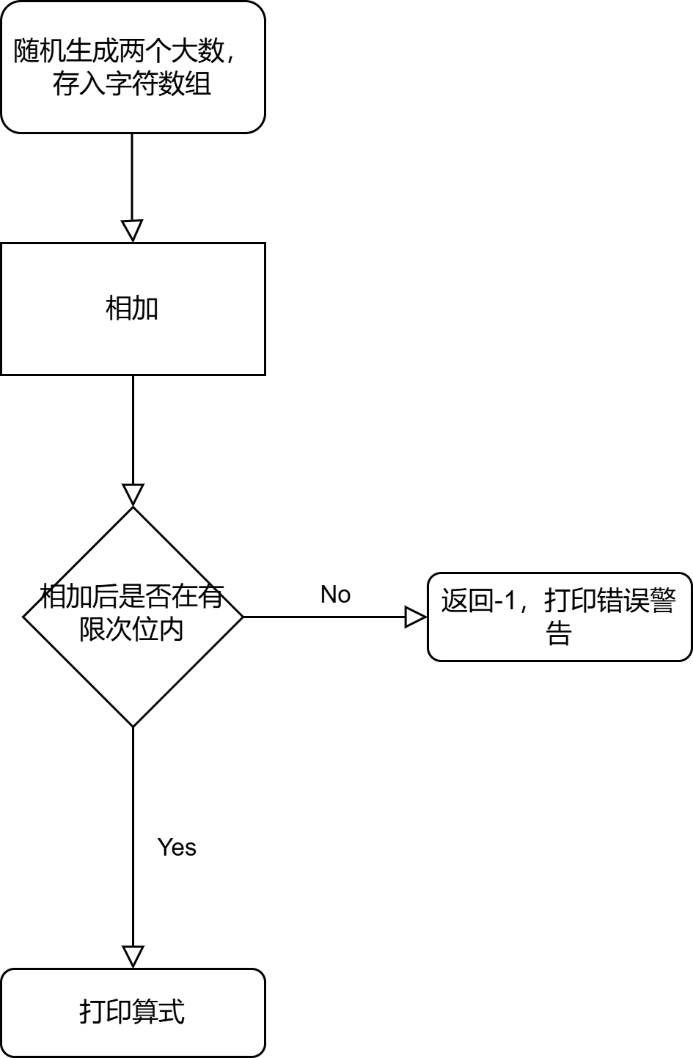
# 实验一

## 实验任务



## 实验步骤

编程思路：



关键代码：

函数void generateLongInt(char longint[]);

int addTwoLongInt(char num1[],char num2[],char numsum[]);

void displayLongInt(char num1[],char num2[],char numsum[]);

## 代码测试

### 测试点1

思路：生成大数后打印大数

预期：打印两个20位大数

结果：

文本

中度可信度描述已自动生成

成功！！！

### 测试点2

思路：运行整个程序

预期：打印两个20位大数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本, 应用程序

描述已自动生成

成功！！！

## 实验结论

代码可以达到功能目标

## 实验总结

实验成功，但编写过程中曾出现数组访问越界的问题，警示我们应当谨慎对待数组的界限，防止越界。

# 实验二

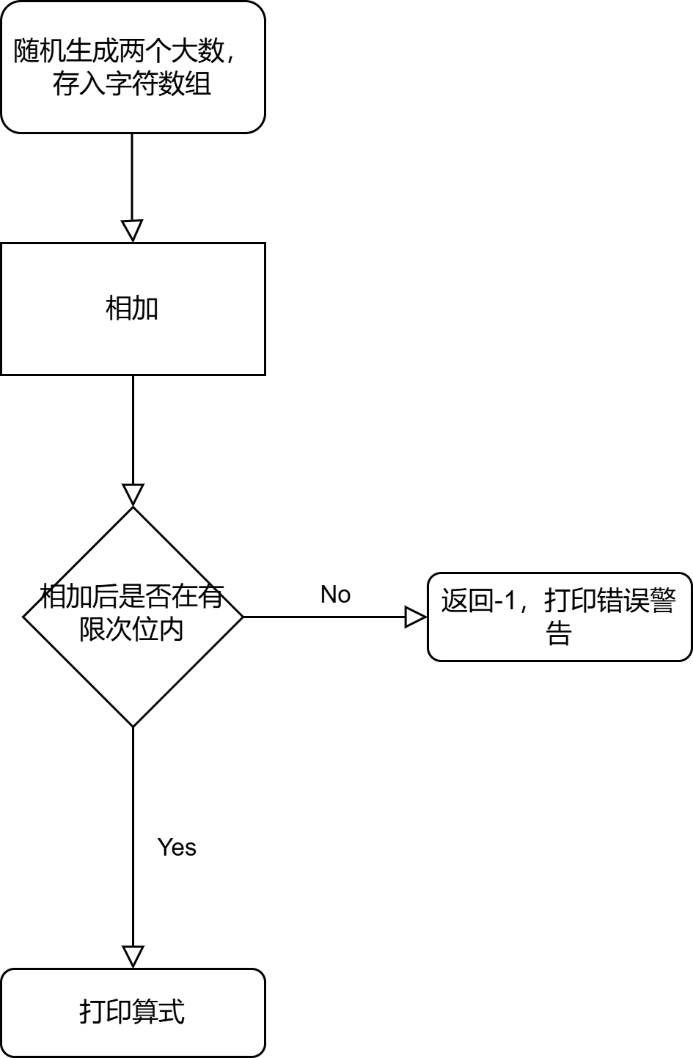
## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：



关键代码：

函数：**void** generateLongReal(**char** longreal[]);

**int** addTwoLongReal(**char** num1[],**char** num2[],**char** numsum[]);

**void** displayLongReal(**char** num1[],**char** num2[],**char** numsum[]);

## 代码测试

### 测试点1

思路：生成大数后打印大数

预期：打印两个38位实数

结果：

文本

描述已自动生成

成功！！！

### 测试点2

思路：运行整个程序

预期：打印两个38位实数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

成功！！！

## 实验结论

代码可以达到功能目标

## 实验总结

实验成功，本实验是在实验一的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验三

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图形用户界面

描述已自动生成

关键代码：

结构体定义：typedef struct{

char num[REAL\_WIDTH+1];

int length;

}Num;

函数：

Num generateLongReal(void);

Num addTwoLongReal(Num num1,Num num2);

void displayLongReal(Num num1,Num num2,Num numsum);

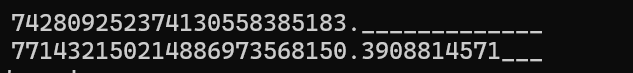
## 代码测试

### 测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：



成功！！！

### 测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

文本

描述已自动生成

文本

中度可信度描述已自动生成

成功！！！

## 实验结论

代码可以达到功能目标

## 实验总结

实验成功，本实验是在实验二的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验四

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图示

描述已自动生成

关键代码：

结构体定义：typedef struct{

char num[REAL\_WIDTH+1];

int pointpos;

int length;

}Num;

函数：

## Num generateLongReal(void);

## Num addTwoLongReal(Num num1,Num num2);

## void displayLongReal(Num num1,Num num2,Num numsum);

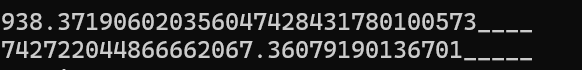
## void shiftDigitsToRight(char number[],int shiftLength);代码测试

### 测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：



成功！！！

### 测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

成功！！！

## 实验结论

代码可以达到功能目标

## 实验总结

实验成功，本实验是在实验三的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验五

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图示

描述已自动生成

关键代码：

结构体定义：typedef struct{

char num[REAL\_WIDTH+1];

int pointpos;

int length;

}Num;

函数：

## Num generateLongReal(void);

## Num addTwoLongReal(Num num1,Num num2);

## void displayLongReal(Num num1,Num num2,Num numsum);

## void shiftDigitsToRight(char number[],int shiftLength);

## 7.3代码测试

### 7.3.1测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：

文本

描述已自动生成

成功！！！

### 7.3.2测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

成功！！！

## 7.4实验结论

代码可以达到功能目标

## 7.5实验总结

实验成功，本实验是在实验四的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验六

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图示

描述已自动生成

关键代码：

结构体定义：typedef struct{

char num[REAL\_WIDTH+1];

int pointpos;

int length;

}Num;

函数：

## void generateLongReal(Num \*iptr);

## void addTwoLongReal(const Num \*num1,const Num \*num2,Num \*numadd);

## void displayLongReal(const Num \*num1,const Num \*num2,const Num \*numsum);

## void shiftDigitsToRight(const char \*number,int shiftLength);

## 8.3代码测试

### 8.3.1测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：

文本

中度可信度描述已自动生成

成功！！！

### 8.3.2测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

文本

中度可信度描述已自动生成

文本

描述已自动生成

成功！！！

## 8.4实验结论

代码可以达到功能目标

## 8.5实验总结

实验成功，本实验是在实验五的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验七

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图示

描述已自动生成

关键代码：

结构体定义：typedef struct{

char num[REAL\_WIDTH+1];

int pointpos;

int length;

}Num;

函数：Num \*generateLongReal(void);

Num \*addTwoLongReal(const Num \*num1,const Num \*num2);

void displayLongReal(const Num \*num1,const Num \*num2,const Num \*numsum);

void shiftDigitsToRight(const char \*number,int shiftLength);

void destroyLongNum(Num \*ptr);

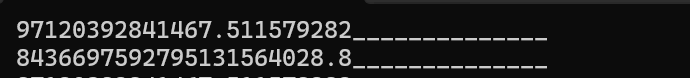
## 9.3代码测试

### 9.3.1测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：



成功！！！

### 9.3.2测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本, 电子邮件

描述已自动生成

文本

描述已自动生成

成功！！！

## 9.4实验结论

代码可以达到功能目标

## 9.5实验总结

实验成功，本实验是在实验六的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 实验八

## 实验任务

文本

描述已自动生成

## 实验步骤

编程思路：

图片包含 图示

描述已自动生成

关键代码：

结构体定义：typedef struct{

char \*num;

int pointpos;

int length;

} Num;

函数：Num \*generateLongReal(void);

Num \*addTwoLongReal(const Num \*num1,const Num \*num2);

void displayLongReal(const Num \*num1,const Num \*num2,const Num \*numsum);

void shiftDigitsToRight(Num \*number,int shiftLength);

void destroyLongNum(Num \*ptr);

10.3代码测试

### 10.3.1测试点1

思路：生成大数后打印大数

预期：打印两个固定宽度实数

结果：

文本

描述已自动生成

成功！！！

### 10.3.2测试点2

思路：运行整个程序

预期：打印两个固定宽度实数相加的算式或者显示出错讯息

结果：

图形用户界面, 文本

描述已自动生成

成功！！！

## 10.4实验结论

代码可以达到功能目标

## 10.5实验总结

实验成功，本实验是在实验七的基础上修改而成，以后进行工作时也可参考从易到难步步推进的实验思路

# 本课程学习总结

学习C语言的历程和总结

初识C语言，只觉得新奇好玩，随着渐渐的了解深入，慢慢体会到C语言的精妙绝伦之处，从一个小小的变量开始，到一个完整的函数，再到能够实现一个完整功能的模块，一步步的努力，一步步的探索，一步步对程序bug的修复和对程序功能的丰富和完善，一处处收获，一幕幕感动（特指找了半天终于找出问题在哪），回首忽觉来路不长，却处处鲜花（bug）盛开，空手而来，满载而归。

程序中出现的问题和改进

大数相加问题中常常出现数组越界的问题，而通过严格限制数组存放的界限以及打印时的结束条件可以很好的解决这个问题。

小数点对齐时常常容易出现偏移越界，所以通过程序事先计算，在发现可能会越界时及时终止并报错有效防止程序的崩溃（而这个问题通过分配动态内存也可以很好的解决）

代码规范与调试技巧

代码编写时应该注意代码的间隔，尽量一行只存放一句话，保持代码的美观度和可读性，要养成写注释的良好习惯，方便此后的修改或者复用，尽量以函数等模块化的方式编写代码，方便使用，省时省力

调试时可以一个个函数进行调试

学习记录和心得

编写代码是一件痛苦的事情，寻找bug也是一件令人痛不欲生的事情，但在努力过后，看见自己那流畅运行的代码一定是世间最棒的事情，有bug时，是乌云密布，万籁俱尽，修改完后，是晴空万里，万物美好。总之，我爱（应该）C语言！！！

# 附录

完整实验代码附在此处

1. 实验一
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <time.h>
6. #define INT\_WIDTH 20  //定义长度
8. **void** generateLongInt(**char** longint[]);  //生成随机数
9. **int** addTwoLongInt(**char** num1[],**char** num2[],**char** numsum[]);  //相加
10. **void** displayLongInt(**char** num1[],**char** num2[],**char** numsum[]);  //打印算式
12. **int** main()
13. {
14. srand(time(NULL));
15. **char** num1[INT\_WIDTH+1] = {'\0'};
17. **char** num2[INT\_WIDTH+1] = {'\0'};
18. **char** numAdd[INT\_WIDTH+1] = {'\0'};
19. generateLongInt(num1);//printf(" %s\n",num1);
20. generateLongInt(num2);//printf(" %s\n",num2);
21. **if**(addTwoLongInt(num1,num2,numAdd)==-1)  //判断是否溢出
22. {
23. **return** -1;
24. }
25. **else**
26. {
27. displayLongInt(num1,num2,numAdd) ;
28. }
29. **return** 0;
30. }
32. **void** generateLongInt(**char** longint[])
33. {
34. **int** i = 0;
35. **for**(;i < INT\_WIDTH;i++)
36. {
37. **if**(i == 0)
38. {
39. longint[i] = rand()%9 + 1+'0';  //第一位不为0，单独生成
40. }
41. **else**
42. {
43. longint[i] = rand()%10+'0';
44. }
45. }
46. **return**;
47. }
49. **int** addTwoLongInt(**char** num1[],**char** num2[],**char** numsum[])
50. {
51. **int** count1 = INT\_WIDTH-1;
52. **int** addFront = 0;
53. **for**(;count1>=0;count1--)
54. {
55. **int** add = num1[count1]-'0'+num2[count1]-'0'+addFront;
56. addFront = add / 10;
57. numsum[count1] = add%10 + '0';
58. }
59. **if**(addFront != 0)
60. {
61. printf("Wrong!");  //溢出，打印错误信息
62. **return** -1;
64. }
65. **return** 0;
66. }
68. **void** displayLongInt(**char** num1[],**char** num2[],**char** numsum[])
69. {
70. printf(" %s\n",num1);
71. printf("+%s\n",num2);
72. printf("=%s",numsum);
73. **return**;
74. }
75. 实验二
76. #include <stdio.h>
77. #include <stdlib.h>
78. #include <time.h>


82. #define REAL\_WIDTH 38
83. #define POINT\_POSITION  25
85. **void** generateLongReal(**char** longreal[]);  //生成随机数
86. **int** addTwoLongReal(**char** num1[],**char** num2[],**char** numsum[]);  //相加
87. **void** displayLongReal(**char** num1[],**char** num2[],**char** numsum[]);  //打印算式

90. **int** main()
91. {
92. srand(time(NULL));
93. **char** num1[REAL\_WIDTH+1] = {'\0'};
95. **char** num2[REAL\_WIDTH+1] = {'\0'};
96. **char** numAdd[REAL\_WIDTH+1] = {'\0'};
97. generateLongReal(num1);//printf(" %s\n",num1);
98. generateLongReal(num2);//printf(" %s\n",num2);
99. **if**(addTwoLongReal(num1,num2,numAdd)==-1)
100. {
101. **return** -1;
102. }
103. **else**
104. {
105. displayLongReal(num1,num2,numAdd) ;
106. }
107. **return** 0;
108. }
110. **void** generateLongReal(**char** longreal[])
111. {
112. **int** i = 0;
113. **for**(;i < REAL\_WIDTH;i++)
114. {
115. **if**(i == 0)
116. {
117. longreal[i] = rand()%9 + 1+'0';
118. }
119. **else** **if**(i == POINT\_POSITION-1)
120. {
121. longreal[i] = '.';
122. }
123. **else**
124. {
125. longreal[i] = rand()%10+'0';
126. }
127. }
128. **return**;
129. }
131. **int** addTwoLongReal(**char** num1[],**char** num2[],**char** numsum[])
132. {
133. **int** count1 = REAL\_WIDTH-1;
134. **int** addFront = 0;
135. **for**(;count1>=0;count1--)
136. {
137. **if**(count1 == POINT\_POSITION-1)  //单独处理小数点
138. {
139. numsum[count1] = '.';
140. **continue**;
141. }
142. **int** add = num1[count1]-'0'+num2[count1]-'0'+addFront;
143. addFront = add / 10;
144. numsum[count1] = add%10 + '0';
145. }
146. **if**(addFront != 0)
147. {
148. printf("Wrong!");  //溢出报错
149. **return** -1;
151. }
152. **return** 0;
153. }
155. **void** displayLongReal(**char** num1[],**char** num2[],**char** numsum[])
156. {
157. printf(" %s\n",num1);
158. printf("+%s\n",num2);
159. printf("=%s",numsum);
160. **return**;
161. }

三 、实验三

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>

6. #define REAL\_WIDTH 38  //最大长度
7. #define POINT\_POSITION  25  //小数点位置
9. **typedef** **struct**{
10. **char** num[REAL\_WIDTH+1];
11. **int**  length;
13. }Num;  //定义结构体
15. Num generateLongReal(**void**);  //生成并返回结构体
16. Num addTwoLongReal(Num num1,Num num2);  //相加返回相加后的结构体
17. **void** displayLongReal(Num num1,Num num2,Num numsum);  //打印算式

20. **int** main()
21. {
22. srand(time(NULL));
23. Num num1 = generateLongReal();
24. Num num2 = generateLongReal();
25. printf(" %s\n",num1.num);
26. printf(" %s\n",num2.num);
27. Num numAdd = addTwoLongReal(num1,num2);
29. **if**(numAdd.length == 0)
30. {
31. **return** -1;
32. }
33. **else**
34. {
35. displayLongReal(num1,num2,numAdd) ;
36. }
37. **return** 0;
38. }
40. Num generateLongReal(**void**)
41. {
43. **int** i = 0;
44. Num temp;
45. temp.length = rand()%(REAL\_WIDTH-POINT\_POSITION)+POINT\_POSITION;
46. **for**(;i < REAL\_WIDTH;i++)
47. {
48. **if**(i == 0)
49. {
50. temp.num[i] = rand()%9 + 1+'0';
51. }
52. **else** **if**(i == POINT\_POSITION-1)  //单独处理小数点
53. {
54. temp.num[i] = '.';
55. }
56. **else** **if**(i > temp.length-1)
57. {
58. temp.num[i] = '\_';
59. }
60. **else**
61. {
62. temp.num[i] = rand()%10+'0';
63. }
64. }
65. temp.num[i] = 0;
66. **return** temp;
67. }
69. Num addTwoLongReal(Num num1,Num num2)
70. {
71. **int** count1 = REAL\_WIDTH-1;
72. Num temp;
73. **int** addFront = 0;
74. temp.num[REAL\_WIDTH] = 0;
75. **for**(;count1>=0;count1--)
76. {
77. **if**(count1 == POINT\_POSITION-1)
78. {
79. temp.num[count1] = '.';
80. **continue**;
81. }
82. **if**(num1.num[count1] == '\_'||num2.num[count1] == '\_')  //若有空位单独计算
83. {
84. temp.num[count1] = num1.num[count1] + num2.num[count1] - '\_';
85. **continue**;
86. }
87. **int** add = num1.num[count1]-'0'+num2.num[count1]-'0'+addFront;
88. addFront = add / 10;
89. temp.num[count1] = add%10 + '0';
90. }
91. **if**(addFront != 0)
92. {
93. printf("Wrong!");
94. temp.length = 0;
95. **return** temp;
97. }
98. temp.length = num1.length>num2.length?num1.length:num2.length;  //长度为二者最大长度
99. **return** temp;
100. }
102. **void** displayLongReal(Num num1,Num num2,Num numsum)
103. {
104. printf(" %s\n",num1.num);
105. printf("+%s\n",num2.num);
106. printf("=%s",numsum.num);
107. **return**;
108. }

四、实验四

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define REAL\_WIDTH 38
7. **typedef** **struct**{
8. **char** num[REAL\_WIDTH+1];
9. **int** pointpos;
10. **int**  length;
12. }Num;  //定义结构体
14. Num generateLongReal(**void**);  //生成随机结构体
15. Num addTwoLongReal(Num num1,Num num2);  //相加
16. **void** displayLongReal(Num num1,Num num2,Num numsum);
17. **void** shiftDigitsToRight(**char** number[],**int** shiftLength);  //将小数右移使二者小数点对齐
19. **int** main()
20. {
21. srand(time(NULL));
22. Num num1 = generateLongReal();
23. Num num2 = generateLongReal();
24. printf(" %s\n",num1.num);
25. printf(" %s\n",num2.num);
26. Num numAdd ;
27. numAdd.pointpos=num1.pointpos>num2.pointpos?num1.pointpos:num2.pointpos;
28. **int** shift;
29. **if**(num1.pointpos>num2.pointpos)
30. {
31. shift = num1.pointpos-num2.pointpos;
32. **if**(num2.length+shift >= REAL\_WIDTH)
33. {
34. printf("wrong!");
35. **return** -1;
36. }
37. shiftDigitsToRight(num2.num,shift);
38. }
39. **else**
40. {
41. shift = num2.pointpos-num1.pointpos;
42. **if**(num1.length+shift >= REAL\_WIDTH)
43. {
44. printf("wrong!");
45. **return** -1;
46. }
47. shiftDigitsToRight(num1.num,shift);
48. }
49. numAdd = addTwoLongReal(num1,num2);
50. **if**(numAdd.length == 0)
51. {
52. **return** -1;
53. }
54. **else**
55. {
56. displayLongReal(num1,num2,numAdd) ;
57. }
58. **return** 0;
59. }
61. Num generateLongReal(**void**)
62. {
64. **int** i = 0;
65. Num temp;
66. temp.pointpos = rand()%REAL\_WIDTH+1;
67. temp.length = rand()%(REAL\_WIDTH-temp.pointpos)+temp.pointpos;
68. **for**(;i < REAL\_WIDTH;i++)
69. {
70. **if**(i == 0)
71. {
72. temp.num[i] = rand()%9 + 1+'0';
73. }
74. **else** **if**(i == temp.pointpos-1)
75. {
76. temp.num[i] = '.';
77. }
78. **else** **if**(i > temp.length-1)
79. {
80. temp.num[i] = '\_';
81. }
82. **else**
83. {
84. temp.num[i] = rand()%10+'0';
85. }
86. }
87. temp.num[i] = 0;
88. **return** temp;
89. }
91. Num addTwoLongReal(Num num1,Num num2)
92. {
93. **int** count1 = REAL\_WIDTH-1;
94. Num temp;
95. **int** addFront = 0;
96. temp.num[REAL\_WIDTH] = 0;
97. **for**(;count1>=0;count1--)
98. {
99. **if**(num1.num[count1] == '.')
100. {
101. temp.num[count1] = '.';
102. **continue**;
103. }
104. **if**(num1.num[count1] == '\_'||num2.num[count1] == '\_')  //单独处理空位
105. {
106. temp.num[count1] = num1.num[count1] + num2.num[count1] - '\_';
107. **continue**;
108. }
109. **int** add = num1.num[count1]-'0'+num2.num[count1]-'0'+addFront;
110. addFront = add / 10;
111. temp.num[count1] = add%10 + '0';
112. }
113. **if**(addFront != 0)
114. {
115. printf("Wrong!");  //溢出报错
116. temp.length = 0;
117. **return** temp;
119. }
120. temp.length = num1.length>num2.length?num1.length:num2.length;
121. **return** temp;
122. }
124. **void** displayLongReal(Num num1,Num num2,Num numsum)
125. {
126. printf(" %s\n",num1.num);
127. printf("+%s\n",num2.num);
128. printf("=%s",numsum.num);
129. **return**;
130. }
132. **void** shiftDigitsToRight(**char** number[],**int** shiftLength)
133. {
134. **int** i = REAL\_WIDTH-shiftLength-1;
135. **for**(;i>=0;i--)
136. {
137. number[i+shiftLength] = number[i];
138. }
139. **for**(i=0;i<=shiftLength;i++)
140. {
141. number[i] = '\_';
142. }
143. **return**;
144. }

五、实验五

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define REAL\_WIDTH 38
7. **typedef** **struct**{
8. **char** num[REAL\_WIDTH+1];
9. **int** pointpos;
10. **int**  length;
12. }Num;
14. Num generateLongReal(**void**);  //生成随机数，结构体存储相关信息
15. Num addTwoLongReal(Num num1,Num num2);  //相加
16. **void** displayLongReal(Num num1,Num num2,Num numsum);  /打印算式
17. **void** shiftDigitsToRight(**char** number[],**int** shiftLength);  //将小数右移使两个数小数点对齐
19. **int** main()
20. {
21. srand(time(NULL));
22. Num num1 = generateLongReal();
23. Num num2 = generateLongReal();
24. printf(" %s\n",num1.num);
25. printf(" %s\n",num2.num);
26. Num numAdd ;
27. numAdd.pointpos=num1.pointpos>num2.pointpos?num1.pointpos:num2.pointpos;
28. **int** shift;
29. **if**(num1.pointpos>num2.pointpos)
30. {
31. shift = num1.pointpos-num2.pointpos;
32. **if**(num2.length+shift >= REAL\_WIDTH)
33. {
34. printf("wrong!");
35. **return** -1;
36. }
37. shiftDigitsToRight(num2.num,shift);
38. }
39. **else**
40. {
41. shift = num2.pointpos-num1.pointpos;
42. **if**(num1.length+shift >= REAL\_WIDTH)
43. {
44. printf("wrong!");
45. **return** -1;
46. }
47. shiftDigitsToRight(num1.num,shift);
48. }
49. numAdd = addTwoLongReal(num1,num2);
50. **if**(numAdd.length == 0)
51. {
52. **return** -1;
53. }
54. **else**
55. {
56. displayLongReal(num1,num2,numAdd) ;  //未溢出就打印算式
57. }
58. **return** 0;
59. }
61. Num generateLongReal(**void**)
62. {
64. **char** \*i = NULL;
65. Num temp;
66. temp.pointpos = rand()%REAL\_WIDTH+1;
67. temp.length = rand()%(REAL\_WIDTH-temp.pointpos)+temp.pointpos;
68. **for**(i = temp.num;i-temp.num < REAL\_WIDTH;i++)
69. {
70. **if**(i == temp.num)
71. {
72. \*i = rand()%9 + 1+'0';
73. }
74. **else** **if**(i-temp.num == temp.pointpos-1)
75. {
76. \*i = '.';
77. }
78. **else** **if**(i-temp.num > temp.length-1)
79. {
80. \*i = '\_';
81. }
82. **else**
83. {
84. \*i = rand()%10+'0';
85. }
86. }
87. \*i = 0;
88. **return** temp;
89. }
91. Num addTwoLongReal(Num num1,Num num2)
92. {
93. Num temp;
94. **char** \*count1 = temp.num+REAL\_WIDTH-1;
95. **int** addFront = 0;
96. temp.num[REAL\_WIDTH] = 0;
97. **for**(;count1>=temp.num;count1--)
98. {
99. **if**(num1.num[count1-temp.num] == '.')
100. {
101. \*count1 = '.';
102. **continue**;
103. }
104. **if**(num1.num[count1-temp.num] == '\_'||num2.num[count1-temp.num] == '\_')
105. {
106. \*count1 = (num1.num[count1-temp.num] + num2.num[count1-temp.num] - '\_'+addFront);
107. addFront = (\*count1 -'0')/10;
108. \*count1 = (\*count1 -'0')%10+'0';
110. **continue**;
111. }
112. **int** add = num1.num[count1-temp.num]-'0'+num2.num[count1-temp.num]-'0'+addFront;
113. addFront = add / 10;
114. \*count1 = add%10 + '0';
115. }
116. **if**(addFront != 0)  //第一位相加进位不为0则报错
117. {
118. printf("Wrong!");
119. temp.length = 0;
120. **return** temp;
122. }
123. temp.length = num1.length>num2.length?num1.length:num2.length;
124. **return** temp;
125. }
127. **void** displayLongReal(Num num1,Num num2,Num numsum)
128. {
129. printf(" %s\n",num1.num);
130. printf("+%s\n",num2.num);
131. printf("=%s",numsum.num);
132. **return**;
133. }
135. **void** shiftDigitsToRight(**char** number[],**int** shiftLength)
136. {
137. **char** \*i = number+REAL\_WIDTH-shiftLength-1;  //指针操作数组元素
138. **for**(;i-number>=0;i--)
139. {
140. \*(i+shiftLength) = \*i;
141. }
142. **for**(i=number;i-number<=shiftLength;i++)
143. {
144. \*i = '\_';
145. }
146. **return**;
147. }

六、实验六

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define REAL\_WIDTH 38
7. **typedef** **struct**{
8. **char** num[REAL\_WIDTH+1];
9. **int** pointpos;
10. **int**  length;
12. } Num;  //定义结构体
14. **void** generateLongReal(Num \*iptr);  //基于地址的传递，生成结构体
15. **void** addTwoLongReal(**const** Num \*num1,**const** Num \*num2,Num \*numadd);  //利用指针传递结构体并相加两个大数
16. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum);  //打印算式
17. **void** shiftDigitsToRight(**const** **char** \*number,**int** shiftLength);  //对齐
19. **int** main()
20. {
21. srand(time(NULL));
22. Num num1,num2,numAdd;
23. Num \*ptr1=&num1,\*ptr2=&num2,\*ptr3=&numAdd;
24. generateLongReal(ptr1);
25. generateLongReal(ptr2);
26. printf(" %s\n",ptr1->num);
27. printf(" %s\n",ptr2->num);
28. ptr3->pointpos=ptr1->pointpos>ptr2->pointpos?ptr1->pointpos:ptr2->pointpos;
29. **int** shift;
30. **if**(ptr1->pointpos>ptr2->pointpos)
31. {
32. shift = ptr1->pointpos-ptr2->pointpos;
33. **if**(ptr2->length+shift >= REAL\_WIDTH)
34. {
35. printf("wrong!");
36. **return** -1;
37. }
38. shiftDigitsToRight(&(ptr1->num),shift);
39. }
40. **else**
41. {
42. shift = ptr2->pointpos-ptr1->pointpos;
43. **if**(ptr1->length+shift >= REAL\_WIDTH)
44. {
45. printf("wrong!");
46. **return** -1;
47. }
48. shiftDigitsToRight(&(ptr2->num),shift);
49. }
50. addTwoLongReal(ptr1,ptr2,ptr3);
51. **if**(ptr3->length == 0)
52. {
53. **return** -1;
54. }
55. **else**
56. {
57. displayLongReal(ptr1,ptr2,ptr3) ;
58. }
59. **return** 0;
60. }
62. **void** generateLongReal(Num \*iptr)
63. {
65. **char** \*i = NULL;
66. iptr->pointpos = rand()%REAL\_WIDTH+1;
67. iptr->length = rand()%(REAL\_WIDTH-iptr->pointpos)+iptr->pointpos;
68. **for**(i = iptr->num;i-iptr->num < REAL\_WIDTH;i++)
69. {
70. **if**(i == iptr->num)
71. {
72. \*i = rand()%9 + 1+'0';
73. }
74. **else** **if**(i-iptr->num == iptr->pointpos-1)
75. {
76. \*i = '.';
77. }
78. **else** **if**(i-iptr->num > iptr->length-1)
79. {
80. \*i = '\_';
81. }
82. **else**
83. {
84. \*i = rand()%10+'0';
85. }
86. }
87. \*i = 0;
88. **return** ;
89. }
91. **void** addTwoLongReal(**const** Num \*num1,**const** Num \*num2,Num \*numadd)
92. {
93. **char** \*count1 = numadd->num+REAL\_WIDTH-1;
94. **int** addFront = 0;
95. numadd->num[REAL\_WIDTH] = 0;
96. **for**(;count1>=numadd->num;count1--)  //从小位加到大位方便记录进位
97. {
98. **if**(numadd->num[count1-numadd->num] == '.')
99. {
100. \*count1 = '.';
101. **continue**;
102. }
103. **if**(num1->num[count1-numadd->num] == '\_'||num2->num[count1-numadd->num] == '\_')
104. {
105. \*count1 = (num1->num[count1-numadd->num] + num2->num[count1-numadd->num] - '\_'+addFront);
106. addFront = (\*count1 -'0')/10;
107. \*count1 = (\*count1 -'0')%10+'0';
109. **continue**;
110. }
111. **int** add = num1->num[count1-numadd->num]-'0'+num2->num[count1-numadd->num]-'0'+addFront;
112. addFront = add / 10;
113. \*count1 = add%10 + '0';
114. }
115. **if**(addFront != 0)
116. {
117. printf("Wrong!");
118. numadd->length = 0;
119. **return** ;
121. }
122. numadd->length = num1->length>num2->length?num1->length:num2->length;
123. **return** ;
124. }
126. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum)
127. {
128. printf(" %s\n",num1->num);
129. printf("+%s\n",num2->num);
130. printf("=%s",numsum->num);
131. **return**;
132. }
134. **void** shiftDigitsToRight(**const** **char** \*number,**int** shiftLength)
135. {
136. **char** \*i = number+REAL\_WIDTH-shiftLength-1;
137. **for**(;i-number>=0;i--)  //循环使数字右移
138. {
139. \*(i+shiftLength) = \*i;
140. }
141. **for**(i=number;i-number<=shiftLength;i++)
142. {
143. \*i = '\_';
144. }
145. **return**;
146. }

七、实验七

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define REAL\_WIDTH 38
7. **typedef** **struct**{
8. **char** num[REAL\_WIDTH+1];
9. **int** pointpos;
10. **int**  length;
12. } Num;  //定义结构体
14. Num \*generateLongReal(**void**);  //结构体指针返回生成随机数地址
15. Num \*addTwoLongReal(**const** Num \*num1,**const** Num \*num2);  //返回相加后生成随机数地址
16. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum);  //打印算式
17. **void** shiftDigitsToRight(**const** **char** \*number,**int** shiftLength);  //对齐
18. **void** destroyLongNum(Num \*ptr);  //释放动态内存
20. **int** main()
21. {
22. srand(time(NULL));
23. Num num1,num2,numAdd;
24. Num \*ptr1=&num1,\*ptr2=&num2,\*ptr3=&numAdd;
25. ptr1 = generateLongReal();  //生成随机数
26. ptr2 = generateLongReal();
27. printf(" %s\n",ptr1->num);
28. printf(" %s\n",ptr2->num);
29. ptr3->pointpos=ptr1->pointpos>ptr2->pointpos?ptr1->pointpos:ptr2->pointpos;  //判断相加后数字小数点位置
30. **int** shift;
31. **if**(ptr1->pointpos>ptr2->pointpos)  //判断右移哪个数
32. {
33. shift = ptr1->pointpos-ptr2->pointpos;
34. **if**(ptr2->length+shift >= REAL\_WIDTH)
35. {
36. printf("wrong!");
37. **return** -1;
38. }
39. shiftDigitsToRight(&(ptr1->num),shift);
40. }
41. **else**
42. {
43. shift = ptr2->pointpos-ptr1->pointpos;
44. **if**(ptr1->length+shift >= REAL\_WIDTH)
45. {
46. printf("wrong!");
47. **return** -1;
48. }
49. shiftDigitsToRight(&(ptr2->num),shift);
50. }
51. ptr3 = addTwoLongReal(ptr1,ptr2);
52. **if**(ptr3->length == 0)
53. {
54. **return** -1;
55. }
56. **else**
57. {
58. displayLongReal(ptr1,ptr2,ptr3) ;
59. }
60. destroyLongNum(ptr1);  //释放内存，防止崩溃
61. destroyLongNum(ptr2);
62. destroyLongNum(ptr3);
63. **return** 0;
64. }
66. Num \*generateLongReal(**void**)
67. {
69. **char** \*i = NULL;
70. Num \*iptr = (Num \*) malloc(**sizeof**(Num));
71. iptr->pointpos = rand()%REAL\_WIDTH+1;
72. iptr->length = rand()%(REAL\_WIDTH-iptr->pointpos)+iptr->pointpos;
73. **for**(i = iptr->num;i-iptr->num < REAL\_WIDTH;i++)
74. {
75. **if**(i == iptr->num)
76. {
77. \*i = rand()%9 + 1+'0';
78. }
79. **else** **if**(i-iptr->num == iptr->pointpos-1)
80. {
81. \*i = '.';
82. }
83. **else** **if**(i-iptr->num > iptr->length-1)
84. {
85. \*i = '\_';
86. }
87. **else**
88. {
89. \*i = rand()%10+'0';
90. }
91. }
92. \*i = 0;
93. **return** iptr;
94. }
96. Num \*addTwoLongReal(**const** Num \*num1,**const** Num \*num2)
97. {
98. Num \*numadd = (Num \*)malloc(**sizeof**(Num));
99. **char** \*count1 = numadd->num+REAL\_WIDTH-1;
100. **int** addFront = 0;
101. numadd->num[REAL\_WIDTH] = 0;
102. **for**(;count1>=numadd->num;count1--)
103. {
104. **if**(numadd->num[count1-numadd->num] == '.')
105. {
106. \*count1 = '.';
107. **continue**;
108. }
109. **if**(num1->num[count1-numadd->num] == '\_'||num2->num[count1-numadd->num] == '\_')
110. {
111. \*count1 = (num1->num[count1-numadd->num] + num2->num[count1-numadd->num] - '\_'+addFront);
112. addFront = (\*count1 -'0')/10;
113. \*count1 = (\*count1 -'0')%10+'0';
115. **continue**;
116. }
117. **int** add = num1->num[count1-numadd->num]-'0'+num2->num[count1-numadd->num]-'0'+addFront;
118. addFront = add / 10;
119. \*count1 = add%10 + '0';
120. }
121. **if**(addFront != 0)
122. {
123. printf("Wrong!");
124. numadd->length = 0;
125. **return** ;
127. }
128. numadd->length = num1->length>num2->length?num1->length:num2->length;
129. **return** numadd;
130. }
132. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum)
133. {
134. printf(" %s\n",num1->num);
135. printf("+%s\n",num2->num);
136. printf("=%s",numsum->num);
137. **return**;
138. }
140. **void** shiftDigitsToRight(**const** **char** \*number,**int** shiftLength)
141. {
142. **char** \*i = number+REAL\_WIDTH-shiftLength-1;
143. **for**(;i-number>=0;i--)
144. {
145. \*(i+shiftLength) = \*i;
146. }
147. **for**(i=number;i-number<=shiftLength;i++)
148. {
149. \*i = '\_';
150. }
151. **return**;
152. }
154. **void** destroyLongNum(Num \*ptr)
155. {
156. free(ptr);
157. ptr = NULL;
158. **return**;
159. }

八、实验八

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define REAL\_WIDTH 38
7. **typedef** **struct**{
8. **char** \*num;
9. **int** pointpos;
10. **int**  length;
12. } Num;  //定义结构体
14. Num \*generateLongReal(**void**);  //生成随机数，返回地址
15. Num \*addTwoLongReal(**const** Num \*num1,**const** Num \*num2);  //相加，返回结果所在地址
16. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum);  //展示算式
17. **void** shiftDigitsToRight(Num \*number,**int** shiftLength); //移动使小数点对齐
18. **void** destroyLongNum(Num \*ptr);  //释放缓存区
20. **int** main()
21. {
22. srand(time(NULL));
23. Num num1,num2,numAdd;
24. Num \*ptr1=&num1,\*ptr2=&num2,\*ptr3=&numAdd;
25. ptr1 = generateLongReal();
26. ptr2 = generateLongReal();
27. printf(" %s\n",ptr1->num);
28. printf(" %s\n",ptr2->num);
29. ptr3->pointpos=ptr1->pointpos>ptr2->pointpos?ptr1->pointpos:ptr2->pointpos;  //初步判断小数点位置
30. **int** shift,n;
31. **if**(ptr1->pointpos>ptr2->pointpos)  //判断右移哪个数
32. {
33. shift = ptr1->pointpos-ptr2->pointpos;
34. shiftDigitsToRight(ptr2,shift);
35. **if**(ptr2->length>REAL\_WIDTH)
36. {
37. ptr1->num = (**char** \*)realloc(ptr1,(ptr2->length-REAL\_WIDTH)\***sizeof**(**char**));
38. **for**(n=0;n<(ptr2->length-ptr1->length);n++)
39. {
40. \*(ptr1->num+ptr1->length+n)=0;
41. }
42. ptr1->length = ptr2->length;
43. }
45. }
46. **else**
47. {
48. shift = ptr2->pointpos-ptr1->pointpos;
49. shiftDigitsToRight(ptr1,shift);
50. **if**(ptr1->length>REAL\_WIDTH)
51. {
52. ptr2->num = (**char** \*)realloc(ptr2,(ptr1->length-REAL\_WIDTH)\***sizeof**(**char**));
53. **for**(n=0;n<(ptr1->length-ptr2->length);n++)
54. {
55. \*(ptr2->num+ptr2->length+n)=0;
56. }
57. ptr2->length = ptr1->length;
58. }
59. }
60. //printf(" %s\n",ptr1->num);
61. //printf(" %s\n",ptr2->num);
62. ptr3 = addTwoLongReal(ptr1,ptr2);
63. displayLongReal(ptr1,ptr2,ptr3) ;
64. destroyLongNum(ptr1);  /释放缓存区
65. destroyLongNum(ptr2);
66. destroyLongNum(ptr3);
67. **return** 0;
68. }
70. Num \*generateLongReal(**void**)
71. {
73. **char** \*i = NULL;
74. Num \*iptr = (Num \*) malloc(**sizeof**(Num));
75. iptr->pointpos = rand()%REAL\_WIDTH+1;
76. iptr->length = rand()%(REAL\_WIDTH-iptr->pointpos)+iptr->pointpos;
77. iptr->num = (**char**\*)malloc((REAL\_WIDTH+1)\***sizeof**(**char**));
78. **for**(i = iptr->num;i-iptr->num < REAL\_WIDTH;i++)
79. {
80. **if**(i == iptr->num)
81. {
82. \*i = rand()%9 + 1+'0';
83. }
84. **else** **if**(i-iptr->num == iptr->pointpos-1)
85. {
86. \*i = '.';
87. }
88. **else** **if**(i-iptr->num > iptr->length-1)
89. {
90. \*i = '\_';
91. }
92. **else**
93. {
94. \*i = rand()%10+'0';
95. }
96. }
97. \*i = 0;
98. **return** iptr;
99. }
101. Num \*addTwoLongReal(**const** Num \*num1,**const** Num \*num2)
102. {
103. Num \*numadd = (Num \*)malloc(**sizeof**(Num));
104. numadd->num = (**char**\*)malloc((num1->length+1)\***sizeof**(**char**));  //分配动态内存
105. **char** \*count1 = numadd->num+num1->length-1;
106. **int** addFront = 0;
107. \*(numadd->num+num1->length) = 0;
108. **for**(;count1>=numadd->num;count1--)
109. {
110. **if**(num1->num[count1-numadd->num] == '.')
111. {
112. \*count1 = '.';
113. **continue**;
114. }
115. **if**(num1->num[count1-numadd->num] == '\_'||num2->num[count1-numadd->num] == '\_')
116. {
117. \*count1 = (num1->num[count1-numadd->num] + num2->num[count1-numadd->num] - '\_'+addFront);
118. addFront = (\*count1 -'0')/10;
119. \*count1 = (\*count1 -'0')%10+'0';
121. **continue**;
122. }
123. **int** add = num1->num[count1-numadd->num]-'0'+num2->num[count1-numadd->num]-'0'+addFront;
124. addFront = add / 10;
125. \*count1 = add%10 + '0';
126. }
127. numadd->length = num1->length>num2->length?num1->length:num2->length;
128. **return** numadd;
129. }
131. **void** displayLongReal(**const** Num \*num1,**const** Num \*num2,**const** Num \*numsum)
132. {
133. printf(" %s\n",num1->num);
134. printf("+%s\n",num2->num);
135. printf("=%s",numsum->num);
136. **return**;
137. }
139. **void** shiftDigitsToRight(Num \*number,**int** shiftLength)
140. {
141. **char** \*i = number->num+number->length-1;
142. number->length=number->length+shiftLength;
143. **if**(number->length >= REAL\_WIDTH)
144. {
146. number->num = (**char**\*)realloc((number->num),(number->length-REAL\_WIDTH)\***sizeof**(**char**));  //若溢出，增加动态内存分配
147. \*(number->num+number->length) = 0;
148. }
150. **for**(;i-number->num>=0;i--)
151. {
152. \*(i+shiftLength) = \*i;
153. }
154. **for**(i=number->num;i-number->num<shiftLength;i++)
155. {
156. \*i = '\_';
157. }
158. **return**;
159. }
161. **void** destroyLongNum(Num \*ptr)
162. {
163. **if**(ptr->num != NULL)  //先释放内部指针所占内存，再释放外部结构体所占用内存
164. {
165. free(ptr->num);
166. ptr->num = NULL;
167. }
168. free(ptr);
169. ptr = NULL;
170. **return**;
171. }