

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 | 刘宏远 | 学号 | U202413662 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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。

# 实验环境

操作系统：Windows 11

编程工具：CodeBlocks 16.01

# 实验

## 实验任务

·在结构体内，用一维字符数组记录长实数的每个数位，通过ASCII字符值转换获取每个数位，用一个数位来存储小数点‘.’，用一个整数记录长实数的长度，用一个整数记录小数点位置

·随机产生两个长整数，约定其最高位不得为‘0’，其长度、小数点位置都可以不同

·对两个长度不同的长实数进行求和

·打印两个长实数求和的竖式计算过程，需要根据小数点位置不同调整输出格式

·存储数位的字符数组为无最大位宽参数的动态数组

·需要完成的函数有：

Real\* generateLongReal(void); // 生成一个随机的长实数

void destroyLongReal(Real \*rPtr); // 释放 Real 结构体的内存

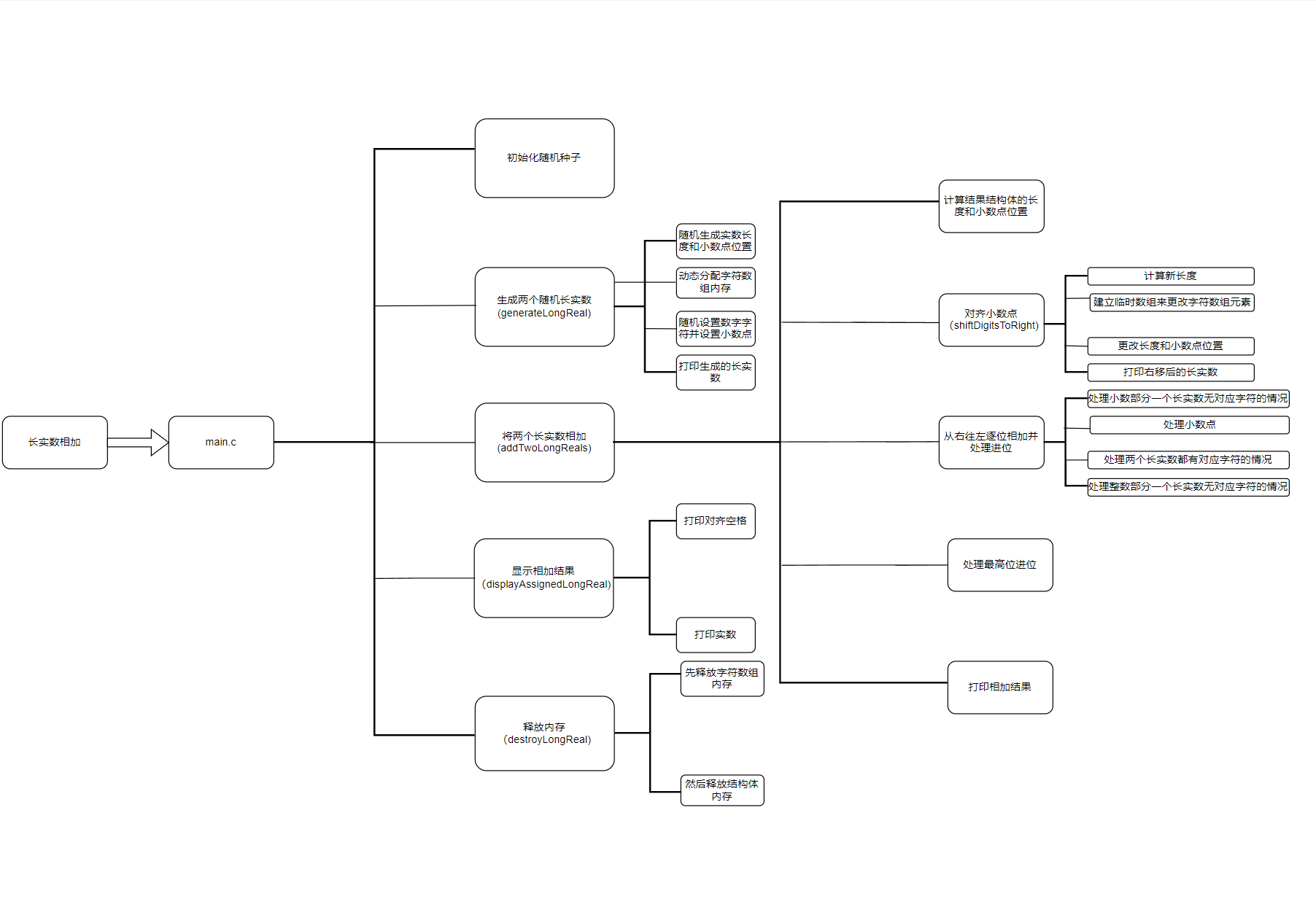
Real\* addTwoLongReals(const Real \*rPtr1, const Real \*rPtr2); // 将两个长实数相加

void displayAssignedLongReal(int assignedLength, const Real \*rPtr); // 对齐打印长实数

void shiftDigitsToRight(Real \*rPtr, int shiftLength); // 将数字字符向右移动指定的位数

## 实验步骤

·编程思路：



·关键代码：长实数相加部分

// 从右到左逐位相加

    for (i = rPtrSum->length - 1, j = 1, k = 1; i >= 0; i--) {

        if (i > rPtrSum->pointPos + rPtr1->length - rPtr1->pointPos - 1 ||

            i > rPtrSum->pointPos + rPtr2->length - rPtr2->pointPos - 1) {

            // 处理小数部分一个长实数无对应字符的情况

            if (rPtr1->length - rPtr1->pointPos >= rPtr2->length - rPtr2->pointPos) {

                rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j];

                j++;

            } else {

                rPtrSum->digits[i] = rPtr2->digits[rPtr2->length - k];

                k++;

            }

        } else if (i == rPtrSum->pointPos - 1) {

            // 处理小数点

            rPtrSum->digits[i] = '.';

            j++, k++;

        } else {

            if (rPtr2->length - k >= 0 && rPtr1->length - j >= 0) {

            // 处理两个长实数都有对应字符的情况

                rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j] + rPtr2->digits[rPtr2->length - k] - '0' + carry;

                carry = 0;

                j++, k++;

            } else {

            // 处理整数部分一个长实数无对应字符的情况

                if ( rPtr1->length - j < 0 ) {

                    rPtrSum->digits[i] = rPtr2->digits[rPtr2->length - k] + carry;

                    carry = 0;

                    k++;

                } else {

                    rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j] + carry;

                    carry = 0;

                    j++;

                }

            }

            // 处理进位

            if (rPtrSum->digits[i] > '9') {

                rPtrSum->digits[i] -= 10;

                carry = 1;

            }

        }

    }

## 代码测试

### 函数的测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试点 | 测试步骤 | 预期测试结果 | 实际测试结果（截图） | 测试结论 |
| Real \*generateLongReal(void)函数，看是否能生成符合要求的长实数 | 随机生成 | 生成符合要求的的长实数 |  | 该函数能生成符合要求的的长实数 |
| Real\* addTwoLongReals(const Real \*rPtr1, const Real \*rPtr2)函数，看能否正确得到两长实数相加的结果 | 观察得到的结果是否正确 | 得到的结果正确 |  | 该函数能正确计算出两个长实数相加的结果 |
| void shiftDigitsToRight(Real \*rPtr, int shiftLength)函数，看能否正确将长实数右移，使小数点对齐 | 观察程序运行后小数点是否对齐 | 小数点对齐 |  | 该函数能正确将长实数右移使小数点对齐 |
| void displayAssignedLongReal(int assignedLength, const Real \*rPtr)函数，看该函数是否能正确打印出对齐的长实数 | 观察打印出来的长实数是否对齐 | 打印的长实数对齐 |  | 该函数能正确地将长实数对齐打印 |

## 实验结论

代码能正确地实现两个无大小限制的长实数相加的功能，并将流程与结果显示出来。

## 实验总结

·结构体字符数组的设置并不是C字符串，并不以‘\0’为结束标志，打印时不能用printf函数整体打印

·两长实数相加时要充分考虑各种情况，对于某一个数位，要考虑两个长实数在这个数位上的数字字符情况（有或者没有）

·通过指针进行操作时要充分理解其含义，否则写代码时容易把自己弄混

# 本课程学习总结

·学习C语言的历程和总结（心酸血泪史）

1.初学阶段：从“Hello, World!”开始，理解基础语法，但是经常犯各种错误让人头大，有时还忘记切换中英输入法。频繁遇到各种错误，代码要改很久才能成功运行。

2.逐渐掌握：通过老师上课的讲解，课后自己看教材和上网查资料，对C语言逐渐掌握，能比较顺利完成老师上课布置的任务（如日历系列、大数计算系列）。

3.收获与成长：学习C语言让我培养了严谨的编程思维，还让我的逻辑更严密。

虽然过程艰辛，但收获满满！

·程序中出现的问题和改进

起初程序需要固定位宽数组的部分，功能有限；通过指针申请和释放内存的方法，成功实现了无最大位宽参数的动态数组，功能拓展到可以进行任意两实数相加。

·代码规范与调试技巧

1.代码规范

命名规范：变量、函数、结构体名称应具有描述性，要使用驼峰命名法等

注释：关键代码处要添加注释，解释逻辑和功能，便于理解

缩进与格式：保持一致的缩进和代码格式

函数职责单一：每个函数只完成一个明确的任务

2.调试技巧

分段调试：将程序分成小模块，逐个测试

打印调试信息：使用 printf 函数输出，定位问题

·学习记录和心得

学好C语言，一定要做到：

1.基础扎实：从变量、数据类型、控制结构到函数，打好基础是关键

2.充分理解指针：指针是C的核心，理解内存分配与释放至关重要

3.调试技巧：要学会一些调试技巧，如使用printf函数调试，逐步排查问题

4.项目实践：通过小项目（如日历系列、大数计算系列）实战来巩固知识

5.持续学习：阅读教材和老师的ppt，要自己动手编程序

# 附录

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
5. #define MAX\_REAL\_WIDTH 25
7. // 定义 Real 结构体，用于表示一个实数
8. **typedef** **struct** {
9. **char** \*digits;  // 指向存储数字字符的动态数组
10. **int** pointPos;  // 小数点的位置
11. **int** length;    // 数字的总长度（包括小数点）
12. } Real;
14. // 函数声明
15. Real\* generateLongReal(**void**);  // 生成一个随机的长实数
16. **void** destroyLongReal(Real \*rPtr);  // 释放 Real 结构体的内存
17. Real\* addTwoLongReals(**const** Real \*rPtr1, **const** Real \*rPtr2);  // 将两个长实数相加
18. **void** displayAssignedLongReal(**int** assignedLength, **const** Real \*rPtr);  // 显示长实数
19. **void** shiftDigitsToRight(Real \*rPtr, **int** shiftLength);  // 将数字字符向右移动指定的位数
21. **int** main(**void**) {
22. srand(time(NULL));  // 初始化随机数种子
24. // 生成两个随机的长实数
25. Real \*rPtr1 = generateLongReal();
26. Real \*rPtr2 = generateLongReal();
28. // 将两个长实数相加
29. Real \*rPtrSum = addTwoLongReals(rPtr1, rPtr2);
31. // 显示相加的结果
32. printf("\n  ");
33. displayAssignedLongReal(rPtrSum->pointPos - rPtr1->pointPos, rPtr1);
34. printf("+ ");
35. displayAssignedLongReal(rPtrSum->pointPos - rPtr2->pointPos, rPtr2);
36. printf("= ");
37. displayAssignedLongReal(0, rPtrSum);
38. printf("\n");
40. // 释放内存
41. destroyLongReal(rPtr1);
42. destroyLongReal(rPtr2);
43. destroyLongReal(rPtrSum);
45. **return** 0;
46. }
48. // 显示长实数，assignedLength 用于对齐输出
49. **void** displayAssignedLongReal(**int** assignedLength, **const** Real \*rPtr) {
50. **int** i;
51. **for** (i = 1; i <= assignedLength; i++) {
52. printf(" ");
53. }
55. **for** (i = 0; i < rPtr->length; i++) {
56. printf("%c", rPtr->digits[i]);
57. }
59. printf("\n");
60. }
62. // 生成一个随机的长实数
63. Real \*generateLongReal(**void**) {
64. Real \*rPtr = (Real\*) malloc(**sizeof**(Real));  // 分配 Real 结构体的内存
66. // 随机生成数字的长度和小数点位置
67. rPtr->length = 2 + rand() % (MAX\_REAL\_WIDTH - 1);
68. rPtr->pointPos = 1 + rand() % (rPtr->length);
70. // 分配 digits 数组的内存
71. rPtr->digits = (**char**\*) calloc(rPtr->length, **sizeof**(**char**));
72. rPtr->digits[0] = '1' + rand() % 9;  // 第一位是 1-9 之间的随机数字
74. **int** i;
75. **for** (i = 1; i < rPtr->length - 1; i++) {
76. rPtr->digits[i] = '0' + rand() % 10;  // 中间位是 0-9 之间的随机数字
77. }
78. rPtr->digits[rPtr->length - 1] = '1' + rand() % 9;  // 最后一位是 1-9 之间的随机数字
79. rPtr->digits[rPtr->pointPos - 1] = '.';  // 设置小数点
81. // 打印生成的随机数
82. printf("Randomize(%2d.%2d) :", rPtr->length, rPtr->length - rPtr->pointPos);
83. displayAssignedLongReal(0, rPtr);
85. **return** rPtr;
86. }
88. // 释放 Real 结构体的内存
89. **void** destroyLongReal(Real \*rPtr) {
90. **if** (rPtr->digits != NULL) {
91. free(rPtr->digits);  // 释放 digits 数组的内存
92. }
93. free(rPtr);  // 释放 Real 结构体的内存
94. }
96. // 将两个长实数相加
97. Real\* addTwoLongReals(**const** Real \*rPtr1, **const** Real \*rPtr2) {
98. Real \*rPtrSum = (Real\*) malloc(**sizeof**(Real));  // 分配结果结构体的内存
100. // 计算结果的小数部分长度以及小数点位置
101. **int** fracPartOfSum = (rPtr1->length - rPtr1->pointPos > rPtr2->length - rPtr2->pointPos)
102. ? rPtr1->length - rPtr1->pointPos
103. : rPtr2->length - rPtr2->pointPos;
104. rPtrSum->pointPos = (rPtr1->pointPos > rPtr2->pointPos)
105. ? rPtr1->pointPos
106. : rPtr2->pointPos;
107. rPtrSum->length = fracPartOfSum + rPtrSum->pointPos;  // 计算结果的总长度
109. **int** i, j, k;
110. **int** carry = 0;  // 是否进位
112. // 从右到左逐位相加
113. **for** (i = rPtrSum->length - 1, j = 1, k = 1; i >= 0; i--) {
114. **if** (i > rPtrSum->pointPos + rPtr1->length - rPtr1->pointPos - 1 ||
115. i > rPtrSum->pointPos + rPtr2->length - rPtr2->pointPos - 1) {
116. // 处理小数部分一个长实数无对应字符的情况
117. **if** (rPtr1->length - rPtr1->pointPos >= rPtr2->length - rPtr2->pointPos) {
118. rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j];
119. j++;
120. } **else** {
121. rPtrSum->digits[i] = rPtr2->digits[rPtr2->length - k];
122. k++;
123. }
124. } **else** **if** (i == rPtrSum->pointPos - 1) {
125. // 处理小数点
126. rPtrSum->digits[i] = '.';
127. j++, k++;
128. } **else** {
129. **if** (rPtr2->length - k >= 0 && rPtr1->length - j >= 0) {
130. // 处理两个长实数都有对应字符的情况
131. rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j] + rPtr2->digits[rPtr2->length - k] - '0' + carry;
132. carry = 0;
133. j++, k++;
134. } **else** {
135. // 处理整数部分一个长实数无对应字符的情况
136. **if** ( rPtr1->length - j < 0 ) {
137. rPtrSum->digits[i] = rPtr2->digits[rPtr2->length - k] + carry;
138. carry = 0;
139. k++;
140. } **else** {
141. rPtrSum->digits[i] = rPtr1->digits[rPtr1->length - j] + carry;
142. carry = 0;
143. j++;
144. }
145. }
147. // 处理进位
148. **if** (rPtrSum->digits[i] > '9') {
149. rPtrSum->digits[i] -= 10;
150. carry = 1;
151. }
152. }
153. }
155. **int** shiftLength = 0;
157. // 对齐小数点
158. **if** (rPtr1->pointPos < rPtr2->pointPos) {
159. shiftLength = rPtr2->pointPos - rPtr1->pointPos;
160. shiftDigitsToRight(rPtr1, shiftLength);
161. } **else** **if** (rPtr1->pointPos > rPtr2->pointPos) {
162. shiftLength = rPtr1->pointPos - rPtr2->pointPos;
163. shiftDigitsToRight(rPtr2, shiftLength);
164. }
166. // 处理最高位的进位
167. **if** (carry == 1) {
168. shiftDigitsToRight(rPtrSum, 1);
169. rPtrSum->digits[0] = '1';
170. }
172. // 打印相加的结果
173. printf("Sum Result(%2d.%2d):", rPtrSum->length, fracPartOfSum);
174. displayAssignedLongReal(0, rPtrSum);
176. **return** rPtrSum;
177. }
179. // 将数字向右移动指定的位数
180. **void** shiftDigitsToRight(Real \*rPtr, **int** shiftLength) {
181. **int** newLength = rPtr->length + shiftLength;  // 计算新的长度
182. **char** newDigits[newLength];  // 创建临时数组
184. **int** i;
185. **for** (i = 0; i < shiftLength; i++) {
186. newDigits[i] =' ';  // 填充左侧的空位
187. }
188. **for** (i = shiftLength; i < newLength; i++) {
189. newDigits[i] = rPtr->digits[i - shiftLength];  // 复制原始数据
190. }
192. **for** (i = 0; i < newLength; i++ ) {
193. rPtr->digits[i] = newDigits[i];  // 将新数据复制回原数组
194. }
195. rPtr->length = newLength;  // 更新长度
196. rPtr->pointPos += shiftLength;  // 更新小数点位置
198. // 打印移动后的数字
199. printf("shiftRight (%2d)  :", shiftLength);
200. **for** (i = 0; i < rPtr->length; i++) {
201. printf("%c", rPtr->digits[i]);
202. }
203. printf("\n");
204. }