

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 | 傅子赫 | 学号 | U202410990 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 10

编程工具：CodeBlocks 16.01

# 实验一

## 实验任务

-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代表的数值

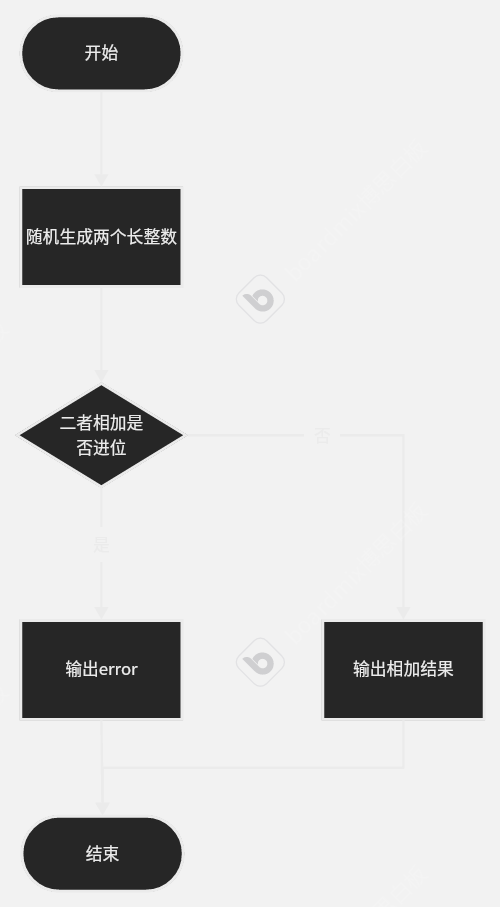
-用一个数位来存储小数点'.

-随机产生两个长实数，约定其最高位不得为'0’

-对两个相同长度的长实数进行求和，有加法进位时显示报错信息

-打印两个长实数求和的竖式计算过程

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否完成 |
| 随机生成随机数函数 | 多次调用函数 |  | 完成 |
| 两数相加函数 | 多次调用函数 |  | 完成 |

## 实验结论

代码能够达到功能目标

## 实验总结

该实验初步的了解了数组的运用

# 实验二

## 实验任务

-用一维字符数组记录长整数的每个数位，通过ASCII字符值转换获取每个数位代表的数值

-随机产生两个长整数，约定其最高位不得为'0'

-对两个相同长度的长整数进行求和，有加法进位导致存储位数不足时显示报错信息

-打印两个长整数求和的竖式计算过程

## 实验步骤



需要编写随机生成长实数的函数与将两者相加的函数

### 测试xxx的测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否完成 |
| 随机生成随机数函数 | 多次调用函数 |  | 完成 |
| 两数相加函数 | 多次调用函数 |  | 完成 |

## 实验结论

代码能够达到功能目标

## 实验总结

该实验进一步的了解了数组

# 实验三

## 实验任务

-结构体内，用一维字符数组记录长实数的每个数位，通过ASCI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

-随机产生两个长整数，约定其最高位不得为0，其长度可以不同，但小数点位置相同

-对两个整数部分长度相同的长实数进行求和，有加法进位时显示报错信息

-打印两个长整数求和的竖式计算过程

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否能实现功能 |
| 随机产生小数的函数 | 多次调用函数 |  | 是 |
| 两数相加的函数 | 多次调用函数 |  | 是 |

## 实验结论

代码能够达到功能目标

## 实验总结

该实验融入了结构体部分，对结构体有了了解。

# 实验四

## 实验任务

- 结构体内，用一维字符数组记录长实数的每个数位，通过ASCI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

- 对两个整数部分长度不同的长实数进行求和，有加法进位时显示报错信息

- 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

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否能实现功能 |
| 随机设置小数点后的位数与小数点前的位数的函数 | 多次调用函数 |  | 是 |
| 两数相加的函数 | 多次调用函数 |  | 是 |

## 实验结论

代码能够达到功能目标

## 实验总结

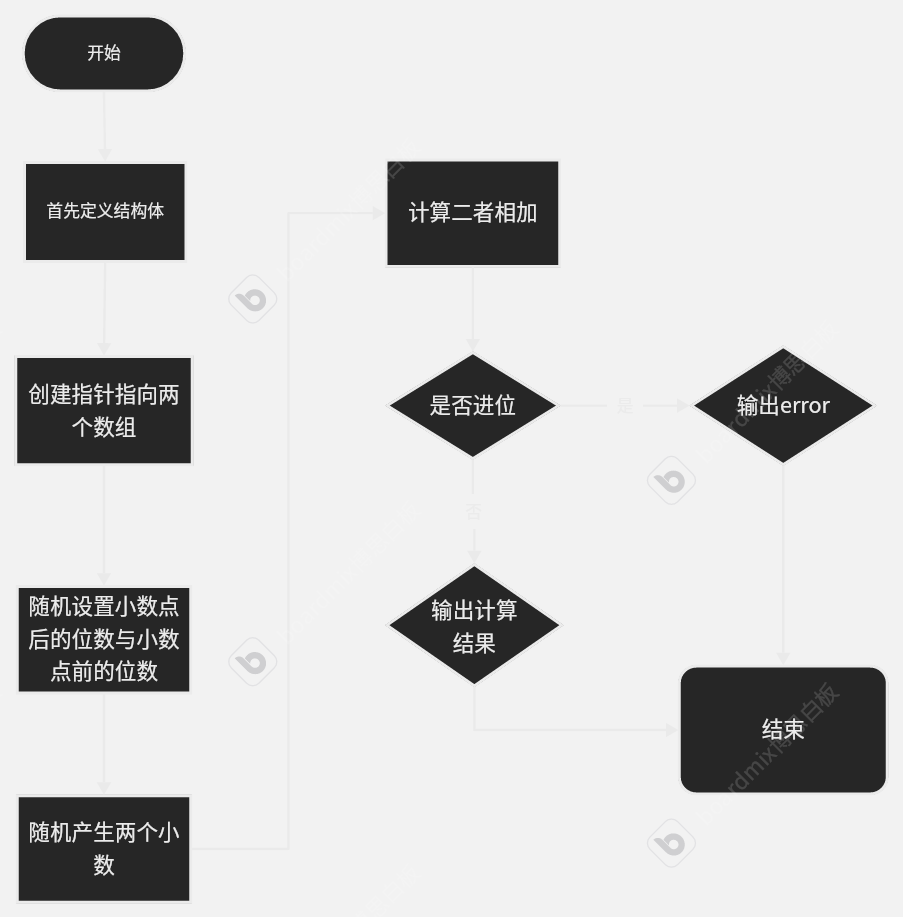
进一步加深了对有关结构体的操作的理解

# 实验五

## 实验任务

预期功能与前一个实验相同，要求修改对应的数组操作的语句，通过指针访问数组元素

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否能实现功能 |
| 随机设置小数点后的位数与小数点前的位数的函数 | 多次调用函数 |  | 是 |
| 两数相加的函数 | 多次调用函数 |  | 是 |

## 实验结论

代码能够达到功能目标

## 实验总结

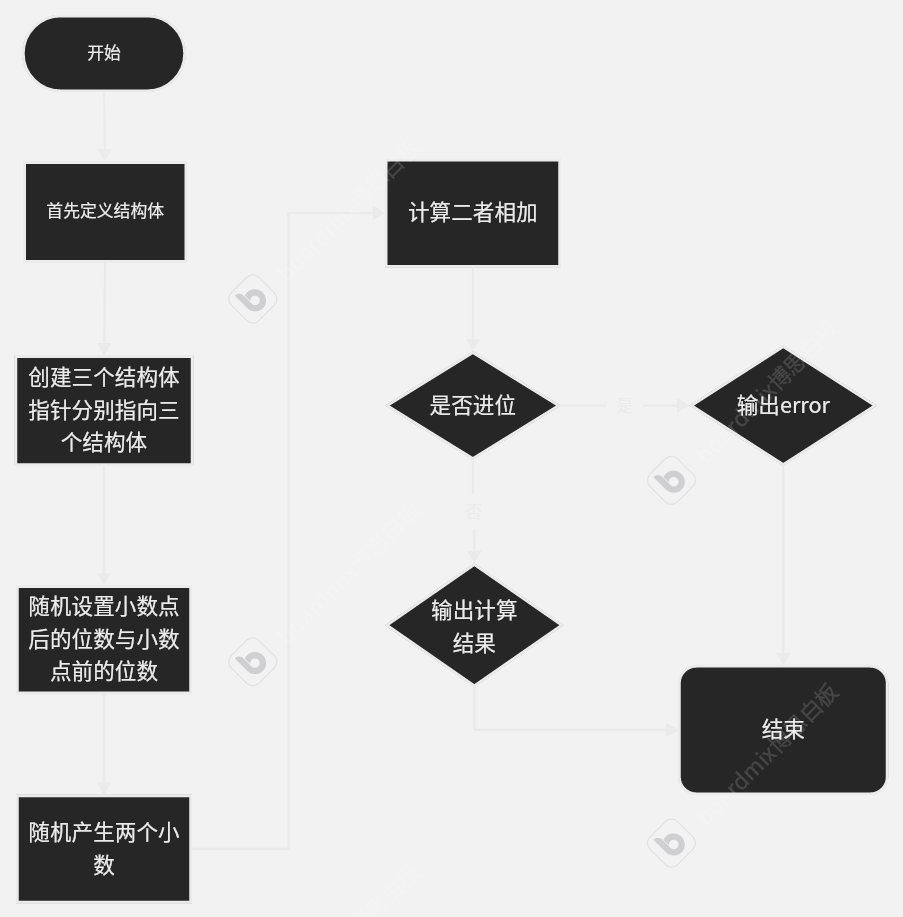
对比上一个实验，本次实验增加了有关指针的运用

# 实验六

## 实验任务

修改函数，通过指针操作相关数组、结构体的函数，实现基于引用的传递

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否能实现功能 |
| 随机设置小数点后的位数与小数点前的位数的函数 | 多次调用函数 |  | 是 |
| 两数相加的函数 | 多次调用函数 |  | 是 |

## 实验结论

代码能够达到功能目标

## 实验总结

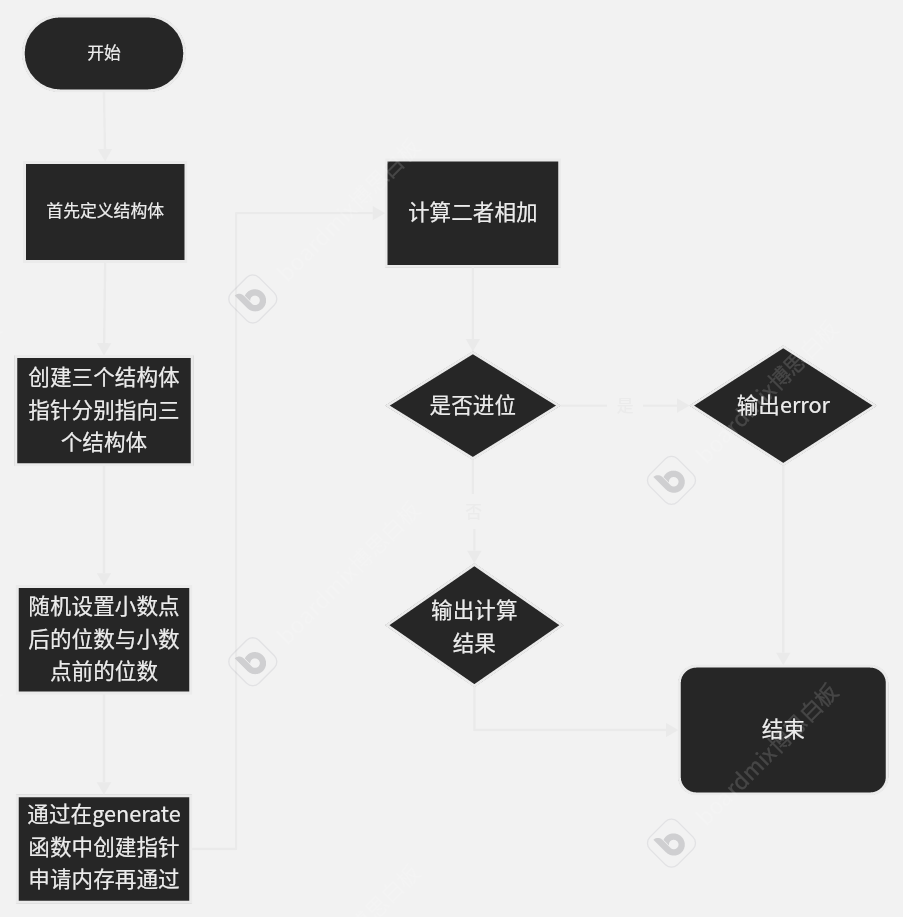
引入了有关结构体指针的各种操作。

# 实验七

## 实验任务

修改函数，通过指针申请内存创建结构体对象，通过返回结构体指针带回

## 实验步骤



## 代码测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试部分 | 测试方法 | 测试结果 | 是否能实现功能 |
| 随机设置小数点后的位数与小数点前的位数的函数 | 多次调用函数 |  | 是 |
| 两数相加的函数 | 多次调用函数 |  | 是 |

## 实验结论

代码能够达到功能目标

## 实验总结

加强了对动态分配内存的编程的掌握，还有通过指针返回值带回返回对象的方法

# 本课程学习总结

从对c语言一窍不通到现在略懂一二，我经历了很多，我也收获了很多。很多上课没有听懂的地方，在空余时我要东找找西找找才能一些资料从而将其搞懂，十分费力。对于程序中出现的错误。我认为有许多错误都是我犯过的，例如取地址&忘记了，数组越界了，内存未能够释放等等。同时，代码的书写规范我也是深受其害，经常会图一时方便而随意设置变量名等等。总而言之，一个学期c语言的学习带我进入了新的编程世界，我的编程能力也得到了长足进步，c语言的学习是挑战，但其中也充满着机遇。

# 附录

实验1：

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

void generateLongReal(char longReal[]);

int addTwoLongReals(char longReal1[], char longReal2[], char longRealSum[]);

void displayLongReal(char longReal[]);

int main(void) {

srand((unsigned)time(NULL));

char longReal1[REAL\_WIDTH] = {0};

generateLongReal(longReal1);

char longReal2[REAL\_WIDTH] = {0};

generateLongReal(longReal2);

char longRealSum[REAL\_WIDTH] = {0};

int isOverflow = addTwoLongReals(longReal1, longReal2, longRealSum);

printf("\n ");

displayLongReal(longReal1);

printf("+ ");

displayLongReal(longReal2);

if(isOverflow == 1) {

printf("Error! Addition overflow!\n");

}

else {

printf("= ");

displayLongReal(longRealSum);

}

printf("\n");

return 0;

}

void generateLongReal(char longReal[])

{

int i;

longReal[0] = '1' + rand() % 9;

for (i = 1; i < REAL\_WIDTH; i++) {

longReal[i] = '0' + rand() % 10;

}

}

int addTwoLongReals(char longReal1[], char longReal2[], char longRealSum[])

{

int i;

int go = 0;

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

int digit1 = longReal1[i] - '0';

int digit2 = longReal2[i] - '0';

int sum = digit1 + digit2 + go;

if (sum >= 10)

{

go = 1;

sum -= 10;

}

else

{

go = 0;

}

longRealSum[i] = sum + '0';

}

if (go == 1)

{

return 1;

}

return 0;

}

void displayLongReal(char longReal[])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

实验2：

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

void generateLongReal(char longReal[]);

int addTwoLongReals(char longReal1[], char longReal2[], char longRealSum[]);

void displayLongReal(char longReal[]);

int main(void) {

srand((unsigned)time(NULL));

char longReal1[REAL\_WIDTH] = {0};

generateLongReal(longReal1);

char longReal2[REAL\_WIDTH] = {0};

generateLongReal(longReal2);

char longRealSum[REAL\_WIDTH] = {0};

int isOverflow = addTwoLongReals(longReal1, longReal2, longRealSum);

printf("\n ");

displayLongReal(longReal1);

printf("+ ");

displayLongReal(longReal2);

if(isOverflow == 1) {

printf("Error! Addition overflow!\n");

}

else {

printf("= ");

displayLongReal(longRealSum);

}

printf("\n");

return 0;

}

void generateLongReal(char longReal[])

{

int i;

longReal[0] = '1' + rand() % 9;

for (i = 1; i < REAL\_WIDTH; i++) {

longReal[i] = '0' + rand() % 10;

}

longReal[POINT\_POSITION - 1] = '.';

}

int addTwoLongReals(char longReal1[], char longReal2[], char longRealSum[])

{

int i;

int go = 0;

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if (i == POINT\_POSITION - 1 )

{

longRealSum[i] = '.';

continue;

}

int digit1 = longReal1[i] - '0';

int digit2 = longReal2[i] - '0';

int sum = digit1 + digit2 + go;

if (sum >= 10)

{

go = 1;

sum -= 10;

}

else

{

go = 0;

}

longRealSum[i] = sum + '0';

}

if (go == 1)

{

return 1;

}

return 0;

}

void displayLongReal(char longReal[])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

实验3：

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

typedef struct

{

char num[ REAL\_WIDTH ];

int float\_Size;

} add;

void disPlay(char longReal[ ]);

void generate ( char longReal[ ] , int float\_Size );

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ]);

int main()

{

//定义三个add结构体,前两个给两个加数,后一个给和

add longReal[3];

srand(time(NULL));

// 产生随机小数位

longReal[0].float\_Size = rand( ) % (REAL\_WIDTH - POINT\_POSITION - 1) + 1 ; // 1 到 12 位小数,

longReal[1].float\_Size = rand( ) % (REAL\_WIDTH - POINT\_POSITION - 1) + 1 ; // 小数部分第13位为'\0'

// 打印两个长实数

printf("Generated (%d,%2d):",longReal[0].float\_Size + 12 , longReal[0].float\_Size ); //12代表11位整数加上一位小数点

generate( longReal[0].num , longReal[0].float\_Size);

disPlay( longReal[0].num ); // longReal.num 作为数组

printf("Generated (%d,%2d):",longReal[1].float\_Size + 12 , longReal[1].float\_Size );

generate( longReal[1].num ,longReal[1].float\_Size);

disPlay( longReal[1].num ); // longReal.num 作为数组

int isOut = addTwoLongReals(longReal[0].num ,longReal[1].num ,longReal[2].num);

if (isOut)

{

printf("Error! addition overflow!");

return 1 ;

}

int i; // 求出和数的小数位

int float\_size = 0;

for ( i = REAL\_WIDTH - 2 ; longReal[2].num[i] == '\_' ; i --) // REAL\_WIDTH-1是\0

{

float\_size++;

}

float\_size = REAL\_WIDTH - POINT\_POSITION - 1 - float\_size ; // 这么做的目的就是得到 和数的 小数位

printf("Sum Result(%d,%2d):",float\_size + 12 , float\_size);

disPlay(longReal[2].num);

printf("\n ");

disPlay(longReal[0].num);

printf("+ ");

disPlay(longReal[1].num);

printf("= ");

disPlay(longReal[2].num);

return 0;

}

void disPlay(char longReal[ ])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

void generate ( char longReal[ ] , int float\_Size )

{

int i;

// 首位设置为1-9之间的随机数，避免以0开头

longReal[0] = '1' + rand() % 9;

// 生成0-9之间的随机数

for (i = 1; i < 12 + float\_Size; i++)

{

longReal[i] = '0' + rand() % 10;

}

// 在指定位置放置小数点

longReal[POINT\_POSITION - 1] = '.';// 覆盖掉了[11]的值

for(; i < REAL\_WIDTH ; i++)

{

longReal[i] = '\_';

}

longReal[REAL\_WIDTH - 1] = '\0'; // 覆盖掉了[REAL\_WIDTH - 1] 的 \_ 字符

}

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ])

{

int i;

int go = 0; // 进位

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if(i == POINT\_POSITION - 1)

{

longRealSum[i] = '.'; // 添加小数点

continue;

}

int digit1;

int digit2;

if (longReal1[i] == '\_')

{

digit1 = 0;

}

else

{

digit1 = longReal1[i] - '0'; // 变成数位(int)

}

if (longReal2[i] == '\_')

{

digit2 = 0;

}

else

{

digit2 = longReal2[i] - '0';

}

if(longReal1[i] == '\_' && longReal2[i] == '\_')

{

longRealSum[i] = '\_';

continue; // 空数位,直接忽略

}

int sum = digit1 + digit2 + go; // 两数位相加再加进位

if (sum >= 10)

{

go = 1; // 进1,否则不进1

sum -= 10;

}

else

{

go = 0;

}

longRealSum[i] = sum + '0';

}

if (go == 1)

{

return 1; // 溢出了

}

return 0;

}

实验4：

#include <stdio.h>

#include <stdlib.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

typedef struct

{

char \*cptr;

char num[ REAL\_WIDTH ];

int float\_Size, size;

} add;

void disPlay(char longReal[ ]);

void generate ( char longReal[ ] , int float\_Size , int size);

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] ,int small);

int main()

{

//定义三个add结构体,前两个给两个加数,后一个给和

add longReal[3];

srand(time(NULL));

// 产生随机小数位

longReal[0].float\_Size = rand( ) % 12 + 1 ;

longReal[1].float\_Size = rand( ) % 12 + 1 ;

longReal[0].size = rand( ) % 12 + 1 ;

longReal[1].size = rand( ) % 12 + 1 ;

// 打印两个长实数

printf("Generated (%d,%02d):",longReal[0].float\_Size + 12 - longReal[0].size , longReal[0].float\_Size );

generate( longReal[0].num , longReal[0].float\_Size, longReal[0].size);

disPlay( longReal[0].num ); // longReal.num 作为数组

printf("Generated (%d,%02d):",longReal[1].float\_Size + 12 - longReal[1].size , longReal[1].float\_Size );

generate( longReal[1].num ,longReal[1].float\_Size, longReal[1].size);

disPlay( longReal[1].num ); // longReal.num 作为数组

int big = (longReal[0].float\_Size > longReal[1].float\_Size) ?

longReal[0].float\_Size : longReal[1].float\_Size ;

int small = (longReal[0].float\_Size < longReal[1].float\_Size) ?

longReal[0].float\_Size : longReal[1].float\_Size ;

int isOut = addTwoLongReals(longReal[0].num ,longReal[1].num ,longReal[2].num , small );

if (isOut)

{

printf("Error! addition overflow!");

return 1 ;

}

printf("Sum Result(%d,%02d):",big + 12 , big);

disPlay(longReal[2].num);

printf("\n ");

disPlay(longReal[0].num);

printf("+ ");

disPlay(longReal[1].num);

printf("= ");

disPlay(longReal[2].num);

return 0;

}

void disPlay(char longReal[ ])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

void generate ( char longReal[ ] , int float\_Size , int size)

{

int i;

// 首位设置为1-9之间的随机数，避免以0开头

//longReal[0] = '1' + rand() % 9;

// 生成0-9之间的随机数

for(i = 0; i < size; i++)

{

\*(longReal+i) = '\_';

}

\*(longReal + i) = '1' + rand() % 9;

for (i = size + 1; i < 12 + float\_Size; i++)

{

\*(longReal + i) = '0' + rand() % 10;

}

// 在指定位置放置小数点

longReal[POINT\_POSITION - 1] = '.';

for(; i < REAL\_WIDTH; i++)

{

\*(longReal + i) = '\_';

}

\*(longReal + REAL\_WIDTH - 1) = '\0';

}

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] , int small)

{

int i;

int go = 0; // 进位

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if(i == POINT\_POSITION - 1)

{

\*(longRealSum + i) = '.';

continue;

}

int digit1;

int digit2;

if (\*(longReal1 + i) == '\_')

{

digit1 = 0;

}

else

{

digit1= \*(longReal1 + i) - '0'; // 变成数位(int)

}

if (\*(longReal2 + i) == '\_')

{

digit2 = 0;

}

else

{

digit2= longReal2[i] - '0';

}

if(\*(longReal1 + i) == '\_' && \*(longReal2 + i) == '\_' && go == 0)

{

longRealSum[i] = '\_';

continue;

}

int sum = digit1 + digit2 + go; // 两数位相加再加进位

if (sum >= 10)

{

go = 1; // 进1,否则不进1

sum -= 10;

}

else

{

go = 0;

}

\*(longRealSum + i) = sum + '0';

}

if (go == 1)

{

return 1; // 溢出了

}

return 0;

}

实验5：

#include <stdio.h>

#include <stdlib.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

typedef struct

{

char \*cptr;

char num[ REAL\_WIDTH ];

int float\_Size, size;

} add;

void disPlay(char longReal[ ]);

void generate ( char longReal[ ] , int float\_Size , int size);

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] ,int small);

int main()

{

add longReal[3];

srand(time(NULL));

longReal[0].float\_Size = rand( ) % 12 + 1 ;

longReal[1].float\_Size = rand( ) % 12 + 1 ;

longReal[0].size = rand( ) % 12 + 1 ;

longReal[1].size = rand( ) % 12 + 1 ;

printf("Generated (%d,%02d):",longReal[0].float\_Size + 12 - longReal[0].size , longReal[0].float\_Size );

generate( longReal[0].num , longReal[0].float\_Size, longReal[0].size);

disPlay( longReal[0].num );

printf("Generated (%d,%02d):",longReal[1].float\_Size + 12 - longReal[1].size , longReal[1].float\_Size );

generate( longReal[1].num ,longReal[1].float\_Size, longReal[1].size);

disPlay( longReal[1].num );

int big = (longReal[0].float\_Size > longReal[1].float\_Size) ?

longReal[0].float\_Size : longReal[1].float\_Size ;

int small = (longReal[0].float\_Size < longReal[1].float\_Size) ?

longReal[0].float\_Size : longReal[1].float\_Size ;

int isOut = addTwoLongReals(longReal[0].num ,longReal[1].num ,longReal[2].num , small );

if (isOut)

{

printf("Error! addition overflow!");

return 1 ;

}

printf("Sum Result(%d,%02d):",big + 12 , big);

disPlay(longReal[2].num);

printf("\n ");

disPlay(longReal[0].num);

printf("+ ");

disPlay(longReal[1].num);

printf("= ");

disPlay(longReal[2].num);

return 0;

}

void disPlay(char longReal[ ])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

void generate ( char longReal[ ] , int float\_Size , int size)

{

int i;

for(i = 0; i < size; i++)

{

\*(longReal+i) = '\_';

}

\*(longReal + i) = '1' + rand() % 9;

for (i = size + 1; i < 12 + float\_Size; i++)

{

\*(longReal + i) = '0' + rand() % 10;

}

longReal[POINT\_POSITION - 1] = '.';

for(; i < REAL\_WIDTH; i++)

{

\*(longReal + i) = '\_';

}

\*(longReal + REAL\_WIDTH - 1) = '\0';

}

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] , int small)

{

int i;

int go = 0;

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if(i == POINT\_POSITION - 1)

{

\*(longRealSum + i) = '.';

continue;

}

int digit1;

int digit2;

if (\*(longReal1 + i) == '\_')

{

digit1 = 0;

}

else

{

digit1= \*(longReal1 + i) - '0';

}

if (\*(longReal2 + i) == '\_')

{

digit2 = 0;

}

else

{

digit2= longReal2[i] - '0';

}

if(\*(longReal1 + i) == '\_' && \*(longReal2 + i) == '\_' && go == 0)

{

longRealSum[i] = '\_';

continue;

}

int sum = digit1 + digit2 + go;

if (sum >= 10)

{

go = 1;

sum -= 10;

}

else

{

go = 0;

}

\*(longRealSum + i) = sum + '0';

}

if (go == 1)

{

return 1;

}

return 0;

}

实验6：

#include <stdio.h>

#include <stdlib.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

typedef struct

{

char \*cptr;

char num[ REAL\_WIDTH ];

int float\_Size, size;

} add;

void disPlay(char longReal[ ]);

void generate ( char longReal[ ] , int float\_Size , int size);

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] ,int small);

int main()

{

add longReal[3];

add \*Ptr1 = &longReal[0];

add \*Ptr2 = &longReal[1];

add \*Ptr3 = &longReal[2];

srand(time(NULL));

(\*Ptr1).float\_Size = rand( ) % 12 + 1 ;

(\*Ptr2).float\_Size = rand( ) % 12 + 1 ;

(\*Ptr1).size = rand( ) % 12 + 1 ;

(\*Ptr2).size = rand( ) % 12 + 1 ;

printf("Generated (%d,%02d):",(\*Ptr1).float\_Size + 12 - (\*Ptr1).size , (\*Ptr1).float\_Size );

generate( (\*Ptr1).num , (\*Ptr1).float\_Size, (\*Ptr1).size);

disPlay( (\*Ptr1).num );

printf("Generated (%d,%02d):",(\*Ptr2).float\_Size + 12 - (\*Ptr2).size , (\*Ptr2).float\_Size );

generate( (\*Ptr2).num ,(\*Ptr2).float\_Size, (\*Ptr2).size);

disPlay( (\*Ptr2).num );

int big = ((\*Ptr1).float\_Size > (\*Ptr2).float\_Size) ?

(\*Ptr1).float\_Size : (\*Ptr2).float\_Size ;

int small = ((\*Ptr1).float\_Size < (\*Ptr2).float\_Size) ?

(\*Ptr1).float\_Size : (\*Ptr2).float\_Size ;

int isOut = addTwoLongReals((\*Ptr1).num ,(\*Ptr2).num ,(\*Ptr3).num , small );

if (isOut)

{

printf("Error! addition overflow!");

return 1 ;

}

printf("Sum Result(%d,%02d):",big + 12 , big);

disPlay(longReal[2].num);

printf("\n ");

disPlay((\*Ptr1).num);

printf("+ ");

disPlay((\*Ptr2).num);

printf("= ");

disPlay((\*Ptr3).num);

return 0;

}

void disPlay(char longReal[ ])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

void generate ( char longReal[ ] , int float\_Size , int size)

{

int i;

for(i = 0; i < size; i++)

{

\*(longReal+i) = '\_';

}

\*(longReal + i) = '1' + rand() % 9;

for (i = size + 1; i < 12 + float\_Size; i++)

{

\*(longReal + i) = '0' + rand() % 10;

}

longReal[POINT\_POSITION - 1] = '.';

for(; i < REAL\_WIDTH; i++)

{

\*(longReal + i) = '\_';

}

\*(longReal + REAL\_WIDTH - 1) = '\0';

}

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] , int small)

{

int i;

int go = 0;

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if(i == POINT\_POSITION - 1)

{

\*(longRealSum + i) = '.';

continue;

}

int digit1;

int digit2;

if (\*(longReal1 + i) == '\_')

{

digit1 = 0;

}

else

{

digit1= \*(longReal1 + i) - '0';

}

if (\*(longReal2 + i) == '\_')

{

digit2 = 0;

}

else

{

digit2= longReal2[i] - '0';

}

if(\*(longReal1 + i) == '\_' && \*(longReal2 + i) == '\_' && go == 0)

{

longRealSum[i] = '\_';

continue;

}

int sum = digit1 + digit2 + go;

if (sum >= 10)

{

go = 1;

sum -= 10;

}

else

{

go = 0;

}

\*(longRealSum + i) = sum + '0';

}

if (go == 1)

{

return 1;

}

return 0;

}

实验7：

#include <stdio.h>

#include <stdlib.h>

#define REAL\_WIDTH 25

#define POINT\_POSITION 12

typedef struct

{

char \*cptr;

char num[ REAL\_WIDTH ];

int float\_Size, size;

} add;

void disPlay(char longReal[ ]);

add \*generate (void );

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] ,int small);

int main()

{

srand(time(NULL));

add longReal[3];

add \*Ptr1 = generate();

add \*Ptr2 = generate();

add \*Ptr3 = generate();

printf("Generated (%d,%02d):",(\*Ptr1).float\_Size + 12 - (\*Ptr1).size , (\*Ptr1).float\_Size );

disPlay( (\*Ptr1).num );

printf("Generated (%d,%02d):",(\*Ptr2).float\_Size + 12 - (\*Ptr2).size , (\*Ptr2).float\_Size );

disPlay( (\*Ptr2).num );

int big = ((\*Ptr1).float\_Size > (\*Ptr2).float\_Size) ?

(\*Ptr1).float\_Size : (\*Ptr2).float\_Size ;

int small = ((\*Ptr1).float\_Size < (\*Ptr2).float\_Size) ?

(\*Ptr1).float\_Size : (\*Ptr2).float\_Size ;

int isOut = addTwoLongReals((\*Ptr1).num ,(\*Ptr2).num ,(\*Ptr3).num , small );

if (isOut)

{

printf("Error! addition overflow!");

return 1 ;

}

printf("Sum Result(%d,%02d):",big + 12 , big);

disPlay((\*Ptr3).num);

printf("\n ");

disPlay((\*Ptr1).num);

printf("+ ");

disPlay((\*Ptr2).num);

printf("= ");

disPlay((\*Ptr3).num);

return 0;

}

void disPlay(char longReal[ ])

{

int i;

for (i = 0; i < REAL\_WIDTH; i++) {

printf("%c", longReal[i]);

}

printf("\n");

}

add \*generate (void)

{

add \*p = (add \*)malloc(sizeof(add));

(\*p).float\_Size = rand( ) % 12 + 1 ;

(\*p).size = rand( ) % 12 + 1 ;

int i;

for(i = 0; i < (\*p).size; i++)

{

(\*p).num[i] = '\_';

}

(\*p).num[i] = '1' + rand() % 9;

for (i = (\*p).size + 1; i < 12 + (\*p).float\_Size; i++)

{

(\*p).num[i] = '0' + rand() % 10;

}

(\*p).num[POINT\_POSITION - 1] = '.';

for(; i < REAL\_WIDTH; i++)

{

(\*p).num[i] = '\_';

}

(\*p).num[REAL\_WIDTH - 1] = '\0';

return p;

}

int addTwoLongReals(char longReal1[ ], char longReal2[ ], char longRealSum[ ] , int small)

{

int i;

int go = 0;

for (i = REAL\_WIDTH - 1; i >= 0 ; i --)

{

if(i == POINT\_POSITION - 1)

{

\*(longRealSum + i) = '.';

continue;

}

int digit1;

int digit2;

if (\*(longReal1 + i) == '\_')

{

digit1 = 0;

}

else

{

digit1= \*(longReal1 + i) - '0';

}

if (\*(longReal2 + i) == '\_')

{

digit2 = 0;

}

else

{

digit2= longReal2[i] - '0';

}

if(\*(longReal1 + i) == '\_' && \*(longReal2 + i) == '\_' && go == 0)

{

longRealSum[i] = '\_';

continue;

}

int sum = digit1 + digit2 + go;

if (sum >= 10)

{

go = 1;

sum -= 10;

}

else

{

go = 0;

}

\*(longRealSum + i) = sum + '0';

}

if (go == 1)

{

return 1;

}

return 0;

}