

电子信息与通信学院

实 验 报 告

|  |  |
| --- | --- |
| 实验名称 | 大数计算系列 |
| 课程名称 | 计算机基础  与程序设计(C) |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 吴佳瑞 | 学号 | U202415480 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2025.1 | 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：Dev-C++ 5.11

# 实验一

## 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

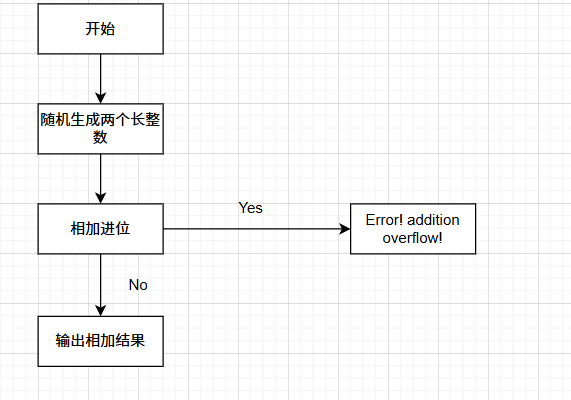
--打印两个长整数求和的竖式计算过程

要求完成的函数：

// Sum two long integers

int addTwoLongInts(char longInt1[], char longInt2[], char longIntSum[]);

## 实验步骤



关键代码：

int addTwoLongInts(char longInt1[], char longInt2[], char longIntSum[])

{

int carry = 0;

int i;

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

{

int digit1 = longInt1[i] - '0';

int digit2 = longInt2[i] - '0';

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longIntSum[i] = sum % 10 + '0';

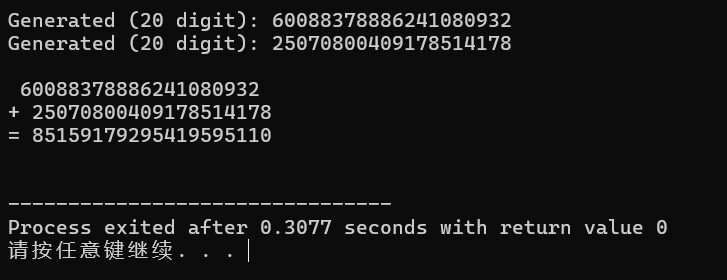
}

return carry;

}

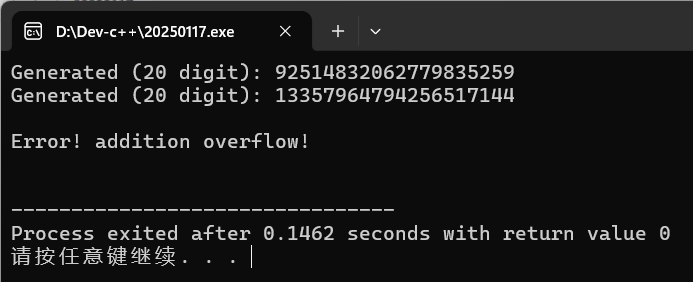
## 代码测试

1.运行程序，测试对两个相同长度的长整数进行求和功能，打印两个长整数求和的竖式计算过程



结论：测试通过

2.运行程序，测试有加法进位是否报错



结论：测试通过

## 实验结论

代码达到功能目标

## 实验总结

针对实验中出现的问题进行总结

gnerateLongInt时要注意把’0’是首位的情况排除掉

计算这种大数相加，应从后往前加，并用一个数代表是否进位

# 实验二

## 实验任务

--一堆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进行求和，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

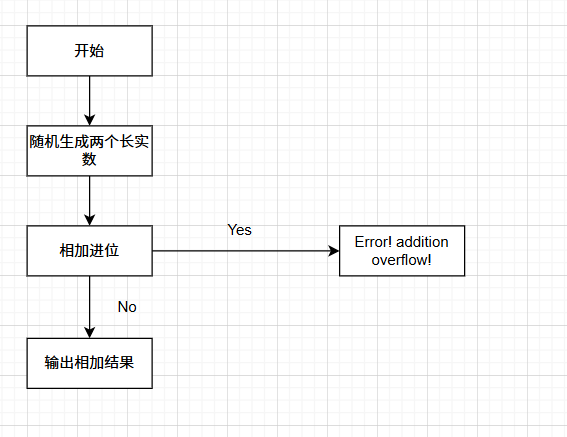
--打印两个长实数求和的竖式计算过程

要求完成的函数：

// sum two long real numbers

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

## 实验步骤



关键代码：

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

{

int carry = 0;

int i;

int pointPosition1 = -1, pointPosition2 = -1, pointPositionSum = -1;

// Find the positions of decimal points in the two numbers and the sum

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

{

if (longReal1[i] == '.')

pointPosition1 = i;

if (longReal2[i] == '.')

pointPosition2 = i;

if (longRealSum[i] == '.')

pointPositionSum = i;

}

// Add the integer parts before the decimal point

for (i = pointPosition1 - 1; i >= 0; i--)

{

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

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

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longRealSum[i] = sum % 10 + '0';

}

if (carry > 0)

return 1;

longRealSum[pointPosition1] = '.';

// Add the fractional parts after the decimal point

carry = 0;

for (i = pointPosition1 + 1; i < REAL\_WIDTH; i++)

{

int digit1 = longReal1[i] == '.'? 0 : longReal1[i] - '0';

int digit2 = longReal2[i] == '.'? 0 : longReal2[i] - '0';

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longRealSum[i] = sum % 10 + '0';

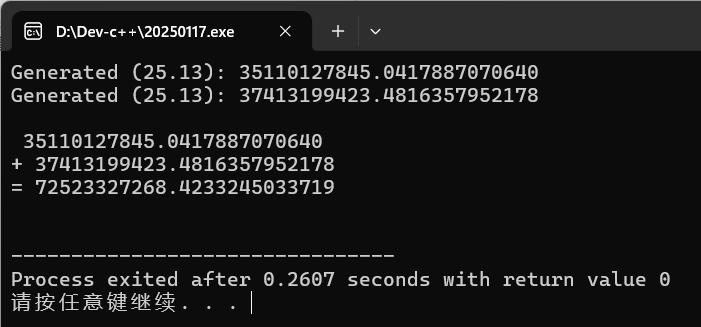
}

return 0;

}

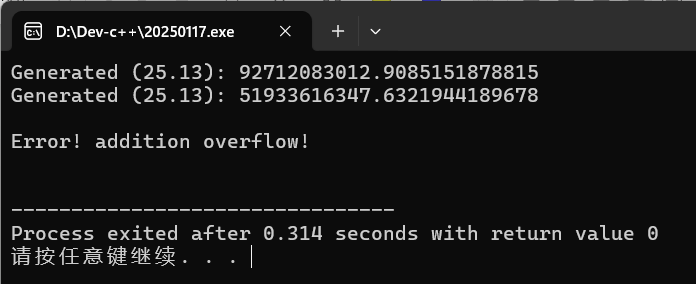
## 代码测试

1. 运行程序，观测是否实现将两个长实数相加并展示竖式计算过程



结论：测试通过

1. 运行程序，测试有加法进位是否报错



结论：测试通过

## 实验结论

代码达到功能目标

## 实验总结

在addTwoLongReals自定义函数里要注意分别去出整数部分和小数部分进行加减，并在最后的数组里中间插入小数点

# 实验三

## 实验任务

--结构体内，用一堆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

--随机产生两个长实数，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

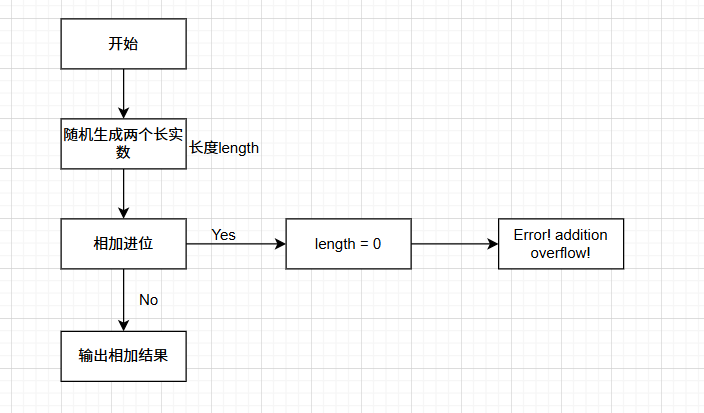
--打印两个长整数求和的竖式计算过程

要求完成的函数：

// Sum two long real numbers

Real addTwoLongReals(Real r1, Real r2);

## 实验步骤



关键代码：

Real addTwoLongReals(Real r1, Real r2)

{

Real sum;

sum.length = (r1.length > r2.length)? r1.length : r2.length;

int carry = 0;

int i;

int pointPosition1 = -1, pointPosition2 = -1;

// Find the positions of decimal points in the two numbers

for (i = 0; i < MAX\_REAL\_WIDTH; i++)

{

if (r1.digits[i] == '.')

pointPosition1 = i;

if (r2.digits[i] == '.')

pointPosition2 = i;

}

// Add the integer parts before the decimal point

for (i = pointPosition1 - 1; i >= 0; i--)

{

int digit1 = r1.digits[i] - '0';

int digit2 = r2.digits[i] - '0';

int sumDigit = digit1 + digit2 + carry;

carry = sumDigit / 10;

sum.digits[i] = sumDigit % 10 + '0';

}

sum.digits[pointPosition1] = '.';

if (carry > 0)

{

sum.length = 0;

return sum;

}

// Add the fractional parts after the decimal point

carry = 0;

for (i = pointPosition1 + 1; i < sum.length; i++)

{

int digit1 = (i < r1.length && r1.digits[i]!= '.')? r1.digits[i] - '0' : 0;

int digit2 = (i < r2.length && r2.digits[i]!= '.')? r2.digits[i] - '0' : 0;

int sumDigit = digit1 + digit2 + carry;

carry = sumDigit / 10;

sum.digits[i] = sumDigit % 10 + '0';

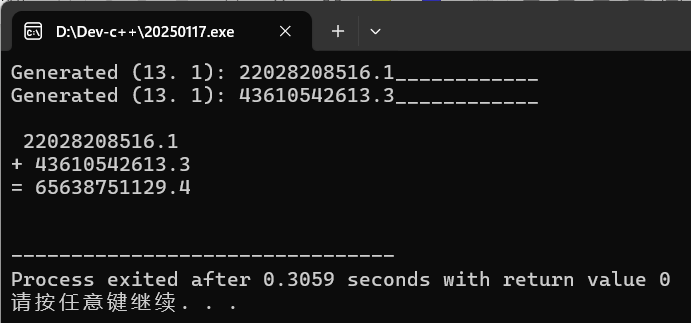
}

return sum;

}

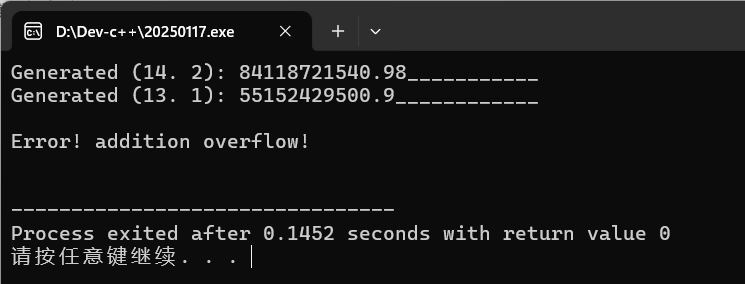
## 代码测试

1.运行程序，观测是否实现将两个长实数相加并展示竖式计算过程



结论：测试通过

2.运行程序，运行程序，测试有加法进位是否报错



结论：测试通过

## 实验结论

代码达到功能目标

## 实验总结

在addTwoLongReals自定义函数里面要注意分别去出整数部分和小数部分进行加减，并在最后的数组里中间插入小数点，而且通过改变sum包含的length来判断是否error

# 本课程学习总结

C语言是一种美妙的编程语言，与较简单粗暴的python不同，C语言蕴含了更多的”大道法则“，比如不管怎样都需要引入stdio这个库来做到实质的输出。刚开始学C语言有点不适应，因为跟python虽然思想一样，但是语句风格和细节方面差距过大，让我学的很难受，时不时就想用python粗暴咔咔了结。后面学到日历部分更让我十分痛苦，到编写日历这个部分难度对我而言就上来了。虽然我试图认真听讲并尽可能吸收老师讲述的知识，但是还是一知半解，后面学指针,动态内存更是心态爆炸，感受到很大的压力。一道题目，可能看上去简单，但是需要投入的时间可能不会如看上去那般少，因为我首先要保证基本功能，又要排除用户输入的各种情况，而且编程的过程中可能还会犯下一些小错误，虽然小但却很致命。所以完成一个程序，我谨慎的打代码，检查回车，大括号成对等潜在的问题（后来熟练了就快了），先完成主程序，再考虑所需要的自定义函数。代码命名等需要规范，而且最好需要一定的注释，增加可读性（虽然我老是忘记写注释）。调试代码时，最好有针对性的输入各类数据，多运行几次，这样能尽快找出错误的地方。学习C语言，需要认真听老师讲的课，需要从网上寻找课程加深理解，还很需要自己上手实践，提高熟练度的同时更产生对C语言的一种”亲和感“，那么在后续的学习中即可保持激情，感悟精心完成一个程序的快乐和C语言的魅力。

# 附录

完整实验代码附在此处

1. 长整数相加实验
2. main.c

int main(void)

{

srand(time(NULL));

char longInt1[INT\_WIDTH] = {0};

generateLongInt(longInt1);

char longInt2[INT\_WIDTH] = {0};

generateLongInt(longInt2);

char longIntSum[INT\_WIDTH] = {0};

int isOverflow = addTwoLongInts(longInt1, longInt2, longIntSum);

if (isOverflow == 0)

{

printf("\n ");

displayLongInt(longInt1);

printf("+ ");

displayLongInt(longInt2);

printf("= ");

displayLongInt(longIntSum);

}

else if (isOverflow == 1)

{

printf("\nError! addition overflow!\n");

}

printf("\n");

return 0;

}

1. generateLongInt.h

void generateLongInt(char longInt[])

{

int i;

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

for (i = 1; i < INT\_WIDTH; i++)

{

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

}

printf("Generated (%d digit): ", INT\_WIDTH);

displayLongInt(longInt);

return;

}

1. displayLongInt.h

void displayLongInt(char longInt[])

{

int i;

for (i = 0; i < INT\_WIDTH; i++)

{

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

}

printf("\n");

return;

}

1. addTwoLongInts.h

int addTwoLongInts(char longInt1[], char longInt2[], char longIntSum[])

{

int carry = 0;

int i;

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

{

int digit1 = longInt1[i] - '0';

int digit2 = longInt2[i] - '0';

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longIntSum[i] = sum % 10 + '0';

}

return carry;

}

1. 长实数相加实验
2. main.c

int main(void)

{

srand(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);

if (isOverflow == 0)

{

printf("\n ");

displayLongReal(longReal1);

printf("+ ");

displayLongReal(longReal2);

printf("= ");

displayLongReal(longRealSum);

}

else if (isOverflow == 1)

{

printf("\nError! addition overflow!\n");

}

printf("\n");

return 0;

}

1. generateLongReal.h

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] = '.';

printf("Generated (%d.%d): ", REAL\_WIDTH, REAL\_WIDTH - POINT\_POSITION);

displayLongReal(longReal);

return;

}

1. displayLongReal.h

void displayLongReal(char longReal[])

{

int i;

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

{

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

}

printf("\n");

return;

}

1. addTwoLongReals.h

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

{

int carry = 0;

int i;

int pointPosition1 = -1, pointPosition2 = -1, pointPositionSum = -1;

// Find the positions of decimal points in the two numbers and the sum

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

{

if (longReal1[i] == '.')

pointPosition1 = i;

if (longReal2[i] == '.')

pointPosition2 = i;

if (longRealSum[i] == '.')

pointPositionSum = i;

}

// Add the integer parts before the decimal point

for (i = pointPosition1 - 1; i >= 0; i--)

{

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

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

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longRealSum[i] = sum % 10 + '0';

}

if (carry > 0)

return 1;

longRealSum[pointPosition1] = '.';

// Add the fractional parts after the decimal point

carry = 0;

for (i = pointPosition1 + 1; i < REAL\_WIDTH; i++)

{

int digit1 = longReal1[i] == '.'? 0 : longReal1[i] - '0';

int digit2 = longReal2[i] == '.'? 0 : longReal2[i] - '0';

int sum = digit1 + digit2 + carry;

carry = sum / 10;

longRealSum[i] = sum % 10 + '0';

}

return 0;

}

1. 固定点小数相加实验
2. main.c

int main(void)

{

srand(time(NULL));

Real real1 = generateLongReal();

Real real2 = generateLongReal();

Real realSum = addTwoLongReals(real1, real2);

if (realSum.length > 0)

{

printf("\n ");

displayLongReal(real1);

printf("+ ");

displayLongReal(real2);

printf("= ");

displayLongReal(realSum);

}

else

{

printf("\nError! addition overflow!\n");

}

printf("\n");

return 0;

}

1. structed type

typedef struct {

char digits[MAX\_REAL\_WIDTH];

int length;

} Real;

1. generateLongReal.h

Real generateLongReal(void)

{

Real r;

int fractionLength = rand() % (MAX\_REAL\_WIDTH - POINT\_POSITION + 1);

r.length = POINT\_POSITION + fractionLength;

int i;

r.digits[0] = '1' + rand() % 9;

for (i = 1; i < r.length - 1; i++)

{

r.digits[i] = '0' + rand() % 10;

}

r.digits[r.length - 1] = '1' + rand() % 9;

r.digits[POINT\_POSITION - 1] = '.';

for (i = r.length; i < MAX\_REAL\_WIDTH; i++)

{

r.digits[i] = '\0';

}

printf("Generated (%2d.%2d): ", r.length, r.length - POINT\_POSITION);

displayFixedWidthNumber(r.digits, MAX\_REAL\_WIDTH);

return r;

}

1. addTwoLongReals.h

Real addTwoLongReals(Real r1, Real r2)

{

Real sum;

sum.length = (r1.length > r2.length)? r1.length : r2.length;

int carry = 0;

int i;

int pointPosition1 = -1, pointPosition2 = -1;

// Find the positions of decimal points in the two numbers

for (i = 0; i < MAX\_REAL\_WIDTH; i++)

{

if (r1.digits[i] == '.')

pointPosition1 = i;

if (r2.digits[i] == '.')

pointPosition2 = i;

}

// Add the integer parts before the decimal point

for (i = pointPosition1 - 1; i >= 0; i--)

{

int digit1 = r1.digits[i] - '0';

int digit2 = r2.digits[i] - '0';

int sumDigit = digit1 + digit2 + carry;

carry = sumDigit / 10;

sum.digits[i] = sumDigit % 10 + '0';

}

sum.digits[pointPosition1] = '.';

if (carry > 0)

{

sum.length = 0;

return sum;

}

// Add the fractional parts after the decimal point

carry = 0;

for (i = pointPosition1 + 1; i < sum.length; i++)

{

int digit1 = (i < r1.length && r1.digits[i]!= '.')? r1.digits[i] - '0' : 0;

int digit2 = (i < r2.length && r2.digits[i]!= '.')? r2.digits[i] - '0' : 0;

int sumDigit = digit1 + digit2 + carry;

carry = sumDigit / 10;

sum.digits[i] = sumDigit % 10 + '0';

}

return sum;

}

1. displayLongReal.h

void displayLongReal(Real r)

{

int i;

for (i = 0; i < r.length; i++)

{

printf("%c", r.digits[i]);

}

printf("\n");

return;

}

1. displayFixedWidthNumber.h

void displayFixedWidthNumber(char number[], const int width)

{

int i;

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

{

if (number[i] == '\0')

{

printf("\_");

}

else

{

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

}

}

printf("\n");

return;

}