1. **实验目的：（字体大小：仿宋 四号 加粗）**

1.主要目的：通过完成有重复元素的排列问题与集合划分问题，深入理解递归算法在解决组合数学问题中的核心作用，掌握利用递归思想构建算法模型的方法，提升运用编程解决复杂逻辑问题的能力

2.基本原理和流程：掌握排列组合问题中重复元素处理及集合划分的数学原理，剖析递归函数在实现这些算法时的执行流程，包括递归边界条件的设定与递归调用过程，明晰算法的逻辑架构。

3.通过实践，熟悉特定工具的使用：在编程实践中，熟练运用 C++ 语言的标准库函数（如 next\_permutation、set）以及自定义函数，理解其功能与使用场景，提升编程工具的操作熟练度，增强代码编写的效率与规范性。

4.培养解决实际问题的能力：将实际问题抽象为数学模型，再转化为编程实现，锻炼问题分析、算法设计、代码调试等综合能力，学会运用递归算法解决实际生活中类似的组合优化问题。

**二、实验环境**

1.windows11家庭与中文版 2.CLion 2024.3.4版本

1. **实验原理及相关知识**

（一）相关技术概述

排列问题：有重复元素的排列问题属于组合数学领域，其技术特点在于需要在计算排列数时消除重复元素带来的冗余排列。核心挑战在于如何准确计算重复元素对排列总数的影响，通常采用组合数学中的公式，通过全排列数除以重复元素的阶乘来得到不重复的排列个数。常用处理框架为构建递归函数，结合回溯算法思想，在生成排列的过程中判断并处理重复情况

集合划分问题：集合划分问题涉及计算将一个集合划分为指定个数子集的方案数，其核心是计算斯特林数。技术特点是通过递归关系进行计算，挑战在于理解递归公式中各参数的含义与计算逻辑。常用的处理框架是基于斯特林数的递归定义，编写递归函数实现计算

（二）环境或系统概述

本实验基于 C++ 编程环境实现。系统核心组件包括输入输出流（iostream）用于数据的读取与输出，标准模板库（STL）中的相关容器（如 string、set）和算法（如 next\_permutation）用于数据存储与处理，自定义函数用于实现具体的算法逻辑。输入输出流负责与用户交互，接收输入数据并展示计算结果；容器和算法提供高效的数据操作方式；自定义函数根据算法原理完成计算任务，各组件相互协作，实现实验功能

（三）工具或框架介绍

C++ 语言：C++ 是一种通用的、高效的编程语言，具有强大的性能和丰富的库函数。其优势在于支持面向对象编程、泛型编程等多种编程范式，能够灵活地构建复杂的程序结构。在本实验中，利用 C++ 语言可以方便地定义函数、操作数据结构，通过调用标准库函数简化编程过程，适用于解决各类算法问题。

STL 库：标准模板库（STL）为 C++ 提供了大量的模板类和函数，如 set 用于存储不重复元素，可自动进行去重操作；next\_permutation 函数能够生成给定序列的下一个排列，极大地简化了排列问题的实现过程。这些工具的特点是高度抽象和复用性强，优势在于减少代码编写量，提高程序的可读性和可维护性，广泛应用于算法设计、数据处理等场景。

1. **实验内容及结果**

**（一）**有重复元素的排列问题

1.实验内容：有重复元素的排列问题

2.实验步骤：

（1）主要模块

int get(char b[], int xx[], int n, string c)函数：

遍历输入的字符串 c，记录每个不同字符及其对应的出现次数，存储到数组中，并返回不同字符的个数

long factorial(int n) 函数：计算 n!

int count(int xx[], int n)函数：

计算不同排列数量，考虑重复字符的影响，假设字符串有n个字符，则每个位置有n种选择，再消去重复字符对应的阶乘，即可得到总共的排列个数

void issort(char b[], int xx[], int n) 函数：

构造待排列字符串并排序；使用 next\_permutation 生成所有可能的排列；用set 去重，并输出最终结果。

main() 函数：

读取文件内容，调用各个函数处理数据。计算不同排列个数并输出。

（2）错误纠正

A.不同字符获取：最初没有引入布尔变量标注已处理的字符，无法进行正确输出

void get(char b[],int xx[],int n,string c){

int ct=0;

for(int i=0;i<n;i++){

b[ct]=c[i];

ct++;

for(int j=0;j<=ct;j++){

if(c[i]==b[j]) {ct--;}

}

}

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(b[i]==c[j]) {

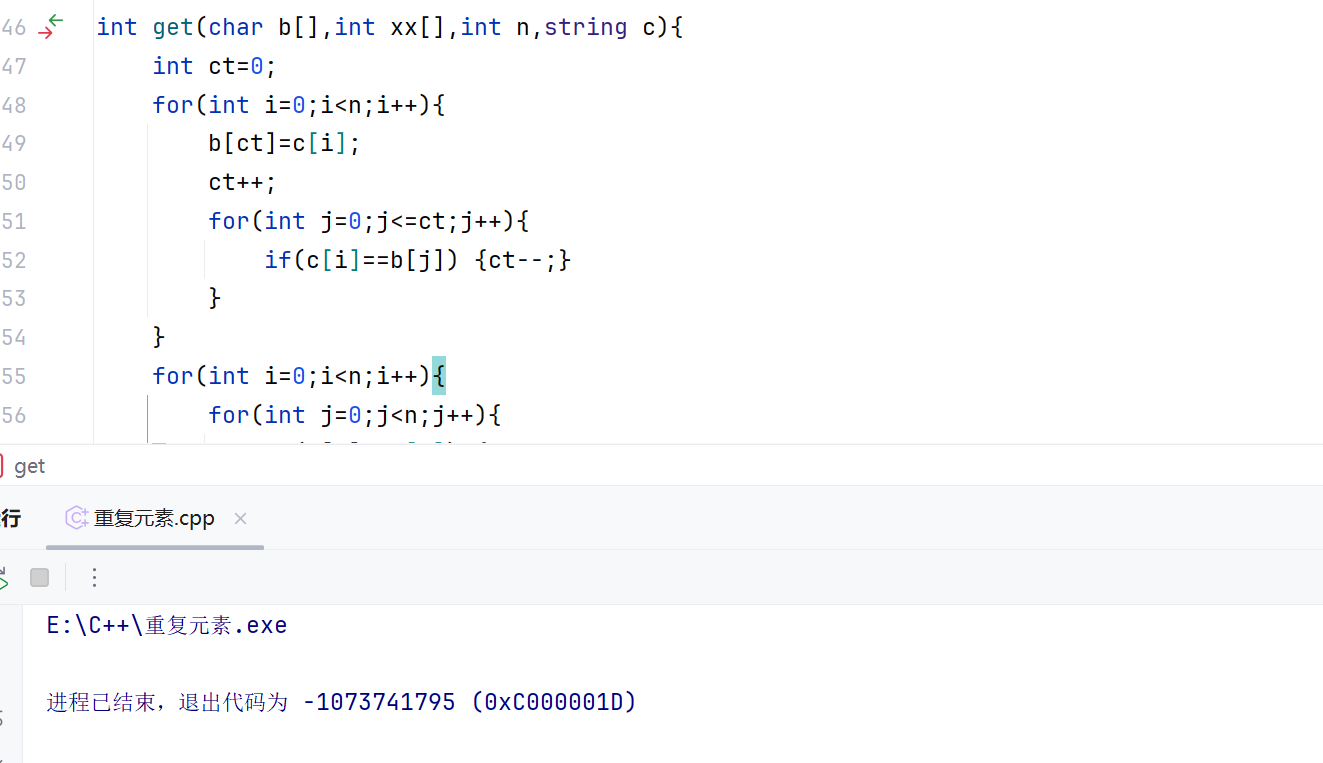
xx[i]++;

}

}

}

}



B. 计数模块

错误写法：应该用字符串总长度的阶乘进行消序，但错误的选择了用不同字符的个数来求阶乘消序，造成结果不正确。

int count(int xx[], int ct) {

long sum = factorial(ct+1);

for (int i = 0; i < ct+1; i++) {

if (xx[i]>=2 && xx[i] <= ct+1) {

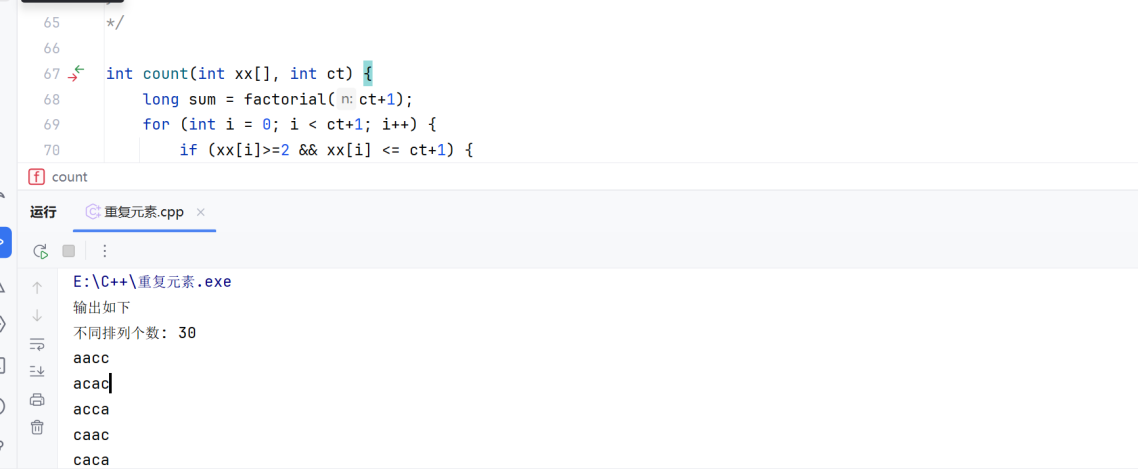
sum /= factorial(xx[i]);

}

}

return sum; // 添加返回值

}



C.字符串排列出错

没有正确的递归思路，此部分最终由AI生成，采用字典排序的方法

void issort(char b[],int xx[],int n){

int num=0;

char k[n];

for(int i=0;i<n;i++){

if(xx[i]>0)

{ k[num]=b[i];

xx[i]--; } num++; }

if(num==n-1){

for(int i = 0; i < n; i++) {

cout << b[i];

}

cout << endl;

return; }

}

issort() 采用 next\_permutation 生成排列，并用 set 去重

sort(s.begin(), s.end()) 确保 next\_permutation 从字典序最小排列开始

set 结构保证了唯一性，但可以优化为直接使用 next\_permutation 输出

正确写法：

void issort(char b[],int xx[],int n){

int num=0;

char k[n];

for(int i=0;i<n;i++){

if(xx[i]>0){

k[num]=b[i];

xx[i]--;

}

num++;

}

if(num==n-1){

for(int i = 0; i < n; i++) {

cout << b[i];

}

cout << endl;

return;

}

}

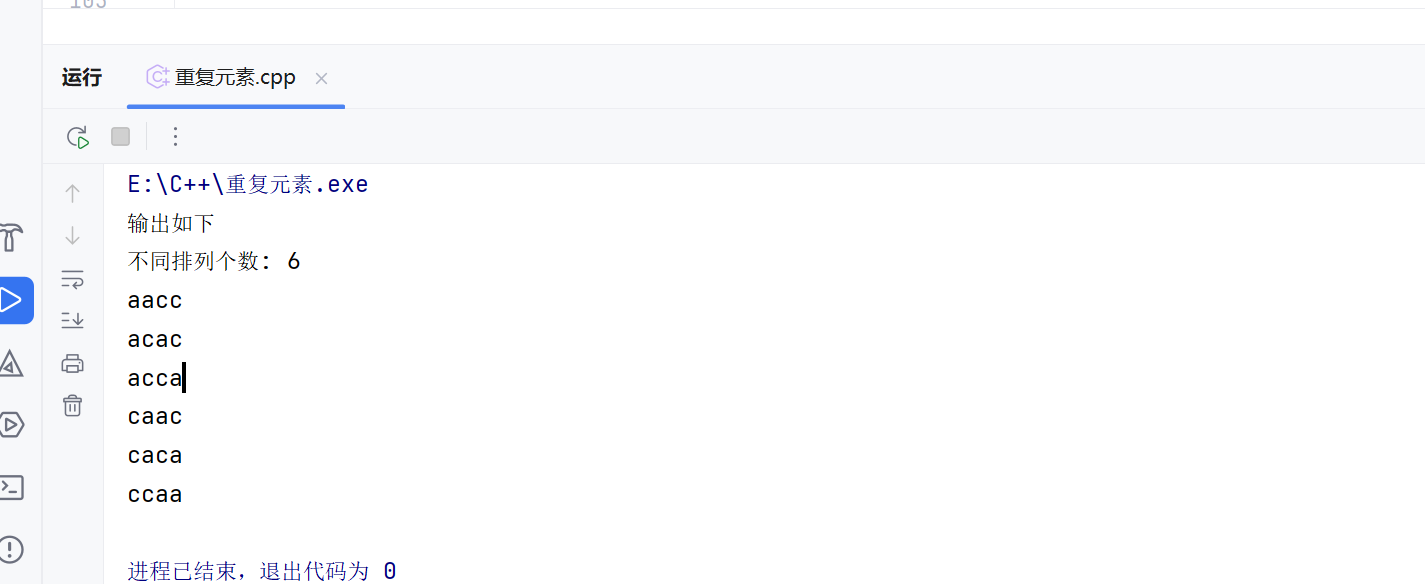
3.实验结果

测试结果正确

输入： 4

aacc

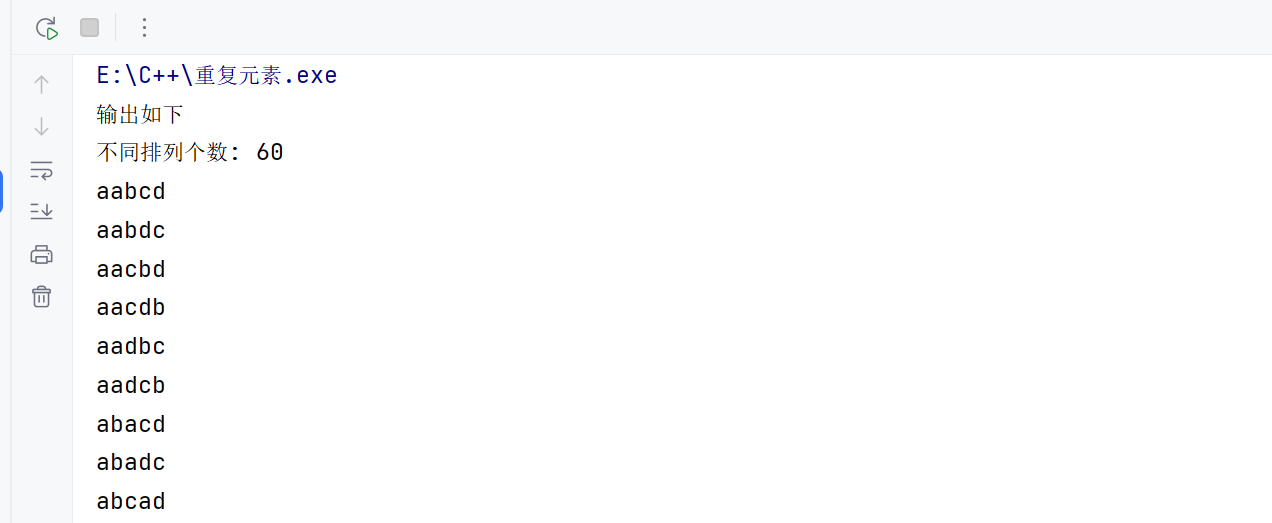
输出：



输入：5

abcda

输出：



aabcd aabdc aacbd aacdb aadbc aadcb abacd abadc abcad abcda abdac abdca acabd acadb acbad acbda acdab acdba adabc adacb

adbac adbca adcab adcba baacd baadc bacad bacda badac badca bcaad bcada bcdaa bdaac bdaca bdcaa caabd caadb cabad cabda

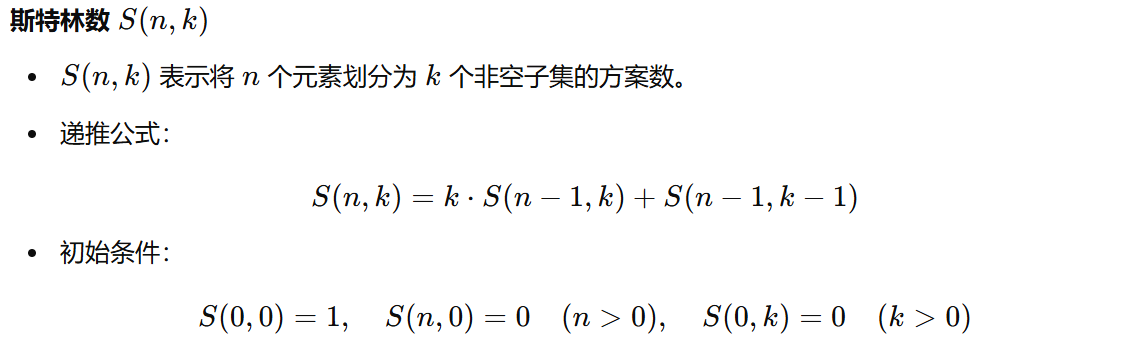
cadab cadba cbaad cbada cbdaa cdaab cdaba cdbaa daabc daacb dabac dabca dacab dacba dbaac dbaca dbcaa dcaab dcaba dcbaa

**（二）**集合划分问题

1.实验内容：集合划分问题

2.实验步骤：

（1）主要模块：



int ss(int n,int k)函数：求斯塔林数

int sum(int n)函数：求斯塔林数总和

main()函数：进行输入输出

（3）错误修正

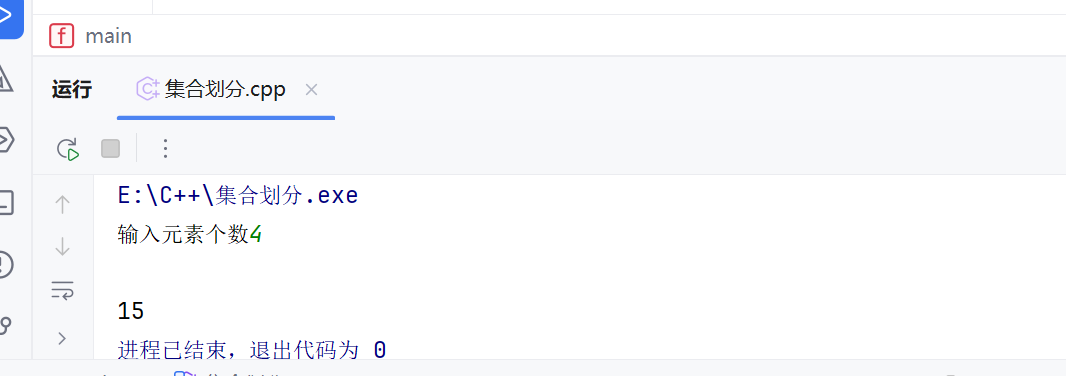


A.求斯塔林数时返回错误，递归书写有误，特殊情况和最后递归的返回值存在混淆

B.Int sum函数为整型，有返回参数，但忘记书写

3.实验结果：





**五、讨论、心得**

1.有重复元素的排列问题采用通过 next\_permutation 实现全排列，并利用 set 去重，提高了正确性

但除了字典排序，还可以采用递归的思路：

使用回溯（Backtracking）的方法，字符的不同位置对应不同递归，每一层递归中选择一个未被使用的字符，并记录已经选择的字符，递归地生成排列，并在到达 n 长度时存储到 set 或 vector 中。最后进行回溯，恢复可选字符，找到所有排列

采用递归可更加直接生成排列，避免去重带来的消耗。

2.集合划分问题相对简单，就是基本的递归函数实现，对斯塔林数的递归公式用代码实现，并进行求和。书写时需要注意有返回值，return一定要写对。

整体难度不高，能够自行进行正确实现