离散数学期末项目



**课程项目设计说明书**

项 目 名 称 ： 组合数学函数

项 目 成 员 ：卢倚桐、陈梦琳、陈泓佳、吴海铭、 霍信波

开 发 日 期 ： 2022 年11 月 21 日

指 导 教 师 ： 付晓倩 目录

[一、相关理论知识 2](#_Toc6054)

[二、功能模块概要 2](#_Toc27159)

[三、功能模块分析 2](#_Toc21020)

[四、项目总结与展望 3](#_Toc25373)

### 一、相关理论知识

1.阶乘函数：N!=1\*2\*3\*….\*n.

2.二项式系数函数：对于任意实数a和b以及非负整数n，二项式定理可以表示为：(a + b)^n = C(n, 0) \* a^n \* b^0 + C(n, 1) \* a^(n-1) \* b^1 + ... + C(n, n) \* a^0 \* b^n。其中C(n, k)表示二项式系数，表示从n个元素中选择k个元素的组合数。二项式系数函数可以实现组合数的计算、排列数的计算、二项分布的概率计算、杨辉三角形。

（1）组合数的计算：C(n, k) = n! / (k! \* (n-k)!)，其中n!表示n的阶乘。

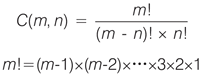
（2）排列数的计算：P(n, k) = n! / (n-k)!。

（3）二项分布的概率计算：P(X = k) = C(n, k) \* p^k \* (1-p)^(n-k)，其中X表示成功次数，n表示试验次数，k表示成功次数，p表示每次试验成功的概率。

（4）多项式展开：

（5）杨辉三角形中，每个数等于它上方两数之和。

3.组合数函数: 组合数 C(m,n) 可以理解为从 m 个数中任意取出 n 个数的所有情况数。在数学中，求组合数 C(m,n) 的值可以借助 m 和 n 的阶乘来计算，计算公式为：



4.插板法函数: 基本题型:有n个相同的元素，要求分到m组中，并且要求每组中至少有一个元素问有多少种分法？

方法：将n个相同的元素排成一行，n个元素之间出现了（n-1）个空档，现在我们用（m-1）个“档板”插入（n-1）个空档中，就把n个元素隔成有序的m份，每个组依次按组序号分到对应位置的几个元素（可能是1个、2个、3个、4个、….），这样不同的插入办法就对应着n个相同的元素分到m组的一种分法，这种借助于这样的虚拟“档板”分配元素的方法称之为插板法。

5.斐波那契数列函数：第一和第二个数均为 1，之后每个数都等于其前面两个数之和，形如：1, 1, 2, 3, 5, 8, 13, 21, ...。

6.卡特兰数函数：C(n) = C(0) \* C(n - 1) + C(1) \* C(n - 2) + ....C(n - 2) \* C(1) + C(n - 1) \* C(0).(n >= 2)

  化简得 C(n) = c(2 \* n, n) / (n + 1). 形如： 1, 1, 2, 5, 14, 42, 132,。

7.钩子公式函数: 钩子函数是在系统在进行消息的传递处理时候利用钩子机制截取消息，可以对一些特定的消息进行处理。

像一些系统的开发功能可以基于消息处理机制，对消息进行初始化、动态创建分配空间，然后转发消息，对各个模块的消息进行实时监听，而钩子函数就是监听消息在传递到制定的模块窗口时候的操作，如果消息没有满足条件，钩子函数就可以把它进行处理然后让它无法最后成功达到目的地。如果满足了条件，就可以继续往后面传递。

8.拆分数函数: 拆分数函数就是把n划分成所有元素不大于n的组合。

以n=4为例，4=1+1+1+1=2+1+1=2+2=3+1\n");

无重复元素为:{1,3}

有重复元素为{1，1，1，1}，{2，1，1}，{2，2}

### 二、功能模块概要

说明项目开发主要解决的问题，做出概要介绍并明确团队分工。

项目主要解决的是帮助、计算各类函数，减少在计算时的琐碎的过程，比如输入数值计算阶乘的结果，二项式的各类运算，组合数的计算，插板函数计算时间，计算斐波那契数列的各类问题，构造卡特兰函数，钩子函数分析，拆分数函数分析。

各项功能编辑

陈梦琳同学：阶乘、拆分数

霍信波：二项式

吴海铭：组合数、插板函数计算时间

卢倚桐：斐波那契数列、构造卡特兰函数

陈泓佳：钩子函数

### 功能模块分析

功能1：阶乘函数功能分析

·算法描述：

通过for语句循环实现阶乘n！的运算，并实现阶乘求和。

·主要代码：for (int i = 1; i < n + 1; i++) {

m = m \* i;

}

printf("n!=%d\n", m);

for (int i = 1; i < n + 1; i++) {

a = 1;

for (int j = 1; j < i + 1; j++) {

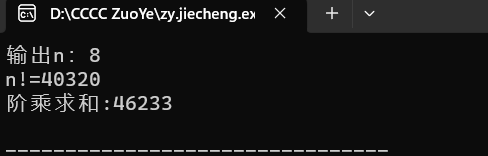
a = a \* j;

}

sum = sum + a;

}

·运行效果抓图：

****

功能2：二项式系数系统模块分析

·算法描述：

计算二项式系数。整合了计算组合数、排列数、计算二项分布的概率、多项式展开、打印杨辉三角的功能。

//定义选择变量名

int choice;

printf("\n");

printf("\033[1;33m");// 设置文本颜色为亮黄色，1表示将后续文本的样式设置为加粗或高亮样式，33为黄色

printf("\*\*\*\*\*\* 二项式系数函数功能 \*\*\*\*\*\*\n");

printf("\033[0m");// 恢复默认文本颜色，避免影响下面内容

do{

printf("\n");

//打印用户选项

printf("1. 计算组合数\n");

printf("2. 计算排列数\n");

printf("3. 计算二项分布的概率\n");

printf("4. 多项式展开\n");

printf("5. 打印杨辉三角形\n");

printf("0. 退出程序\n");

printf("\n");

printf("请选择要执行的功能：\n");

scanf("%d", &choice);

//根据用户选项执行相应的功能

switch (choice) {

case 1:

calculateCombination();// 计算组合数

break;

case 2:

calculatePermutation();// 计算排列数

break;

case 3:

calculateBinomialDistribution();// 计算二项分布的概率

break;

case 4:

polynomialExpansion();// 多项式展开

break;

case 5:

printPascalTriangle();// 打印杨辉三角形

break;

case 0:

printf("已退出二项式系数函数功能。");

break;

default:

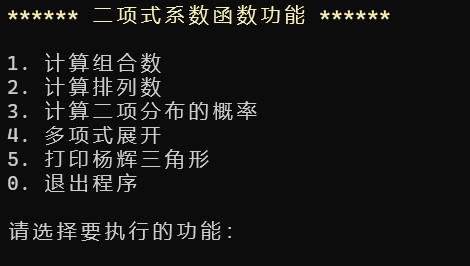
printf("无效的选项\n");

break;

}

} while (choice != 0);

运行效果：

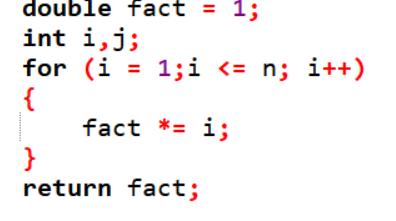


功能3：组合数模块分析

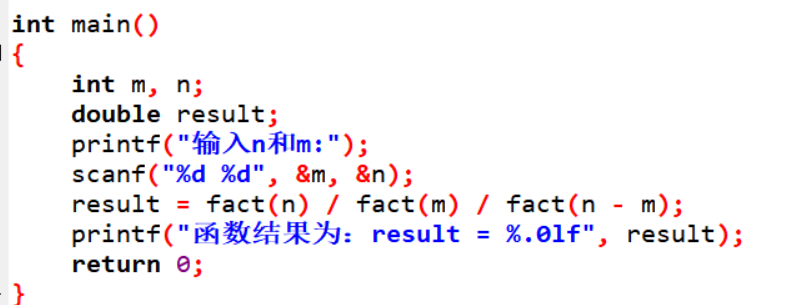
·算法描述：

在函数中可以快速输出求解的组合数

用for循环算出n的阶乘



再根据组合数公式来计算、并输出结果。



·主要代码

//组合数函数

#include <stdio.h>

double fact(int n);

int main()

{

int m, n;

double result;

printf("输入n和m:");

scanf("%d %d", &m, &n);

result = fact(n) / fact(m) / fact(n - m);

printf("函数结果为：result = %.0lf", result);

return 0;

}

double fact(int n){

double fact = 1;

int i,j;

for (i = 1;i <= n; i++)

{

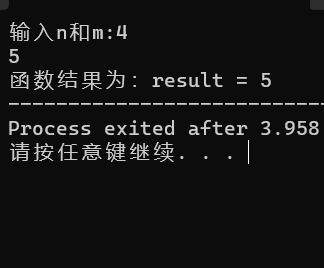
fact \*= i;

}

return fact;

}

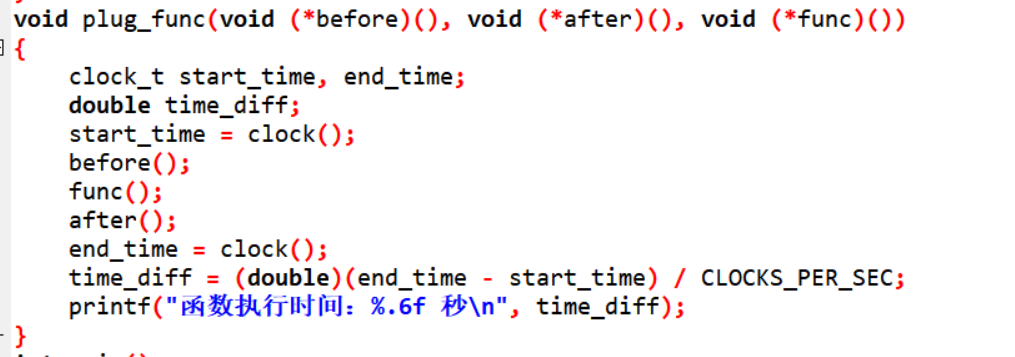
运行效果：



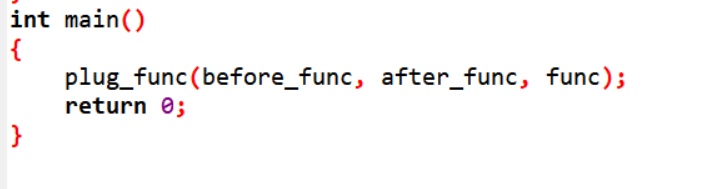
功能4：通过插板函数实现函数运行时间的计算

·算法描述：

plug\_func()` 的插板函数，该函数接收三个函数作为参数：`before()`、`after()` 和 `func()`。其中 `func()` 是被插板的函数，`before()` 和 `after()` 分别是插板函数的前置和后置操作，用于输出一些信息。在 `plug\_func()` 中，使用了 `clock()` 函数获取函数执行前后的时间，并计算出执行时间差。



在 `main()` 中调用 `plug\_func()` 函数，以 `func()` 作为被插板的函数。运行程序后，会输出以下信息：



·主要代码

//插板

#include <stdio.h>

#include <time.h>

void before\_func()

{

printf("函数开始执行...\n");

}

void after\_func()

{

printf("函数执行结束...\n");

}

void func()

{

int i, sum = 0;

for (i = 0; i < 100000000; i++)

{

sum += i;

}

printf("sum: %d\n", sum);

}

void plug\_func(void (\*before)(), void (\*after)(), void (\*func)())

{

clock\_t start\_time, end\_time;

double time\_diff;

start\_time = clock();

before();

func();

after();

end\_time = clock();

time\_diff = (double)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

printf("函数执行时间：%.6f 秒\n", time\_diff);

}

int main()

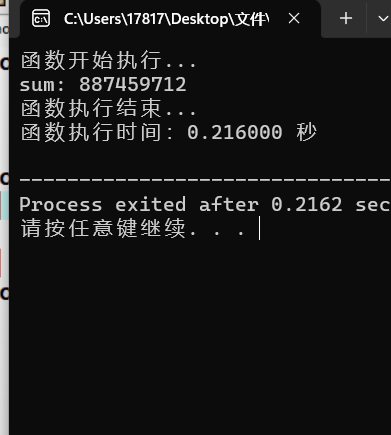
{

plug\_func(before\_func, after\_func, func);

return 0;

}

运行结果：

功能5：构造斐波那契函数功能模块分析

* 算法描述：

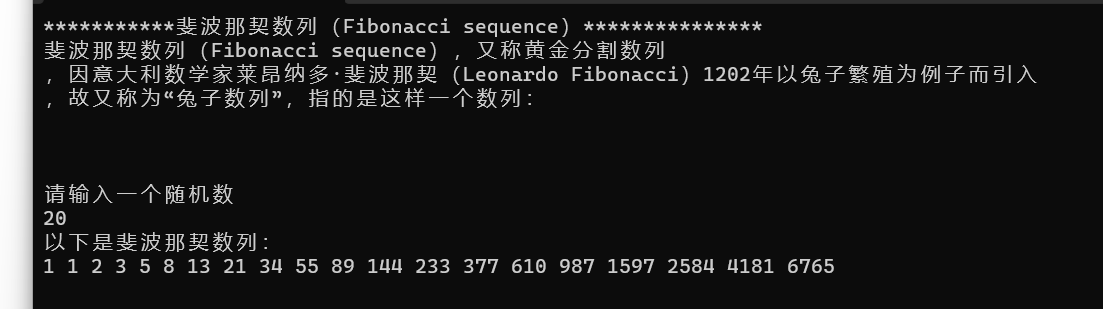
功能：用C语言构造函数斐波那契函数fib（函数的封装）。先输入具体的数进行测试，后面再通过用户输入的数和调用以构造的函数解决应用问题。

作用：解决：(1)兔子繁殖问题

(2)矩形覆盖问题

(3)青蛙跳台阶问题

结果：



* 主要代码：

//构造斐波那契函数

// index 表示求数列中第 index 个位置上的数的值

int fib(int index) {

// 设置结束递归的限制条件

if (index == 1 || index == 2) {

return 1;

}

return fib(index - 1) + fib(index - 2);

}

//兔子繁殖问题

int month;

int young = 1;

int adult = 0;

int total = 0;

printf("请输入兔子要经过的月份数：\n");

scanf("%d",&month);

printf("经过月数\t幼崽对数\t成兔对数\t总兔对数\n");

for(int i = 0;i < month;i++)

{

if(i > 0 && i % 3 == 0)

{

//每3个月幼兔到成兔

adult += young;

young = 0;

}

total = young+adult;

printf("%d\t\t%d\t\t%d\t\t%d\n",i + 1,young,adult,total);

young = total;

}

//矩形覆盖

int count;

printf("请输入矩形长度\n");

scanf("%d",&count);

int total = fib(count);

printf("2 x %d 的矩形可以用 %d 种方法进行覆盖", count, total);

break;

//青蛙跳台

int step;

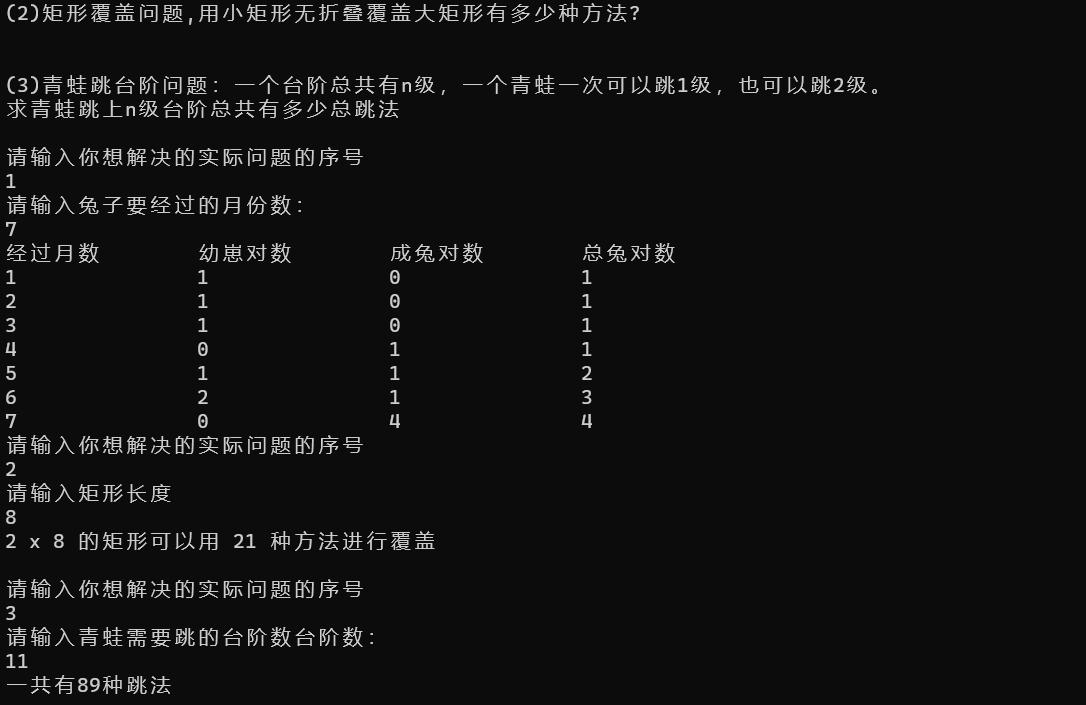
printf("请输入青蛙需要跳的台阶数台阶数：\n");

scanf("%d", &n);

int result = fib(n);

printf("一共有%d种跳法\n", result);

* 运行效果抓图：



功能6：构造卡特兰函数功能模块分析

* 算法描述：

功能：用C语言构造函数卡特兰函数Catalan（函数的封装）。先输入具体的数进行测试，后面再通过用户输入的数和调用以构造的函数解决应用问题。

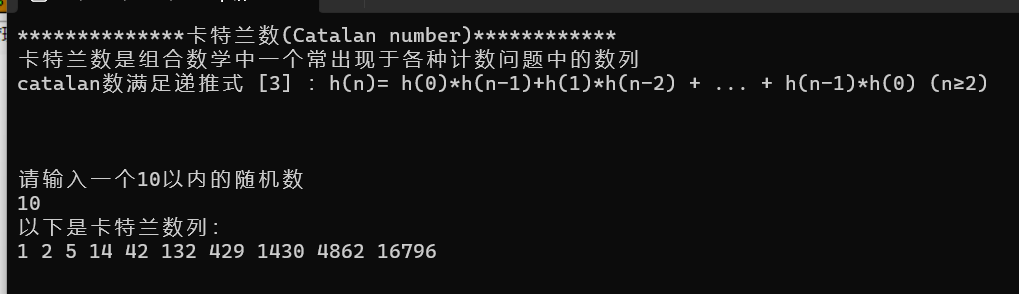
作用：解决： (1)结点二叉树问题

(2)括号匹配问题

(3)凸出栈序列

(4)找零问题

**结果：**

****

* 主要代码：

//卡特兰函数

long long catalan(int n)

{

if (n <= 1)

{

return 1;

}

long long res = 0;

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

{

res += catalan(i) \* catalan(n - i - 1);

}

return res;

}

typedef struct {

int x, y;

} Point;

//两个向量的叉积

int cross\_product(Point a, Point b, Point c)

{

int ac\_x = c.x - a.x;

int ac\_y = c.y - a.y;

int ab\_x = b.x - a.x;

int ab\_y = b.y - a.y;

return ac\_x \* ab\_y - ab\_x \* ac\_y;

}

switch(choice)

{

case 1:

{

//二叉树

int point;

printf("输入需要多少个节点构成的二叉树\n ");

scanf("%d", &point);

printf("%d个节点构成的二叉树共有%lu种情形\n\n", point, catalan(n));

break;

}

case 2:

//括号 {

int bracket;

printf("输入括号数\n ");

scanf("%d", &bracket);

printf("%d个括号数%lu种匹配方式\n\n", bracket, catalan(n));

break;

}

//凸出栈序列

case 3:

{

int degree;

printf("请输入栈的大小：");

scanf("%d", &degree);

printf("不同的凸出栈序列的数量为：%llu\n\n",catalan(degree) );

break;

}

//找零问题

case 4:

{

int num;

printf("请输入排成一行进入剧场的人数\n\n");

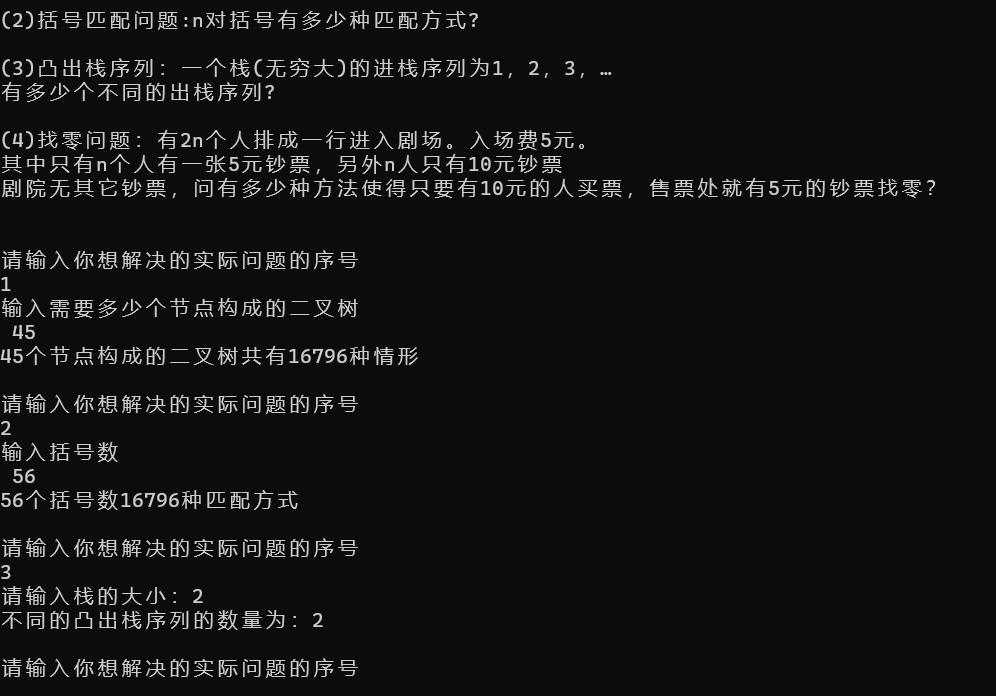
scanf("%d", &num);

printf("有%lld种方法使得只要有10元的人买票，售票处就有5元的钞票找零\n\n",catalan(num) );

break;

}

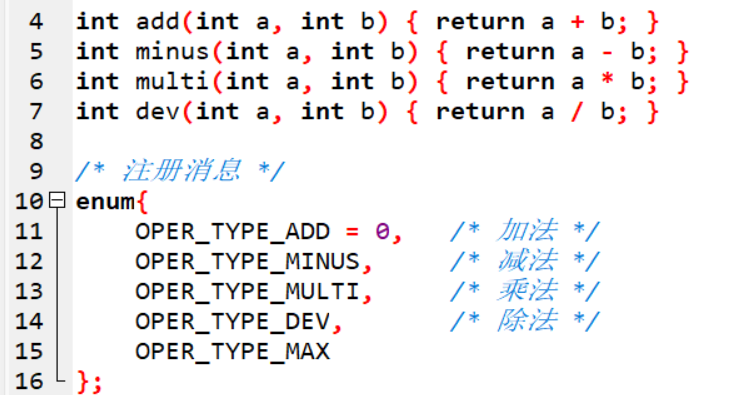
* 运行效果抓图：



**功能7：**钩子函数功能分析

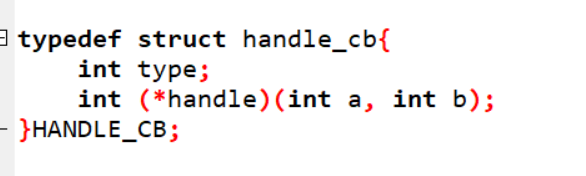
* 算法描述：

1.



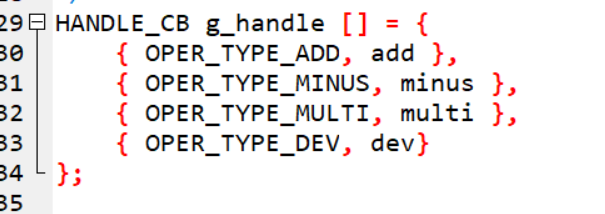
此部份为方法定义;这里确定了加减乘除的四种运算方法，后续利用钩子函数特性进行调用；

2.

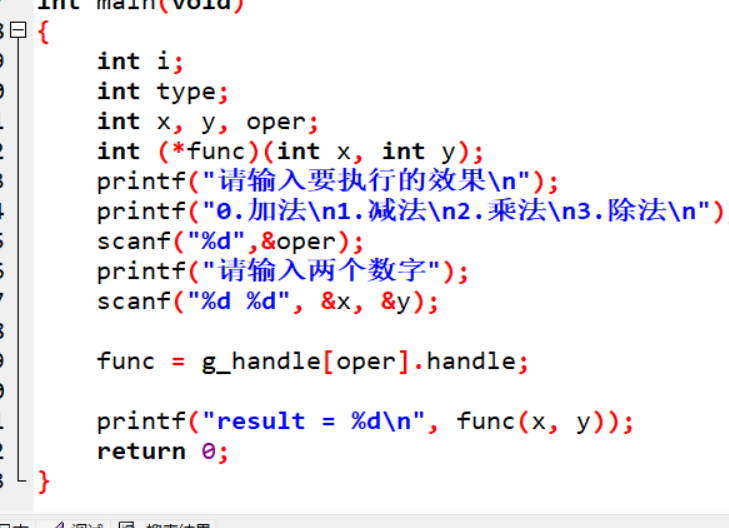


此部分为指针应用，利用指针的特性寻找函数；

3.



此部分为函数的主体内容，指针找到该函数，然后进行调用执行。



此部分为主题函数；

* 主要代码：

#include <stdio.h>

int add(int a, int b) { return a + b; }

int minus(int a, int b) { return a - b; }

int multi(int a, int b) { return a \* b; }

int dev(int a, int b) { return a / b; }

/\* 注册消息 \*/

enum{

OPER\_TYPE\_ADD = 0, /\* 加法 \*/

OPER\_TYPE\_MINUS, /\* 减法 \*/

OPER\_TYPE\_MULTI, /\* 乘法 \*/

OPER\_TYPE\_DEV, /\* 除法 \*/

OPER\_TYPE\_MAX

};

typedef struct handle\_cb{

int type;

int (\*handle)(int a, int b);

}HANDLE\_CB;

/\*

注册钩子，这里显式地把消息类型和钩子函数对应关系体现出来了，

handle\_cb里面的type可以不要的，注意维护好注册消息表中

每个枚举值先后顺序和下面钩子函数表中的先后顺序一致，能对应上

就行了

\*/

HANDLE\_CB g\_handle [] = {

{ OPER\_TYPE\_ADD, add },

{ OPER\_TYPE\_MINUS, minus },

{ OPER\_TYPE\_MULTI, multi },

{ OPER\_TYPE\_DEV, dev}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main程序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

int i;

int type;

int x, y, oper;

int (\*func)(int x, int y);

printf("请输入要执行的效果\n");

printf("0.加法\n1.减法\n2.乘法\n3.除法\n");

scanf("%d",&oper);

printf("请输入两个数字");

scanf("%d %d", &x, &y);

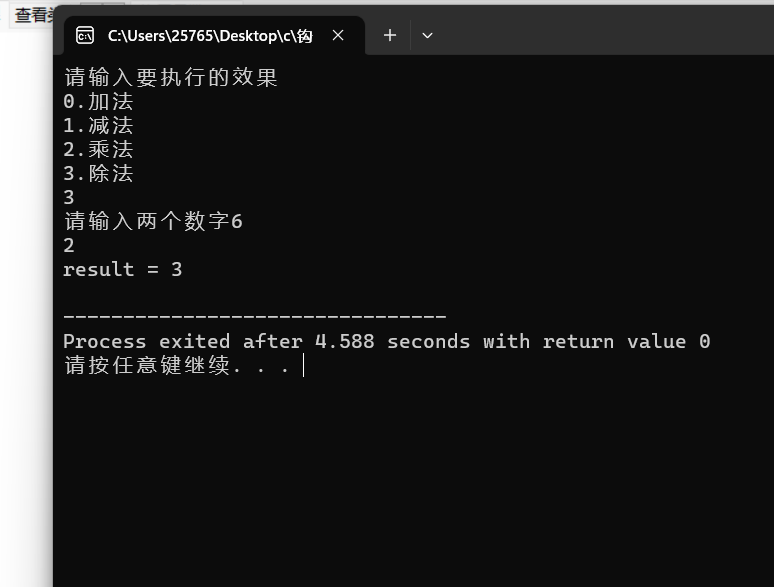
func = g\_handle[oper].handle;

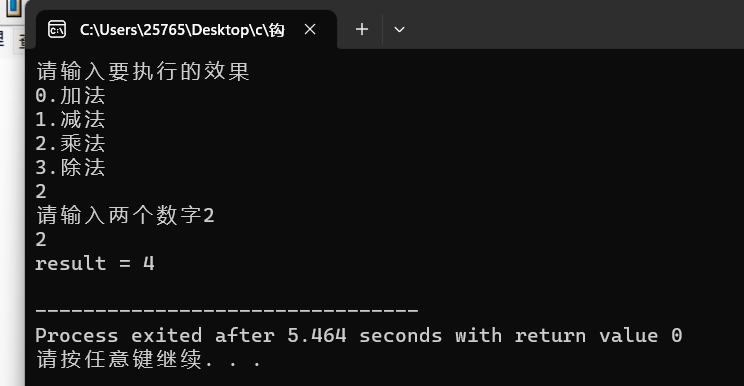
printf("result = %d\n", func(x, y));

return 0;

}

运行效果：





**功能8：**拆分数函数功能分析

·算法描述：

通过自定义函数，定义pl函数（运行结果没有相同元素）和plsame函数（运行结果有相同元素）。

pl函数：运用for语句实现拆分功能。通过表达式，在循环体语句中，调用pl函数本身，实现递归，改变一维数组a[]并且通过指针，间接调用、改变count变量。

plsame函数：在pl函数的基础上，通过在递归调用plsame自身函数时，改变实参，实现含有相同元素组合的运算。

·主要代码：

void pl(int n, int m, int a[], int o, int \*p) {

for (int i = m; n - i > i; i++) {

(\*p)++;

a[o] = i;

a[o + 1] = n - i;

for (int j = 0; j <= o + 1; j++)

printf("%d ", a[j]);

printf("\n");

pl(n - i, i + 1, a, o + 1, p);

}

}

void plsame(int n, int m, int a[], int o, int \*p) {

for (int i = m; n - i >= i; i++) {

(\*p)++;

a[o] = i;

a[o + 1] = n - i;

for (int j = 0; j <= o + 1; j++)

printf("%d ", a[j]);

printf("\n");

plsame(n - i, i, a, o + 1, p);

}

}

·运行效果抓图：

**功能9：**退出函数功能

### 项目总结与展望

1. 总结：

本次项目旨在通过构造组合函数，了解并巩固基本算法概念，并初步接触递归的思想。在项目实践中，学习递归函数调用的原理，以及递归函数的编写方法。通过实现一系列组合函数，对递归进入更加深入的了解，并且掌握了处理常见问题的递归技巧:记忆化搜索和化简函数。

递归算法在计算机科学领域中非常重要，尤其是在算法分析、优化和解决问题中。目前我们还需要进一步加深对递归的理解，研究更多高效的递归算法，以便在实际问题中更好地使用它。

2.展望：

二项式系数函数作是一个广泛应用于组合数学和概率统计等领域的重要函数，需要通过不断完善和扩展功能，使得这个函数更加实用和多样化，满足更多领域的需求。

（1）. 增加更多的数学计算功能：可以考虑添加其他数学计算功能，如计算排列组合的问题、计算数列、解方程等，使得函数更加全面和实用。

（2）. 增加参数范围的限制：对于输入参数的范围进行限制和验证，避免非法输入和计算溢出的问题。

（3）. 增加异常处理和错误提示：在用户界面模块中，可以进一步完善异常处理机制，提供更详细的错误提示和用户引导，增强用户体验。

（4）. 添加更多的交互方式：除了命令行交互，可以考虑采用图形界面或网页界面等更友好的交互方式，提供更直观和便捷的使用体验。

（5）. 进一步优化性能：对于涉及大数计算的功能模块，可以考虑优化算法和计算方式，提高计算速度和效率。