《程序设计进阶与实践》实验报告

姓名	王昱	学号	PB21030814	日期	2022.04.23
实验名称	24 点问题的求解				

实验环境：
CPU：Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz 2.21 GHz
内存：16.0 GB (15.8 GB 可用)
操作系统：Windows 11 家庭中文版
软件平台：Visual Studio Code 1.65.2

一、 问题分析与求解思路

24 点问题一个比较容易的方法就是暴力枚举法，本次实验采用**暴力枚举**的方法来解决。

定义**两个二维数组**，一个存 4 个数的排列组合，另一个存三个运算符的排列组合。

考虑加括号的情况，加括号的方式有五种，故对于 4 个数和 3 个运算符的一种排列组合，一共有**五个表达式**。

通过一个**双层循环**来计算五个表达式的值是否为 24，是则直接输出，不是则继续判定。最终将所有的结果全部输出。

二、核心代码说明

```
int main()
{
    double num[LEN1];
    printf("请输入四个不大于20的正整数:(以空格作为间隔)\n");
    scanf("%lf%lf%lf%lf", &num[0], &num[1], &num[2], &num[3]);
    arrangedNumber(num, number);
    arrangedOperationalCharacter(sign);
    //通过双层循环实现数和运算符的排列组合
    for (int i = 0; i < VALUE; i++)
    {
        for (int j = 0; j < LEN3; j++)
        {
            expression1(number[i], sign[j]);
            expression2(number[i], sign[j]);
            expression3(number[i], sign[j]);
            expression4(number[i], sign[j]);
            expression5(number[i], sign[j]);
        }
    }
    if (!isASolution)
        printf("No Answer!\n");
    else
    {
        printf("The Answer Is Above!\nSome Answers may be repeated!\n");
    }
    system("pause");
    return 0;
}
```

① 主函数：

依次调用 arrangedNumber 和 arrangedOperationalCharacter 函数来求出数和运算符的排列组合，然后通过一个双层循环求出每个表达式的值并判断。

```
void arrangedNumber(double num[], double number[VALUE][LEN1])
{
    int count = 0;
    for (int a = 0; a < LEN1; a++)
    {
        for (int b = 0; b < LEN1; b++)
        {
            if (b == a)
                continue;
            for (int c = 0; c < LEN1; c++)
            {
                if (c == a || c == b)
                    continue;
                int d = 6 - a - b - c; // a+b+c+d=6 (四个下标)
                number[count][0] = num[a];
                number[count][1] = num[b];
                number[count][2] = num[c];
                number[count][3] = num[d];
                count++;
            }
        }
    }
}
```

② arrangedNumber 函数：

图中 a b c d 是数组的下标，满足关系 a+b+c+d=6，通过一个三层循环将 a、b、c、d 进行排列，最后将 num 数组(也就是输入的 4 个数所存放的数组)赋给 number 数组

```
void arrangedOperationalCharacter(char sign[LEN3][LEN2])
{
    int count = 0;
    for (int a = 0; a < LEN1; a++)
    {
        for (int b = 0; b < LEN1; b++)
        {
            for (int c = 0; c < LEN1; c++)
            {
                sign[count][0] = Sign[a];
                sign[count][1] = Sign[b];
                sign[count][2] = Sign[c];
                count++;
            }
        }
    }
}
```

③ arrangedOperationalCharacter 函数：

与 arrangedNumber 函数处理方法一致。

```
//用A、B、C、D表示数，￥表示运算符
// ( (A￥B) ￥C) ￥D
void expression1(double number[], char sign[])
{
    double expression = 0;
    expression = calculate(number[0], number[1], sign[0]);
    expression = calculate(expression, number[2], sign[1]);
    expression = calculate(expression, number[3], sign[2]);
    if (equal(expression, VALUE))
    {
        isASolution = true;
        printf("((%.0lf%c%.0lf)%c%.0lf)%c%.0lf\n", number[0], sign[0], number[1], sign[1], number[2], sign[2], number[3]);
    }
}
```

```
double calculate(double a, double b, char sign)
{
    switch (sign)
    {
    case '+':
        return a + b;
        break;
    case '-':
        return a - b;
        break;
    case '*':
        return a * b;
        break;
    case '/':
        return a / b;
        break;
    }
}
```

```
#define EPSILON 1e-6
#define VALUE 24
#define LEN1 4
#define LEN2 3
#define LEN3 64
//这里使用带参宏equal来表征值为24的情况
#define equal(a, b) fabs(a - b) <= EPSILON
```

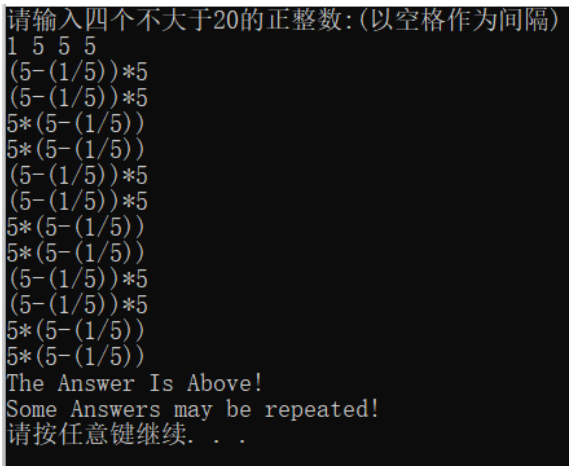
④ expression 函数：

根据（）的位置，判断运算的优先级，再根据优先级的顺序依次算出各个值。最后判断表达式是否为 24。由于 **double 数据类型不能用==来比较是否相等**，故这里采用带参宏定义 equal 取代==来判断否等于 24。

三、测试、运行与分析

```
请输入四个不大于20的正整数:(以空格作为间隔)
3 6 8 11
No Answer!
请按任意键继续. . .
```

```
请输入四个不大于20的正整数:(以空格作为间隔)
3 4 5 6
((3-4)+5)*6
(3-(4-5))*6
((3+5)-4)*6
(3+(5-4))*6
((5+3)-4)*6
(5+(3-4))*6
((5-4)+3)*6
(5-(4-3))*6
6*((3-4)+5)
6*(3-(4-5))
6*((3+5)-4)
6*(3+(5-4))
6*((5+3)-4)
6*(5+(3-4))
6*((5-4)+3)
6*(5-(4-3))
The Answer Is Above!
Some Answers may be repeated!
请按任意键继续. . .
```



运行结果如上图所示，输入时仅需要输入四个待判断的数即可，如果有解则输出全部解(但是没有去重,即显示的解可能出现重复,但是解的个数可以保证正确)；如果没有解，则输出 No Answer！

总结

本次实验相较于前两次实验来说比较简单，求解问题的方法容易想到。通过本次实验，回顾了枚举法在解题中的应用，进一步规范了代码风格（如尽量不出现立即数，尽量使用函数作为一个个模块等）。

本次实验依旧使用的是面向过程的 C 语言，还没有完全掌握面向过程的 python 语言的语法，希望下一次实验能够用面向过程的 python 语言来写。

另附（源代码文件名称）	twenty-four points.c