西北农林科技大学信息工程学院

数据结构实习报告

题　目：算24

学　　号　2020012249

姓　　名　张宇晨

专业班级　软件工程 2001 班

指导教师　冯妍

实践日期　2022 年 7 月 11 日-7 月 22 日

目　录

一、综合训练目的与要求

本综合训练是计算机科学与技术、软件工程、数据科学与大数据技术专业重要的实践性环节之一，是在学生学习完《程序设计语言(C++)》、《数据结构》课程后进行的一次全面的综合练习。本课综合训练的目的和任务：

1. 巩固和加深学生对 C++语言、数据结构课程的基本知识的理解和掌握
2. 掌握 C++语言编程和程序调试的基本技能
3. 利用 C++语言进行基本的软件设计
4. 掌握书写程序设计说明文档的能力
5. 提高运用 C++语言、数据结构解决实际问题的能力

二、综合训练任务

每局4个整数，运用四则运算（可以加括号），但为了降低程序设计的难度，除法的结果只保留整数部分，即5/2=2。给出计算出24的方法。

输入：本题包含多个测例。数据的第一行有一个整数 N（0从第二行开始的 N 行，各包含4个不大于15的以空格分隔的非零整数。

输出：对于每个测例，如果可以计算得到24，则输出“Yes”，否则输出“No”。每个输出占一行。

输入样例：

2

2 2 3 3

2 4 9 10

输出样例：

Yes

Yes

主要功能：

(1)从文件中读出题目的输入；

(2)向屏幕上打印出题目的计算结果；

新增功能：

（1）将每种可能的情况打印出来。

（2）利用栈操作将中缀表达式转换为后缀表达式

（3）利用栈计算后缀表达式的值，验证结果的正确性

（4）统计一共有多少种结果。

三、总体设计

（1） 文件读取

本项目的测试样例存储在文件中，一个文件可以有多组测试样例，程序会如每一组的测试样例并顺寻存放在一维数组 number 中，直到读到程序末尾，如图 1。

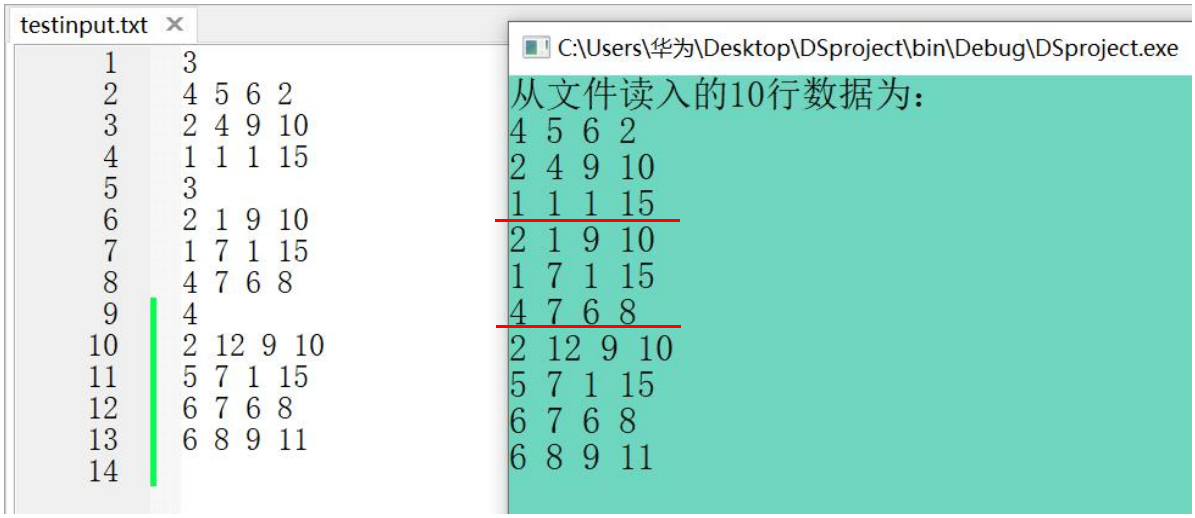

图 1 文件读取

Fig. 1 read from document

（2） 递归求解

项目采用递归回溯法求解，其解空间是二叉树，对每一个结点进行递归回溯操作，搜索满足要求的运算组合，搜索过程如下：

1. 在四个数遍历选两个数;
2. 计算该组合在此运算符下的结果;
3. 将步骤 2 的结果放入原数组第 i 个,将最后一个放入原数组的第 j 个，并对字符串数组赋对应的 表达式;
4. 对新数组递归调用 f，找到一个表达式则返回;
5. 将步骤 2 的结果移除，并将该组合中的两个数重新放回该数组，字符串数组也同理。

搜索流程图如图 2：

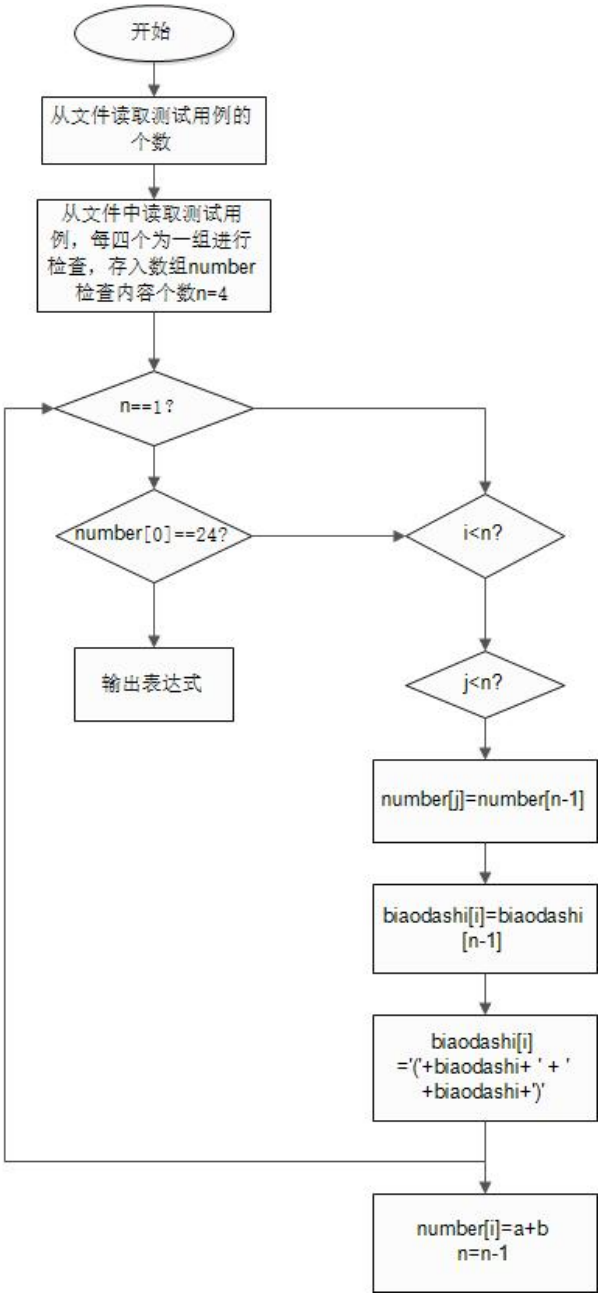

图 2 文件读取

Fig. 2 Search flow chart

（3） 栈的实现

为了提升程序结果的可信度，项目对每一个搜索到的表达式做计算验证。为了便于计算，项目使用自建栈类，类的属性声明如图 3，利用栈将中缀表达式转换为后缀表达式，并计算每一个中缀表达式的后缀表达式的值，以验证搜索结果的正确性。

mystack.h

```
#ifndef MYSTACK_H_INCLUDED
#define MYSTACK_H_INCLUDED

#include<iostream>
#include<assert.h>
using namespace std;
enum { STACK_INIT_SIZE = 10, STACK_INC_SIZE = 2 };
//给定初始容量为10，且按照2倍的增量来扩容
class MyStack
{

private:
    int* _data; //方便扩容
    int _capc;  //容量
    int _top;   //栈顶
public:
    MyStack(int sz = STACK_INIT_SIZE);
    ~MyStack();
    int Size()  const;
    int Capc() const;
    bool Empty() const;
    bool Full() const;
    bool Resize(int newsz);
    bool Push(const int val);
    int Top() const;
    void Pop();
    int GetPop();
};
using namespace std;

#endif

```

图 3 自建栈

Fig. 3 Self built stack

（4） 计算后缀表达式

利用栈，从左（中缀表达式始端）到右扫描，扫描到数字，压栈；扫描到运算符，出栈。出站顺序为从右向左，第一个出栈的数字作为运算符的第二操作数，第二个出栈的作为第一运算数，计算运算结果，将运算结果压栈。重复上述操作，直到栈空，最后一个出栈的数字即为此后缀表达式的运算结果，转换流程如图 4.

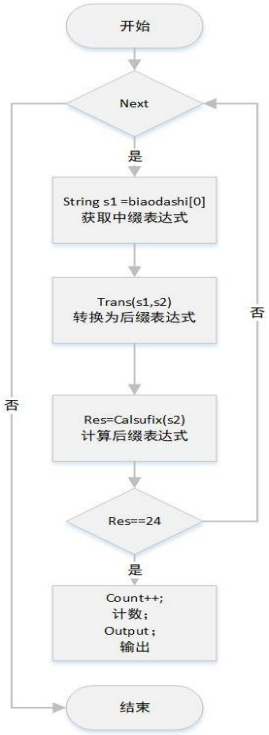

图 4 后缀转换

Fig. 4 Suffix conversion

（5） 图形界面

本项目主要包括云服务器模块和 web 前端模块。云服务器模块包括参数接收、参数计算和结果反馈，web 前端模块负责读入用户需求，并传参给云端服务器，使用 Apache 框架搭建 web 服务器，实现前后端通信，项目信息交互如图 5。

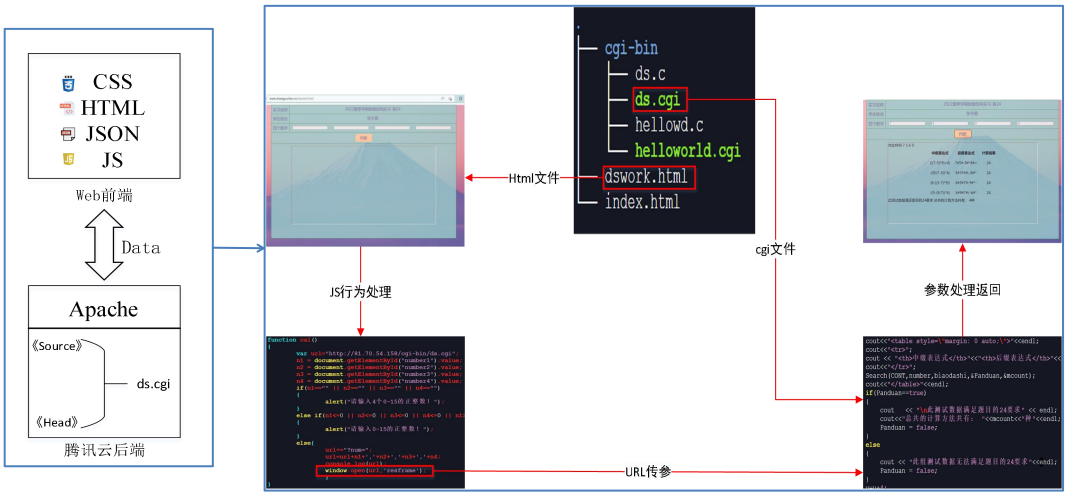

图 5 信息交互图

Fig. 5 Information interaction diagram

四、详细设计说明

（1） 创建流

```
 ifstream input;
//打开文件，将流与文件相关联
input.open("number.txt");
//从文件读入数据
int n;
input >> n;
//读取测试用例个数后循环读入每组测试内容，四个为一组，并存入到数组中
int u=0;
for(int i=0; i<n; i++)
{
    for(int j=u; j<u+4; j++)
    {
        input >> number[j];
    }
    u=u+4;
}
```

（2） 判断输入的数据是否符合要求

```
for(int i=0; i<4*n; i++)
{   if(number[i]>15||number[i]<=0)
    cout<< number[i]<<"数字不符合要求，请输入不大于 15 的以空格分隔的非零整数 ";
}
```

递归搜索

```
 void Search(int n)
{
    if (n == 1)
    {
        if ( number[0] - VALUE == 0    )
        {
            cout << biaodashi[0] << "\t\t";
            Panduan = true;
            count ++;
            if((count % 3)==0) //使输出时每行三个表达式

                cout<<endl;
        }
    }
```

```
//从数组中任意取出两个数的组合
   for(int i=0; i < n; i++)//查找
   {
      for (int j = i + 1; j < n; j++)//与其后面的查找进行计算
      {
         int  a, b;
         string  expa, expb;
         a = number[i];
         b = number[j];
         number[j] = number[n - 1];
//将最后一位赋给空出来的 j
         expa = biaodashi[i];
         expb = biaodashi[j];
         biaodashi[j] = biaodashi[n - 1];
//最后一位的数字放入第 j 个
         biaodashi[i]= &apos;(&apos;+ expa + &apos;+&apos; + expb + &apos;)&apos;;
         number[i] = a + b;
//第一个空间保存前两个的运算结果
         Search(n-1);
         biaodashi[i]=&apos;(&apos;+ expa+ &apos;-&apos; + expb + &apos;)&apos;;
  //减法应该分顺序
         number[i] = a - b;
         Search(n-1);
         biaodashi[i] = &apos;(&apos;+expb + &apos;-&apos; + expa + &apos;)&apos;;
         number[i] = b -a;
         Search(n-1);
         biaodashi[i]= &apos;(&apos;+ expa +&apos;*&apos;+ expb+ &apos;)&apos;;
  //乘法
         number[i]=a*b;
         Search(n-1);
   //除法要分顺序，并且判断分母不为 0
         if (b != 0)
         {
            biaodashi[i] =&apos;(&apos;+expa+&apos;/&apos; + expb + &apos;)&apos;;
            number[i] = a / b;
            Search(n-1);
         }
         if (a != 0)
         {
            biaodashi[i]=&apos;(&apos;+expb + &apos;/&apos;+ expa + &apos;)&apos;;
            number[i] = b / a;
```

```
                Search(n-1);
            }
            //当以上四则运算的结果都不满足条件时
            //为了方便下一个 for 循环，需要将之前的 i 和 j 上的值都重新找回
            number[i] =a;
            number[j]=b;
            biaodashi[i] = expa;
            biaodashi[j] = expb;
        }
    }
}
```

此函数完成对四个数能否计算出 24 的判断。主要用的递归方法，文中标蓝色部分为递归的出口，下面的循环主要是递归体，将两个数字的各种组合结果（包括符号和括号）组合成一个新的字符串，并将结果保存，如果可以递归出结果，就将最终的组合结果的字符串输出并进行统计。

（3）栈的实现

```
MyStack::MyStack(int sz) : _data(nullptr), _capc(sz), _top(-1)  //栈的构造函数，获取资源，并进行初始化
{
    _data=nullptr;
    _capc=sz;
    _top=-1;
    _data = new(nothrow) int[_capc]; //抛出异常判断
    if (nullptr == _data)
    {
        exit(1);
    }

}

MyStack::~MyStack() //栈的析构函数，释放资源
{
    delete[]_data;
    _data = nullptr;//指针置为空
    _capc = 0;
    _top = -1;
}
int MyStack::Size()  const//当前元素的个数
{
    return _top + 1;//返回栈顶元素个数
}
```

```
int MyStack::Capc() const//容量
{
	return _capc;
}
bool MyStack::Empty() const//判空
{
	return Size() == 0;
}
bool MyStack::Full() const//判满
{
	return Size() == Capc();
}
bool MyStack::Resize(int newsz)//扩容操作
{
	if (newsz <= _capc) //如果当前容量小于原来的容量，则不需要扩容
	{
		return true;
	}
	else
	{
		int* newdata = new(nothrow) int[newsz];
		if(nullptr==newdata)
		{
			return false;
		}
		memmove(newdata, _data, sizeof(int) * Size());//将数据 _data 移动到 newdata，一个一个字节拷贝
		delete[]_data; //将原有空间释放
		_data = newdata;
		_capc = newsz;
		return true;
	}
}
bool MyStack::Push(const int val)
{
	if (Full()&&!Resize(Capc()*STACK_INC_SIZE))//如果栈满或者扩容失败时，则返回 false
	{
		return false;
	}
	else
	{
```

```
        _top += 1;
        _data[_top] = val;
        //cout<<"入栈元素"<<val<<endl;
        return true;
    }
}
int MyStack::Top() const //返回栈顶元素
{
    if (!Empty())
    {
        return _data[_top];
    }
}
void MyStack::Pop()   //出栈
{
    _top -= 1;
}
int MyStack::GetPop()//取栈顶元素并出栈
{
    assert(!Empty());
    return _data[_top--];//_先将_top 指向的数据取出，再移动栈顶指针
}
```

（4） 中缀转后缀

```
int prio(char op)                          //给运算符优先级排序
{
    int priority=0;
    if (op == '*' || op == '/')
        priority = 2;
    if (op == '+' || op == '-')
        priority = 1;
    if (op == '(')
        priority = 0;
    return priority;
}

bool Trans(string& str1, string& str2)       //引用传递
{
    MyStack s;
    s.Resize(100);                         //定义一个 char 类型的栈 s
```

```
int i;
for (i = 0; i < str1.size(); i++)
{
    if (str1[i] >= '0' && str1[i] <= '9')        //如果是数字，直接入栈
    {
        while(str1[i] >= '0' && str1[i] <= '9')
        {
            str2 += str1[i++];
        }
        str2+='#';
        i--;
    }
    else      //否则不是数字
    {

        if (s.Empty())                  //栈空则入站
            s.Push(str1[i]);
        else if (str1[i] == '(')    //左括号入栈
            s.Push(str1[i]);
        else if (str1[i] == ')')     //如果是右括号，只要栈顶不是左括号，就弹出并输出
        {
            while (s.Top() != '(')
            {
                str2 += s.Top();
                s.Pop();
            }
            s.Pop();                        //弹出左括号，但不输出
        }
        else            //当栈顶非空时
        {

            while (prio(str1[i]) <= prio(s.Top()))    //当栈顶优先级大于等于当前运算符，输出
            {

                str2 += s.Top();
                s.Pop();
                if (s.Empty())        //栈为空，停止
                    break;
            }
            s.Push(str1[i]);    //把当前运算符入栈
        }
```

```
        }
    }
    while (!s.Empty())          //最后，如果栈不空，则弹出所有元素并输出
    {
        str2 += s.Top();
        s.Pop();
    }
    return true;
}

/*
```

（5）计算后缀表达式

```
int calsufix(string tokens)
{
    MyStack numbers;
    numbers.Resize(128);

    for(int i = 0 ; i < tokens.size() ; i++)
    {
        //若为运算符，则弹出两个栈顶元素，进行运算并将结果放回栈
        if(tokens[i] == '+' || tokens[i] == '-' || tokens[i] == '*' || tokens[i] == '/')
        {
            int res;
            int n2 = (int)numbers.Top();
            numbers.Pop();
            int n1 = (int)numbers.Top();
            numbers.Pop();

            if(tokens[i] == '+')
                res = n1 + n2;
            else if(tokens[i] == '-')
                res = n1 - n2;
            else if(tokens[i] == '/')
                res = n1 / n2;
            else
                res = n1 * n2;
            numbers.Push(res);
        }
        else if(tokens[i] == '#')
```

```
            {
                continue;
            }
            else
            {
                int temp = 0;
                while(tokens[i] >= '0' && tokens[i] <= '9')
                {
                    temp = temp *10 + int(tokens[i] - '0');
                    i++;
                }
                numbers.Push(temp);
            }
        }
        return numbers.Top();
}
```

(6) Cgi 程序实现前后端通信

```
int main()
{
    int t=0;
    int numt=0;
    int tcount =0;
    //char *s_input = "14,11,7,9";
    char *ss=getenv("QUERY_STRING");
                    //读取环境变量 QUERY_STRING 的值，获得前端传过来的网页参数
    char s_input[1024] = {'\0'};
    cout<<"Content-Type: text/html;charset=utf-8\n\n";
    sscanf(ss,"num=%s",s_input);
    //cout<<"获得的环境变量参数为"<<s_input<<endl;
    string temp="";
    if(s_input!=NULL)
   {
    for(int i=0;; ++i)
    {

//          cout<<"i="<<i<<"s_input[i]="<<s_input[i]<<endl;
            if(s_input[i]=='\0') break;
            if(s_input[i]==',')
            {
```

```
            //number[t++]=numt;
            tcount=0;
            numt=0;
            continue;
        }
        else
        {
            int temp = 0;
            while(s_input[i] >= '0' && s_input[i] <= '9')
            {
                temp = temp *10 + int(s_input[i] - '0');
                i++;
            }
                                        //字符串转数字
            number[t++]=temp;
           // cout<<"temp 加入 number，temp="<<temp<<endl;
        }
    }
    int u=0;
    int n=1;
                                        //开始写网页
cout<<"<html>"<<endl;
cout<<"<head>"<<endl;
cout<<"<style>";
cout<<"th,td{padding: 10px;text-align:center;}";
cout<<"</style>";
cout<<"</head>"<<endl;
cout<<"<body style=\"margin: 0 auto;\">"<<endl;
    for(int i=0; i<n; i++)
    {
        int k=u;
        number[0]=number[k];
        number[1]=number[k+1];
        number[2]=number[k+2];
        number[3]=number[k+3];
        cout<<endl<<"对此样例\t";
        cout<< number[0]<<" ";
        cout<< number[1]<<" ";
        cout<< number[2]<<" ";
        cout<< number[3]<<" ";
        cout<<endl;
```

```
            mcount=0;
            for (int i = 0; i < CONT; i++)
            {
                char ch[20];
                //cout<<number[i]<<"    ";
               // itoa(number[i],ch, 10);
                snprintf(ch,sizeof(ch),"%d",number[i]);
                biaodashi[i] = ch;
            }
            cout<<"<table style=\"margin: 0 auto;\">"<<endl;
            cout<<"<tr>";
            cout << "<th>中缀表达式</th>"<<"<th>后缀表达式</th>"<<"<th>计算结果</th>";
            cout<<"</tr>";
            Search(CONT,number,biaodashi,&Panduan,&mcount);
            cout<<"</table>"<<endl;
            if(Panduan==true)
            {
                cout     << "\n 此测试数据满足题目的 24 要求" << endl;
                cout<<"总共的计算方法共有：  "<<mcount<<"种"<<endl;
                Panduan = false;
            }
            else
            {
                cout << "此组测试数据无法满足题目的 24 要求"     <<     endl;
                Panduan = false;
            }
            u=u+4;
        }
      }
        cout<<"</body>"<<endl;
        cout<<"</html>"<<endl;
        return 0;
}
```

五、调试与测试

（1）、控制台操作流程

1. 将测试数据写入文件 input.txt，可写入多行测试数据，程序可以读入多组测试数据中的多行数据，如图 6 测试输入

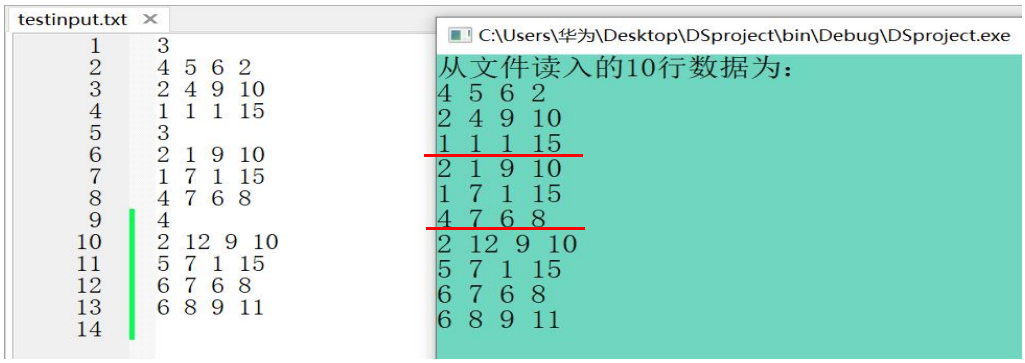

图 6 测试输入

Fig. 6 Test input

2. 输出，输出满足要求的中缀表达式和后缀表达式，并输出后缀表达式的计算结果。为了便于分割多位数，在后缀表达式中以‘#’作为数字分割符，输出情况如图 7 测试输出。

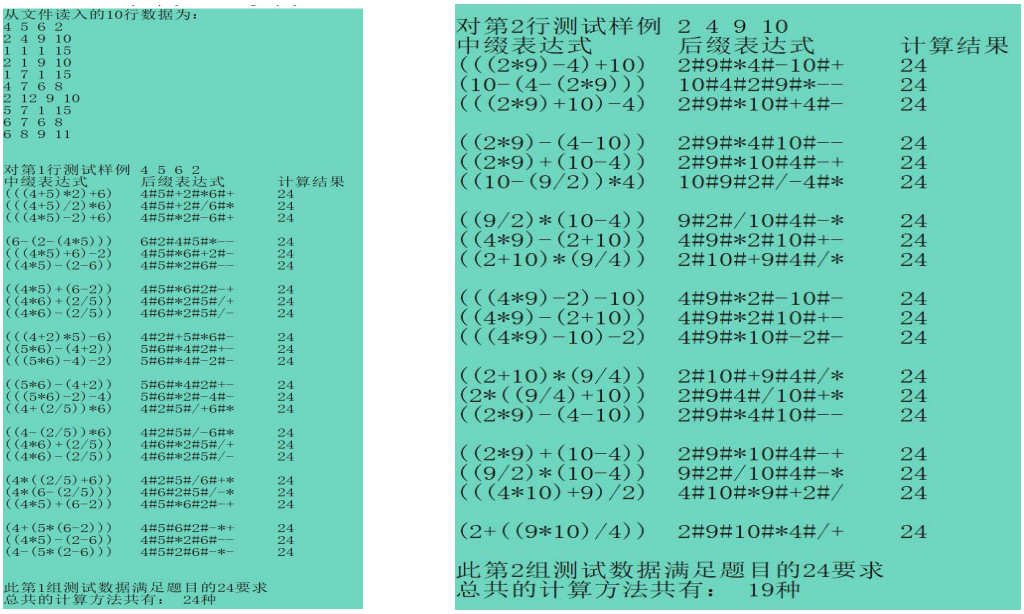

图 7 测试输出

Fig. 7 Test output

（2）、网页端操作流程

网页操作相对简单，只需访问 www.zhangyuchen.cn/dswork.html，然后在界面中四个框中输入测试数据，然后点击按钮，即可在输出框输出次测试样例的结果，如图 8 网页测试。

图 8 网页测试
Fig. 8 Test web

六、实习日志

7 月 11 日 星期一
今天主要是对题目的熟悉，确定自己的选题，了解题目的具体要求，确立解题思路，并编写实施计划书。
7 月 12 日 星期二
查阅相关资料，彻底深入理解问题，确定好解决问题的算法，进行结局问题的初步尝试。
7 月 13 日 星期三
今天主要完成了对文件的操作的代码的编写，如何读取文件内容，并将文件中获取的内容可以进行相应的操作。
7 月 14 日 星期四
今天准备实现判断的功能，之前确定了递归的方法，但是递归体的编写还未完成，正在完成中。
7 月 15 日 星期五
今天将昨天的递归体进行了完善，有了初步的结果，但是判断条件出了问题，有的测试用例是正确的但没有通过，正在修改中。

7 月 18 日 星期一

今天主要任务还是继续完善代码功能，已经可以判断出四个数的结果能否产生 24，但是新增功能还在调试中。

7 月 19 日 星期二

今天主要任务是将结果为 24 的表达式输出出来，但是由于字符串数组重复调用，出现了重复的情况，所以正在调试修改中。

7 月 20 日 星期三

基本实现了测试的所有功能，进一步测试检查，开始实习报告的撰写。

7 月 21 日 星期四

继续完成对论文的撰写，并制作答辩需要的 PPT。

7 月 22 日 星期五

完成整个实习的答辩，根据老师提出的意见进行修改，同时将自己的实习报告修改更加完善。

七、实习总结

（1） 总结

通过本次数据结构实习，不仅对之前所学进行了巩固复习，更是一个非常好的机会能够提升自己的能力，如何熟练的将所学知识知识运用到实践当中，尤其加深了对递归法和文件操作的理解，而且再碰到问题时，基本上都可以独立的解决。但学无止境，所以仍要继续努力前行。

（2） 心得

无论做什么都要脚踏实地，学什么东西不能只浮在表面，而需要不断地尝试动手，真正沉下去，一步一个脚印，不能光靠脑子想，俗话说得好，眼过千遍不如手过一遍。其次，遇到问题一定要及时解决，不能拖拖拉拉，多和老师同学交流，这样才能深入理解问题并解决问题，从而提高自己。最重要的一点是：一定要听老师的话，明确老师的任务要求，避免南辕北辙的情况，或者强调的东西，转眼就忘掉，只有紧跟老师的步伐，才能将自己的工作做的更加漂亮。

（3） 致谢

衷心感谢实习过程中冯妍老师的悉心指导与帮助！

八、附录：核心代码清单

（1） 栈的实现

头文件：

```
#ifndef MYSTACK_H_INCLUDED
#define MYSTACK_H_INCLUDED

#include<iostream>
#include<assert.h>
using namespace std;
enum { STACK_INIT_SIZE = 10, STACK_INC_SIZE = 2 };
//给定初始容量为 10，且按照 2 倍的增量来扩容
class MyStack
{

private:
    int* _data; //方便扩容
    int _capc;   //容量
    int _top;    //栈顶
public:
    MyStack(int sz = STACK_INIT_SIZE);
    ~MyStack();
    int Size()   const;
    int Capc() const;
    bool Empty() const;
    bool Full() const;
    bool Resize(int newsz);
    bool Push(const int val);
    int Top() const;
    void Pop();
    int GetPop();
};
using namespace std;

#endif
```

Cpp 文件：

```
#include"mystack.h"

MyStack::MyStack(int sz) : _data(nullptr), _capc(sz), _top(-1)  //栈的构造函数，获取资源，并进行初始化
{
    _data=nullptr;
    _capc=sz;
    _top=-1;
    _data = new(nothrow) int[_capc]; //抛出异常判断
    if (nullptr == _data)
    {
        exit(1);
    }

}

MyStack::~MyStack() //栈的析构函数，释放资源
{
    delete[]_data;
    _data = nullptr;//指针置为空
    _capc = 0;
    _top = -1;
}
int MyStack::Size()  const//当前元素的个数
{
    return _top + 1;//返回栈顶元素个数
}
int MyStack::Capc() const//容量
{
    return _capc;
}
bool MyStack::Empty() const//判空
{
    return Size() == 0;
}
bool MyStack::Full() const//判满
{
    return Size() == Capc();
}
bool MyStack::Resize(int newsz)//扩容操作
```

```
    {
        if (newsz <= _capc) //如果当前容量小于原来的容量，则不需要扩容
        {
            return true;
        }
        else
        {
            int* newdata = new(nothrow) int[newsz];
            if(nullptr==newdata)
            {
                return false;
            }
            memmove(newdata, _data, sizeof(int) * Size());//将数据 _data 移动到 newdata，一个一个字节拷贝
            delete[]_data; //将原有空间释放
            _data = newdata;
            _capc = newsz;
            return true;
        }
    }
    bool MyStack::Push(const int val)
    {
        if (Full()&&!Resize(Capc()*STACK_INC_SIZE))//如果栈满或者扩容失败时，则返回false
        {
            return false;
        }
        else
        {
            _top += 1;
            _data[_top] = val;
            //cout<<"入栈元素"<<val<<endl;
            return true;
        }
    }
    int MyStack::Top() const //返回栈顶元素
    {
        if (!Empty())
        {
            return _data[_top];
        }
```

```
}
void MyStack::Pop()   //出栈
{
    _top -= 1;
}
int MyStack::GetPop()//取栈顶元素并出栈
{
    assert(!Empty());
    return _data[_top--];//_先将_top 指向的数据取出，再移动栈顶指针
}


/* 测试栈的功能
int main()
{
MyStack mys;
mys.Resize(10000);//扩容栈到 1 万
for (int i = 0; i < 10000; i++)
{
    mys.Push(i); //入栈
}
int x;
while (!mys.Empty())
{
    x = mys.GetPop();
    cout << x << endl;
}
return 0;
}
*/
```

（2） 中缀转后缀

头文件：

```
#ifndef INFIXTOPOSTFIX_H_INCLUDED
#define INFIXTOPOSTFIX_H_INCLUDED
#include<iostream>
#include<string>
#include"mystack.h"
int prio(char);
bool Trans(string&, string&);
```

```
#endif // INFIXTOPOSTFIX_H_INCLUDED
```

Cpp 文件：

```
#include"InfixToPostfix.h"
int prio(char op)                          //给运算符优先级排序
{
    int priority=0;
    if (op == '*' || op == '/')
        priority = 2;
    if (op == '+' || op == '-')
        priority = 1;
    if (op == '(')
        priority = 0;
    return priority;
}

bool Trans(string& str1, string& str2)      //引用传递
{
    MyStack s;
    s.Resize(100);                          //定义一个 char 类型的栈 s
    int i;
    for (i = 0; i < str1.size(); i++)
    {
        if (str1[i] >= '0' && str1[i] <= '9')       //如果是数字，直接入栈
        {
            while(str1[i] >= '0' && str1[i] <= '9')
            {
                str2 += str1[i++];
            }
            str2+='#';
            i--;
        }
        else      //否则不是数字
        {

            if (s.Empty())                  //栈空则入站
                s.Push(str1[i]);
            else if (str1[i] == '(')    //左括号入栈
                s.Push(str1[i]);
            else if (str1[i] == ')')     //如果是右括号，只要栈顶不是左括号，就弹出并输
```

```
出
                {
                    while (s.Top() != '(')
                    {
                        str2 += s.Top();
                        s.Pop();
                    }
                    s.Pop();                        //弹出左括号，但不输出
                }
                else            //当栈顶非空时
                {

                    while (prio(str1[i]) <= prio(s.Top()))    //当栈顶优先级大于等于当前运算
符，输出
                    {

                        str2 += s.Top();
                        s.Pop();
                        if (s.Empty())          //栈为空，停止
                            break;
                    }
                    s.Push(str1[i]);    //把当前运算符入栈
                }
            }
        }
        while (!s.Empty())          //最后，如果栈不空，则弹出所有元素并输出
        {
            str2 += s.Top();
            s.Pop();
        }
        return true;
    }

    /*

    int main()                      //测试一下
    {
        string infix;   //保存输入的中缀表达式
        string postfix;
        cout << "请输入中缀表达式：" << infix << endl;
        cin >> infix;
```

```
    Trans(infix, postfix);
    cout << "后缀表达式为：" << postfix << endl;
    return 0;
}
*/
```

（3） 后缀表达式计算

头文件：

```
#ifndef CALPOSTFIX_H_INCLUDED
    #define CALPOSTFIX_H_INCLUDED
    #include<stdio.h>
    #include<stdlib.h>
    #include<string>
    #include"mystack.h"
    using namespace std;
    int calsufix(string);
    #endif // CALPOSTFIX_H_INCLUDED
```

Cpp 文件：

```
    #include"calpostfix.h"
    int calsufix(string tokens)
    {
        MyStack numbers;
        numbers.Resize(128);

        for(int i = 0 ; i < tokens.size() ; i++)
        {
            //若为运算符，则弹出两个栈顶元素，进行运算并将结果放回栈
            if(tokens[i] == '+' || tokens[i] == '-' || tokens[i] == '*' || tokens[i] == '/')
            {
                int res;
                int n2 = (int)numbers.Top();
                numbers.Pop();
                int n1 = (int)numbers.Top();
                numbers.Pop();

                if(tokens[i] == '+')
                    res = n1 + n2;
                else if(tokens[i] == '-')
                    res = n1 - n2;
                else if(tokens[i] == '/')
```

```
                res = n1 / n2;
            else
                res = n1 * n2;
            numbers.Push(res);
        }
        else if(tokens[i] == '#')
        {
            continue;
        }
        else
        {
            int temp = 0;
            while(tokens[i] >= '0' && tokens[i] <= '9')
            {
                temp = temp *10 + int(tokens[i] - '0');
                i++;
            }
            numbers.Push(temp);
        }
    }
    return numbers.Top();
}

/*
int main()//测试一下
{
    int x = calsufix("3639/+*");
    cout<<"x 是："<<x<<endl;
}
*/
```

（4） 递归查找实现

头文件：

```
#ifndef SEARCH_H_INCLUDED
#define SEARCH_H_INCLUDED
using namespace std;
#include"G24.h"
#include"yanzheng.h"
void Search(int,int *r,string*,bool*,int*);
#endif
```

Cpp 文件：

```
#include"search.h"
void Search(int n,int * number,string* biaodashi,bool* panduan,int* mcount)
{
    if (n == 1)
    {
        if ( number[0] - VALUE == 0     )//可以算出 24,输出表达式
        {

            cout << biaodashi[0] << "\t";
            yanzheng(biaodashi[0]);//中缀转后缀并计算验证
            *panduan = true;
            (*mcount)++;
            if(((*mcount) % 3)==0)

                cout<<endl;
        }
    }

    for(int i=0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            int    a, b;
            string    expa, expb;
            a = number[i];
            b = number[j];
            number[j] = number[n - 1];
            expa = biaodashi[i];
            expb = biaodashi[j];
            biaodashi[j] = biaodashi[n - 1];

            //+
            biaodashi[i]= '('+ expa + '+' + expb + ')';
            number[i] = a + b;
            Search(n-1,number,biaodashi,panduan,mcount);
            //-
            biaodashi[i]='('+ expa+ '-' + expb + ')';
            number[i] = a - b;
            Search(n-1,number,biaodashi,panduan,mcount);
            //*
```

```
                biaodashi[i] = '('+expb + '-' + expa + ')';
                number[i] = b -a;
                Search(n-1,number,biaodashi,panduan,mcount);
                //除法
                biaodashi[i]= '('+ expa +'*'+ expb+ ')';
                number[i]=a*b;
                Search(n-1,number,biaodashi,panduan,mcount);
                    //分情况判断
                if (b != 0)
                {
                    biaodashi[i] ='('+expa+'/' + expb + ')';
                    number[i] = a / b;
                    Search(n-1,number,biaodashi,panduan,mcount);
                }
                if (a != 0)
                {
                    biaodashi[i]='('+expb + '/'+ expa + ')';
                    number[i] = b / a;
                    Search(n-1,number,biaodashi,panduan,mcount);
                }

                number[i] =a;
                number[j]=b;
                biaodashi[i] = expa;
                biaodashi[j] = expb;
            }
        }
    }
```

（5） 验证

头文件：

```
#ifndef YANZHENG_H_INCLUDED
#define YANZHENG_H_INCLUDED
#include"calpostfix.h"
#include"InfixToPostfix.h"
#include<stdio.h>
#include<string>
using namespace std;
void yanzheng(string);
#endif // YANZHENG_H_INCLUDED
```

Cpp 文件：

```
#include"yanzheng.h"
void yanzheng(string infix)
{
    string sufix="";
    Trans(infix, sufix);
    int res = calsufix(sufix);
    cout<<sufix<<"\t"<<res<<"\t\n";
}
```

（6） 主函数

头文件：

```
#ifndef MAIN_H_INCLUDED
#define MAIN_H_INCLUDED
#include <string>
#include <math.h>
#include<cstring>
#include <fstream>
#include<iostream>
#include"mystack.h"
using namespace std;
const int CONT = 4;
const int VALUE = 24;
void Search(int n,int*,string *,bool*,int*);
int number[4096];
string biaodashi[20];
bool Panduan = false;
int mcount = 0;
#endif // MAIN_H_INCLUDED
```

Cpp 文件：

```
#include"main.h"
int main()
{
    //1. 创建流
    ifstream input;
    //2. 打开文件，将流与文件相关联
    input.open("testinput.txt");
    //3. 从文件读入数据
```

```
int u=0;
int n=0;
int t=0;
while(1)
{
    if(input.peek()==EOF) break;
    else
    {
        t=0;
        input >> t;
        //cout<<"t"<<t<<endl;
        n+=t;
        for(int i=0; i<t; i++)
        {
            for(int j=u; j<u+4; j++)
            {
                input >> number[j];
            }
            u=u+4;
        }
    }
}

u=0;
cout<<"从文件读入的"<<n<<"行数据为："<<endl;
for(int i=0; i<n; i++)
{
    for(int j=u; j<u+4; j++)
    {
        cout<< number[j]<<" ";
    }
    u=u+4;
    cout<<endl;
}
for(int i=0; i<4*n; i++)
{
    if(number[i]>15||number[i]<=0)
        cout<< number[i]<<"数字不符合要求，请输入不大于 15 的以空格分隔的非零
整数 ";
}
cout<<endl;
```

```
u=0;
for(int i=0; i<n; i++)
{
    int k=u;
    number[0]=number[k];
    number[1]=number[k+1];
    number[2]=number[k+2];
    number[3]=number[k+3];
    cout<<endl<<"对第"<<i+1<<"行测试样例\t";
    cout<< number[0]<<" ";
    cout<< number[1]<<" ";
    cout<< number[2]<<" ";
    cout<< number[3]<<" ";
    cout<<endl;
    mcount=0;
    for (int i = 0; i < CONT; i++)
    {
        char ch[20];
        //cout<<number[i]<<"    ";
        itoa(number[i],ch, 10);

        biaodashi[i] = ch;
    }

    cout << "中缀表达式"<< "\t"<<"后缀表达式\t"<<"计算结果\t\n";
    Search(CONT,number,biaodashi,&Panduan,&mcount);
    if(Panduan==true)
    {
        cout    << "\n 此第"<<i+1<<"组测试数据满足题目的 24 要求" << endl;
        cout<<"总共的计算方法共有：  "<<mcount<<"种"<<endl;
        Panduan = false;
    }
    else
    {
        cout << "此第"<<i+1<<"组测试数据无法满足题目的 24 要求"    <<    endl;
        Panduan = false;
    }
    u=u+4;
}
input.close();
printf("计算完毕！ ");
```

```
    return 0;
}
```

（7）网页源码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link href="css/4-2.css" rel="stylesheet" type="text/css" />
        <title>实习 4-2</title>
        <style>
        body{
            display: flex;
            flex-direction: column;
            justify-content: center;
        }
        #main{
        /*    margin: auto; */
        /*    display: flex;
            flex-direction: column; */
        /*    width: 100%;
            height: 100%; */
            color: #666699;
            font-size: large;

        }
        #charnot,#charuse,#code{
            display: flex;
            flex-direction: row;
            align-items: center;
        }
        #tres{
            text-align: center;
        }
        button{
            background-color: skyblue;
            justify-content: center;
            margin: auto;
        }
        input[type=text]{
```

```
        color: dimgrey;
        height: 10px;
        opacity: 0.8;
}

#d1,#d2{
        display: flex;
        flex-direction: row;
        /* padding: 8px; */
        border-bottom: darkgreen 1.1px solid;
}
.check{
        background-color: #99cccc;
        padding: 8px;
        border-right: darkgreen 1.1px solid;
}

.check0{
        /* background-color: #C0C0C0; */
        padding: 8px;
}
#wei{
        padding-right: 80px;
}
#d3{
        display: flex;
        flex-direction: row;
        /* padding: 8px; */
        align-items: stretch;
        /* border: mediumseagreen 1px solid; */
        border-bottom: darkgreen 1.1px solid;
}
#button{
        display: flex;
        flex-direction: row;
        text-align: center;
        justify-content: center;
        padding: 7px;
        border-bottom: darkgreen 1.1px solid;
}
form{
```

```
        border: darkgreen 1.1px solid;
}
#tbutton{
        border: white 0px solid;
        height: 35px;
        color: #666699;
        width: 80px;
        /* font-weight: 700; */
        background-color: #ffcc99;
        border-radius: 8%;
        font-size: large;
        align-items: center;
}
.labl{
        text-align: center;
          background-color: silver;
}
#res{
        justify-content: center;
        display: flex;
        flex-direction: row;
        padding: 10px;
}
.labl_r{
        display: flex;
        flex-direction: row;
        justify-content: flex-start;
        padding:5px 18px;
        text-align: center;
}
th,td{
        padding: 0;
        height: 1px;
        text-align: right;
}
table{
        padding: 0;
}
#ex{
        width: 66px;
        margin: 0 ;
```

```
                border-left: darkgreen 1.1px solid;
                border-right: darkgreen 1.1px solid;
            }
            #tex{
                margin: 0;
                width: 66px;
            }
            #number{
                display: flex;
                flex-direction: row;
                align-items: center;
                justify-content: center;
            }
    .numberin{
        height: 10000px;
        margin: 5px 15px;
    }
            </style>
        </head>

        <body>
            <div id="main">
                <div                                                    style="background-image:
url(https://s2.loli.net/2022/07/20/45zIsUHfCRbSatY.jpg);
                position: absolute;
                z-index:        -1;height:        100%;width:        100%;background-repeat:
no-repeat;background-size: cover;filter: blur(1px);">
                </div>
                <form style="margin: auto; width: 900px; height: 600px;background-color:
#99cccc;opacity: 0.8;">
                    <div id="d1">
                        <div class="check">
                            实习名称
                        </div>
                        <div class="labl_r" style="margin: 0 auto;">
                            2022 夏季学期数据结构实习-算 24
                        </div>
                    </div>
                    <div id="d2">
                        <div class="check">学生姓名</div>
                        <div class="labl_r" style="margin: 0 auto;">
```

```
                张宇晨
                </div>
            </div>
            <div id="d3">
                <div class="check" id="len">四个数字</div>
                <div id="number">
                    <input type="text" class="numberin" id="number1">
                    <input type="text" class="numberin" id="number2">
                    <input type="text" class="numberin" id="number3">
                    <input type="text" class="numberin" id="number4">
                </div>


            </div>
            <div id="button">
                    <input id="tbutton" type="button" onclick="cal()" value="判断
">
                </div>

                <div id="res">
                        <iframe      height="350"   width="700"   title="结果"
name="resframe"></iframe>
                </div>
        </form>
        </div>
        <script>
        function cal()
        {
            var url="http://81.70.54.158/cgi-bin/ds.cgi";
            n1 = document.getElementById("number1").value;
            n2 = document.getElementById("number2").value;
            n3 = document.getElementById("number3").value;
            n4 = document.getElementById("number4").value;
            if(n1=="" || n2=="" || n3=="" || n4=="")
            {
                alert("请输入 4 个 0-15 的正整数！");
            }
            else if(n1<=0 || n2<=0 || n3<=0 || n4<=0 || n1>15 || n2>15 || n3>15 || n4>15)
            {
                alert("请输入 0-15 的正整数！");
            }
```

```
                else{
                    url+="?num=";
                    url=url+n1+','+n2+','+n3+','+n4;
                    console.log(url);
                    window.open(url,'resframe');
                    }
            }
            </script>
        </body>
</html>
```