专业：微电子科学与技术
姓名：焦天晟
学号：3220105664
日期：2023.05.24

浙江大学实验报告

课程名称：C 程序设计专题　　指导老师：翁恺　　成绩：

实验名称：作业 3：　表达式计算

一、实验题目要求：

Write a program that reads an expression in a line as input and prints out the result. Only integers and operators below are allowed in the expression:

+ - * / % ( )

输入格式:

A line of expression.

输出格式:

The result.

二、实验思路

这是一个利用递归实现+-*/%()运算的程序，首先由于涉及到运算与数字的储存，所以考虑用链表来实现

```
//定义链表结点结构体
typedef struct node {
    char type; //类型，'N'表示数字，'O'表示操作符
    double value; //值，如果是数字则存储数字的值，如果是操作符则存储操作符的ASCII码
    struct node *next; //指向下一个结点的指针
} node;
```

```
//定义链表结构体
typedef struct list {
    node *head; //指向链表头结点的指针
    node *tail; //指向链表尾结点的指针
} list;
```

在链表的新建，插入与删除运用如下函数：

```
//创建一个空链表
list *create_list() {
    list *l = (list *)malloc(sizeof(list));
    l->head = NULL;
    l->tail = NULL;
    return l;
}

//在链表尾部插入一个结点
void append(list *l, char type, double value) {
    node *n = (node *)malloc(sizeof(node));
    n->type = type;
    n->value = value;
    n->next = NULL;
    if (l->head == NULL) { //如果链表为空，则头尾都指向新结点
        l->head = n;
        l->tail = n;
    } else { //否则，尾结点的 next 指向新结点，然后更新尾结点为新结点
        l->tail->next = n;
        l->tail = n;
    }
}

//从链表头部删除一个结点，并返回它的值
double pop(list *l) {
    if (l->head == NULL) { //如果链表为空，则返回 0
        return 0;
```

```
    } else { //否则，保存头结点的值和类型，然后更新头结点为下一个结点，并释放原头结点的内存
        double value = l->head->value;
        char type = l->head->type;
        node *temp = l->head;
        l->head = l->head->next;
        free(temp);
        if (type == 'N') { //如果是数字，则直接返回值
            return value;
        } else { //如果是操作符，则返回负的 ASCII 码，以便于区分
            return -value;
        }
    }
}
```

由于本次实验还要求识别空格等，所以在 read a line 中做如下考虑：

```
list *read_line() {

    list *l = create_list();

    char c;

    while ((c=getchar()) != '\n' && c != EOF) {

        if (isspace(c)) continue;

        if (isdigit(c) || (c == '-' && l->tail == NULL) || (c == '-' && l->tail->type == 'O' && l->tail->value != ')')) {
            //如果是数字，或者是开头的负号，或者是括号或操作符后面的负号，则视为一个数字的开始

            double num=0;
            int sign = 1; //记录正负号
```

```
            if (c == '-') { //如果是负号，则将符号设为-1，并读取下一个字符
                sign = -1;
                c = getchar();
            }

            while(isdigit(c)) { //读取数字的整数部分
                num=num*10+(c-'0');
                c=getchar();
            }

            if(c=='.') { //如果有小数点，则读取小数部分
                c=getchar();
                double frac=1;
                while(isdigit(c)) {
                    frac/=10;
                    num+=frac*(c-'0');
                    c=getchar();
                }

            }

            ungetc(c,stdin); //将最后一个非数字字符放回输入流中

            append(l,'N',sign*num); //将数字乘以符号后插入链表

        } else {

            append(l,'O',(double)c); //将操作符插入链表

        }

    }
```

```
    return l;

}
```

利用 factor 函数来处理文字与符号：

```
//处理括号和数字
double factor() {
    double result;
    if (current == NULL) {
        error = 1;
        return 0;
    }
    char type = current->type;
    if (type == 'N') {
        result = current->value;
        current = current->next;
    } else if (type == 'O') {
        char op = (char)current->value;
        if (op == '(') {
            current = current->next;
            result = expr();
            if (current == NULL || current->type != 'O' ||
current->value != ')') {
                error = 1;
                return 0;
            } else {
                current = current->next;
            }
        } else if (op == '%') {
            current = current->next;
            double right = factor();
            result = fmod(result, right);
        } else {
            error = 1;
```

```
            return 0;
        }
    } else {
        error = 1;
        return 0;
    }
    return result;
}
```

利用两个函数来处理运算符：

```
//处理加减法运算
double expr() {
    double left = term(); //获取左操作数，即一个乘除法运算的结果
    while (current != NULL && !error) { //循环处理右操作数和操作符，直到链表结束或出错
        char op = (char)current->value; //获取当前操作符
        if (op == '+' || op == '-') { //如果是加号或减号，则继续处理
            current = current->next; //移动到下一个结点
            double right = term(); //获取右操作数，即一个乘除法运算的结果
            if (op == '+') { //根据操作符进行相应的计算，并更新左操作数的值
                left += right;
            } else {
                left -= right;
            }
        } else { //如果不是加号或减号，则退出循环
            break;
        }
    }
    return left; //返回最终的结果
}

//处理乘除法运算
double term() {
```

```
    double result = factor();
    if (error) {
        return 0;
    }
    while (current != NULL && current->type == 'O') {
        char op = (char)current->value;
        if (op == '*' || op == '/' || op == '%') {
            current = current->next;
            double right = factor();
            if (error) {
                return 0;
            }
            if (op == '*') {
                result *= right;
            } else if (op == '/') {
                result /= right;
            } else if (op == '%') {
                result = fmod(result, right);
            }
        } else {
            break;
        }
    }
    return result;
}
```

最后用 main 函数来处理：

```
int main() {

    printf("请输入一个中缀表达式（支持+ - / 和 () 运算符），以回车结束：
\n");

    list *l=read_line();
```

```
    current=l->head;

    error=0;

    double result=expr();

    if (error || current != NULL) {
        printf("错误：无效的表达式\n");
    } else {
        printf("结果：%f\n", result);
    }
    free_list(l); //释放链表占用的内存
    return 0;
}
```

四、实验体会与心得：

这次大程用递归来完成一个简单运算，且由于一开始没好好审阅题目写成了可以实现小数运算的程序，在提交 PTA 的程序中用了（int）处理。没有太过复杂的地方，主要就是一点一点码出来，可能就是对链表的理解要求稍微多一点。