C 语言课程设计

多功能计算器 A 难度

课程设计报告

072040127 樊卓铭

1. 程序功能及限制

1.1 程序功能

该多功能计算器能进行只包含加减乘除和括号的运算式的计算，并且能区分符号与减号，忽略空格。在检测到表达式有误时能输出报错信息。程序可循环执行，程序界面如下：

图 1 程序运行的 cmd 窗口

输入完整运算表达式后，点击回车。若检测到表达式有误，会在左侧界面显示表达式 error（错误），若表达式正确则会在左侧界面显示表达式及计算结果。计算结果保留四位小数。

1.2 程序的局限性

该多功能计算器输入运算式时必须使用英文输入法，中文字符会按照报错处理。在程序源代码中，给接收字符串输入的数组只保留了 1000 字符，因此如果输入的表达式字符量超过此数量结果将不可预知。并且左侧“界面”保留的空间更为有限，如果输入过多的字符看起来也会比较难看。

2. 程序数据结构与逻辑

2.1 主函数的数据结构

```
char c[1000],b[1000]={0};
double a[1000]={0};
int len=-1,lenm=-1,judge=0;
```

数组 c 用于接收用户输入的原始的字符串。

数组 a 和 b 并列存放处理好的运算式，a 装数据，b 装运算符。当 a 其中的值出现-1 时，表示数据有误，条件分支判断后会直接输出报错。当 b 其中值为 0 时表示空，如果在有效长度之内则表示其相对地址在 a 中为数据。

len 和 lenm 都是表示数组长度，存放的是当前数组最后一个有效位的相对地址（从 0 开始算）。len 表示 a 和 b 的长度，lenm 表示 c 的原始长度。当这两个变量值等于-1 时，表示对应的数组数据有误，条件分支判断后会转到输出

报错。
judge 是专门给 judger 函数存放“布尔”的变量。1 为真表示运算式合法，0 为假表示运算式不正确

数据结构示例如下：假设当前运算式为 1.1-(2.2+ -3)*3

	0	1	2	3	4	5	6	7	8
a	1.1			2.2		-3			3
b	0	‘-‘	‘(‘	0	‘+’	0	‘)’	‘*’	0

此时 len=8，lenm 必然大于 8。

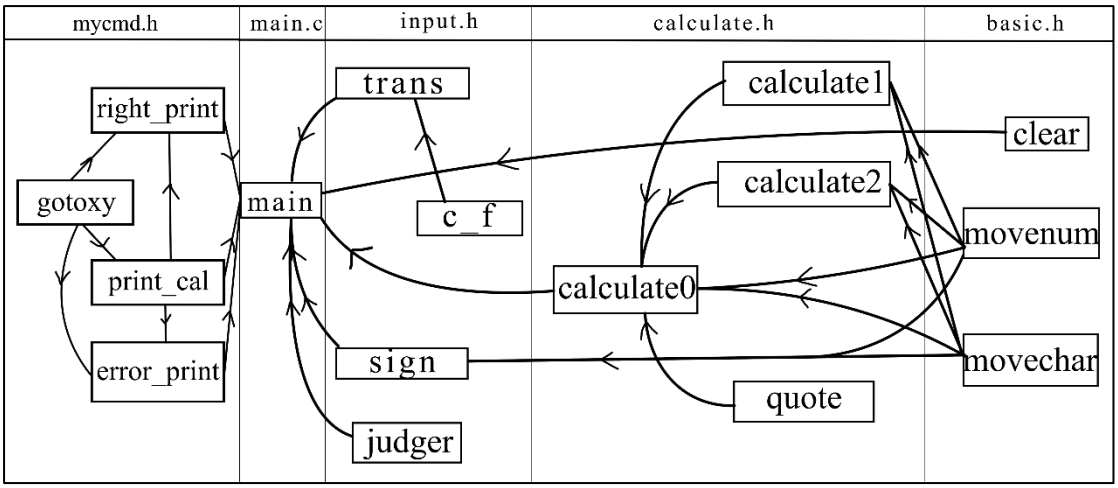
2.2 所有函数及调用关系

所有函数功能如下表

函数名称	输入	输出	功能
c_f	char a[],int d,int l	double	输入字符串数组转换成浮点数
trans	char c[],double a[],char b[]	int	将输入字符分解到二数组 a 和 b，并输出处理后数组长度
sign	double a[],char b[],int l	int	判断是否为负号，并乘进数据，并输出处理后数组长度
movechar	char a[] int adr1 int adr2 int l	void	移动数组从 adr2 到 adr1
movenum	double a[] int adr1 int adr2 int l	void	移动数组从 adr2 到 adr1
quote	char b[],int l,char section[2]	int	寻找最内层括号的区间，并判断其合法性，并输出情况-1,0,1
calculate0	double a[],char b[],int l	int	完成一次最里层括号的运算与数据替换，移动，并输出处理后数组长度
judger	char b[] int l	int	判断表达式合法性，输出布尔
calculate1	double a[],char b[] int p[2]	void	计算括号内每一次的加减运算并替换，移动
calculate2	同上	void	计算括号内每一次的乘除运算并替换，移动
clear	double a[] char b[],int len	void	抹除数组 a 和 b 的数据
gotoxy	int x,int y	void	光标移动到坐标（x,y）
print_cal		void	打印计算器“字符画”
error_print	char a[]	void	报错时输出
right_print	char a[],double result	void	正常时输出
main		int	主函数

表 1 函数功能一览

调用关系如下图：



2.3 主函数及基本算法的实现

```
#include <stdio.h>
#include "input.h"
#include "cauculate.h"
#include "mycmd.h"

int main()
{   char c[1000],b[1000]={0};
    double a[1000]={0};
    int len=-1,lenm=-1,judge=0;
while(1)
{   len=-1;lenm=-1;judge=0;
    while (judge==0 || len<0)
    {   clear(a,b,lenm+1);
        print_cal();
        gotoxy(52,6);
        gets(c);
        lenm=len=trans(c,a,b);
        if(len<0)
        {   error_print(c);
            continue;
        }
        len=sign(a,b,len);
        if(len<0)
        {   error_print(c);
            continue;
        }
        judge=judger(b,len);
        if(judge==0)
        {   error_print(c);
            continue;
        }
    }

    while(len>0)
        len=cauculate0(a,b,len);
    if(len==-1)
        error_print(c);
    else
        right_print(c,a[0]);
}
}
```

如图是主函数的代码。
函数对数组操作时，都会需要当前的数组长度 len。当改变了数组的有效长度时，也会返回操作后的数组长度，方便下一步操作。
在计算过程中，其实只需要多次调用 calculate0，迭代直到数组里只剩一个数，就完成了计算，得到了结果。

一个大循环
{　重置变量
当输入不合法执行输入内层循环
{
清空 ab 数组
展示窗口，等待输入
将输入分至 ab 数组
出现非法字符 continue
判断并将负号乘入数据
浮点数不合法 continue
判断运算式合法与否
运算式不合法 continue
}
当运算式没被算到只剩一个数时执行循环
{计算内层括号结果并替换}
运算过程报错，输出报错。
运算过程正常，输出结果。
}

2.4 核心函数 trans 的算法

```
//i：指向原字符串数组，j：指向两个输出数组，k指向n临时数组，d表示小数点相对地址。
//a为数据，b为运算符
int trans(char c[],double a[],char b[])
{   int i,j=0,k,d;
    char n[40];
    for(i=0;c[i]!='\0';)
    {   if (c[i]=='+' || c[i]=='-' || c[i]=='*' || c[i]=='/' || c[i]=='(' || c[i]==')')
        {   b[j]=c[i];
            j++;
            i++;

        }
        else if((c[i]>='0' && c[i]<='9') || c[i]=='.')
        {   for(k=0,d=-1;(c[i]>='0' && c[i]<='9') || c[i]=='.';k++,i++)
            {   if(c[i]=='.')
                    d=k;
                n[k]=c[i];
            }
            if(d==-1)
                d=k;
            a[j]=c_f(n,d,k-1);
            if (a[j]<0)
                return EOF;
            j++;
        }
        else if(c[i]==' ')
            i++;
        else
            return EOF;

    }
    return j-1;
}
```

如图为该函数的源代码，函数输入输出见前表

本函数是对从主函数传进来的这三个数组操作。将 c 的字符串转换到 double a[]和 char b[]中。Char n[]为单独存放数字字符串的数组，其将被 c_f 转换成浮点数。I，j，k 为数组元素的相对地址，i 指向 c 数组，j 指向 a 和 b 数组，k 指向 n 数组。

当 c 数组没结束时执行循环

{

当 c 中的元素是运算符时

{

放到 b 对应的格子里

ij 都指向下一个元素

}

当 c 中元素是数字时

{

i 扫描到不是数字为止

将扫描到的字符串数据存入 n 数组中

c_f 转换成浮点数存入 a 数组中，若出问题-1 报错

j 指向下一个元素。

}

是空格时：跳过空格，指向 i 下一个；

出现其他字符：返回-1 报错

}

返回 a 和 b 数组长度 j-1

3.5 核心函数 calculate0 的算法

```
//算好一个括号内的数据并替换（迭代后可得出最终结果）
int cauculate0(double a[],char b[],int len)
{   int judge=0,p[2],p1,p2;
    judge=quote(b,len,p);
    p1=p[0];p2=p[1];
    if (judge==-1)
        return -1;
    cauculate2(a,b,p);
    cauculate1(a,b,p);
    if(p[1]-p[0]>2)
        return -1;
    if(p[1]-p[0]==0)
        return 0;
    a[p1]=a[p1+1];
    b[p1]=0;
    movenum(a,p1+1,p2+1,len);
    movechar(b,p1+1,p2+1,len);
    len=len-(p2-p1);
    return len;
}
```

如图为该函数的源代码，函数输入输出见前表。
judge 接收 quote 返回值，用于判断括号合法与否。
p[2]存放最内层括号区间。
p1，p2 记录原括号位置。

找到一个最内层括号
判断其合法性，不合法返回-1 报错
先算好所有乘除运算的结果替换掉
再算好所有加减运算的结果替换掉
若括号中剩下了不只一个数，报错
若连括号都没有了，直接返回 0 长度
一般情况
{
去括号
将后面的数据补位
计算操作后的数组长度并返回
}

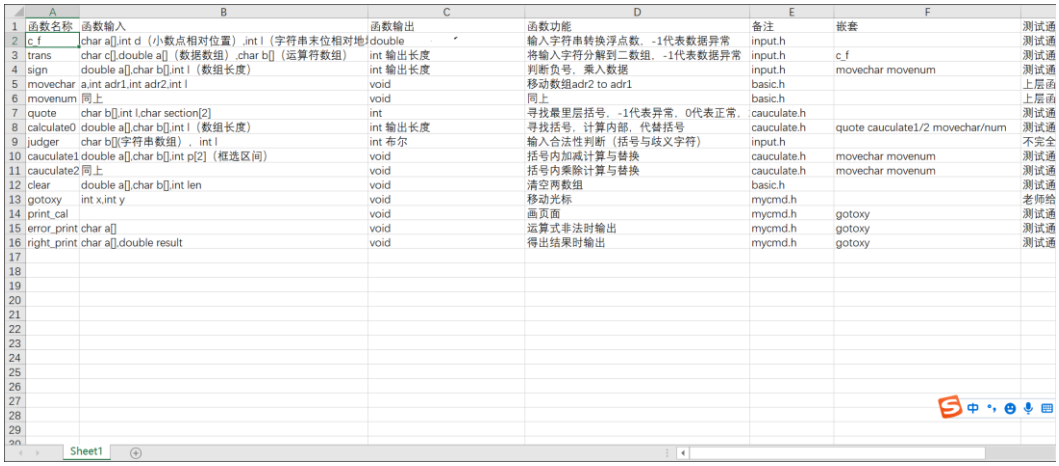
3. 开发过程

本次程序从敲第一个函数到 main 函数测试完成用时七个晚上，当然咕咕咕了好久才开始敲报告。

3.1 开发过程中的记录

产生的记录有：

1. 课设函数与进度表.xlsx，用于记录各个函数的功能，输入输出，编写进度和测试进度。
2. 开发记录.log。反正一个人开发，后来就懒得写了。
3.
4. 一张潦草的电子版草稿纸。边开发边在上面记录思维过程，整理思路，后期给几位女生讲解时被画的更乱了。



	A	B	C	D	E	F	
1	函数名称	函数输入	函数输出	函数功能	备注	嵌套	测试通
2	c_f	char a[],int d（小数点相对位置）,int l（字符串末位相对地址	double	输入字符串转换浮点数，-1代表数据异常	input.h		测试通
3	trans	char c[],double a[]（数据数组）,char b[]（运算符数组）	int 输出长度	将输入字符分解到二数组，-1代表数据异常	input.h	c_f	测试通
4	sign	double a[],char b[],int l（数组长度）	int 输出长度	判断负号，乘入数据	input.h	movechar movenum	测试通
5	movechar	a,int adr1,int adr2,int l	void	移动数组adr2 to adr1	basic.h		上层函
6	movenum	同上	void	同上	basic.h		上层函
7	quote	char b[],int l,char section[2]	int	寻找最里层括号，-1代表异常，0代表正常，	cauculate.h		测试通
8	calculate0	double a[],char b[],int l（数组长度）	int 输出长度	寻找括号，计算内部，代替括号	cauculate.h	quote cauculate1/2 movechar/num	测试通
9	judger	char b[](字符串数组），int l	int 布尔	输入合法性判断（括号与歧义字符）	input.h		不完全
10	cauculate1	double a[],char b[],int p[2]（框选区间）	void	括号内加减计算与替换	cauculate.h	movechar movenum	测试通
11	cauculate2	同上	void	括号内乘除计算与替换	cauculate.h	movechar movenum	测试通
12	clear	double a[],char b[],int len	void	清空两数组	basic.h		测试通
13	gotoxy	int x,int y	void	移动光标	mycmd.h		老师给
14	print_cal		void	画页面	mycmd.h	gotoxy	测试通
15	error_print	char a[]	void	运算式非法时输出	mycmd.h	gotoxy	测试通
16	right_print	char a[],double result	void	得出结果时输出	mycmd.h	gotoxy	测试通
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							

Sheet1

图 3 课设函数与进度表.xlsx

总共编写了 15 个函数，约 300 行代码，四个头文件一个 main.c。许多同学表示我这敲得又快又少又完成了功能。

3.2 开发的基本流程

首先整体构思，如何进行计算。把整体思路画在草稿纸上，之后再根据每一步功能的需要，分解出需要敲的函数。

思考如何实现这个函数的算法→把思路画在那张草稿纸的相应位置上→在函数与进度表上记录该函数→敲代码（*面向草稿的程序设计*）→在 test.c 编写测试函数→运行测试→报错或出 bug 找问题（*可能需要面向百度修改 bug*）→完成后骄傲的在进度表上写上“测试通过”

（*据说在我的 BGM 里敲的代码不会出错*）

循环上述步骤做好每一个函数，最后在 main.c 中根据草稿纸上的思路调用函数，完成最后的代码。之后“*一位测试工程师走进这家酒吧*”，进行大量的测试，把各种非法的、合法的输入都试一遍。

3.3 开发过程中的插曲与经验教训

1. 有一个循环条件中的 l（数组长度）和 1 敲反了，找了这个错误找了一个

晚上最终才发现的（真的差点砸电脑了）。之后立刻把文本编辑器的字号调大，更改字体，并且后边的函数中都用 len 代替 l。
这个事情告诉我们给变量起名的时候不要用一个字母，多敲几个字母累不死人。

图 4　IT 类专业学生表示深有体会

2. 对于如何阻拦非法运算式，“*如何拦住顾客在酒吧点的炒饭*”，的确有点费脑子。我懒得在每个函数中都加个判断条件，所以决定寻找一个函数拦住所有非法运算式。
   一个如今在北邮的高中同学建议我考虑下 antlr 或者 yacc 语法分析，可这现学估计来不及了。不过在搜索了之后，其中代码的 lookahead 和 lookback 的这种方法让我灵光乍现，敲出了那个 judger 函数。事实证明那个函数除了反括号这种阴险的错误外，其他的都妥妥能拦住。那个阴险的错误在后面检索最内层括号的 quote 函数中用一个条件判断轻松解决。当然为保险起见我还是在中间加了些判断。
   后期决定借本编译原理好好看看语法分析和词法分析到底是何方神圣。
3. 在写完了所有代码后，我好奇的搜了一下这个课设，看到知乎网友使用了“堆栈”的方法。不过仔细看了看，本质思想都差不多。只不过我没使用“栈”这个概念。但每步计算后的结果替换进数组中的原位置以及向前移动后面的数据，都已经体现了其本质的思想。
   我个人认为，没必要过分迷信那些专业人士的算法，咱都有脑子，只要逻辑能想通就一定能用代码实现出来。

4. 程序测试

部分测试情况如下表，所有举例均可代表一类输入。

测试输入	预期结果	实际结果
((-5.5))	-5.5000	-5.5000
0.1	0.1000	0.1000

1+ -1	0.0000	0.0000
1++1	error	error
3+2)-4*(2	error	error
1-(2.2*(3/4)-5)-(-2-3)	9.3500	9.3500
（直接回车）	error	error

表 2 代表性测试数据

结论：程序能够完成文章前所述的功能

5. 结语

经过这次 C 语言课程设计，从写程序的习惯，到各类算法的灵活应用都有所提高。真正做到了把 C 语言活学活用到了实际中。写这篇课设报告的过程中，总结中有提高，收获颇丰。