CS205 C/C++ Program Design Project1

> Name:章志轩
>
> SID:12010526

>
> Time:2023/03/05

Abstract:四则运算是数学的重要基础，也是进行复杂逻辑运算的前提条件。在Project1中我将使用C语言实现一个简单的计算器包含+-*/四种运算，并探索大数运算实现过程。

I. 项目介绍

C语言是是一门被广泛使用的高级语言，也是第一个且唯一一个被用来实现了通用操作系统内核的高级语言，是C++，Java等其它高级语言的基础。想学好编程需要扎实的数学功底。做为初学者，在这次Project中我将尝试完成一个计算器，包含简单的四则运算。同时，我将处理各种错误输入，并在命令行显示错误类型。

II.格式/参数

1. 语言：`C`语言，不能使用`C++`；
2. 编译器：`gcc`，不能使用`g++`；
3. 执行文件：命名为`calculator`
4. 入参格式：从命令行输入参数，不同参数间必须有空格分隔；
5. 文件名：`calculator.c`；
6. 输出：在命令行打印结果；
7. 四则运算：加法(`+`),减法(`-`),乘法(`x`),除法(`/`)。注：乘法是字母`x`，因为`*`会被识别为通配符；
8. 命令行命令（项目文件下）：

```
/* 生成可执行文件 */
gcc calculator.c -o calculator
/* 执行可执行文件，如./calculator 1 + 1.2 */
./calculator <数字1> <操作符> <数字2>
```

III.解题思路

1. 为实现命令行输入参数，将main函数设置为：`int main(int argc, const char* argv[])`；
2. 检查入参数，当`argc < 4`时说明参数不足，打印`Error input!`，当`argc>4`时，末尾参数被舍弃；
3. 检查argv[2](对应操作符输入)，当长度不为一时说明输入不符合操作符`+-x/`，打印`Error input!`；

   如此，输入格式满足要求。接下来将`argv[1], argv[2][0], argv[3]`赋值给`num1(char*),op(char),num2(char*)`进行参数格式检查(注：`argv[0]`是`./calculator`)。

4. 检查`num1, num2`，若不是数字(指含有`0~9 . - e`以外的字符或不符合正常数字/科学计数法格式)；
5. 检查`argv[2]`，若不等于`+-x/`，打印`The operator cannot be recognized as + - x or /!`；
6. 为实现大数运算，将`num1`和`num2`用`<stdlib.h>`库里的`strtod(char*, NULL)`转为`double`类型(可处理科学计数法)，并赋值给`long double`类型的`arr3, arr4`(这么命名是历史遗留问题)；
7. 利用`switch`根据`op`选择正确的运算(`+-x/`)，然后将结果赋值给`long double`类型的`ans`，此时存在`default`默认打印`Unknown error!`表示中间出现未知错误；
8. 若`op='/'`，则还要进行除数为0的检查，若为0打印：`A number cannot be divied by zero.`；
9. 直到这一步说明运算不存在问题，打印正确结果:`num1 op num2 = ans`.

IV.运算范围

由于正确输出的`ans`类型是`long double`，只能表示1.18973e+4932 ~ 3.3621e-4932，有效位数(精度)只有18~19个数字，对于输入的`char*`类型数字虽然没有长度限制，但超出这个范围就会显示`±inf`，它们的范围同样是1.18973e+4932 ~ 3.3621e-4932。

V.所有可能结果

按优先级排序：

1. 输入的参数不足3(`argc<4`) || 操作符(`argc[3]`)长度不为1 -> 输出：`Error input!`；
2. 输入的不是数字(argv[1], argv[3]) -> 输出：`The input cannot be interpret as numbers!`；
3. 输入的操作符不是`+ - x /` -> 输出：`The operator cannot be recognized as + - x or /!`；
4. 除数是0 -> 输出：`A number cannot be divied by zero.`；
5. 未知的错误 -> 输出：`Unknown error!`；
5. 正常加减乘除 -> 输出：`num1 op num2 = ans`；

VI.使用的头文件及方法

- `#include <stdio.h>  printf,puts`
- `#include <stdbool.h>  bool`
- `#include <stdlib.h>  strtod`
- `#include <string.h>  strlen`

VII.收获经验

事实上提交版本为第二版。在初版中，由于输入长度不定我打算用`char*`自动适应输入长度，并直接对`char`数组进行操作，获得高精度高范围的计算器。我在初版尝试将所有数都转换成标准的科学计数法(`char`数组)，然后通过下标读取数组，计算结果，并根据结果选择输出格式(科学计数法or普通样式)。事实上以上行为在方法内部都实现的很好，然而将它们组装起来后结果就变的不可预测了。它有时显示乱码，有

时丢失字符，有时前一个数组被替换成了别的数组(我未进行操作)，这令我很烦躁并调试了很久。最后在网络上查找答案时我发现在C中函数返回数组头指针尽管不会报错，却是个极其不智的做法，因为局部变量的数组保存在堆栈，返回指针并不能保护数据，数组内存地址很可能被其它方法覆盖，导致数据丢失或出错。为此我又去探索了全局变量，但结果却总是`Segmentation fault`。最后我放弃了这种激进的想法，选择用`long double`转换数组并进行运算。

C与Java是不一样的，在C中数组的修改和赋值都有严格的限制，对内存的操作必须十分清晰且熟练(本次Project最大的收获)，更不可能像python那样有自由的书写方法。本次Project我发现其它语言语法严重影响C程序在编写时的思路。错误的开头会导致错误的结尾，今后我将更加着重关注C的基础知识和语法，像编程新手那样谨小慎微，稳步学习。

VIII.源码

https://github.com/15775011722/CS205-Sustech-Cpp/blob/main/project/project1/calculator.c