# 实验二（第2次上机）：Pair Project

## 一、实验目的及要求：

1. 体验敏捷开发中的两人合作
2. 进一步提高个人编程技巧与实践

## 二、任务说明：

1. 根据四则运算的问题，练习结对编程实践，完成实验报告。
2. 要求学生两人一组，自由组合。二人共同编码，两人的角色至少切换4次，完成实验要求。

**四则运算问题说明**：

任何编程语言都可以，命令行程序接受一个数字输入，然后输出相应数目的四则运算题目和答案。例如输入数字是 30， 那就输出 30 道题目和答案。 运算式子必须至少有两个运算符，运算数字是在 100 之内的正整数，答案不能是负数。

**问题扩展：**

1. 一次可以出一千道道题目，并且没有重复的，把题目写入一个文件中
2. 支持多个运算符，最多多 10 个运算符，括号的数量不限制

## 程序代码

#include <stdio.h>

#include <stdlib.h>

float operato(int a,int b,char d){

float add;

int c;

c=rand()%100+1;

int x;

x=rand()%4;

if(d=='+'){

switch(x){

case 0:

add=a+b+c;

if(add<0){

break;

}

printf("%d%c%d+%d=\n",a,d,b,c);

return(add);

break;

case 1:

add=a+b-c;

if(add<0){

break;

}

printf("%d%c%d-%d=\n",a,d,b,c);

return(add);

break;

case 2:

add=a+b\*c;

if(add<0){

break;

}

printf("%d%c%d\*%d=\n",a,d,b,c);

return(add);

break;

case 3:

add=a+b/c;

if(add<0){

break;

}

printf("%d%c%d/%d=\n",a,d,b,c);

return(add);

break;

}

}

if(d='-'){

switch(x){

case 0:

add=a-b+c;

if(add<0){

break;

}

printf("%d%c%d+%d=\n",a,d,b,c);

return(add);

break;

case 1:

add=a-b-c;

if(add<0){

break;

}

printf("%d%c%d-%d=\n",a,d,b,c);

return(add);

break;

case 2:

add=a-b\*c;

if(add<0){

break;

}

printf("%d%c%d\*%d=\n",a,d,b,c);

return(add);

break;

case 3:

add=a-b/c;

if(add<0){

break;

}

printf("%d%c%d/%d=\n",a,d,b,c);

return(add);

break;

}

}

if(d='\*'){

switch(x){

case 0:

add=a\*b+c;

if(add<0){

break;

}

printf("%d%c%d+%d=\n",a,d,b,c);

return(add);

break;

case 1:

add=a\*b-c;

if(add<0){

break;

}

printf("%d%c%d-%d=\n",a,d,b,c);

return(add);

break;

case 2:

add=a\*b\*c;

if(add<0){

break;

}

printf("%d%c%d\*%d=\n",a,d,b,c);

return(add);

break;

case 3:

add=a\*b/c;

if(add<0){

break;

}

printf("%d%c%d/%d=\n",a,d,b,c);

return(add);

break;

}

}

if(d='/'){

switch(x){

case 0:

add=a/b+c;

if(add<0){

break;

}

printf("%d%c%d+%d=\n",a,d,b,c);

return(add);

break;

case 1:

add=a/b-c;

if(add<0){

break;

}

printf("%d%c%d-%d=\n",a,d,b,c);

return(add);

break;

case 2:

add=a/b\*c;

if(add<0){

break;

}

printf("%d%c%d\*%d=\n",a,d,b,c);

return(add);

break;

case 3:

add=a/b/c;

if(add<0){

break;

}

printf("%d%c%d/%d=\n",a,d,b,c);

return(add);

break;

}

}

return 0;

}

int main(){

printf("请输入题目个数:");

int i,times=0;

srand(100);

scanf("%d",&i);

do{

int x,a,b;

float add;

char d;

x=rand()%4;

a=rand()%100+1;

switch(x){

case 0:d='+';

break;

case 1:d='-';

break;

case 2:d='\*';

break;

case 3:d='/';

break;

}

b=rand()%100+1;

add=operato(a,b,d);

if(add<0){

i++;

continue;

}

printf("答案:%.2f\n",add);

times++;

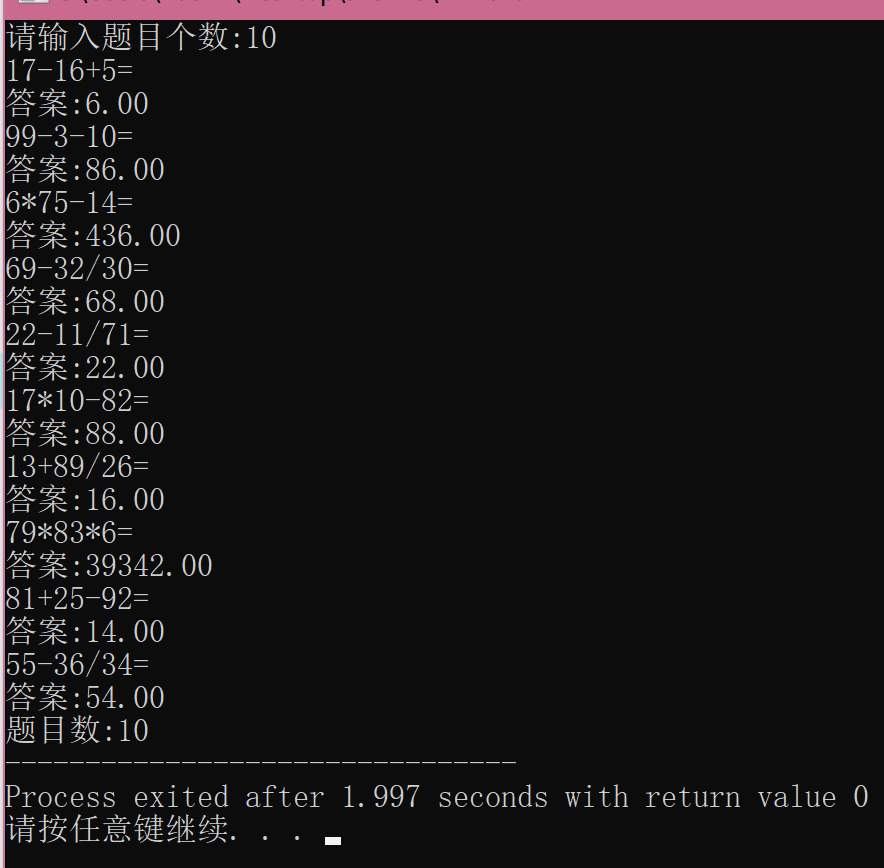
}while(times<i);

printf("题目数:%d",times);

return 0;

}

## 四、设计与结果



## 代码复审表

1．概要部分

（1）代码能符合需求和规格说明么？

能，可以自定义题目题数

1. 代码设计是否有周全的考虑？

否，还有一些没有考虑到的，会继续完善的

1. 代码容易维护么？

容易维护

1. 代码的每一行都执行并检查过了吗？

是的，都可以执行

2．设计规范部分

（1）设计是否遵从已知的设计模式或项目中常用的模式？

是的，因为比较熟悉，更容易操作

（2）代码有没有依赖于某一平台，是否会影响将来的移植（如Win32到Win64）？ 没有，不会影响移植，任何平台都可以

（3）开发者新写的代码能否用已有的Library/SDK/Framework中的功能实现？在 本项目中是否存在类似的功能可以调用而不用全部重新实现？

存在，有些代码是可以调用的

1. 有没有无用的代码可以清除？

没有，已经全部清除完毕了

3．代码规范部分

（1）修改的部分符合代码标准和风格么?

符合，全都按照代码标准修改的

4．具体代码部分

（1）有没有对错误进行处理？对于调用的外部函数，是否检查了返回值或处理了异常？

对错误都进行了处理，没有异常

1. 参数传递有无错误，字符串的长度是字节的长度还是字符（可能是单/双字节）的长度，是以0开始计数还是以1开始计数？

是字节的长度，以1开始计数的

（3）循环有没有可能出现死循环？

没有可能，可以重复抽取

（4）有没有可能导致资源泄露（内存、文件、各种GUI资源、数据库访问的连接，等等）？有没有可能优化？

不会导致资源泄露，可以优化使用断言来保证我们认为不变的条件

5．效能

（1）代码的效能（Performance）如何？

达到了具体任务要求的程度

（2）代码中，特别是循环中是否有明显可优化的部分（C++中反复创建类，C#中 string 的操作是否能用StringBuilder 来优化）？

可以用StringBuilder 来优化，当我们需要多次的对一个字符串进行多次操作的时候，他的效率要远远高与string

1. 对于系统和网络调用是否会超时？如何处理?

目前没有出现超时的现象。假如出现了我们会 杀毒；整理系统，减少运行的进程，释放内存、cpu，释放c盘空间；

6．可读性

代码可读性如何？有没有足够的注释？

可以读取，注释都是最新的，只是有部分没有测试

7．可测试性

代码是否需要更新或创建新的单元测试？

暂时不需要

## 个人软件过程耗时估计与统计表

|  |  |  |  |
| --- | --- | --- | --- |
| Personal Software Process Stages | 个人软件流程阶段 | 估计的时间（分钟） | 实际花费的时间 (分钟) |
| **Planning** | **计划-把工作细化并大致安排次序** | 6 | 10 |
| **Development** | **开发** | 60 | 120 |
| · Analysis | 需求分析  (包括学习新技术) | 10 | 8 |
| Design Spec | 生成设计文档 | 15 | 11 |
| Design Review | 设计复审  (和同事审核设计文档) | 10 | 8 |
| Coding Standard | 代码规范  (制定合适的规范) | 5 | 2 |
| Design | 具体设计 | 5 | 6 |
| Coding | 具体编码 | 8 | 5 |
| Code Review | 代码复审 | 20 | 13 |
| Test | 测试（自我测试，修改代码，提交修改） | 20 | 15 |
| **Reporting** | **总结报告** | 10 | 8 |
| Test Report | 测试报告 | 10 | 7 |
| Size Measurement | 计算工作量 | 5 | 2 |
| Postmortem & Improvement Plan | 事后总结,并提出改进 | 5 | 3 |
| **Total** | **总计** | 189 | 218 |

## 六、总结

这次实验虽然在初学C语言时写过很多，但也许是第一次和同学合作写代码，彼此没有及时沟通，这样导致开发过程中有很多可以避免的问题没有及时解决，从而导致进度滞后。

但是也让我们提前学习到在开发大项目中的及时沟通和交流的重要性，也让我们学到了很多。