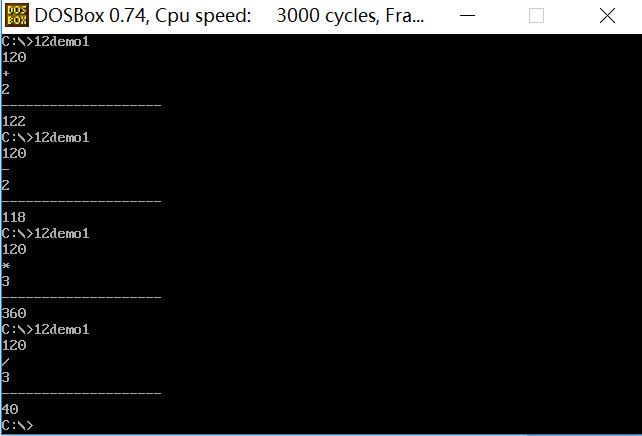
**c语言综合研究十二报告**

1. **研究过程展示**

**1>.程序1部分：**

运行：



整个程序的逻辑就是先定义数组和变量，再输入，再判断输入，再输出。所以这里运算符的地位都是平等的，要将程序扩充为识别“\*”、“/”，要在判断和输出里加上“\*”、“/”，再在输出语句里实现相关算法。修改的程序如下：

程序部分：

main()

{

char a[20];

char b[20];

char ch;

gets(a);

printf("%c\n",ch = getch());

gets(b);

if(ch != '+' && ch != '-' && ch != '\*' && ch != '/' )

{

printf("error!");

return;

}

printf("--------------------\n");

if(ch == '+')

{

printf("%d",atoi(a)+atoi(b));

}

if(ch == '-')

{

printf("%d",atoi(a)-atoi(b));

}

if(ch == '\*')

{

printf("%d",atoi(a)\*atoi(b));

}

if(ch == '/')

{

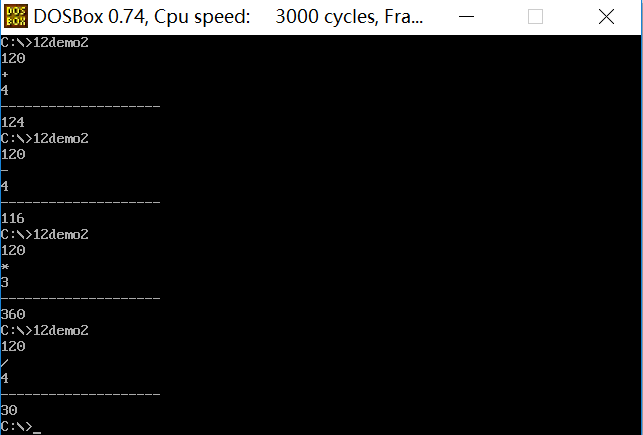
printf("%d",atoi(a)/atoi(b));

}

}

**2>.程序2部分：**

运行：



程序二是定义了两个运算函数add、sub，分别实现将两个数相加和将两个数相减的功能。又定义了一个函数指针数组存放这两个函数的地址。这样可以通过数组来选择并调用函数。另外我们定义了一个字符指针code，并初始化为“+-”，然后在main函数里用for语句判断输入的符号是加号还是减号，如果是加号，则for语句结束后n的值为0，如果是减号，则for语句结束后n的值为1。通过n的值作为参数，可以通过函数指针调用具体要调用的函数。

程序部分：

char\* code = "+-\*/";

int add(int a,int b)

{

return a+b;

}

int sub(int a,int b)

{

return a-b;

}

int mul(int a,int b)

{

return a\*b;

}

int div(int a,int b)

{

return a/b;

}

int (\*func[4])(int,int) = {add,sub,mul,div};

main()

{

char a[20];

char b[20];

char ch;

int n;

gets(a);

printf("%c\n",ch = getch());

gets(b);

for(n = 0;code[n] && code[n] != ch;n++ );

if(!code[n] )

{

printf("error!");

return;

}

printf("--------------------\n");

printf("%d",func[n](atoi(a),atoi(b)));

}

对比程序1和程序2，可以发现程序1的语句都是在main函数里，它使用的方法比较简单，但是要增加操作符比较麻烦，要修改程序需要再增加输出语句并修改判断方法；而程序2是用子函数来实现一个单独的功能，并使用函数指针来进行函数的调用，使用函数指针数组来管理函数，这样所有的输入、输出、判断语句都在main函数中，关于算法的语句都在子函数里，如果我们要增加新的操作符，只需要写一个新的子函数并在函数指针数组里将函数名写入，在字符指针里加入要添加的操作符，修改code数组的大小，不需要修改main函数。

程序2中的共性实现在main函数里，即输入、判断、输出的功能，个性实现在子函数。

1. **已思考研究并已解决问题汇总**
2. **已思考研究并未解决问题汇总**

1.好的程序设计思想是选择适合的，两种程序设计思想分别适合于何种情况？

1. **研究感想（心得体会）**

模块化的设计思想对于大型程序的开发、管理和维护有十分重要的作用。自己需要好好体会。