C1-E1答疑要点整理

关于`scanf`的使用

使用`scanf`读入数字并存入`int`类型变量时，不需要按照题目中的输入数据格式在格式字符串中加入空格或换行符`\n`

因为**使用`scanf`读入`int`类型的变量时**，它会在开始读入时**自动忽略空格、换行等字符**，直到遇到第一个数，执行读入，然后在下一个空格/换行处停止，将读到的数存入参数列表中对应的第一个变量，再根据格式字符串中格式符`%d`的个数，继续执行读入。举例如下：

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);    //使用scanf读入3个数字，分别存入变量a,b,c中，%d之间没有空格
```

使用上面的代码时，无论输入方式是

```
11 22 33
```

还是

```
11 22
33
```

或者是

```
11
22 33
```

都可以实现分别读入数字`11,22,33`并按顺序存入变量`a,b,c`中

当需要读入格式中包含非空白字符的数时，可以在`scanf`的格式化字符串中加入其他字符用于匹配，实现格式化读入，后面遇到时会专门说明，此处先不考虑.

关于初始化变量

当我们在`main`函数中使用类似`int a;`这样的语句声明变量后，变量中的值是**不确定的**，如果直接向该变量中累加其他值，最后得到的结果也是未知的。因此，在使用一个变量前，一定要确保该变量存有确定的值。如在向计数变量中累加前，需要通过赋值的方法设置其初始值为`0`

关于多组输入输出问题

当题目中有**多组数据**时，可以**每读入一组数据就进行处理、输出结果**。OJ系统在评测时，**只会读取**我们提交上去的程序的**输出结果，与测试点比对**，而并不检查这些结果是在程序运行的哪个阶段输出的。所以只要每组数据输入后就立刻处理，并按照格式要求输出结果就可以。注意输出每组数据后的**换行问题**，如果缺少换行，那下一组数据的输出会直接接在上一组数据输出同一行的后面，导致代码不能通过评测（可能返回`PE`或`WA`的结果）

关于不定个数据读入，以特定值（如`-1`）结束的问题

参考 PPT 可知，可以使用如下的语句实现多组数据读入，遇到`-1`时结束：

```
int score;
scanf("%d",&score); //预先读入一次，若为-1则不会进入后面的循环，否则进入
while (score != -1){
    //do something，可以是处理读入的score值操作
    scanf("%d",&score); //在循环中再次读入，以下一个输入数据更新score变量的值，这样才能实现在读到-1之前不断循环读入
    //若缺少该scanf，则score的值不会更新，保持原值，会导致该while循环的条件始终为真，无尽循环！
}
```

`id returned 1 exit status`报错

使用 Dev-C++ 编写并调试代码的同学，如果在编译时出现了`id returned 1 exit status`的报错，可能是因为上一次在 Dev-C++ 中运行的程序没有结束，或者是代码文件/文件夹被其余进程占用，关闭上一次程序运行的黑框或者重启 Dev-C++，重新编译运行即可

关于输出字符串

遇到题目要求输出字符串时，一定要将需要输出的字符串**复制粘贴**到代码中，自己输入的时候很可能会看错或切错输入法，从而导致代码无法通过评测的情况发生。如：`NO`与`NO`（前者为第14个大写英文字母和第1个自然数连接成的字符串），`YE5`与`YES`（前者为第25个大写英文字母、第5个大写英文字母和第6个自然数连接成的字符串），`！`与`!`（前者为全角叹号，后者为半角叹号），`？`与`?`（同为全角半角区别）

关于`if`语句

一个常见的语法错误

判断两数是否相等需要使用**两个连续等号**，即`a == b`，**单个等号**起到的是**赋值**作用，将等号右面的值赋给等号左面。

那么有人就问了：既然意思完全不对，那么为什么编译还能通过呢？

实际上，`if(...)`语句的行为是判断括号里面的表达式的值，`a==b`是逻辑判断表达式，它的数据类型为`int`，值为0或1。而`a=b`同样也是表达式，它叫做赋值表达式，它的值是`a`被赋值以后的值。而`if`只会判断括号里面的表达式的值是不是0，所以编译是可以通过的。

在`E1`的题解中，出现了这样的写法：

```
while(t--){
    //代码
}
```

这里的`t--`叫做“自减表达式”，在处理此表达式时，会将`t`原先的值作为表达式的值参与运算，然后使`t`减一。这样的写法实现的效果是将`//代码`部分执行`t(初始值)`遍。

一个常见的逻辑错误

使用`if`分支语句的时候，需要注意下面两种代码的区别：

```
//这里使用if-else逻辑
//当满足上面的if条件时
//确定不会进入下面else的代码块
//在上面代码块中修改过的a的值不会影响下面的代码块
if (a % 2 == 1){
    a = a + 1;
} else {
    a = a - 1;
}

//这里使用的是2个if并列逻辑
//当满足上面的if条件时
//上面的代码块执行结束后，仍然会判断是否满足下面的if条件
//从而决定是否进入下面代码块
if (a % 2 == 1){
    a = a + 1;
}
if (a % 2 == 0){ //如果在上一个if代码块中修改过a的值，且新值满足这个if的条件，则会进入这个代码块，从而导致a的值被第二次修改
    a = a - 1;
}
```

一个常见的写法错误

使用`if`分支判断语句或`while`、`for`条件循环语句时，建议在后面加上大括号，明确在满足条件时会执行的（被它们所“管着”的）代码是哪部分。如果不使用大括号划定范围，那么在满足条件时会执行的只有紧随其后的一句以分号结尾的代码。例如：

```
int a = 1, b = 1, c = 0;
if (c != 0){
    a = a + 1;
    b = b + 1;
}
//上面代码执行结果为：if的条件为假，a,b的值都保持不变，仍为1

int a = 1, b = 1, c = 0;
if (c != 0)
    a = a + 1;
    b = b + 1;
//上面代码执行结果为：if的条件为假，a值仍为1，b值增为2
```

避免诸多低级错误的小技巧

可以开启编译选项“显示最多警告信息(`-Wall`)”，更为谨慎的同学也可以开启`-Wextra`。 `-Wall`是 gcc 编译器认可的、很有用的警告选项集合。对于这些警告，应该理解其含义，通过修改代码来消除警告。接下来通过具体实例说明它的用处：

1. **`scanf`没加`&`**

   编写代码：

   ```
   #include <stdio.h>

   int main() {
       int a;
       scanf("%d",a);
       return 0;
   }
   ```

   编译后，弹出编译警告：

   ```
   [警告] format '%d' expects argument of type 'int *', but argument 2 has type
   'int' [-Wformat=]
   ```

   即： 格式`%d`的参数类型为`int *`，但参数 2 的类型为`int`。关于这句话是什么意思，涉及到我们日后会学习的“指针”概念。现在你只需要知道：编译器在警告你这行代码写得有问题。

2. **变量没初始化**

   编写代码：

   ```
   #include <stdio.h>

   int main() {
       int a;
       printf("%d",a);
       return 0;
   }
   ```

   编译后，弹出编译警告：

   ```
   [警告] 'a' is used uninitialized in this function [-Wuninitialized]
   ```

即：`a`在该函数中未初始化

3. **`if`没写俩等号**

   编写代码：

```
#include <stdio.h>

int main() {
    int a;
    scanf("%d",&a);
    if(a=0)
        printf("zero");
    else
        printf("%d",a);
    return 0;
}
```

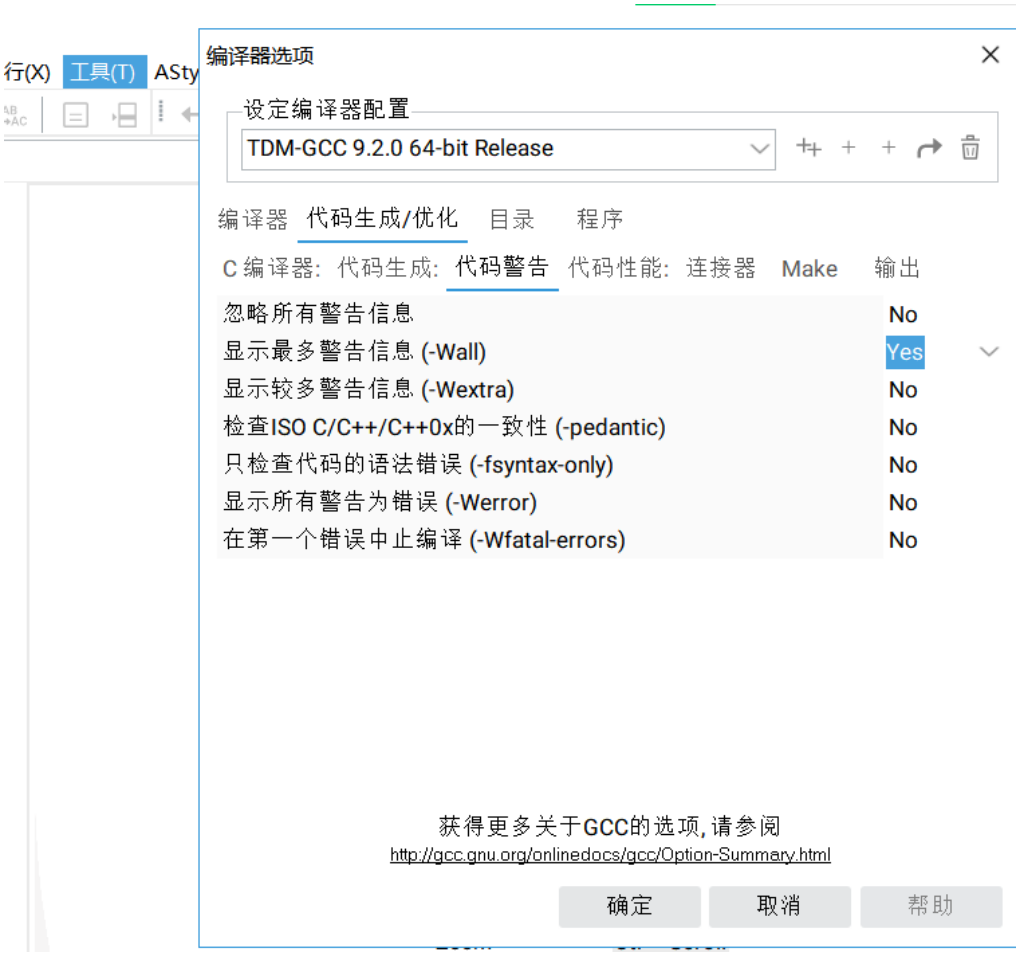
   编译后，弹出编译警告：

```
[警告] suggest parentheses around assignment used as truth value [-
Wparentheses]
```

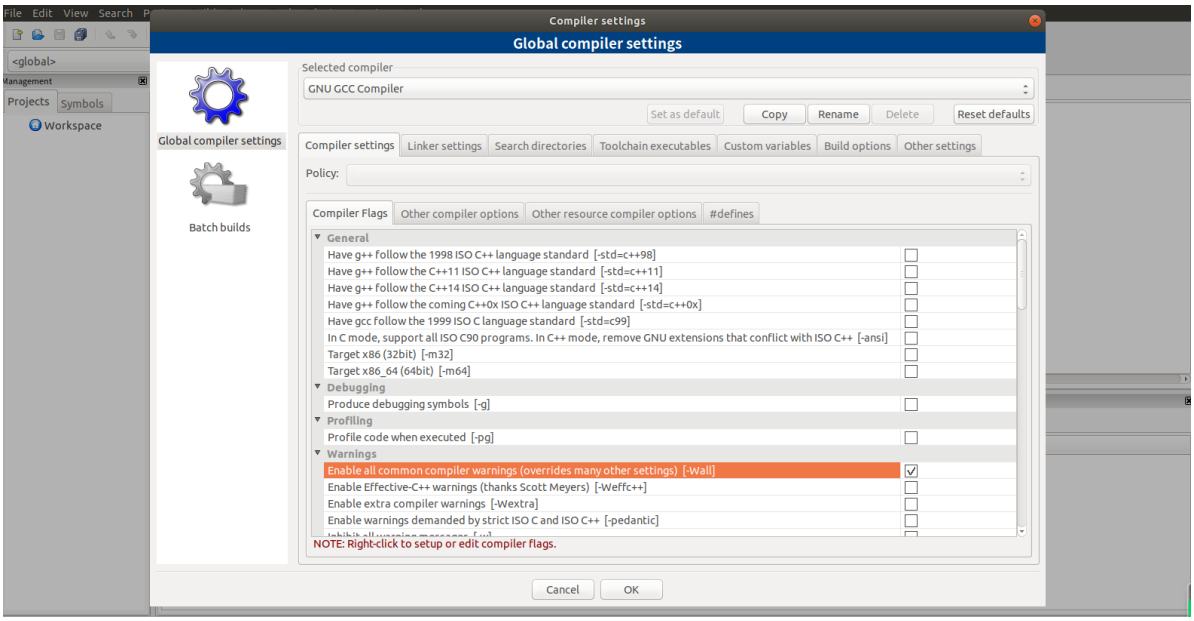
   即：建议在作为真值的赋值周围加上括号

开启方法：

对使用 **Dev C++** 的同学：在 工具-编译器选项-代码生成/优化-代码警告中，将"显示最多警告信息(`-Wall`)"右边的按钮切换为`Yes`，可以将下面的`-Wextra`也切换为`Yes`：



对使用 **Code::Blocks** 的同学：在任务栏上方的 Settings（设置）- Complier（编译器）- Global Complier Settings（全局编译器设置）-Complier settings（编译设置）- Complier Flags（编译标志）中，找到 Warnings（警告）一栏，再在下方找到`-Wall`（`-Wextra`也在下方），右面方格点击勾选，然后点击下方`OK`并重启软件即可启用：



Author：梁秋月 && 逐月的游星

审核：王君臣