实验环境

C语言实验环境：Code::Blocks 13.12、GNU GCC Compiler

Java语言实验环境：

java version "1.8.0\_60"

Java(TM) SE Runtime Environment (build 1.8.0\_60-b27)

Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)

文本语言编写：Notepad++

3.3 在Ada中有loop…exit结构；PL/I中有类似loop…break结构。在C和java中是否有类似结构？这是某种设计规则的一个范例吗？

解答：

Ada中loop…exit结构是一种常用的简单循环结构，反复执行同一操作。

最简单的loop语句格式为：

**loop**

statements

**end** **loop;**

当要退出该循环时，使用exit或exit when语句。exit表示直接退出循环，exit when则在符合when后面的条件才退出。

实例程序：

**procedure** testrange **is**

var **:** integer**;**

**begin**

**loop**

Put**(**"Enter an Integer number to confirm its range:"**);**

Get**(**var**);**

**case** var **is**

**when** Integer**'**First **..** **-**1 **=>**

Put\_Line**(**"It's a negative number"**);**

**when** 1 **..** Integer**'**Last **=>**

Put\_Line**(**"It's a positive number"**);**

**when** **others** **=>**

Put\_Line**(**"It is 0"**);**

**end** **case;**

**exit** **when** var **=** 0**;**

**end** **loop;**

**end** testrange**;**

实例程序循环输出"Enter an Integer number to confirm its range:"，要求输入一个整数；当输入值为0时，输出"It's a positive number"，再退出。

而我们知道，C和Java语言有while语句与break语句，能够实现上述相同功能：

**C语言实验环境：**

**test.c文件源代码**

#include <stdio.h>

int main**()**

**{**

int var **=** 0**;**

**while(**1**)**

**{**

printf**(**"Enter an Integer number to confirm its range:\n"**);**

scanf**(**"%d"**,&**var**);**

**if(**var **<** 0**)**

**{**

printf**(**"It's a negative number\n"**);**

**}**

**else** **if(**var **>** 0**)**

**{**

printf**(**"It's a positive number\n"**);**

**}**

**else**

**{**

printf**(**"It is 0\n"**);**

**break;**

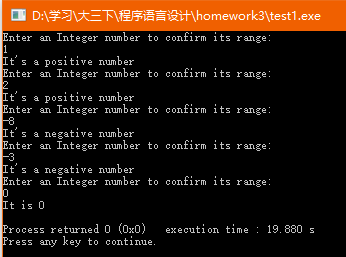
**}**

**}**

**return** 0**;**

**}**

C语言运行结果截图



**test.java文件源代码：**

**import** java**.**util**.\*;**

public class test1**{**

public static void main**(**String**[]** args**){**

Scanner inScan **=** **new** Scanner**(**System**.**in**);**

int var **=** 0**;**

**while(true){**

System**.**out**.**println**(**"Enter an Integer number to confirm its range:"**);**

inScan**.**hasNext**();**

var**=**inScan**.**nextInt**();**

**if(**var**<**0**){**

System**.**out**.**println**(**"It's a negative number"**);**

**}else** **if(**var**>**0**){**

System**.**out**.**println**(**"It's a positive number"**);**

**}else{**

System**.**out**.**println**(**"It is 0"**);**

**break;**

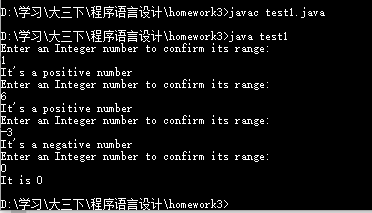
**}**

**}**

**}**

**}**

Java语言程序运行结果截图



**结果分析：** 在C和java中也有类似Ada的loop…exit与PL/I的loop…break结构。并且在C和Java中都在任意时刻能退出循环语句如while、for。在C中exit语句用于终止程序。就单对循环的作用来看，它们是一致的。

但我们注意到，对于break语句，在switch分支语句中，它的作用是退出switch语句防止程序继续执行下面的case。我们做简单的案例：

test1\_1.c文件源代码

#include <stdio.h>

int main**()**

**{**

int var **=** 0**;**

printf**(**"Enter an Integer number:\n"**);**

**while(**1**)**

**{**

scanf**(**"%d"**,&**var**);**

**switch(**var**)**

**{**

**case** 0**:**

printf**(**"It is 0\n"**);**

**break;**

**case** 1**:**

printf**(**"It is 1\n"**);**

**break;**

**default:**

printf**(**"other numbers\n"**);**

**break;**

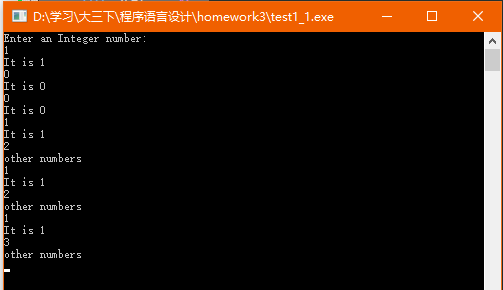
**}**

**}**

**return** 0**;**

**}**

Test1\_1.c程序运行截图



**结果分析：**

可以发现，在这里，break无法跳出循环，而只是跳过其后面的case，跳出switch。当然，当break只是一个在循环里的if条件分支语句中时，它的作业是可以跳出循环的。这里就存在了break语句的不规律性，也可以认为是非正交性的，也不能称之为范例了。

3.4 Java中，整型量可赋给实型量，反之不成立，这违反了什么规则？在C中没有这种限制，这又违反了什么规则？

解答：

Java中，整型量可赋给实型量，反之不行，这违反了正交性原则，这是数据与数据类型之间的交互，一个语言结构不应该有不同的行为。

C语言中，没有实型量赋给整型量的限制，这违反了安全性原则。我们知道，如浮点数类型值赋值给整型值时，很有可能会导致数据丢失，因此会带来不准确信息。

下述实例说明

test2.java中，源程序如下

**import** java**.**util**.\*;**

public class test2**{**

public static void main**(**String**[]** args**){**

int a **=** 5**;**

double a\_ **=** a**;**

double d **=** 5.676543**;**

int d\_ **=** d**;**

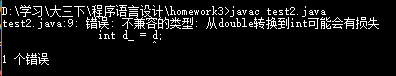
System**.**out**.**println**(**a\_**+**""**);**

System**.**out**.**println**(**d\_**+**""**);**

**}**

**}**

其运行结果：



当实型到整型转换时，编译时报错，“从double转换到int可能会有损失”。

将有关实型到整型转换的语句**注释**之后,如下：



说明整型到实型转换是被允许的。

test2.c中，源程序如下：

#include<stdio.h>

int main**()**

**{**

int a **=** 5**;**

double a\_ **=** a**;**

double d **=** 5.676543**;**

int d\_ **=** d**;**

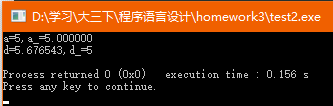
printf**(**"a=%d,a\_=%lf\n"**,** a**,** a\_**);**

printf**(**"d=%lf,d\_=%d\n"**,** d**,** d\_**);**

**return** 0**;**

**}**

test2.c程序运行结果如下



从运行结果可以看到，当整型转为实型（double、float）值的大小不会丢失；而当实型转为整型时，会发生自动截断，导致精度降低，值大小变化，而这对于写者逻辑是不合理的。

3.14 Ada和C是编程语言在注释说明上的两个对立的观点的代表，对于Ada,注释由相邻的连字符开始，并在行尾结束：

--this is an Ada comment

对于C，注释由“/\*”开始，直到对应的“\*/”结束:

/\* this is a C comment \*/

比较两种注释方法的可读性、可写性和可靠性。C++加入了一种很像Ada的注释规定（“//”），为什么C++不完全使用Ada的约定？

解答：

**可读性：** Ada语言中注释很难和其它语言结构混淆，其注释--直接代表其后所有的语句都是无效的。而对于C语言，则不是，它的注释有头有尾，当一条语句开始，我们不知道它将什么时候结束，当注释长度很宽的时候甚至不能在一页宽度中显示，这是不友好的，并且在注释后面可能还会有其它语句信息（虽然这是应该算编码者的错）。C语言中“/”和“\*”分别是除号和乘号算术运算符，这可能带来一些错误。

Ada**:**

2**-**2 --this is a comment

C**:**

2 **/** 3 /\*this is a comment \*/ **+** 6

2 **/** 3 **/** **\***this is an error **\*/ + 6**

/\* A comment

/\* a nested comment

...

but only one comment closer \*/

C语言注释也可以多行注释，与嵌入式注释会导致可读性降低，也会导致写者常只闭合一个注释，这是一个极大的易犯错误。

**结论：**Ada注释可读性强于C语言注释

**可写性：** 因为我们知道注释经常更新，也并不一定只用一行就能说明清楚。在这种情况下，C语言的注释的优势就体现出来了，每当多添一行注释，Ada语言需要重新使用“--”来代表注释，而C语言只需要记录一下开头和结尾，就能写出多行注释。这种情况下C优于Ada。但单行注释来说，C语言需要闭合注释，而Ada直到行尾，Ada优于C。

/\* one line comment C \*/

-- one line comment Ada

/\* C

1st comment

2nd comment

3rd comment

\*/

-- Ada

-- 1st comment

-- 2nd comment

-- 3rd comment

**结论：**可写性中，多行注释C优于Ada；单行注释Ada优于C，总体来说C可写性优于Ada。

**可靠性：**可靠性的表现与可读性相关，一个可读性更好的语言结构，它的可靠性也会更好，因为读者可以迅速的发现写者的逻辑错误或者编写错误。Ada注释的主要特点是，所有注释在一行表现完毕，不需要后续闭合注释（适当匹配符号）。而且在可读性中，我们注意到C语言嵌套注释会导致的问题，可能闭合注释时只闭合了一个注释，这是错误的来源，还有闭合注释后的其它语句容易被遗漏。

结论：Ada注释的可靠性强于C语言。

**C++中“//”引入当行注释的讨论：**

C++向下兼容C语言，首先保证语言统一性，“/\*”与“//”是相似的，它们都用来标注注释，这是具有统一性的表现。然后更为重要的，C++中“--”已经用来表示整数的递减操作，如for语句中常用“--i”表示循环次数递减。如果使用Ada中的约定，会导致“--”符号具有二义性，而对于编译器，二义性语言将不知道被用于哪种用途。