2016“亚马逊•种子杯”

初赛试题

2016/10/23

*本次最终解释权由大赛组委会所有
*更多详情请访问大赛官网 http://dian.hust.edu.cn/seedpk/

1 题目概述

在现实中，遇到不可直接执行的代码时，往往需要在虚拟环境下分析其执行过程以及代码执行次数。

本题中，我们要求选手协助实现一个简单的 C 语言代码分析器，用于分析给定 C 语言代码的执行顺序并输出行号。

2 基本要求

1） 编程语言：C/C++（附加库只允许用标准库）
编译器：CL(VS 自带的编译器)/GCC
测试平台：
Linux：注明编译器及版本
Windows：7/8/10 32/64bit 任意一种即可，请在文档中注明
2） 所完成的功能一律按照需求说明中的标准实现。
3） 不需要考虑错误用例
4） 对全部由大一学生组成的参赛队会酌情加分。
5） 发现抄袭、作弊现象，直接取消比赛资格。
6） 最终解释权归赛事裁判组所有。

3 作品提交

3.1 提交规范

作品请以附件形式发送至大赛官方邮箱：seedcup@dian.org.cn。邮件主题命名方式：**初赛提交-队名**。要求将完整的工程和工程文档打包提交，压缩包命名规则为：初赛提交-队名.zip，提交的目录格式如下：

—[初赛提交-队名]
—Src
—Bin
—Doc

目录说明如下：Src 文件夹放置源代码及工程文件，Bin 文件夹放置最后生成的可执行文件，Doc 文件夹放置说明文档。

说明文档需要包含：

1） 程序的编译运行环境说明，包括使用的编译器类型及版本。
2） 程序的设计结构及各模块的功能说明。
3） 对题目功能要求的完成情况，可以列举自己程序中觉得设计良好的部分，并说明为什么好。
4） 你认为必要的附加信息，以方便评委了解你的想法。

3.2 提交时间

比赛终止时间为 2016/11/1 晚上 10:00，请各位选手注意提交时间（以邮件发送时间为准），我们不接受晚于终止时间提交的作品。

4 评分说明

我们会依据选手提交上的文档和代码编译可执行文件，并将其作为主要评分标准，选手提交的可执行文件仅作为参考。另外，请特别注意输出的格式（参见 *5.需求说明*），**输出格式错误有可能导致该功能点不得分**。

项目	分值	评分标准
文档	10’	逻辑和结构清晰，描述和图例详细，不得超过 10 页。
代码规范	10’	编码的规范程度和代码的设计结构，要求核心代码都能给出注释。
功能点	80’	功能点的实现情况，依据下面的详细需求包括： 顺序结构(10) 分支结构(30) 循环结构(40)
附加项		参见 *2.基本要求*中的第 4 项，我们将会视具体情况酌情给分。

5 需求说明

5.1. 需求简介

- **使用 C/C++实现一个程序，完成对一段 C 语言代码的分析，并依次输出这段代码执行的顺序（行号）。**

5.2. 输入输出规范

- **程序命名：**最终的可执行文件命名为 SeedCup2016.exe。
- **输入：**同目录下存在命名为 input.txt 的输入文本（一段 C 语言代码）。SeedCup2016.exe 需要读取同目录下的 input.txt 输入文本，并对该 C 语言代码进行分析。
- **输出：**SeedCup2016.exe 将上述分析结果输出到同目录下 output.txt 文件（output.txt 文件需要自行生成）。
- **注：所有输入输出文档以 UTF-8 无 BOM 格式编码，换行符为”\r\n”。**

5.3. 详细需求

◆ 顺序结构（10’）

该题目中的顺序结构中的关键字为 **int**。语句类型包含**声明语句，初始化语句，赋值语句，输出语句，空语句**五种。该部分需要识别所有需要执行的代码行，并正确的输出程序运行的顺序（行号）。

顺序结构格式介绍:

```
/* 声明语句 */
int 变量名 1,变量名 2;

/* 初始化语句 */
int 变量名= 值或表达式;

/* 赋值语句 */
变量名 = 值或表达式;

/* 输出语句 */
printf(“xxx is %d\n”, 变量名或表达式);

/* 空语句 */
  ;
```

说明：

1. 表达式只包含整数、变量名及其+，–，*， / ,++（自增符号）,--（自减符号）运算符号，一个表达式中运算符号不多于 10 个。
2. 顺序结构的测试用例可能包含空行，注释。这些均为不运行的内容，不需要输出其行号。
3. 注释包含 /* comment */ 与 // comment 两种形式。
4. 单独的声明语句，空语句，注释不需要输出其行号；
5. 除 printf 函数外，不会调用其余系统函数或自定义函数。
6. 参见 5.4 全局说明。
7. 除上述说明外，其余未说明的合法用例均有可能出现。

用例：

输入文件内容：

```
int pen;
int apple;
int apple_pen;

pen = 1;
```

```
apple = 2;
apple_pen = apple + pen;

int pineapple = 4;
int pineapple_pen = pineapple + pen;

printf("I have a pen. %d\n", pen);
printf("I have an apple. %d\n", apple);
/* PPAP */
printf("Pen-pineapple-apple pen.\n");
```

输出文件内容：

```
5 6 7 9 10 12 13 15
```

◆ 分支结构（30’）

该题目中的分支结构中的关键字为 **if** 及 **else**。使用方法与 C 语言中的用法相同。该部分需要你的程序识别程序分支结构，并正确地输出程序运行的顺序（行号）。

分支结构格式介绍（参见 C 语言 if-else 用法）：

```
//举例 1
if(表达式 1) {
    语句 1；
    语句 2；
}
else if(表达式 2){
    语句 3；
    语句 4；
}
else {
    语句 5；
}
//举例 2
if(表达式 1)
    语句 1；
else
    语句 2；
```

说明：

1. 分支结构中规定的条件语句的表达式格式为：(**变量名 比较符号 值**)，其中比较符号为：>、<、==、!=、>=、<=（例如：apple>8）。
2. 所有的输入中，分支结构的表达式中不会出现非上述比较符号。
3. 所有的输入中，条件语句括号中不会出现多个表达式运算。

4. 所有的输入中，分支结构的嵌套不会多于三层（<=3）。
5. 单独的 else 和花括号均不属于执行语句，不需要输出其行号。
6. 所有的顺序结构语句均遵循上述《顺序结构》中的规则。
7. 参见 5.4 全局说明。
8. 除上述说明外，其余未说明的合法用例均有可能出现。

用例：

输入文件内容：

```
int apple;
int orange;
orange = 2;
apple = 1;
apple = apple + 1;
orange = apple + orange;
if (orange == 2){
    printf("%d", apple);
}
else if(orange == 3){
    printf("%d", apple);
}
else{
    printf("%d", orange);
}
printf("%d", orange);
```

输出文件内容：

```
3 4 5 6 7 10 14 16
```

◆ 循环结构（40’）

该题目中的循环结构中的关键字为 **for、break、while、do**。使用方法与 C 语言中的用法相同。该部分需要你的程序识别程序循环结构，并正确地输出程序运行的顺序（行号）。

循环结构格式介绍：

```
for(表达式 1; 表达式 2; 表达式 3) {
    语句 1；
    语句 2；
    …
}

while (表达式 1){
    语句 1；
    语句 2；
```

```
    …
}

do{
    语句 1；
    语句 2；
    …
} while (表达式 1)
```

说明：

1. 本题中比较符号仅含有：>、<、==、!=、>=、<=
2. 运算符号仅含有为：+，-，*，/,++（自增符号）,--(自减符号)
3. 本题中规定的表达式格式有以下几种形式：
   例如：apple > 8, apple++, apple--, apple = apple + 2
4. 本题中可能包含顺序结构和分支结构。
5. break 语句存在于循环结构中，作用为中断并跳出当前循环，需要输出 break 语句所在的行号。
6. 所有的输入中，循环结构的表达式中不会出现非上述比较符号和运算符号。
7. 所有的输入中，循环结构的嵌套不会多于两层（<=2）。
8. 单独的 do 和花括号均不属于执行语句，不需要输出其行号。
9. 所有的顺序结构语句及分支结构语句均遵循上述《顺序结构》、《分支结构》中的规则。
10. 参见 5.4 全局说明。
11. 除上述说明外，其余未说明的合法用例均有可能出现。

用例：

输入文件内容：

```
int i = 2;
int j = 0;
for ( ; i > 0; i--){
    printf("hello world!");
    while (j <= 2){
        printf("hello world!");
        j++;
    }
}
```

输出文件内容：

```
1 2 3 4 5 6 7 5 6 7 5 6 7 5 3 4 5 3
```

5.4. 全局说明

- 对于所有的输入文件，出现的变量名均不超过 50 个。

- 对于所有的输入文件，每行不超过 255 个字符。
- 对于所有的输入文件，每行不超过 5 条语句。
- 对于所有的输入文件，40%的测试用例不超过 50 行，100%的测试用例不超过 300 行。
- 对于所有的输入文件，花括号会与分支语句和循环语句共同出现，不会单独出现，另外花括号中具有变量作用域。参见 C 语言花括号的用法。
- 不可使用编译器编译执行输入文件的方式完成题目，否则视为作弊。
- 对文档中描述不清楚的地方请参见 2016“亚马逊·种子杯”官方交流群群公告。

5.5. 样例说明

- 样例提供在题目压缩包中，三个文件夹命名为 test1，test2，test3。每个文件夹中存在一个输入文件 input.txt 及一个输出文件 output.txt。
- **取 test2 中的样例作为解析。**

输入文件内容：

```
int apple;
int orange;
orange = 2;
apple = 1;
apple = apple + 1;
orange = apple + orange;
if (orange == 2){
	printf("%d", apple);
}
else if(orange == 3){
	printf("%d", apple);
}
else{
	printf("%d", orange);
}
printf("%d", orange);
```

输出文件内容：

```
3 4 5 6 7 10 14 16
```

SeedCup2016.exe 读取同目录下 input.txt 的内容，并分析过程如下：

前 3-6 行均为顺序执行语句，第 7 行为分支语句，此时判断 orange 是否等于 2，条件为假，因此跳到第 10 行判断 orange 是否等于 3，条件仍为假，跳到 else 花括号中，直接执行第 14 行语句，执行完毕后跳到分支语句外第 16 行执行。因此执行顺序为 3 4 5 6 7 10 14 16

5.6. 测试及评分说明

对于每个功能点，我们会设置不同难度的测试用例，根据程序通过的测试用例给分。

6 参考

种子杯官网：http://dian.hust.edu.cn/seedpk/?cat=15
种子杯交流群：590489892
对于题目的解答及内容的修正将会第一时间在种子杯官网以及种子杯交流群中发出，请关注。