Puntos de secuencia en C | Serie 1

Nivel de dificultad: Medio

Última actualización: 31 de julio de 2018

Traducido por Juan C. Giraldo para el curso de DISPRO de la siguiente fuente de información:

https://www.geeksforgeeks.org/sequence-points-in-c-set-1/

En esta publicación, se intenta cubrir preguntas ambiguas como la siguiente:

¿Cuál el resultado de los siguientes programas?

Programa 1. Orden de impresión de cadenas de caracteres en una expresión de suma.

```
1       /* sequence_point_1.c ******************************************************/
2
3       #include <stdio.h>
4       int f1() { printf ("Geeks"); return 1;}
5       int f2() { printf ("forGeeks"); return 1;}
6
7       int main()
8       {
9          int p = f1() + f2();
10         return 0;
11      }
```

Programa 2. Orden de cómputo aritmético en una expresión de suma.

```
1       /* sequence_point_2.c ******************************************************/
2
3       #include <stdio.h>
4       int x = 20;
5       int f1() { x = x+10; return x;}
6       int f2() { x = x-5;  return x;}
7
8       int main()
9       {
10         int p = f1() + f2();
11         printf ("p = %d", p);
12         return 0;
13      }
```

Programa 3. Resultado de evaluación de expresión con efectos laterales.

```
1       /* sequence_point_3.c ********************************************************/
2
3       #include <stdio.h>
4
5       int main()
6       {
7          int i = 8;
8          int p = i++*i++;
9          printf("%d\n", p);
10      }
```

La salida de todos los programas anteriores no está definida o no está especificada. La salida puede ser diferente para diferentes compiladores y diferentes máquinas. Es como preguntar el valor de una variable automática indefinida.

La razón del comportamiento indefinido en el Programa 1 es que el operador "+" no tiene un orden de evaluación estándar definido para sus operandos. Primero se puede ejecutar f1() ó f2(). Por tanto, la salida puede ser "GeeksforGeeks" o "forGeeksGeeks".

Similar al operador '+', la mayoría de los otros operadores similares como '-', '/', '*', bitwise AND &, bitwise OR |, ... etc, no tienen un orden definido estándar para la evaluación de sus operandos.

La evaluación de una expresión también puede producir efectos laterales. Por ejemplo, en el programa 2 anterior, los valores finales de p son ambigüos. Dependiendo del orden de evaluación de la expresión, si f1() se ejecuta primero, el valor de p será 55, de lo contrario 40.

La salida del programa 3 tampoco está definida. Puede ser 64, 72 o puede ser otra cosa. La subexpresión i++ causa un efecto secundario, modifica el valor de i, lo que conduce a un comportamiento indefinido ya que i también se hace referencia en otra parte de la misma expresión.

A diferencia de los casos anteriores, en ciertos puntos específicos de la secuencia de ejecución denominados puntos de secuencia, se garantiza que todos los efectos secundarios de las evaluaciones anteriores estarán completos. Un punto de secuencia define cualquier punto en la ejecución de un programa informático en el que se garantiza que se habrán realizado todos los efectos secundarios de evaluaciones anteriores y que aún no se han realizado efectos secundarios de evaluaciones posteriores. A continuación se muestran los puntos de secuencia enumerados en el estándar C:

- El final del primer operando de los siguientes operadores:

a) AND && lógico

b) OR lógico ||

c) condicional ?

d) coma,

Por ejemplo, se garantiza que la salida de los siguientes programas será "GeeksforGeeks" en todos los compiladores o máquinas.

Programa 4. Ejemplo de ejecución de una expresión con punto de secuencia garantizada con operador AND lógico &&.

```
1       /* guarantee_sequence_point_1.c ********************************************/
2
3       #include <stdio.h>
4       int f1() { printf ("Geeks"); return 1;}
5       int f2() { printf ("forGeeks"); return 1;}
6
7       int main()
8       {
9          // Since && defines a sequence point after first operand, it is
10         // guaranteed that f1() is completed first.
11         int p = f1() && f2();
12         return 0;
13      }
```

Programa 5. Ejemplo de ejecución de una expresión con punto de secuencia garantizada con operador de secuencia “,”.

```
1       /* guarantee_sequence_point_2.c ********************************************/
2
3       #include <stdio.h>
4       int f1() { printf ("Geeks"); return 1;}
5       int f2() { printf ("forGeeks"); return 1;}
6
7       int main()
8       {
9          // Since comma operator defines a sequence point after first operand, it is
10         // guaranteed that f1() is completed first.
11        int p = (f1(), f2());
12        return 0;
13      }
```

Programa 6. Ejemplo de ejecución de una expresión con punto de secuencia garantizada con operador condicional "?:".

```
1       /* guarantee_sequence_point_3.c ********************************************/
2
3       #include <stdio.h>
4       int f1() { printf ("Geeks"); return 1;}
5       int f2() { printf ("forGeeks"); return 1;}
6
7       int main()
8       {
9          // Since ? operator defines a sequence point after first operand, it is
10         // guaranteed that f1() is completed first.
11        int p = f1() ? f2(): 3;
12        return 0;
13      }
```

- El final de una expresión completa. Esta categoría incluye las siguientes expresiones

a) Cualquier declaración completa termina con un punto y coma como "a = b";

b) Instrucciones de retorno.

c) Las expresiones de control de las instrucciones if, switch, while o do-while.

d) Las tres expresiones en una declaración for.

La lista anterior de puntos de secuencia es parcial. Se cubrirán todos los puntos de secuencia restantes en una próxima publicación sobre Sequence Point.

Referencias:

http://en.wikipedia.org/wiki/Sequence_point

http://c-faq.com/expr/seqpoints.html

http://msdn.microsoft.com/en-us/library/d45c7a5d(v=vs.110).aspx

http://www.open-std.org/jtc1/sc22/wg14/www/docs/n925.htm