Expression evaluation

Martin Kozeny
CSCI 4501: Programming Language Structure
Spring 2011 University of New Orleans

February 23, 2011

1 Expression evaluation in Java

According to chapter 15, part 7 from [1], is Java's evaluation order from left to right, so the result of calling `a*f(1)+a` in code below will be $10 * 1 * 11 + 11 = 121$.

```java
public class Main {
  static int a = 10;
  public static void main(String[] args) {
    System.out.println(String.valueOf(a*f(1)+a));
  }
  public static int f(int x){
    a = a + 1;
    return x*a;
  }
}
```

2 Expression evaluation in C

In C is the situation with evaluating expressions different. According to [2], C uses function first evaluation so the result of calling `a*f(1)+a` in code below will be $11 * 1 * 11 + 11 = 132$. In C exists so called sequence points. According to [3], sequence points are checkpoints which it is guaranteed that all side effects of previous evaluations will have been performed, and no side effects from subsequent evaluations have yet been performed. In our source code there are two sequence points: function-entry after the evaluation of all the function's arguments and before execution of anything the function's body and at function-exit after a returned value has been copied and before the calling function resumes execution. Function-entry sequence point guarantees, that input parameter a is evaluated (in our case parameter `z`) and second function-exit s. p. guarantees incrementation `a` and multiplying by `x` before leaving function.

```c
int f(int x);
int a = 10;
int main(){
  printf("%i",a*f(1)+a);
  return 0;
}
int f(int x){
  a = a + 1;
  return x*a;
}
```

References

[1] Java Language specification, http://java.sun.com/docs/books/jls/.

[2] C Language Reference Manual, April 1999.

[3] Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Sequence_point