Lecture 06: Control Structures

Control structures are the building blocks for constructing program logic. All programming languages therefore need them. This lecture discusses the topic under the following subheadings:

- Introduction
- Selection Structures
- Iteration Structures
- Recursion Structures

6.1 Introduction

In all programming languages, but particularly imperative (procedural) and OO languages, control structures are used to establish program logic. The fundamental control structures are as follows:

- Sequential structures (self-explanatory and needs no further clarification)
- Selection structures
- Iteration structures
- Recursion (this will be discussed in the next lecture)

6.2 Selection Structures

Selection structures facilitate decision based or certain pre-conditions. Two structures are common: the **if-structure** and the **case-structure**. They are represented in figures 6.1 and 6.2. Most programming languages support these structures, but the syntax tends to vary slightly. The syntax for these structures remains the same for C-based languages.

Figure 6.1: Generic Representation of If-Structure

```
If (<condition>)
    <Statement(s)>
End-If
[Else
    <Statement(s)>
End-Else]
```

Figure 6.2: Generic Representation of Case-Structure

```
Case <Variable> | <Expression> is
<Value_1>: <Statement(s)>
<Value_2>: <Statement(s)>
    …..
<Value_N>: <Statement(s)>
Otherwise: <Statement(s)>
End-Case
```

In figure 6.1, the term *condition* is used to represent a Boolean expression (as discussed in the previous lecture) — something that evaluates to true or false. You will see this term used in the upcoming section as well. The **if-structure** is useful an expression may result in a limited number of outcomes, and each requires a different action. The **case-structure** is applicable when the expression in question may result in a more elaborate list of outcomes than the if-structure, and each outcome requires a different action.

Exercise: Conduct a Web search to identify the syntax for the **if-Structure** and the **case-Structure** in languages such as Pascal, C++, Java, C#, and Python. In each case, provide the BNF representation of the required syntax as well as actual code examples.

6.3 Iteration Structures

All imperative and OO languages have iteration structures. The commonly implemented ones are the **while-Structure**, the **repeat-until-structure**, and the **for-structure**. The syntax varies from language to language, but is identical in all C-based languages.

Figure 6.2: Generic Representation of Iteration Structures

The **While- Structure** has the following form:

```
While (<Condition>) Do the following:
        <Statement(s)>
  …
 End-while;
```

The **Repeat-Until-Structure** has the following form:

```
Do the following:
    <Statement(s)>
    ….
Until (<Condition>)
```

The **For- Structure** has the following form:

```
For <Variable>|<Expression>  : = <Value1> To <Value2> With increments of <Value3>,  Do the following:
<Statement(s)>
…
End-For
```

Then **while-structure** is applicable when it is desirable to test a Boolean condition to determine whether a certain set of action(s) is required as long as that condition holds true. The **repeat-until-structure** is similar to the while-structure, except that the test is administered after at least one iteration of the related set of action(s). The **for-structure** is ideally suited for scenarios where on each iteration, an expression increments or decrements towards a target value. However, in several languages, the syntax is flexible enough to replace a **while-structure** of a **repeat-until-structure**.

Exercise: Conduct a Web search to identify the syntax for the iteration structures in languages such as Pascal, C++, Java, C#, and Python. In each case, provide the BNF representation of the required syntax as well as actual code examples.

6.4 Summary and Concluding Remarks

Here are the salient points of this brief lecture:

- High-level languages (HLLs) rely on control structures to manage the logic and flow of the programs developed in these languages.
- The main areas of control are sequential structures, selection structures, iteration structures, and recursion structures.
- Sequential structures refer to the order in which instructions are given. For the most part, this category is self-explanatory (you do not print a file before opening it; nor do you attempt to return a calculated value before executing the calculation).
- Selection structures are of two types: the **if-structure** and the **case-structure**.
- Iteration structures are of three varieties: the **while-structure**, the **repeat-until-structure**, and the **for-structure**.

Recursion may be regarded as a control structure as well. However, since this principle is intricately related to subprograms, we will look at these two related topics in the next lecture.

6.5 Recommended Readings

[Pratt & Zelkowitz 2001] Pratt, Terrence W. and Marvin V. Zelkowits. 2001. *Programming Languages: Design and Implementation* 4th Edition. Upper Saddle River, NJ: Prentice Hall. See chapter 8.

[Sebesta 2012] Sebesta, Robert W. 2012. *Concepts of Programming Languages* 10th Edition. Colorado Springs, Colorado: Pearson. See chapter 8.