The control statements of the imperative languages occur in several categories: selection, multiple selection, iterative, and unconditional branching.

The switch statement of the C-based languages is representative of multiple selection statements. The C# version eliminates the reliability problem of its predecessors by disallowing the implicit fall through from a selected segment to the following selectable segment.

A large number of different loop statements have been invented for high-level languages, starting with Fortran’s counting to. Ada’s for statement is, in terms of complexity, the opposite. It elegantly implements only the most commonly needed counting loop forms. C's for statement is the most flexible iteration construct, although its flexibility leads to some reliability problems.

The C-based languages have exit statements for their loops; these statements take the place of one of the most common uses of goto statements.

Data-based iterators are loop constructs for processing data structures, such as linked lists, hashes, and trees. The for statement of the C-based languages allows the user to create iterators for user-defined data. The foreach statement of Perl and C# is a predefined iterator for standard data structures. In the contemporary object-oriented languages, iterators for collections are specified with standard interfaces, which are implemented by the designers of the collections.

The unconditional branch, or goto, has been part of most imperative languages. Its problems have been widely discussed and debated. The current consensus is that it should remain in most languages but that its dangers should be minimized through programing discipline.

Dijkstra’s guarded commands are alternative control constructs with positive theoretical characteristics. Although they have not been adopted as the control constructs of a language, part of the semantics appear in the concurrency mechanisms of CSP and Ada and the function definitions of Haskell.

1. What is the definition of control structure?
2. What is the definition of block?
3. What are the design issues for selection structures?
4. What are the common solutions to the nesting problem for two-way selectors?
5. What are the design issues for multiple selection statements?
6. What is unusual about C’s multiple selection statement? What design trade-off was made in this design?
7. Explain how C#’s switch statement is safer than that of Java.
8. What are the design issues for counter-controlled loop statements?
9. What is a pretest loop statement? What is a posttest loop statement?
10. What is the difference between the for statement of C++ and that of Java?
11. What are the design issues for logically controlled loop statements?
12. What is the main reason user-located loop control statements were invented?
13. What advantage does C#’s break statement have over C’s break statement?
14. What are the differences between the break statement of C++ and that of Java?
15. What is a user-defined iteration control?
16. What common programming language borrows part of its design from Dijkstra’s guarded commands?
1. Rewrite the following pseudocode segment using a loop structure in the specified languages:
   \[ k \leftarrow \left( j + 13 \right) / 27 \]
   loop:
   if \( k > 10 \) then goto out
   \( k \leftarrow k + 1 \)
   \( i \leftarrow 3 \times k - 1 \)
   goto loop
out: ...
   a. Fortran 95
   b. Ada
   c. C, C++, Java, or C#

Assume all variables are integer type. Discuss which language, for this code, has the best writability, the best readability, and the best combination of the two.

2. Redo Problem 1, except this time make all the variables and constants floating-point type, and change the statement
   \[ k \leftarrow k + 1 \]
   to
   \[ k \leftarrow k + 1.2 \]

3. Rewrite the following code segment using a multiple selection statement in the following languages:
   if \( (k = 1) \) or \( (k = 2) \) then \( j := 2 \times k - 1 \)
   if \( (k = 3) \) or \( (k = 5) \) then \( j := 3 \times k + 1 \)
   if \( (k = 4) \) then \( j := 4 \times k - 1 \)
   if \( (k = 6) \) or \( (k = 7) \) or \( (k = 8) \) then \( j := k - 2 \)
   a. Fortran 95 (you'll have to look this one up)
   b. Ada
   c. C, C++, Java, or C#

Assume all variables are integer type. Discuss the relative merits of the use of these languages for this particular code.

4. Consider the following C program segment. Rewrite it using no gotos or breaks.
   \( j = -3 \);
   for \( (i = 0; i < 3; i++) \) {
     switch \( (j + 2) \) {
     case 3:
     case 2: \( j--; \) break;
     case 0: \( j += 2; \) break;
     default: \( j = 0; \)
     }  
     if \( (j > 0) \) break;
    \( j = 3 - i \)
   }

5. In a letter to the editor of CACM, Rubin (1987) uses the following code segment as evidence that the readability of some code with gotos is better than the equivalent code without gotos. This code finds the first row of an \( n \times n \) integer matrix named \( x \) that has nothing but zero values.
   for \( (i = 1; i <= n; i++) \) {
     for \( (j = 1; j <= n; j++) \)
     if \( (x[i][j] != 0) \)
     goto reject;
     println ('First all-zero row is:', i);
     break;
   }
   reject:

Rewrite this code without gotos in one of the following languages: C, C++, Java, C# or Ada. Compare the readability of your code to that of the code above.

6. Consider the following programming problem: The values of three integer variables, first, second, and third, must be placed in the three variables \( \text{max}, \text{mid}, \text{and} \text{min} \), with the obvious meanings, without using arrays or user-defined or predefined subprograms. Write two solutions to this problem, one that uses nested selections and one that does not. Compare the complexity and expected reliability of the two.