2.19 A New Language for the New Millennium: C#  

C#, along with the new development platform, .NET, was announced by Microsoft in 2000. In January, 2002, production versions of both were released.

2.19.1 Design Process  

C# is based on C++ and Java, but includes some ideas from Delphi and Visual BASIC. Its lead designer, Anders Hejlsberg, also designed Turbo Pascal and Delphi, which explains the Delphi parts of the heritage of C#.

The purpose of C# is to provide a language for component-based software development, specifically for such development in the .NET Framework. In this environment, components from a variety of languages can be easily combined to form systems. All of the .NET languages, which include C#, Visual BASIC.NET, Managed C++, J#, .NET, and JScript.NET use the Common Type System (CTS). The CTS provides a common class library. All types in all five .NET languages inherit from a single class root, System.Object. Compilers that conform to the CTS specification create objects that can be combined into software systems. All five .NET languages are compiled into the same intermediate form, Intermediate Language (IL). Unlike Java, however, the IL is never interpreted. A Just-In-Time compiler is used to translate IL into machine code before it is executed.

2.19.2 Language Overview  

Many believe that one of Java's most important advances over C++ lies in the fact that it excludes some of C++'s features. For example, C++ supports multiple inheritance, pointers, structs, enum types, operator overloading, and a goto statement, but Java includes none of these. The designers of C# obviously disagreed with these wholesale removals of features, for all of these except multiple inheritance have been brought back in the new language.

To the credit of C#s designers, however, in several cases, the C# version of a C++ feature has been improved. For example, the enum types of C# are safer than those of C++, because they are never implicitly converted to integers. This allows them to be more type safe. The struct type was changed significantly, resulting in a truly useful construct, whereas in C++ it serves virtually no purpose. In C#, a struct is a lightweight class that does not support inheritance or subclassing. However, C# structs can implement interfaces and have constructors. They are value types, which means they are allocated on the run-time stack. All C# primitive types are implemented as structs. C# takes a stab at improving the switch statement that is used in C, C++, and Java. In those languages, there is no implicit branch at the end of the selectable segments of code, which can have innumerable program-

2The .NET development system is briefly discussed in Chapter 1.

ming errors. The problem stems from the fact that the programmer must remember to put a break statement in every selectable segment from which flow is to exit the structure, rather than going to the next selectable segment. Unfortunately, this makes the default action one that is rarely used in practice. The C# switch statement adds a single new requirement to the statement: Every nonempty case segment must end with an unconditional branch statement. So, if you want control to flow from one case segment to the next, it is necessary to use a goto statement to branch to the next case segment.

Although C++ includes function pointers, they share the lack of safety that is inherent in C++'s pointers to variables. C# includes a new type, delegates, which are both object-oriented and type-safe method references. Delegates are used for implementing event handlers and callback functions. Callbacks are implemented in Java with interfaces; in C++, method pointers are used. In C#, methods can take a variable number of parameters, as long as they are all the same type. This is specified by the use of a formal parameter of array type, preceded by the params reserved word.

Both C++ and Java use two distinct typing systems, one for primitives and one for objects. In addition to being confusing, this leads to frequent conflicts between the two systems, for example, to put a primitive value into a collection that stores objects. C# makes the conversion between values of the two typing systems partially implicit through the implicit boxing and unboxing operations, which are discussed in detail in Chapter 12.

Among the other features of C# are rectangular arrays, which are not supported in most programming languages, and a foreach statement, which is an iterator for arrays and collection objects. A similar foreach statement is found in Perl and PHP. Also, C# includes properties, which are an alternative to public data members. Properties are specified as data members with get and set methods, which are implicitly called when references and assignments are made to the associated data members.

2.19.3 Evaluation  

C# was meant to be an advance over both C++ and Java as a general purpose programming language. Although it can be argued that some of its features are a step backwards, C# clearly includes some constructs that move it beyond its predecessors.

The primary intended application of C# is as the main language in the .NET environment. It is far too early to say with any certainty that C# will succeed in attracting a large number of users. However, given the huge effort being made by Microsoft to sell .NET, it appears that its chances at becoming widely used are excellent. Furthermore, some of its features will surely become adopted by programming languages of the near-term future.

The ancestry of C# is shown in Figure 2.15.
SUMMARY

We have investigated the development and the development environments of a number of the most important programming languages. This chapter should have given the reader a good perspective on current issues in language design. We hope to have set the stage for an in-depth discussion of the important features of contemporary languages.

BIBLIOGRAPHIC NOTES

Perhaps the most important source of historical information about the development of programming languages is *History of Programming Languages*, edited by Richard Wexelblat (Wexelblat, 1981). It contains the developmental background and environment of 13 important programming languages, as told by the designers themselves. A similar work resulted from a second "history" conference, this time published as a special issue of *ACM SIGPLAN Notices* (ACM, 1993a). In this work, the history and evolution of 13 more programming languages are discussed.

The paper "Early Development of Programming Languages" (Knuth and Pardo, 1977), which is part of the *Encyclopedia of Computer Science and Technology*, is an excellent 85-page work that details the development of languages up to and including Fortran. The paper includes example programs to demonstrate the features of many of those languages.

Another book of great interest is *Programming Languages: History and Fundamentals*, by Jean Sammet (Sammet, 1969). It is a 785-page work filled with details of 80 programming languages of the 1950s and 1960s. Sammet has also published several updates to her book, such as Sammet (1976).

REVIEW QUESTIONS

1. In what year was PL/1 designed? In what year was that design published?
2. What two common data structures were included in PL/1?
3. How were the pseudocodes of the early 1950s implemented?
4. Speedcoding was invented to overcome two significant shortcomings of the computer hardware of the early 1950s. What were these two?
5. Why was the slowness of interpretation of programs acceptable in the early 1950s?
6. What hardware capability that first appeared in IBM 704 computer strongly affected the evolution of programming languages? Explain why.
7. In what year was the Fortran design project begun?
8. What was the primary application area of computers at the time Fortran was designed?
9. What was the source of all of the control flow statements of Fortran I?
10. What was the most significant feature added to Fortran I to get Fortran II?
11. What control flow statements were added to Fortran IV to get Fortran VII?
12. Which version of Fortran was the first to have any sort of dynamic variables?
13. Which version of Fortran was the first to have character string handling?
14. Why were linguists interested in artificial intelligence in the late 1950s?
15. Where was LISP developed? By whom?
16. In what way are Scheme and COMMON LISP opposites of each other?
17. What dialect of LISP is used for introductory programming courses at some universities?
18. What two professional organizations together designed ALGOL 60?
19. In what version of ALGOL did block structure appear?
20. What missing language element of ALGOL 60 damaged its chances for widespread use?
21. What language was designed to describe the syntax of ALGOL 60?
22. On what language was COBOL based?
23. In what year did the COBOL design process begin?
24. What data structure appeared in COBOL that originated with Plankalkül?
25. What organization was most responsible for the early success of COBOL (in terms of extent of use)?
26. What user group was the target of the first version of BASIC?
27. Why was BASIC an important language in the early 1980s?
28. PL/I was designed to replace what two languages?
29. For what new line of computers was PL/I designed?
30. What features of SIMULA 67 are now important parts of some object-oriented languages?
31. What innovation of data structuring was introduced in ALGOL 68 but is often credited to Pascal?
32. What design criterion was used extensively in ALGOL 68?
33. What language introduced the case statement?
34. What operators in C were modeled on similar operators in ALGOL 68?
35. What are two characteristics of C that make it less safe than Pascal?
36. What is a nonprocedural language?
37. What are the two kinds of statements that populate a Prolog database?
38. What is the primary application area for which Ada was designed?
39. What are the concurrent program units of Ada called?
40. What Ada construct provides support for abstract data types?
41. What populates the Smalltalk world?
42. What three concepts are the basis for object-oriented programming?
43. Why does C++ include the features of C that are known to be unsafe?
44. What do the Ada and COBOL languages have in common?
45. What was the first application for Java?
46. What characteristic of Java is most evident in JavaScript?
47. How does the typing system of PHP and JavaScript differ from that of Java?
48. What array structure is included in C#, but not in C, C++, or Java?
49. What feature of Delphi’s classes is included in C#?
50. What deficiency of the switch statement of C is addressed with the changes made by C# to that statement?

**Problem Set**

1. What features of Plankalkül do you think would have had the greatest influence on Fortran 0 if the Fortran designers had been familiar with Plankalkül?
2. Determine the capabilities of Backus’s 701 Speedcoding system, and compare them with those of a contemporary programmable hand calculator.
3. Write a short history of the A-0, A-1, and A-2 systems designed by Grace Hopper and her associates.
4. As a research project, compare the facilities of Fortran 0 with those of the Lancing and Zierler system.
5. Which of the three original goals of the ALGOL design committee, in your opinion, was most difficult to achieve at that time?
6. Make an educated guess as to the most common syntax error in LISP programs.
7. LISP began as a pure functional language but gradually acquired more and more imperative features. Why?
8. Describe in detail the three most important reasons, in your opinion, why ALGOL 60 did not become a very widely used language.
9. Why, in your opinion, did COBOL allow long identifiers when Fortran and ALGOL did not?
10. Outline the major motivation of IBM in developing PL/I.
11. Was IBM’s major motivation for developing PL/I correct, given the history of computers and language developments since 1964?
12. Describe, in your own words, the concept of orthogonality in programming language design.
13. What is the primary reason why PL/I became more widely used than ALGOL 68?
14. What are the arguments both for and against the idea of a typeless language?
15. Are there any logic programming languages, other than Prolog?
16. What is your opinion of the argument that languages that are too complex are too dangerous to use and we should therefore keep all languages small and simple?
17. Do you think language design by committee is a good idea? Support your opinion.
18. Languages continually evolve. What sort of restrictions do you think are appropriate for changes in programming languages? Compare your answers with the evolution of Fortran.
19. Build a table identifying all of the major language developments, together with when they occurred, in what language they first appeared, and the identities of the developers.
20. There have been some public interchanges between Microsoft and Sun concerning the design of Microsoft’s J++ and C# and Sun’s Java. Read some of these documents, which are available on their respective Websites, and write an analysis of the disagreements concerning delegates.