Course Code 007313 (Spring 2019)

프로그래밍언어의 개념

Concepts of Programming Language

(Lecture 04 : Chapter 2- Evolution of Major Programming Languages)

Prof. A. P. Shrestha, Ph.D.

Dept. of Computer Science and Engineering, Sejong University

Last Class

Chapter 2 - Evolution of Major Programming Languages

- Language genealogy and early programming languages
- Fortran
- Lisp

Today

Chapter 2- Evolution of Major Programming Languages

- (continued)

Next class

- Chapter 3- Describing Syntax and Semantics

4.1.1 The First Step Towards Sophistication: ALGOL

- Environment of development
  - FORTRAN had (barely) arrived for IBM 70x
  - Many other languages were being developed, all for specific machines
  - No portable language; all were machine-dependent
  - No universal language for communicating algorithms

- ALGOL was the result of efforts to design a universal language

- ALGOL: universal, international, machine-independent (imperative) language for expressing scientific algorithms
  - Eventually, 3 major designs: ALGOL 58, 60, and 68
  - Developed by increasingly large international committees

ALGOL: Algorithmic Language

4.1.2 ALGOL – Phrase-Level Control

- Early languages used *label-oriented control*:

```
      GO TO 27
30    IF (A-B) 5,6,7
```

- ALGOL supports sufficient *phrase-level control*, such as **if**, **while, switch**, **for**, **until**
  →*structured programming*

- Programming style:
  - Programs consist of blocks of code: blocks → functions → files → directories
  - Easy to develop, read, maintain; make fewer errors

4.1.3 Example of Block Scope

C language: Pseudocode

```
int main()
{
        {
                int x = 10, y = 20;
                {
                        print(x, y);
                        {
                                int y = 40;
                                x++;
                                y++;
                                print(x, y);
                        }
                        print(x, y);
                }
        }
}
```

Q1) Is it correct?

Q2) Predict the output?

4.1.4 Influences of ALGOL

- Virtually all languages after 1958 used ideas pioneered by the Algol designs:
  - Compound statements: **begin** statements **end**
  - BNF definition of syntax
  - Local variables with block scope
  - Static typing with explicit type declarations
  - Nested if-then-else
  - Call by value and Call by name
  - Recursive subroutines and conditional expressions (ex Lisp)
  - Dynamic arrays
  - User-defined operators etc.

4.1.5 A Simple Program in ALGOL 60

```
comment An Algol 60 sorting program;
procedure Sort (A, N)
        value N;
        integer N; real array A;
   begin
        real X;
        integer i, j;
        for i := 2 until N do begin
          X := A[i];
          for j := i-1 step -1 until 1 do
           if X >= A[j] then begin
                A[j+1] := X; goto Found
          end else
                 A[j+1] := A[j];
         A[1] := X;
        Found:
          end
      end
 end Sort
```

4.2 Beginning of Timesharing: BASIC

- Design goals:
  - Easy to learn and use for non-science students
  - Must be “pleasant and friendly”
  - Fast turnaround for homework
  - Free and private access
  - User time is more important than computer time

- First widely used language with time sharing
  - A single machine could divide up its processing time among many users, giving them the illusion of having a slow computer to themselves
  - Simultaneous individual access through terminal

BASIC (Beginner’s All-purpose Symbolic Instruction Code)

4.3 Everything for Everybody: PL/I

- IBM at 1963-1964:
  - Scientific computing: IBM 1620 and 7090, FORTRAN
  - Business computing: IBM 1401 and 7080, COBOL
  - Scientific users began to need elaborate I/O, like in COBOL; business users began to need FP and arrays
- The obvious solution
  - New computer to do both → IBM System/360
  - Design a new language to do both → PL/I
- Results:
  - Concurrently executing subprograms, exception handling, pointer
  - But, too many and too complex

PL/I : Programming Language One

4.4 Early Dynamic Languages: APL and SNOBOL

- APL and SNOBOL share two fundamental characteristics:
  1. dynamic typing and
  2. dynamic storage allocation

- A variable acquires a type when it is assigned a value, at which time it assumes the type of the value assigned.

- Storage is allocated to a variable only when it is assigned a value, because before that there is no way to know the amount of storage that will be needed.

APL : A Programming Language
SNOBOL : StriNg Oriented and symBOlic Language

4.5 Beginning of Data Abstraction

- SIMULA
  - Designed primarily for system simulation in University of Oslo, Norway, by Nygaard and Dahl

- Starting 1961: SIMULA I, SIMULA 67

- Primary contributions
  - *Co-routines*: a kind of subprogram
  - Implemented in a structure called a class, which include both local data and functionality and are the basis for data abstraction -> foundation for object-oriented programming

4.6.1 Programming Based on Logic: Prolog

- Based on formal logic
- Can be summarized as being an intelligent database system that uses an inferencing process to infer the truth of given queries
- The database of a Prolog program consists of two kinds of statements: facts and rules.

Fact statement

```
mother(joanne, jake)
father(vern, joanne)
```

joanne is the mother of jake, and vern is the father of joanne.

Rule statement

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z)
```

PROLOG: Programming in Logic

4.6.2 Simple Programs in Prolog

```
/* Some facts about parent relationships */
parent(sam,mark).
parent(mark,jim).

/* A general rule */
grandparent(GRANDPARENT,CHILD) :-
parent(GRANDPARENT,PARENT),
parent(PARENT,CHILD).
```

Query:

```
| ?-  grandparent(WHO, jim).
```

Q) Predict the output

4.6.3 Simple Programs in Prolog

```
/* Some facts*/

 likes(mary,food).

 likes(mary,wine).

 likes(john,wine).

 likes(john,mary).
```

Query

```
| ?- likes(mary,food).
```

True

Q) Predict Output

```
| ?- likes(john,wine).

| ?- likes(john,mary).

|?- likes(john, food).
```

4.6.4 Simple Programs in Prolog

```
/* Some facts */
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
likes(paul, mary).
likes(paul, food).
likes(paul,flowers).
likes(paul,wine).
```

Query

Q) Predict Output

```
| ?- likes(mary,X),likes(john,X).
```

Meaning: is anything liked by Mary also liked by John?

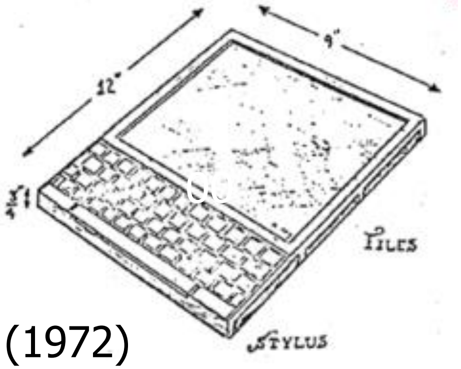
```
| ?- likes(mary,X),likes(paul,X).
```

Meaning: is anything liked by

Mary also liked by John?

4.7.1 Object-Oriented Programming

- Smalltalk: Alan Kay, Xerox PARC, 1972
- First full implementation of an object-oriented language
  - Everything is an object: variables, constants, activation records, classes, etc.
  - All computation is performed by objects sending and receiving messages
  - Data abstraction, inheritance, dynamic type binding
- Also pioneered graphical user interface design



Dynabook (1972)

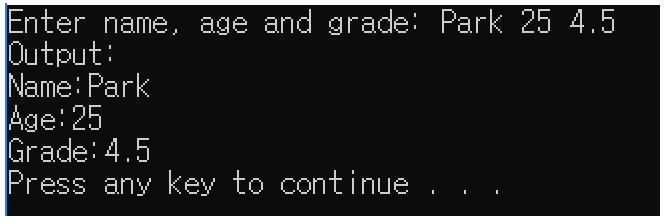
4.7.2 Imperative + Object-Oriented Features: C++

- Developed at Bell Labs by Stroustrup in 1980
- Evolved from C and SIMULA 67
- Facilities for object-oriented programming, taken partially from SIMULA 67
- A large and complex language, in part because it supports both procedural and OO programming

4.7.3 A Simple Program in C++

```
#include<iostream>
using namespace std;
class student
{
private:
        char name[20];
        int age;
        float grade;
public:
        void getInfo()
        {
                cout << "Enter name, age and grade: ";
                cin >> name >> age >> grade;
        }
        void putInfo()
        {
                cout << "Output:\n";
                cout << “Name:" << name << endl;
                cout << “Age:" << age << endl;
                cout << “Grade:" << grade << endl;
        }
};
```

```
int main()
{
        student obj;
        obj.getInfo();
        obj.putInfo();
        system("pause");
}
```



```
Enter name, age and grade: Park 25 4.5
Output:
Name:Park
Age:25
Grade:4.5
Press any key to continue . . .
```

4.7.4 Related OOP Languages

- Objective-C (designed by Brad Cox – early 1980s)
  - C plus support for OOP based on Smalltalk
  - Uses Smalltalk’s method calling syntax
  - Used by Apple for systems programs

- Delphi (Borland)
  - Pascal plus features to support OOP
  - More elegant and safer than C++

- Go (designed at Google - 2009)
  - Loosely based on C, but also quite different
  - Does not support traditional OOP

4.7.5 An Imperative-Based Object-Oriented Language: Java

- Developed at Sun in the early 1990s

  - C and C++ were not satisfactory for embedded electronic devices

- Based on C++

  - Significantly simplified (does not include **`struct, union, enum`**, pointer arithmetic, and half of the assignment coercions of C++)

  - Supports *only* OOP

  - Has references, but not pointers

  - Java uses implicit storage deallocation for its objects, often called **garbage collection**

4.7.6 A Simple Program in Java

```java
public class AddTwoNumbers {

        public static void main(String[] args) {

                int num1 = 5, num2 = 15, sum;
                sum = num1 + num2;

                System.out.println("Sum of these numbers: " + sum);
        }
}
```

4.8.1 Scripting Languages for the Web

- **Perl**

  - Variables are statically typed but implicitly declared
  - Three distinctive namespaces, denoted by the first character of a variable's name

    I. All scalar variable names begin with dollar signs ($),

    II. All array names begin with at signs (@),

    III. All hash names begin with percent signs (%)

- **JavaScript**

  - A client-side HTML-embedded scripting language, often used to create dynamic HTML documents
  - Purely interpreted

- **PHP**

  - PHP: Hypertext Preprocessor, designed by Rasmus Lerdorf
  - A server-side HTML-embedded scripting language, often used for form processing and database access through the Web
  - Purely interpreted

4.8.2 Scripting Languages for the Web

- **Python**
  - An OO interpreted scripting language
  - Type checked but dynamically typed
  - Data structure lists, tuples, and dictionaries

- **Ruby**
  - Began as a replacement for Perl and Python
  - A pure object-oriented scripting language : All data are objects
  - Purely interpreted

- **Lua**
  - An OO interpreted scripting language
  - Type checked but dynamically typed
  - Supports lists, tuples, and hashes, all with its single data structure, the table
  - Easily extendable

4.8.3 A Simple Program in Python

```python
num = 4

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
        print("Sorry, factorial does not exist for negative numbers")
elif num == 0 :
        print("The factorial of 0 is 1")
else:
        for i in range(1, num + 1) :
                factorial = factorial * i
        print("The factorial of", num, "is", factorial)
```

4.9 The Flagship .NET Language: C#

- Part of the .NET development platform (2000)
- Based on C++ , Java, and Delphi
- Includes pointers, delegates, enumeration types, a limited kind of dynamic typing, and anonymous types
- Is evolving rapidly

Q & A