Organization of Programming Languages CS320/520N

Lecture 02

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

A Brief History of Programming Languages

- Assembly languages
- IBM 704 and Fortran – FORmula TRANslation
- LISP – LISt Processing
- ALGOL 60 – International Algorithmic Language
- Simula 67 – first object oriented language
- Ada – history's largest design effort
- C++ – Combining Imperative and Object-Oriented Features
- Java – An Imperative-Based Object-Oriented Language
- Prolog – Logic Programming

Assembly Languages

- Invented by machine designers in early 1950s.
- Machine code is tedious and error-prone.
  - Poor readability.
  - Poor modifiability.
- Shift from machine code to mnemonics.
- First occurrencs of reusable macros & subroutines.

Machine Code (Intel Core2 Quad CPU)

```
int factorial (int n)
{
    int result = 1;
    int i;
    for (i = 2; i <= n; i++)
        result = result * i;

    return result;
}
```

```
55
4889E5
897DEC
C745FC01
C745F802
EB0E
8B45FC
0FAF45F8
8945FC
8345F801
8B45F8
3B45EC
7CEA
8B45FC
C9
C3
```

```
> gcc –c –g –Wa,–aln=factorial.s –c factorial.c
```

Assembly (Intel Core2 Quad CPU)

```
.LFB2:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movl    %edi, -20(%rbp)
        movl    $1, -4(%rbp)
        movl    $2, -8(%rbp)
        jmp     .L2
 .L3:
        movl    -4(%rbp), %eax
        imull   -8(%rbp), %eax
        movl    %eax, -4(%rbp)
        addl    $1, -8(%rbp)
```

```
.L2:
        movl    -8(%rbp), %eax
        cmpl    -20(%rbp), %eax
        jle     .L3
        movl    -4(%rbp), %eax
        leave
        ret
```

> gcc –S factorial.c
> gcc –c –g –Wa,–a,–ad factorial.c

Fortran 0

- Designed by John Backus at IBM in the early 1950's
- First widely accepted compiled high-level language:
  - Designed for the new IBM 704, which had index registers and floating point hardware.
  - This led to the idea of compiled programming languages, because there was no place to hide the cost of interpretation (no need for floating-point software).
- Design influenced by environment:
  - Computers were expensive, slow, with small memory.
  - Primary use of computers was for scientific applications.
  - No existing efficient way to program computers.

Fortran I

- First implemented version of Fortran (1957, 18 worker years of effort):
  - Names could have up to six characters.
  - Post-test counting loop (**`DO`**).
  - Formatted I/O.
  - No dynamic memory allocation.
  - User-defined subprograms (separate compilation added in Fortran II).
  - Three-way selection statement (**`IF`**).
  - No data typing statements (I,J,K,L,M,N integers, rest floating point).
  - Code was very fast => quickly became widely used.

Evolution of Fortran

- Fortran IV, 77, 90, 95, 2003:
  - Explicit type declarations for variables.
  - Subprograms as parameters.
  - Character string handling
  - Logical loop control statements
  - Dynamic arrays, records, pointers
  - Multiple selection statement
  - Modules, recursive subprograms
  - Parametrized data types
  - Support for OOP
  - Procedure pointers, interoperability with C.

Factorial in Fortran

```
function fact(n)
      integer fact, n, p
      p = 1
      do i = 2, n
            p = p * i
      end do
      fact = p
end
```

```
program demo_factorial
      integer fact, n
      print *, “n = “
      read *, n
      print *, n, “! = ", fact(n)
end
```

LISP

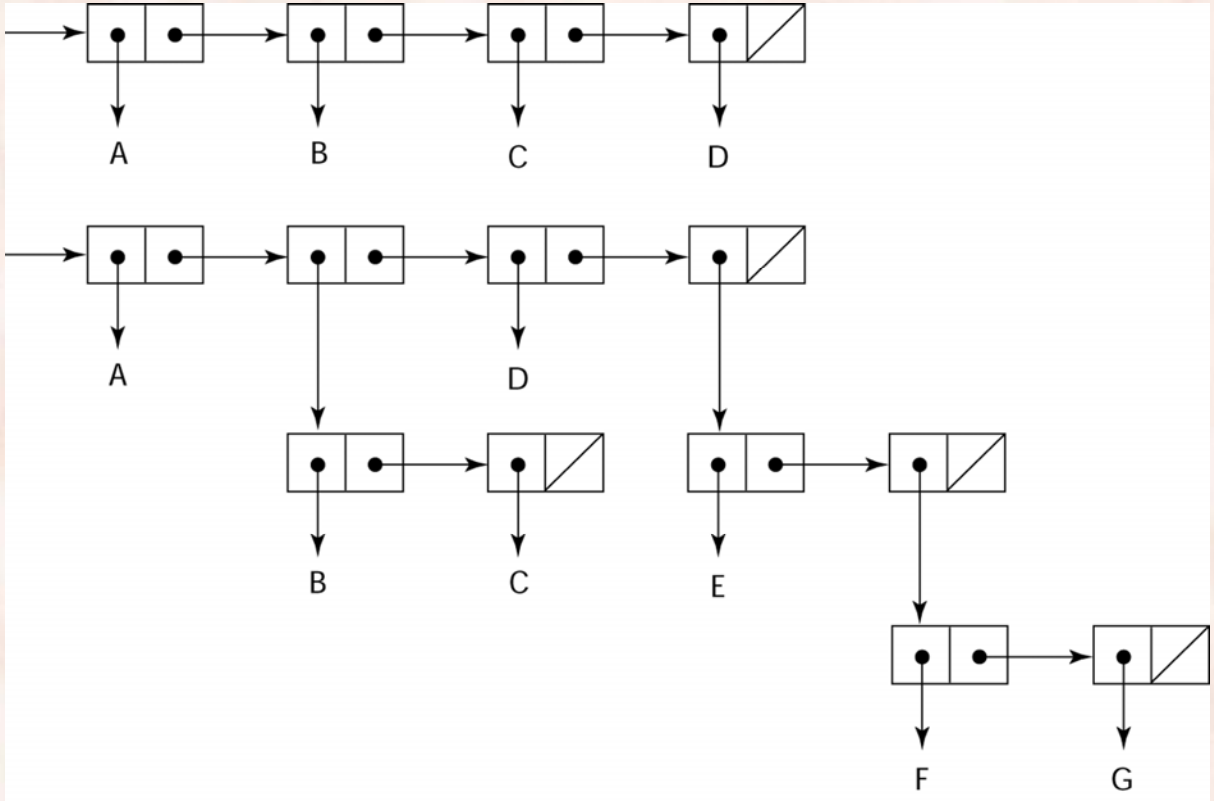
- Designed by John McCarthy at MIT in the late 1950s.
- Design influenced by AI applications:
  - Symbolic computation (rather than numeric).
    - Ex: differentiation of algebraic expressions.
    - Ex: Advice taker.
  - Process data in lists (rather than arrays):
    - Dynamically allocated linked lists.
    - Implicit deallocation of abandoned lists.
- Implemented on IBM 704.

"Pure" LISP

- Purely functional language:
  - No need for variables, assignment, or iteration (loops).
  - Control via recursion and conditional expressions.
  - Syntax is based on *lambda calculus*.
- Only two data types: Atoms and Lists.
  - Atoms are either symbols (identifiers) or numeric literals.
  - Two basic list operations: CAR and CDR

```
(defun factorial (n)
     (if  (<= n 1)
          1
          (* n (fact (- n 1)))))
```

Lists



(A B C D) and **(A (B C) D (E (F G)))**

Related Functional Languages

- Scheme (MIT mid-1970s):
  - Small size, simple syntax and semantics.
  - Exclusive use of static scoping.
  - Functions are first class entities.

```
(define fact
    (lambda (n)
        (if  (<= n 1)
             1
             (* n (fact (- n 1))))))
```

- Common Lisp, Miranda, Haskell, ML.

Algol

- International Algorithmic Language.
- Designed by IFIP working group in 1958-1960:
  - John Backus, Peter Naur, John McCarthy, Alan Perlis & others.
  - Syntax specified formally using the Backus-Naur Form (BNF).
- Goals:
  - Universal language for communicating algorithms.
  - Portable, machine independent.
  - Close to mathematical notation.
  - Must be translatable to machine code.

Algol 58

- Concept of type was formalized (explicit variable type declarations)
- Names could be any length
- Arrays could have any number of dimensions
- Parameters were separated by mode (in & out)
- Subscripts were placed in brackets
- Compound statements (**`begin ... end`**)
- Semicolon as a statement separator, assignment operator was :=
- **`if`** had an **`else-if`** clause
- No I/O - “would make it machine dependent”

Algol 60

- New features:
  - Block structure (local scope).
  - Two parameter passing methods.
  - Recursive subprograms.
  - Stack-dynamic arrays.

  - Still no I/O and no string handling.

Algol 60

- Successes:
  - It was the standard way to publish algorithms for over 20 years.
  - First machine-independent language.
  - First language whose syntax was formally defined (BNF).
  - Significant influence on all of today's modern languages:
    - Pascal, Modula, Ada, C, C++ & Java are direct descendants.
    - Scheme adopted lexical scoping from Algol.

Algol 60

- Failures:
  - Never widely used, especially in U.S.
  - Reasons:
    - Lack of I/O and the character set made programs non-portable.
    - Too flexible => hard to implement.
    - Entrenchment of Fortran.
    - Formal syntax description.
    - Lack of support from IBM.

Simula 67

- Designed by Kristen Nygaard and Ole-Johan Dahl at NCC.
- Superset of Algol 60, for simulations.
- Innovations:
  - Coroutines (subprograms that restart at the position where they previously stopped).
  - First OOP language:
    - Classes (package data structure with manipulating routines).
    - Objects as class instances (local data & code executed at creation).
    - Inheritance, virtual methods.

Simula 67

- Influenced all subsequent OO programming languages:
  - Smalltalk
  - Objective-C
  - C++
  - Eiffel
  - Modula 3
  - Self
  - C#
  - CLOS

Ada

- Designed for DoD as a high-level language for embedded systems applications:
- Huge design effort, involving hundreds of people, much money, and about eight years.
  - Strawman requirements (April 1975)
  - Woodman requirements (August 1975)
  - Tinman requirements (1976)
  - Ironman equipments (1977)
  - Steelman requirements (1978)
- Named Ada after Augusta Ada Byron, the first programmer

Ada

- Major Contributions:
  - Packages - support for data abstraction
  - Exception handling - elaborate
  - Generic program units
  - Concurrency - through the rendezvous synchronization model
- Comments:
  - Competitive design
  - Included all that was then known about software engineering and language design
  - First compilers were very difficult; the first really usable compiler came nearly five years after the language design was completed

Ada 95

- Ada 95 (began in 1988):
  - Support for OOP through type derivation.
  - Better control mechanisms for shared data.
  - New concurrency features.
  - More flexible libraries.
- Popularity suffered because the DoD no longer requires its use but also because of popularity of C++.

Factorial in Ada

```
procedure demo_factorial is
        function factorial (n: Integer) return Integer is
        begin
                if n <= 1 then
                        return 1;
                else
                        return n * factorial(n – 1);
                end if;
        end factorial;

n: Integer;
begin
        get(n);
        put(factorial(n));
end demo_factorial;
```

C: A Portable Systems Language

- Designed by Dennis Ritchie at Bell Labs in 1972.
- Designed for *systems programming*:
  - the development of an OS and its utilities.
  - first Unix written in assembly language.
  - B was first high-level language on UNIX (Ken Thompson, 1970)
  - C was developed as a typed language based on B:
    - int i, *pi, *ppi;
    - int f(), *f(), *(*pf)();
    - int *api[10], (*pai)[10];
    - syntax influenced by Algol 68.
    - also added structs & unions.

C: A Portable Systems Language

- Standardization:
  - K&R book published in 1983.
  - ANSI C standard in 1989 (C89).
    - C++ like function prototypes, const & volatile keywords, …
  - ISO 9899:1999 (C99)
    - C++ like decls, inline functions, bools, variable arrays & more.
- Used as a portable assembly language:
  - Early C++, Modula 3, and Eiffel were translaed to C.
- C compilers available for all kinds of architectures:
  - GNU gcc for more than 70 instruction set architectures.

C++: Combining Imperative and OO Programming

- Developed by Bjarne Stroustrup at Bell Labs in 1980.
- Backward compatible with C:
  - Easy to link C++ code with C code.
- Facilities for OOP related to Simula 67 & Smalltalk:
  - Derived classes & inheritance (1983).
  - Virtual methods, overloaded methods & operators (1984).
  - Multiple inheritance, abstract classes (1989).
  - Templates, exception handling (ISO 1998).

C++: Combining Imperative and OO Programming

- Large & complex language:
  - Supports both procedural and OO programming through functions & methods.
- Very popular:
  - Availability of good & inexpensive compilers.
  - Suitable for large commercial software projects.
- Microsoft's version (released with .NET in 2002):
  - No multiple inheritance, references for garbage collected objects, …

Java: An Imperative-Based OO Language

- Developed by a team headed by James Gosling at Sun in the early 1990s
  - C and C++ were not satisfactory for embedded electronic devices.

- Based on C++:
  - Significantly simplified:
    - no **`struct,union,enum.`**
    - no pointer arithmetic.
    - eliminated half of the assignment coercions of C++ .
    - no multiple inheritance, no operator overloading.
  - Supports *only* OOP (e.g. no stand-alone subprograms).
  - All objects allocated on the heap & garbage collected.

Java: An Imperative-Based OO Language

- Very successful:
  - Eliminated many unsafe features of C++ ⇒ simpler, safer design.
  - Supports concurrency (threads, **synchronize**d methods).
  - Libraries for applets, GUIs, database access.
  - Portable:
    - Java Virtual Machine concept, JIT compilers.
  - Widely used for Web programming.
  - Use increased faster than any previous language.
- Java 5.0:
  - Enumeration class, generics, new iteration construct.

Prolog: Logic Programming

- Developed, by Colmerauer and Roussel (University of Aix-Marseille), with help from Kowalski (University of Edinburgh) in the early 1970s.

- Non-procedural language:
  - describe *What* as opposed to *How*.
  - notation based on predicate calculus (Horn clauses).
  - Inference method based on resolution (Robinson 1965).

- Highly inefficient relative to equivalent imperative progs.
- Small application areas in AI and DBMS.

Prolog: Logic Programming

- Program = a collections of statements:
  - Facts:
    - mother(joanne, jake); father(vern, joanne)
  - Rules:
    - parent(X,Y) :- mother(X,Y).
    - parent(X,Y) :- father(X,Y).
    - grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
  - Queries:
    - grandparent(X,jake).

Factorial in Prolog

```
factorial(0,1).

factorial(1,1).

factorial(N,M) :-  N1 is N – 1,
                   factorial(N1,M1),
                   M is N*M1.
```

Scripting Languages for the Web

- JavaScript
  - Began at Netscape, but later became a joint venture of Netscape and Sun Microsystems
  - A client-side HTML-embedded scripting language, often used to create dynamic HTML documents
  - Purely interpreted
  - Related to Java only through similar syntax
- 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
- Python
  - multiparadigm scripting language:
    - imperative
    - functional
    - object oriented
  - Used for CGI programming and form processing