Programming Languages Advance

Ms. Farzeen Ashfaq

Welcome Back!

Course Plan

1) Theoretical Foundations of Programming Languages
   - Names , Bindings and Scope
   - Data Types
   - Expressions & Assignments
   - Underlying Machine Architecture

2) Imperative Paradigm
   - Procedural
   - Object Oriented

3) Concurrent & Parallel Programming

4) Research & Case Studies

NAMES , BINDINGS & SCOPE

WHAT IS A NAME?

- A character or string
- Most languages names are identifiers (alphanumeric tokens)
- Names allow us to refer to variables , constants, operations or types
- Abstraction????

- ***A name is a string of characters used to identify some entity in a program.***
- The following are the primary design issues for names:
  - Are names case sensitive?
  - Are the special words of the language reserved words or keywords?

Special Names

- Special words in programming languages are used to make programs more readable by naming actions to be performed.
- They also are used to separate the syntactic parts of statements and programs.
- A keyword is a word of a programming language that is special only in certain contexts

- ***What problem could arise if a language keeps a large stock of reserved words?***

Abstraction

- Abstraction is a process by which the programmer associates a name with potentially complicated program fragment
- Variables → Memory Abstraction
- Subroutines → Control Abstraction
- Classes → Data Abstraction

Variable

- A program variable is an abstraction of a computer memory cell or collection of cells. Programmers often think of variables as names for memory locations, but there is much more to a variable than just a name

Associated Concepts

- Name
- Address
- Value
- Type
- Binding Time
- Lifetime
- Scope
- Referencing Environment

Binding

- Association Between Two Things
- Binding time is the time at which a binding is created or, more generally, the time at which any implementation decision is made

Binding Time

- **Language design time:**

  In most languages, the control flow constructs, the set of fundamental (primitive) types, the available constructors for creating complex types, and many other aspects of language semantics are chosen when the language is designed

Binding Time

- **Language implementation time:**

  Most language manuals leave a variety of issues to the discretion of the language implementor. Typical (though by no means universal) examples include the precision (number of bits) of the fundamental types, the coupling of I/O to the operating system’s notion of files, the orga nization and maximum sizes of stack and heap, and the handling of run-time exceptions such as arithmetic overflow

Binding Time

- **Program writing time:**

  Programmers, of course, choose algorithms, data structures, and names.

- **Compile time:**

  Compilers choose the mapping of high-level constructs to machine code, including the layout of statically defined data in memory

Binding Time

- **Link time:**

  Since most compilers support separate compilation—compiling different modules of a program at different times—and depend on the availability of a library of standard subroutines, a program is usually not complete until the various modules are joined together by a linker. The linker chooses the overall layout of the modules with respect to one another, and resolves intermodule references. When a name in one module refers to an object in another module, the binding between the two is not finalized until link time.

Binding Time

- **Load time:**

  Load time refers to the point at which the operating system loads the program into memory so that it can run. In primitive operating systems, the choice of machine addresses for objects within the program was not finalized until load time. Most modern operating systems distinguish between virtual and physical addresses. Virtual addresses are chosen at link time; physical addresses can actually change at run time. The processor’s memory management hardware translates virtual addresses into physical addresses during each individual instruction at run time

Binding Time

- **Run time:**

  Run time is actually a very broad term that covers the entire span from the beginning to the end of execution. Bindings of values to variables occur at run time, as do a host of other decisions that vary from language to language.