CS 350: Programming Language Design

Lecture 19

Sub-programs, re-usable code that can be used at different steps or in different situations within a program.

Two Fundamental Abstractions

Process

* Advantages: abstract details, improve logical structure, better readability
* Reuse code, save code time, and memory

Data

* Developed with Object oriented programming

Subprograms are most basically, groups of statements that define parameterized computations

Each subprogram will have a single entry point, when called, the calling program suspends itself until the execution of the subprogram has completed or terminates.

Two Categories of subprogram

* Procedure vs. Functions
* Function
  + Returns value
  + They are expected to produce no side effects
  + No changes to parameters or outside variables
  + Functions truly have side effects in vivo
* Procedure
  + Doesn’t return a value
  + Can produce results in the calling program through two methods
  + Through variables that are not formal parameters but are still visible in both the procedure and the calling unit.
  + Through formal parameters that allow the transfer of data to the caller, those parameters can be changed.

A subprogram header is the first part of the definition

* This includes the name, kind of subprogram, and formal parameters

The parameter profile is the number, order, and types of its parameters

The Protocol is the subprogram’s parameter profile and, if it is a function, it’s return type

A subprogram describes the interface to and the actions of the subprogram abstraction.

A subprogram call is an explicit request that the subprogram be executed.

Subprograms typically describe computations that require data.

There are two ways to provide that data,

* Direct access, using globals etc.
* Parameter provision (more reliable)
  + Formal vs. actual (reference vs. address)

Actual/Formal Parameter correspondence

Positional

* The binding of actual parameters to formal parameters is by position
* The first actual parameter is bound to the first formal parameter and so forth
* This is safe and reliable.

Keywords

* The name of the formal parameter to which an actual parameter is to be bound is specified with the actual parameter.
* Advantage: Parameters may appear in any order, thus avoiding parameter correspondence errors is easy
* Disadvantage, user must know the formal parameters names.

C# accepts a variable number of parameters, as long as they are all of the same type.

Design Issues for Subprograms

* Are local variables static or dynamic
* What parameter passing methods are provided
* Are parameter types checked?
* Can subprograms be passed as parameters? What is the referencing environment of a passed subprogram?
* Can subprograms be nested? What is the referencing environment of a passed subprogram?
* Can subprograms be overloaded?
* Can subprogram be generic?

Local Referencing Environments

* Local variables are defined inside subprograms
* Local variables can be
  + Stack-dynamic
  + Flexible, recursive, storage for locals is shared
  + Allocation/De-allocation, initialization time, indirect addressing, subprograms can’t be history sensitive
* Static
  + Advantages and disadvantages are the opposite of those for stack-dynamic local variables