CHAPTER 5

1. List and discuss the design issues for names?

* **Are names case-sensitive:**If the names are case-sensitive, then it affects the readability because the names that look alike will be different.
* **Are the special words reserved words or the keywords:** Reserved words are special words that cannot be used as a user defined word. And keywords act as special words under certain context. So, if a word is a special word, it’s difficult to determine whether it is reserved word or keyword.

1. What is a scalar variable?

* A variable is an abstraction of a memory cell and the scalar or primitive variable is a variable that stores a single value.

1. List and discuss the six attributes associated with a variable.

* There are six attributes associated with a variable:
  + Names: The name given to a variable. Not all variables have names
  + Address: The address of a memory cell with which it is associated.
  + Type: determines the range of values of a variable and set of operation that are defined for the values of those type. For floating point numbers, type determines precision.
  + Value: The contents of the cell with which the variable is associated.
  + Lifetime: The time during which the variable is bound to a perticular memory cell.
  + Scope: The range of the program statement over which the variable is visible.

1. What is an alias?

* If two variable names can be used to access same memory address then they are called aliases.

1. Define *binding* and *binding time*.

* Binding: It is an association of an entity and an attribute. For instance, association of a variable and its type or value, or association of an operation and symbol
* Binding time: It is the time at which the binding takes place.

1. What are the five times bindings can take place in a program?

* Language design time: bind operator symbols to operations
* Language implementation: bind floating point type to representation
* Compile time: bind variable to its type in C or java.
* Load time: bind C or C++ static variable to memory cell
* Run time: bind a local non-static variable to memory cell

1. Define *static binding* and *dynamic binding*.

* Static binding: It is the binding that occurs before the run time and remains unchanged during the program execution
* Dynamic binding: It is the binding that occurs during the run time and can change during the execution of the program.

1. Discuss static type binding and explicit and implicit type declarations.

* Static type binding is the binding of variable to its type before the program execution and it remains unchanged during the program execution. There are two types of declarations in static type binding. They are:
  + Explicit declarations: A program statement used to declare the type of the variable.
  + Implicit declaration: A default mechanism used to declare the type of variable using default conventions, rather than declaration statements.

1. What are the advantages and disadvantages of implicit declarations?

* Advantages: Writability (a minor convenience)
* Disadvantage: reliability (little trouble with pearls)

1. What is type inferencing?

* Type inferencing is the automatic deduction of type of a variable using the context of the appearance of the variable.

1. What are the advantages and disadvantages of dynamic type binding?

* Dynamic type binding occurs during program execution and can change during the run time.
  + Advantages: Flexibility – It is easy to write generic code
  + Disadvantages: High cost (Dynamic type checking, interpretation)

: type error detection by compiler is difficult.

1. Define *static*, *stack-dynamic*, *explicit heap-dynamic*, and *implicit heap-dynamic variables*. What are their advantages and disadvantages?

* Static variable: It is bound to a memory cell before run time and remains bound to the same memory cell during the program execution.
  + Advantage:
    - efficiency (direct addressing)
    - history sensitive sub program is supported
  + Disadvantage:
    - Lack of flexibility (no recursion)
* Stack-dynamic: Storage binding for a variable occurs when the declaration statement is elaborated. A statement is elaborated when an executable code associated with the statement is executed.
  + Advantage:
    - Flexibility (supports recursion)
    - Saves storage
  + Disadvantage:
    - Overhead for allocation and deallocation.
    - History sensitive subprogram is not supported
    - Inefficient (indirect addressing)
* Explicit heap-dynamic: Allocation and deallocation for a variable occurs with explicit directive, specified by programmer, which takes into effect during program execution
  + Advantages:
    - Provides for dynamic storage management
  + Disadvantage:
    - Inefficient and unreliable
* Implicit heap-dynamic : Allocation and deallocation of a variable occurs by assignment statement.
  + Advantage:
    - Flexibility (generic code)
  + Disadvantage:
    - Inefficient (all attributes are dynamic)
    - Loss of error detection

1. Define *lifetime*, *scope*, *static scope*, and *dynamic scope*.

* The lifetime is the time during which a variable is bound to a particular memory cell.
* Scope of a variable is the range of program statements over which the variable is visible.
* Static scope is based on program text. To connect a name reference to a variable, the programmer (or compiler) must find the declaration. The search process search the declaration, first locally, then in increasingly larger enclosing scope, until one is found for the given name.
* Dynamic scope is based on calling sequences of program units and not their textual layout. Reference to a variable is connected to the declaration by searching back through the chain of subprogram calls that forced the execution to this point.

1. Define local, nonlocal, and global variables.

* Local variable of a program unit are those variables that are defined in that unit.
* Non-local variable of a program unit are those variables that are defined outside that unit but are visible.
* Global variables: They are a non local variable which can be accessed from all program unit.

1. How is a reference to a nonlocal variable in a static-scoped program connected to its definition?

* To connect a name reference to a non local variable in a static -scoped program the programmer (or compiler) must first find the declaration.
* Search process search the declaration, first locally, then in increasingly larger enclosing scope until one is found for the given name.

1. **What is a static ancestor of a subprogram? What is a dynamic ancestor of a subprogram?**

* The static enclosing scope to a particular scope is its static ancestor. The nearest static ancestor is its static parent.
* The sub program calls that forced the execution to the current point of execution are the dynamic ancestor.

1. What is a block and what does it have to do with scope?

* The section of code that can have its own local variables whose scope is minimized.
* The scope created by blocks which can be nested to larger blocks are treated exactly like those created by subprograms.

1. What is a block-structured language?

* A programming language that allows the creating of blocks, including the blocks nested within blocks which can be treated as subprograms are called block-structured language.

1. What are the advantages and disadvantages of static scoping?

* Advantages:
  + Works well in many cases
* Disadvantages:
  + In most cases too much access is possible
  + As a program evolves, the intial structure is destroyed and local variables become global and subprograms gravitate towards becoming global rather than nested.

1. What are the advantages and disadvantages of dynamic scoping?

* Advantages:
  + Convenient
* Disadvantages:
  + The subprogram that is executing has its variable visible by all the subprograms that it executes.
  + It is impossible to statically type check
  + Poor readability: It is not possible to check the data type of a variable.

1. What is the referencing environment of a statement? In a static scoped language? In a dynamic scoped language?

* The referencing environment of a statement is the collection of all names that are visible in the statement.
* In a static scoped language, the referencing environment is local variables plus all visible variables in all enclosing scopes.
* In a dynamic scoped language, the referencing environment is local variable plus all visible variable in all the active subprograms.

1. What is an active subprogram?

* A subprogram is active if the execution has begun but not terminated yet.

1. What is a named constant? What are the advantages of named constants

* Named constant is a variable that is bound to a value only when it is bound to storage. The advantages of named constants are readability and modifiability.

1. Most computer languages have types. Discuss the advantages and disadvantages of a language being:

CHAPTER 9

1. **What are the three general characteristics of subprograms?**

* Each subprogram must have a single entry point.
* The calling program is suspended during the execution of the called subprogram
* Control always return to the caller after the exectution of the called subprogram is terminated.

1. **What does it mean for a subprogram to be active?**

* A subprogram is said to be active if after being called, the execution of the subprogram has started but is ye t to be terminated.

1. **What is a subprogram definition?**

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

1. **What is given in the header of a subprogram?**

* A subprogram header is the first part of definition, including the name, type of subprogram and formal parameters.

1. **What is a parameter profile?**

* A parameter profile (aka signmature) of a subprogram is the number, order and type of its parameters.

1. **What are formal parameters? What are actual parameters?**

* Formal parameters are dummy variables listed in the header of the subprogram and used in the subprogram
* Actual parameter is the value or the address that are used in the subprogram call statement.

1. **What are the advantages and disadvantages of positional parameters?**

* Positional parameters: The binding of an actual parameter to a formal parameter is by position. The first actual parameter is bound to the first formal parameter and so on.
  + Advantage:
    - Safe and effective
  + Disadvantage:
    - The order of the actual parameters must match to that of formal parameters

1. **What are the advantages and disadvantages of keyword parameters?**

* Keyword Parameters: The name of the formal parameter to which an actual parameter is to be bound is specified in the actual parameter.
  + Advantage:
    - Can place the parameters in any order, avoiding the parameter correspondence error.
  + Disadvantage:
    - User must remember the names of the formal parameters

1. **What are the differences between a function and a procedure?**

* Procedures: Procedures are collection of statements that define parameterized computations. They don’t return value.
* Function: Functions are structurally similar to procedures but are semantically modelled after mathematical functions. They return value.

1. **What are the design issues for subprograms?**

* Are local variables static or dynamic?
* Can subprogram definition appear in another subprogram definition?
* What parameter passing methods are provided?
* Are parameter types checked
* If a subprogram can be passed as a parameter and the subprogram can be nested, what is the referencing environment for the passed subprogram.
* Are functional side effects allowed?
* What kind of values can a function return?
* How many values can a function return?
* Can subprogram be generic
* Can subprogram be overloaded?
* If the language allows subprogram to be nested, is closure allowed.

1. **Define local referencing environment.**

* Subprograms are allowed to define their own variables, thereby creating local referencing environment.
* Mostly local variables are stack dynamic or static.

1. **Compare and contrast static and dynamic local variables?**

* Static:
  + Advantage:
    - Efficient (direct addressing)
    - History sensitive subprograms are supported
  + Disadvantage:
    - Lack of flexibility (no recursion)
* Stack dynamic:
  + Advantage:
    - Recursion allowed
    - Saves storage
  + Disadvantage:
    - Overhead for allocation and deallocation of varaiables
    - Not efficient (indirect addressing)
    - History sensitive subprogram not supported

1. **What are the three semantic models of parameter passing?**

* In mode
* Out mode
* Inout mode

1. **What are the modes, the conceptual models of transfer, the advantages, and the disadvantages of pass-by-value, pass-by-result, pass-by-value-result, and pass-by-reference parameter-passing methods?**

* Pass by value:
  + Modes: In mode
  + The value of the actual parameter is used to initialize the corresponding formal parameter. Normally implemented by copying the value.
  + Advantages:
    - Faster access to the formal parameter.
  + Disadvantage:
    - Require extra storage
    - Copy operation can be costly for large parameters
* Pass by result:
  + Modes: out mode
  + When a parameter is passed by result, no value is transmitted to the subprogram. The formal parameter of the subprogram acts as a local variable and its value is transmitted to the actual parameter of the caller when control is passed to caller.
  + Advantage:
    - Faster access to actual parameter
  + Disadvantage:
    - Need extra storage
    - Copying the parameter can cost heavy for large parameters
* Pass by value-result
  + Mode: Inout mode
  + Uses both pass by value and pass by result. Have local storage
  + Advantage:
    - Have local storage
    - Faster to access both actual and formal parameter.
  + Disadvantage:
    - Extra storage
    - Cost heavy for copying
* Pass by reference:
  + Mode: Inout
  + Use access path instead of value.
  + Advantage:
    - Efficient because no copying and no extra storage
  + Disadvantage:
    - Slower
    - Possibility of creating aliases
    - Possibility of collision

1. **Describe the ways that aliases can occur with pass-by-reference parameters.**

* In subprogram, we can refer to the value in any ways since we are passing an address. For example:
  + Fun(total, total); fun(list[i],list[j]); fun(list[i],i);
  + They all refer to the same memory location.
* Aliases can be occurring because pass by reference makes access paths available to the called subprograms.

1. **What are two fundamental design considerations for parameter-passing methods?**

* Efficiency
* One way or two way data transfer

1. **What are the two issues that arise when subprogram names are parameters?**

* Are the parameters type checked?
* What is the correct referencinig environment for the subprogram that was passed as a parameter?

1. **Define shallow, deep, and ad hoc binding for referencing environments of subprograms passed as parameters.**

* Shallow binding: The environment of the call statement that enacts the passed subprogram. Most natural in dynamic scoped languages
* Deep binding: The environment of the definition of the passed subprogram Most natural in static scoped languages
* Ad hoc binding: The environment of the call statement that passed the subprogram.

1. **What are the design issues for functions?**

* Are side effects allowed?
* What type of return value is allowed?

1. **What is an overloaded subprogram?**

* An overloaded subprogram is a subprogram that has the same name as other subprogram in the same referencing environment.

1. **What is a generic or polymorphic subprogram?**

* A generic or polymorphic subprogram takes in different types of parameters in different activations.

**UNIT 10**

1. **What is the definition used in this chapter for “simple” subprograms?**

* Simple subprograms do not have nested loops and the local variables are static.

1. **Which of the caller or callee saves execution status information?**

* The last three actions of the call is clearly done by the caller. Saving the execution status information of the caller can be done by either.

1. **What must be stored for the linkage to a subprogram?**

* Execution status information, parameters, return address, return value of the function, temporaries.

1. **What is the task of a linker?**

* The first task of linker is to find files that contain translated subprograms referenced to that program and load it to the memory. Then it must set the target addresses of all calls to those subprograms in the main program to the entry addresses of those subprograms.

1. **What are the two reasons why implementing subprograms with stack-dynamic local variables is more difficult than implementing simple subprograms?**

* The compilers must generate code that causes the implicit allocation and deallocation of local variables.
* Recursion must be allowed (which adds the possibility of multiple simultaneous activation subprograms)

1. **What is the difference between an activation record and an activation record instance?**

* An activation record format is static but its size may be dynamic
* An activation record instance reside on the run time stack and is dynamically created during execution of subprogram.

1. **Why are the return address, dynamic link, and parameters placed in the bottom of the activation record?**

* Return address: After the execution of subprogram where to return and start the continue of execution.
* Dynamic link: To figure out the referencing environment
* Parameter: Local variable which exists only while the subprogram is being executed

1. **What are the two steps in locating a nonlocal variable in a static-scoped language with stack-dynamic local variables and nested subprograms?**

* First, activation record instance of a specific stack must be located
* Then, using the local offset the activation record instance can be accessed an inspected.

1. **Define static chain, static\_depth, nesting\_depth, and chain\_offset.**

* Static chain: It is the chain of static links that connect certain activation record instances.
* Static depth: It is an integer associated with a scope whose value is the depth of the nesting of that scope.
* Chain offset or nesting depth of a non local reference is the difference between the static depth of the reference and that of scope when it is declared.

1. What is an EP, and what is its purpose?

* Environment pointer must always be maintained by run-time system. It always points to the base of the activation record instance of the currently executing program unit.

1. How are references to variables represented in the static-chain method?

* Reference to variables can be represented by a pair

(chain\_offset, local\_offset), where local\_offset is the offset in the activation record of the variable being referenced.

1. What are the two potential problems with the static-chain method?

* Non-local reference is slow if the nesting depth is high
* Time critical code is difficult:
  + Cost of Non local references is difficult to determin
  + Chage in code changes the nesting depth which changes the cost.

1. Describe the deep-access method of implementing dynamic scoping.

* The non local references are found by searching the activation record instances of dynamic chain
* Length of the chain cannot be statically determined
* Each activation record instances must have variable names.

1. Describe the shallow-access method of implementing dynamic scoping.

* Put locals in a central place:
  + One stack for each variable name
  + Central table with an entry for each variable name