REVIEW QUESTIONS

1. What are the design issues for names?

- Case sensitive
- Special words, reserved words, keywords

2. What is the potential danger of case-sensitive names?

- readability = names that look very similar in fact denote different entities.

3. What is an alias?

- one variable name that can be used to access the same memory location

4. Which category of C++ reference variables always produces aliases?

- Union / pointer

5. What is the l-value of a variable? What is the r-value?

- l-value of a variable is the address of a variable
- r-value of a variable is a variable's value itself.

6. Define binding and binding time.

- Binding is an association such as between an attribute and an entity, or between an operation and a symbol.
- Binding time is the time at which a binding takes place.

7. After language design and implementation, what are the four times bindings can take place in a program?

- Compile time
- Load time
- Link time
- Run time

8. Define static binding and dynamic binding.

- Static binding: if it first occurs before run time and remains unchanged throughout program execution.
- Dynamic binding: if it first occurs during execution or can change during execution of the program.

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

- Advantage: writability
- Disadvantage: reliability and readability

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

- Advantages: flexibility and no definite types declared(PHP, JavaScript).

- Disadvantages: adds to the cost of implementation and type error detection by the compiler is difficult.

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

- Static: bound to memory cells before execution begins and remains bound to the same memory cell throughout the execution.
- Stack-dynamic: storage bindings are created for variables when their declaration statements are elaborated.
- Explicit heap-dynamic: allocated and deallocated by explicit directives, specified by the programmer, which take effect during execution.
- Implicit heap-dynamic variables: Allocation and deallocation caused by assignment statements.

12. Define lifetime, scope, static scope, and dynamic scope.

- Lifetime of variable is time during which the variable is bound to a specific men location.
- Scope of variable is the range of statements over which it is visible.
- Static scope is binding names to non-local variables.
- Dynamic scope is based on the calling sequence of subprograms, not on their spatial relationship to each other. Thus the scope can be determined only at run time.

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

- through the static chain, also known as the access link or the static link.

14. What is the general problem with static scoping?

- Mistakenly call subprogram which should not be callable
- Too much data accesses and verifcations
- Encourages the use of more global than necessary

15. What is the referencing environment of a statement?

- the collection of all variables that are visible in the statement

16. What is a static ancestor of a subprogram? What is a dynamic ancestor of a subprogram?

- The static ancestors of a subprogram sub() are all the procedures in the program within which the procedure sub() is defined, i.e., the definition of the procedure sub() is nested. The definition of a procedure may be directly nested within only one procedure, called its static parent procedure. However, this static parent procedure may itself be nested within another procedure, and so on up to the main() program. All these procedures are considered to be static ancestors of the procedure sub().

Simply put, the static ancestors are those that strictly contain the subprogram in question.

- The dynamic ancestors of a subprogram sub() are all the procedures called before sub() during the execution of a program, that have not yet finished executing. These are the procedures that are waiting for procedure sub() to finish executing before they can terminate. Simply put, dynamic ancestors are those that are called to reach the subprogram in question.

17. What is a block?

- Section of code to have its own local variables whose scope is minimized. Such variables are typically stack dynamic, so their storage is allocated when the section is entered and deallocated when the section is exited. Such a section of code is called a block.

18. What is the purpose of the let constructs in functional languages?

- allows defining local variables and functions within an expression or a block of code. The purpose of let constructs is to support functional programming concepts such as abstraction, modularity, and immutability.

19. What is the difference between the names defined in an ML let construct from the variables declared in a C block?

- Binding time: In ML, the names defined in a let construct are bound at compile-time and remain constant throughout the execution of the program. In contrast, the variables declared in a C block are typically allocated at run-time and can be modified or deallocated during the execution of the program.
- Typing: In ML, the names defined in a let construct have statically inferred types that are checked at compile-time, ensuring type safety and reducing runtime errors. In C, the variables declared in a block can have either static or dynamic types, depending on whether they are declared with or without the "auto" keyword, and their types can only be checked at run-time.
- Scope: In ML, the names defined in a let construct have lexical scope, meaning that they can only be accessed within the block where they are defined and any nested blocks. In C, the variables declared in a block have block scope, meaning that they can only be accessed within the block where they are declared and any nested blocks, but not outside of the block.
- Memory management: In ML, the names defined in a let construct are typically managed by the garbage collector, which automatically deallocates any unused memory. In C, the variables declared in a block are typically managed by the programmer, who must explicitly allocate and deallocate memory using functions such as malloc() and free().

20. Describe the encapsulation of an F# let inside a function and outside all functions.

- Inside: local scope and only visible within the body of the function
- Outside: global scope and can be visible from any part of the program

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

- Advantages: Writability and flexibility
- Disadvantages: Readability, reliability

1. Inability to statically check for references to nonlocal variables.

2. Dynamics scoping also makes program difficult to read because the calling sequence of subprograms must be known to determine the meaning of references to nonlocal variables.

3. There's no way to protect local variables from being accessed to by subprograms because local variables of subprogram are all visible to all other executing subprograms, regardless of textual proximity.

4. Accessing to nonlocal variables in dynamic scoping takes far longer than accesses to nonlocal variables when static scoping is used.

22. What are the advantages of named constants?

- Aids to readability and program reliability
- Readability by using pi instead of the constant 3.14159265
- Parameterize a program