Queens College of CUNY
Department of Computer Science
Programming Languages
(CSCI 316)
Winter 2026

Assignment #13
"Final Exam Review Sheet"
Due: January 22, 2026

Introduction and Instructions:

In this assignment, we use "crowd-sourcing" (i.e. class participation and partnership) to populate and prepare this review sheet for tomorrow's Final Exam, just as we did before the Midterm Exam. .

Using your own previously obtained knowledge; referring back to the book, slides, and lecture notes;or using online tools such as internet search and AI chats, complete as many (unanswered) review questions as you can *in this document*. Next to each answer you provide, place your first and last name in parentheses to get credited for your contributions.

To review different areas, it is recommended that you answer at least one question from each of the nine chapters. You may submit an answer to a question already answered - based on independent research or resource - only if all review questions from that chapter have already been answered at least once each.

While the multiple-choice portion of the Final Exam includes core concepts from the entire course, this "Final Exam Review Sheet" consists only of summaries and review questions from the latter part of the course, i.e. Chapters 8-16. For the summaries and review questions for Chapters 1-7, see the Midterm Exam Review Sheet.

You do not need to submit anything in the Google form for this assignment. The assignment will be scored on the basis of the quantity and quality of the review questions you have answered in this document during the in-class activity.

Chapter 8: Statement-Level Control Structures

Summary

Control statements occur in several categories: selection, multiple selection, iterative, and unconditional branching.

The switch statement of the C-based languages is representative of multiple selection statements. The C# version eliminates the reliability problem of its predecessors by disallowing the implicit continuation from a selected segment to the following selectable segment.

A large number of different loop statements have been invented for high level languages. C's for statement is the most flexible iteration statement, although its flexibility leads to some reliability problems.

Most languages have exit statements for their loops; these statements take the place of one of the most common uses of goto statements.

Data-based iterators are loop statements for processing data structures, such as linked lists, hashes, and trees. The for statement of the C-based languages allows the user to create iterators for user-defined data. The foreach statement of Perl and C# is a predefined iterator for standard data structures.

In the contemporary object-oriented languages, iterators for collections are specified with standard interfaces, which are implemented by the designers of the collections.

Ruby includes iterators that are a special form of methods that are sent to various objects. The language predefines iterators for common uses, but also allows user-defined iterators.

The unconditional branch, or goto, has been part of most imperative languages. Its problems have been widely discussed and debated. The current consensus is that it should remain in most languages but that its dangers
should be minimized through programming discipline.
Dijkstra's guarded commands are alternative control statements with positive theoretical characteristics. Although they have not been adopted as the control statements of a language, part of the semantics appear in the concurrency mechanisms of CSP and the function definitions of Haskell.

Review Questions

1. What is the definition of control structure?

A Control Structure is a Control Statement, and the statements whose execution it controls
(Nicholas Roberts)

2. What did Böhm and Jocopini prove about flowcharts?

They were able to show that goto was not necessary to write programs resulting in a major foundation in structured Programming (Andrew Laboy)
It was proven that all algorithms represented by flowcharts can be coded with only two way selections and pretest logical loops. (Jackie Lin)

3. What is the definition of block?

The definition of a block is a group of statements that are treated as single unit of execution (Vincent Comaianni)

4. What is/are the design issue(s) for all selection and iteration control statements?

The main design issues for all selection and iteration control statements are the scope of variables and the evaluation of the control expression. Languages must define where variables declared in the construct are visible and when and how often the control expression is evaluated. (Jahid Hasan)

5. What are the design issues for selection structures?

The design issues for selection structure are:
Form of control expression whether single or multiple statements can be selected.
Support for multi- way selection, restriction on case labels, fall through behaviour and overall
Readability and safety. (Sonu Khadgi)

6. What is unusual about Python's design of compound statements?

What's unusual about Python's design of compound statements is that they have no explicit block delimiters, meaning there are no specific tokens that mark the beginning and end of a block of code, its done by indentation in Python. (Vincent Comaianni)

7. Under what circumstances must an F# selector have an else clause?

If there is a selector expression(if statement) (Ahmed Jawhar)

8. What are the common solutions to the nesting problem for two-way selectors?

Language-specific syntax like requiring braces/compound statements (Java, C++), using indentation (Python), or specific rules (Perl's mandatory compound clauses) aim to define scope and prevent mismatching. (Weitong Chen)

9. What are the design issues for multiple-selection statements?

How segments are specified, and whether more than one segment can execute (Jeaneth Lliguicota)

10. Between what two language characteristics is a trade-off made when deciding whether more than one selectable segment is executed in one execution of a multiple selection statement?

The trade-off is made between efficiency(executing multiple segments can be faster) and readability(executing one segment is safer and easier to understand).(Kaiwen Liu)

11. What is unusual about C's multiple-selection statement?

C's multiple-selection statements are not mutually exclusive by default. Also, case labels must be constant expressions. (Sonu Khadgi)

12. On what previous language was C's switch statement based?

ALGOL68(Junbao Zhang)

13. Explain how C#'s switch statement is safer than that of C. d

They get rid of the implicit "fall thru" of C (If you dont break when you match a case) (Jeremy Haft)

14. What are the design issues for all iterative control statements?

The design issues for all iterative control statements include the method of loop control (pretest vs posttest), form of the loop condition, termination mechanisms, restrictions and scope of loop control variables, evaluation of loop parameters, allowance of side effects, and support for nested or labeled loops. (Arsalan Ansari)

15. What are the design issues for counter-controlled loop statements?

Some design issue for counter-controlled loop statements include:

- What is the type and scope of the variable?
- Can the loop variable/parameters be changed in the loop, and how is it affected?
- Should the loop parameters be evaluated once or per iteration?
- What is the value of the variable after the loop terminates?

(Jackie Lin)

16. What is a pretest loop statement? What is a posttest loop statement?

A pretest loop statement means the conditions that are to be true before execution. A posttest loop are conditions that are true after execution. - (Dariel Urena)

17. What is the difference between the for statement of C++ and that of Java?

Java's for statement is more restrictive and type safe, while C++ 's for loop is more flexible and expressive. In Java, all variables need to be the safe type during declaration. Arbitrary declarations or expressions are not allowed. In C++, condition can be any expression convertible to 'bool'. And non-boolean values like integers and pointers are allowed. (Kaiwen Liu)

18. In what way is C's for statement more flexible than that of many other languages?

It allows any expression in its three parts such as init, test, and increment and it can be used to create user-defined iterators (Jeaneth Lliguicota)

19. What does the range function in Python do?

The range function in Python generates a sequence of numbers, it's most commonly used in for loops. (Oscar Yang)

20. What contemporary languages do not include a goto?

Java, Python…(thasneem mohamed)

21. What are the design issues for logically controlled loop statements?

A logically controlled loop statement uses depiction control based on a Boolean expression
some design issues include

1. Pretest or Posttest
    a. pretest means the condition is tested before the loop runs so if the condition starts as false, it will never execute
    b. posttest means the condition is tested after the loop body runs. So the body runs at least once even if the condition is false at the first run
2. Should the logically controlled loop be a special case of the counting loop statement or a separate statement

(Nicholas Roberts)

22. What is the main reason user-located loop control statements were invented?

To allow the user to dictate the location of a loop and allows for greater flexibility and control. - (Dariel Urena)

23. What are the design issues for user-located loop control mechanisms?

User-located loop control mechanisms raise design issues about where a loop can be exited or restarted. Another issue is whether these controls make the code clear or confusing by breaking the normal loop structure. (Jahid Hasan)

24. What advantage does Java's break statement have over C's break statement? Java's breaks allow you to break out of multiple nested loops at once. (Tony Chen)

25. What are the differences between the break statement of C++ and that of Java?

Java supports breaks to labels where you can exit to outside a loop while in C++ you can only break out of the innermost loop. (Joseph Pasqualicchio)

26. What is a user-defined iteration control?

A user-defined iteration control lets the programmer decide how a loop repeats and when it stops. (Adeel Asaf)

27. What Scheme function implements a multiple selection statement?

Cond: same function as if, elif, and else chain in python. - (Vegus Tao)

28. How does a functional language implement repetition?

Ans - A functional language implements repetition using recursion instead of traditional loop statements. (Adeel Asaf)

29. How are iterators implemented in Ruby?

Ruby iterators are implemented as methods that take blocks, using yield to pass each element to the block (commonly via the Enumerable module) (Gyeonghwan Do)

30. What language predefines iterators that can be explicitly called to iterate over its predefined data structures?

Python — it predefines iterators using the iterator protocol (iter() and next()). (Gyeonghwan Do)

31. What common programming language borrows part of its design from Dijkstra's guarded commands?

Ada (notably in its if and select constructs). (Gyeonghwan Do)

Chapter 9: Subprograms

Summary

Process abstractions are represented in programming languages by subprograms. A subprogram definition describes the actions represented by the subprogram. A subprogram call enacts those actions. A subprogram header identifies a subprogram definition and provides its interface, which is called its protocol.

Formal parameters are the names that subprograms use to refer to the actual parameters given in subprogram calls. In Python and Ruby, array and hash formal parameters are used to support variable numbers of parameters.

JavaScript also supports variable numbers of parameters. Actual parameters can be associated with formal parameters by position or by keyword.

Parameters can have default values.

Subprograms can be either functions, which model mathematical functions and are used to define new operations, or procedures, which define new statements.

Local variables in subprograms can be stack dynamic, providing support for recursion, or static, providing efficiency and history-sensitive local variables.

JavaScript, Python, Ruby, and Swift allow subprogram definitions to be nested.

There are three fundamental semantics models of parameter passing—in mode, out mode, and inout mode—and a number of approaches to implement them. These are pass-by-value, pass-by-result, pass-by-value-result, pass-byreference, and pass-by-name. In most languages, parameters are passed in the run-time stack.

Aliasing can occur when pass-by-reference parameters are used, both among two or more parameters and between a parameter and an accessible nonlocal variable.

Parameters that are multidimensioned arrays pose some issues for the language designer, because the called subprogram needs to know how to compute the storage mapping function for them. This requires more than just the name of the array.

Parameters that are subprogram names provide a necessary service but can be difficult to understand. The opacity lies in the referencing environment that is available when a subprogram that has been passed as a parameter is executed.

C and C++ support pointers to functions. C# has delegates, which are objects that can store references to methods. Delegates can support multicast calls by storing more than one method reference.

Ada, C++, C#, Ruby, and Python allow both subprogram and operator overloading. Subprograms can be overloaded as long as the various versions can be disambiguated by the types of their parameters or returned values.

Function definitions can be used to build additional meanings for operators.

Subprograms in C++, Java 5.0, and C# 2005 can be generic, using parametric polymorphism, so the desired types of their data objects can be passed to the compiler, which then can construct units for the requested types.

The designer of a function facility in a language must decide what restrictions will be placed on the returned values, as well as the number of return values.

A closure is a subprogram and its referencing environment. Closures are useful in languages that allow nested subprograms, are static-scoped, and allow subprograms to be returned from functions and assigned to variables.

A coroutine is a special subprogram that has multiple entries. It can be used to provide interleaved execution of subprograms.

Review Questions

1. What are the three general characteristics of subprograms?
    - Each subprogram has a single entry point
    - The calling program is suspended during execution of the called subprogram
    - Control always returns to the Caller when the called subprograms execution terminates (Nicholas Roberts)

2. What does it mean for a subprogram to be active?
A subprogram is active when it has been called and its execution has started but isn't completed yet (ahmed jawhar)

3. What is given in the header of a subprogram?
The header of a subprogram specifies the subprogram name, its formal parameters (including number, order, and types), and, for functions, the return type. (Arsalan Ansari)
The subprogram header includes the name, kind of subprogram, and the formal parameters. (Jackie Lin)

4. What characteristic of Python subprograms sets them apart from those of other languages?
Python subprograms are first class objects so they
-Can be assigned to variables
-Can be passed as arguments to a function
-Can be returned from functions
-Can be stored in data structures
(Ahmed Jawhar)

5. What languages allow a variable number of parameters?
Python, Ruby, and JavaScript (Jeaneth Lliguicota)

6. What is a Ruby array formal parameter?

A Ruby array formal parameter is a parameter which receives multiple arguments that are packaged into an array, i.e : def f (*args), args is bound to an array, and any additional parameters are stored in said array. (Vincent Comaianni)

7. What is a parameter profile? What is a subprogram protocol?
parameter profile：the number of subprogram,order,and kind of subprogram
subprogram protocol: subprogram parameter and return type(Junbao Zhang)

8. What are formal parameters? What are actual parameters?

Formal parameters are the variables listed in a subprogram's definition that receive values when the subprogram is called. Actual parameters (arguments) are the values or expressions provided in the subprogram call that are passed to the formal parameters. (Jahid Hasan)

9. What are the advantages and disadvantages of keyword parameters?

Advantage of keyword parameters is the readability and order independence.
Disadvantage is that the caller must know parameters names

10. What are the differences between a function and a procedure?
A function returns a value and can be used in expressions. A procedure only perform tasks so it does not return values (Lily Zheng)
A function returns values while a procedure defines new statements; it doesn't return a value.

11. What are the design issues for subprograms?
The design issues for subprograms are local variable allocations, whether or not it will be allocated in the stack or static. As well as parameter passing method and nested definitions. (Jeaneth Lliguicota)
The design issues for subprograms include parameter passing methods, parameter types and number, return values, local variable scope and lifetime, and whether subprograms can be nested or recursive. (Adeel Asaf)

12. What are the advantages and disadvantages of dynamic local variables?
The advantage of dynamic local variables is the support for recursion and the disadvantage is slower access and allocate overhead (Jeaneth Lliguicota)

13. What are the advantages and disadvantages of static local variables?
They are initialized only once They are only accessible within the scope they are declared. They are also persistent upon a function finishing execution. Meaning if I declare (inside a function) static int counter = 0; Then I do counter++, if I leave the function and then reenter it again at a different point, the value is still 1. The value is stored within the scope and the variable isn't "freed." One of the disadvantages is that it could cause something known as a hidden persistent state. This means that we may not know that the state (variable/value stored within) is still alive even after the function's execution is complete! It might be essentially hidden and not obviously realized that this value is remaining the same despite the function ending its execution which may cause odd behavior if the function is called again with the thought that its value was reset or not initialized. (Jeremy Haft).

14. What languages allow subprogram definitions to be nested?
Python, JavaScript (Jeremy Haft)

15. What are the three semantics models of parameter passing?
The three semantics models of parameter passing are pass by value, pass by result, pass by value-result. (Oscar Yang)

16. 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?
Answer:

a) Pass-by-value

- i) Mode
  - 1) In-mode
    - (a) Receives data from the caller
- ii) Advantages
  - 1) Actual parameter cannot be modified by subprogram
  - 2) Efficient access to formal parameter
- iii) Disadvantages
  - 1) Additional storage for copying the parameter
  - 2) Copying large data structures
    - (a) Time-consuming
    - (b) inefficient

b) Pass-by-result
- i) Mode
  - 1) out-mode
    - (a) transmits data back to the caller
- ii) Advantages
  - 1) Avoids value copying cost
    - (a) From actual parameter
      - (i) To formal parameter upon function entry
- iii) Disadvantages
  - 1) Additional storage for formal parameter
    - (a) As a local variable
  - 2) Unpredictable behavior
    - (a) Aliasing collision
      - (i) Same actual parameter passed
        - (1) To two different formal parameters
  - 3) Slow
    - (a) Due to copy-back operation

c) Pass-by-value-result
- i) Mode
  - 1) In-out-mode
    - (a) Receives data and transmits data back
- ii) Advantages
  - 1) Avoids aliasing problem
  - 2) Fast access to formal parameter
    - (a) Ex: local variable
- iii) Disadvantages
  - 1) Additional memory for local copy
  - 2) Slow copying of large parameters
  - 3) Order of parameter evaluation matters

d) Pass-by-reference
- i) Mode
  - 1) In-out-mode
    - (a) Allows both reading and writing
- ii) Advantages
  - 1) Time and space efficiency
  - 2) Original data modified by the subprogram
- iii) Disadvantages
  - 1) Indirecting addressing
    - (a) Leads to slower access to formal parameters
  - 2) Changes to actual parameters
    - (a) Leading to side effects
  - 3) Aliasing

(a) Changes to one formal parameter
    (i) Affects another
(b) Programs becomes hard to…
    (i) Read
    (ii) Debug
    (iii) Verify (Monika Luchowska)

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

Pass by reference uses a variable's address instead of duplicating its value and working with that instead. As such, pass by reference functions would be using aliases to reference other variables which can become exacerbated with global variables in the function. -Kristopher Garcia

18. What is the difference between the way original C and C89 deal with an actual parameter whose type is not identical to that of the corresponding formal parameter?

Answer:

a) C
    i) No type checking
        1) No warning or error
        2) Undefined behavior at runtime
    ii) Default argument promotions
        1) Ex1: char and short -> promoted into -> int
        2) Ex2: float -> promoted into -> double
b) C89
    i) Compile-time type checking
    ii) Implicit conversion
        1) Type casting
            (a) Of actual parameter's val
                (i) To formal parameter type
    iii) Mismatch diagnostics
        1) Types are incompatible
            (a) Warning or error (Monika Luchowska)

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

The two fundamental design considerations for parameter-passing methods are efficiency, which concerns about the time and the space cost of passing parameter, and safety/clarity, which concerns about side affect, aliasing, and how parameter passing affects program reasoning.(Kaiwen Liu)

20. Describe the problem of passing multidimensioned arrays as parameters.

The compiler must know array dimension sizes to build address mapping. (Shiwlee Rahman)

21. What is the name of the parameter-passing method used in Ruby?

Pass-by-assignment. (Shiwlee Rahman)

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

1. Type checking
2. Referencing environment binding (Shiwlee Rahman)

23. Define shallow and deep binding for referencing environments of subprograms that have been passed as parameters.

1. Shallow binding: environment of the call
2. Deep binding: environment of the definition (Shiwlee Rahman)

24. What is an overloaded subprogram?

A subprogram that shares the same name as another subprogram but has different parameters. - (Dariel Urena)

25. What is parametric polymorphism?
By using type variables, you can allow functions and data types to work with values of any type. - (Dariel Urena)

26. What causes a C++ template function to be instantiated?
C++ template is instantiated when the compiler needs a concrete version of the code to exist in memory. This happens in two ways:
Implicit Instantiation (Automatic): Triggered when you use the template, such as calling the function or taking its address. The compiler "fills in" the template arguments based on your usage.
Explicit Instantiation (Manual): Triggered when you command the compiler to create it using specific syntax (e.g., template void func<int>(int);), even if the function isn't called in that file.
Crucial Rule: The compiler must be able to see the entire source code (the definition) of the template at the moment of use, which is why templates are almost always placed in header files. (Ameen Alhubaishi)

27. In what fundamental ways do the generic parameters to a Java 5.0generic method differ from those of C++ methods?
Java 5.0 generic parameters differ from C++ parameters in that Java uses type erasure, so only one version of a method/class is generated and generic types are not available at runtime, whereas C++ generates a separate version for each type at compile time.(Kaiwen Liu)

28. If a Java 5.0 method returns a generic type, what type of object is actually returned?
After compilation, the generic type becomes “Object” and implicitly casted when called. (Tony Chen)

29. If a Java 5.0 generic method is called with three different generic parameters, how many versions of the method will be generated by the compiler?
Only one version of a Java 5.0 generic method is generated by the compiler, because type erasure removes generic type information at compile time.(Kaiwen Liu)

30. What are the design issues for functions?
1. Functional side effects are allowed or not.
2. Number of values returned by function.
3. Types of values returned by the function.

31. Name two languages that allow multiple values to be returned from a function.
Go and Python - (Vegus Tao)

32. What exactly is a delegate?
- A delegate is a type safe reference to a method. It allows you to pass methods as parameters, store them in variables, and call them dynamically. (Oscar Yang)

33. What is the main drawback of generic functions in F#?
The main drawback of generic functions in F# is limited support for generic arithmetic.
Unlike some other languages, standard F# generic functions cannot use operators like + or - across different types by default. If you write a generic function to add two numbers, F# will "lock" it to a specific type (usually int) unless you use a more complex feature called Statically Resolved Type Parameters (SRTP).

Key Issues:
Type Lock-in: The compiler's type inference often forces a "generic" function to become specific to one type (e.g., only working for int and not float).
Complexity: To fix this, you must use the inline keyword and "hat" types (e.g., ^T), which are much harder to read and maintain.
Binary Bloat: Using the inline workaround causes the compiler to copy the function code into every location it is called, increasing the final file size.(Ameen Alhubaishi)

34. What is a closure?
A closure is a function along with the environment that remembers the values of variables it uses from where it was defined. (Adeel Asaf)

35. What are the language characteristics that make closures useful?
The languages characteristics that make closure useful are:
Nested loops, First class functions, Lexical (static) scoping, and support for accessing non-local variables.(Sonu Khadgi)

36. What languages allow the user to overload operators?
C++, Python, Scala, Kotlin, and Swift (Adeel Asaf)

37. In what ways are coroutines different from conventional subprograms?
Coroutines have multiple checkpoints, can suspend execution and later resume suspension, and transfer control cooperatively rather than strictly through call-and-return. Unlike conventional subprograms use one-way -call -and -return. (Sonu Khadgi)

Chapter 10: Implementing Subprograms

Summary

Subprogram linkage semantics requires many actions by the implementation.

In the case of “simple” subprograms, these actions are relatively uncomplicated. At the call, the status of execution must be saved, parameters and the return address must be passed to the called subprogram, and control must be transferred. At the return, the values of pass-by-result and pass-byvalue-result parameters must be transferred back, as well as the return value

if it is a function, execution status must be restored, and control transferred back to the caller. In languages with stack-dynamic local variables and nested subprograms, subprogram linkage is more complex. There may be more than one activation record instance, those instances must be stored on the run-time stack, and static and dynamic links must be maintained in the activation record instances. The static link is to allow references to nonlocal variables in static-scoped languages.

Subprograms in languages with stack-dynamic local variables and nested subprograms have two components: the actual code, which is static, and the activation record, which is stack dynamic. Activation record instances contain the formal parameters and local variables, among other things.

Access to nonlocal variables is implemented with a chain of static parent pointers.

Access to nonlocal variables in a dynamic-scoped language can be implemented by use of the dynamic chain or through some central variable table method. Dynamic chains provide slow accesses but fast calls and returns. The central table methods provide fast accesses but slow calls and returns.

Review Questions

1. What is the definition used in this chapter for “simple” subprograms?
“Simple” subprograms:
    1. Cannot be nested
    2. All of their local variables are static

Therefore, if all their local variables are static they can not have a stack instead they have an activation record.

(Jeaneth Lliguicota)

2. Which of the caller or callee saves execution status information?
The callee saves execution status information (Ahmed Jawhar)

3. What must be stored for the linkage to a subprogram?
The return address , parameters, local variable, and the caller's execution status within an Activation Record. - (Dariel Urena)

4. What is the task of a linker?
A linker combines object modules, resolves external references, links libraries, performs address relocation, and produces an executable program. (Arsalan Ansari).

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

a) Implicit allocation & deallocation
    i) Stack-dynamic locals
        1) Compilers must generate code
            (a) to push new activation records (AR)
                (i) Onto the stack
    ii) Simple Subprograms
        1) Memory is fixed / static
b) Support for recursion & multiple instances
    i) Stack-dynamic locals
        1) Call themselves
            (a) Multiple active calls exist simultaneously
            (b) Own AR on the stack
            (c) Complex linking happening!
    ii) Simple subprograms
        1) Don't have to worry about this (Monika Luchowska)

6. What is the difference between an activation record and an activation record instance?
Activation record is the template/layout for a function call. It describes what fields exist. The activation record instance is a runtime occurrence of that activation record. (Oscar Yang)

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

The return address, dynamic link, and parameters are placed at the bottom of the activation record so they have a fixed, predictable location that is known at call time. This allows the calling and called subprograms to access control information easily, while local variables—whose size may vary—are placed above them. (Jahid Hasan)

8. What kind of machines often use registers to pass parameters?
Machines which often use registers to pass parameters are register machines, and often machines which use RISC architectures. They pass function parameters in CPU registers rather than on the stack. (Vincent Comaianni)

9. What are the two steps in locating a nonlocal variable in a static-scoped language with stack-dynamic local variables and nested subprograms?
First we find the correct activation record which is usually the hardest part. Then We determine the offset within the activation record.(Andrew Laboy)

10. Define static chain, static_depth, nesting_depth, and chain_offset.

Static chain: a Static Chain is a chain of static links that connect certain activation record instances. The static chain from an activation record instance connects it to all of its static ancestors
Static_depth: Static_Depth is an integer associated with a static scope whose value is the depth of nesting of that scope
nesting_depth: the nesting depth is how many levels deep a subprogram is inside other subprograms. In this example below, the main program has a nesting depth of 0, procedure A has a nesting depth of 1, and procedure B has a nesting depth of 2
Level 0: Main program
    Level 1: procedure A
        Level 2: procedure B
Chain_offset: The Chain_Offset is the number of links you have to follow in a static chain to reach the correct enclosing scope's activation record. In simple terms, it tells you how far up the nested cope ladder you need to go in order to find a variable that was declared in an outer function
(Nicholas Roberts)

11. What is an EP, and what is its purpose?
The environment pointer always points to the base of the activation record instance of the current executed program unit.(Jackie Lin)

12. How are references to variables represented in the static-chain method? Represented using a pair of static links and offsets within the activation record. (Tony Chen)

13. Name three widely used programming languages that do not allow nested subprograms.
Three widely used programming languages that do not allow nested subprograms are C, C++, and Java. (Kaiwen Liu)

14. What are the two potential problems with the static-chain method?
The two potential problems with static-chain method are:
Nonlocal variables can be slow because it may require several static links at runtime.
Extra storage and maintenance overhead as it requires to store and manage static-chain pointers in each activation record. (Sonu Khadgi)

15. Explain the two methods of implementing blocks.
Blocks can be implemented in two ways: using static allocation, where memory for variables is fixed at compile time, and using dynamic allocation, where memory is created at runtime each time the block is entered and destroyed when it exits. (Adeel Asaf)

16. Describe the deep-access method of implementing dynamic scoping
Answer:Nonlocal variables are found by searching the dynamic chain at run time. (Shiwlee Rahman)

17. Describe the shallow-access method of implementing dynamic scoping.
Answer :Uses a central table or stack per variable name, allowing direct access (Shiwlee Rahman)

18. What are the two differences between the deep-access method for nonlocal access in dynamic-scoped languages and the static-chain method for static-scoped languages?
Answer:

- Deep access searches the dynamic chain, static chain follows static links
- Dynamic access cannot be determined at compile time; static can (Shiwlee Rahman)

19. Compare the efficiency of the deep-access method to that of the shallow-access method, in terms of both calls and nonlocal accesses.

Answer: 1.Deep access: faster calls, slower nonlocal access
2. Shallow access: slower calls, faster nonlocal access (Shiwlee Rahman)

Chapter 11: Abstract Data Types and Encapsulation Constructs

Summary

The concept of abstract data types, and their use in program design, was a milestone in the development of programming as an engineering discipline. Although the concept is relatively simple, its use did not become convenient and safe until languages were designed to support it.

The two primary features of abstract data types are the packaging of data objects with their associated operations and information hiding. A language may support abstract data types directly or simulate them with more general encapsulations.

C++ data abstraction is provided by classes. Classes are types, and instances can be either stack or heap dynamic. A member function (method) can have its complete definition appear in the class or have only the protocol given in the class and the definition placed in another file, which can be separately compiled. C++ classes can have two clauses, each prefixed with an access modifier: private or public. Both constructors and destructors can be given in class definitions.

Heap-allocated objects must be explicitly deallocated with delete.

Java data abstractions are similar to those of C++, except all Java objects are allocated from the heap and are accessed through reference variables. Also, all objects are garbage collected. Rather than having access modifiers attached to clauses, in Java the modifiers appear on individual declarations (or definitions).

C# supports abstract data types with both classes and structs. Its structs are value types and do not support inheritance. C# classes are similar to those of Java.
Ruby supports abstract data types with its classes. Ruby's classes differ from those of most other languages in that they are dynamic—members can be added, deleted, or changed during execution.

C++, Java 5.0, and C# 2005 allow their abstract data types to be - parameterized—Ada through its generic packages,

C++ through its templated classes, and Java 5.0 and C# through their collection classes and interfaces and user-defined generic classes.

To support the construction of large programs, some contemporary languages include multiple-type encapsulation constructs, which can contain a collection of logically related types. An encapsulation may also provide access control to its entities. Encapsulations provide the programmer with a method of organizing programs that also facilitates recompilation.

C++, C#, Java, and Ruby provide naming encapsulations. For Ada and Java, they are named packages; for C++ and C#, they are namespaces; for Ruby, they are modules. Partially because of the availability of packages, Java does not have friend functions or friend classes.

Review Questions

1. What are the two kinds of abstractions in programming languages? → (thasneem mohamed) Process abstraction and data abstraction

2. Define abstract data type.--> is a data structure that is defined by its operators and not by implementations(thasneem mohamed)

3. What are the advantages of the two parts of the definition of abstract data type?--> reliability by hiding data representation, less name conflict, provides a method of program organization, and a separate compilation (thasneem mohamed)

4. What are the language design requirements for a language that supports abstract data types?
Encapsulation and information hiding. - (Dariel Urena)

5. What are the language design issues for abstract data types?
The main design issues for abstract data types include encapsulation and information hiding, scope and visibility control, mechanism for defining ADTs, instantiation and lifetime management, operation binding, parameter passing, and support for generics and operator overloading.(Kaiwen Liu)

6. From where are C++ objects allocated?
C++ objects can be allocated from three different places:
-Static Storage
-Stack (automatic storage)
-Heap (dynamic storage)
(Ahmed Jawhar)

7. In what different places can the definition of a C++ member function appear?
Either inside the class definition or outside the class definition (ahmed jawhar)

8. What is the purpose of a C++ constructor?
Ans - The purpose of a C++ constructor is to initialize an object by setting its data members when the object is created. (Adeel Asaf)

To create an object based on given input parameters (Jeremy Haft)
9. What are the legal return types of a constructor- ?
10. Where are all Java methods defined?
All Java methods are defined inside classes (or interfaces); Java does not support free-standing functions. (Arsalan Ansari).

11. How are C++ class instances created?
C++ class instances are created in two different ways depending on where they are stored in memory:

1. Stack-Dynamic (Automatic) Allocation: Instances are created on the stack, often as a local variables within a function
2. Heap Dynamic Allocation: Instances are created on the heap using the new operator

(Jeaneth Lliguicota)

12. From where are Java class instances allocated?
From the heap (large pool of memory used for dynamic allocation) - (Vegus Tao)

13. Why does Java not have destructors? Java has an automatic GC. (Tony Chen)

14. Where are all Java methods defined? In their respective classes/interface. (Tony Chen)

15. Where are Java classes allocated?
In the metaspace (memory managed by the OS) - (Vegus Tao)

16. Why are destructors not as frequently needed in Java as they are in C++? Java prioritizes memory safety over determinism hence the usage of an automatic GC. (Tony Chen)

17. What is a friend function? What is a friend class?
A friend function is a non-member function that is allowed to access private or protected members of a class by being declared with the friend keyword inside that class. A friend class is a class whose member functions can access the private and protected members of another class that declares it as a friend. (Oscar Yang)

18. What is one reason Java does not have friend functions or friend classes?
Java has less need for explicit friend declarations. (Andrew Laboy)

19. Describe the fundamental differences between C# structs and its classes.
C# Structs are more commonly used as a collection of data rather than used for behaviors. Classes are used for functionality and methods. (Jeremy Haft)

20. How is a struct object in C# created?
A struct object in C# can be created by instantiating a value type which is defined with the struct. An example of this is: struct h { public int hi; } , then instantiating it by: h ourstruct; ourstruct.hi = 2; (Vincent Comaianni)

21. Explain the three reasons accessors to private types are better than making the types public.
1. They protect the data
2. you can change the internal representation without breaking code
3. allows extra flexibility without exposing internals

(Nicholas Roberts)

22. What are the differences between a C++ struct and a C# struct?
The differences between a C++ struct and a C# struct are:
C++ struct supports inheritance whereas C# struct does not.
C++ struct can define and constructions whereas C# can not define a parameterless constructor.
C++ struct is a value or a reference type depending on the usage whereas C# struct is strictly a value type. (Sonu Khadgi)

23. What is the name of all Ruby constructors?
Initialize, if more constructors are required they must have different names and must explicitly call new.(Jackie Lin)

24. What is the fundamental difference between the classes of Ruby and those of C++ and Java?
Ruby classes are dynamic and can be modified at run time. (Shiwlee Rahman)

25. How are instances of C++ template classes created?
By instantiating the template with a type, e.g. Stack<int>. (Shiwlee Rahman)

26. Describe the two problems that appear in the construction of large programs that led to the development of encapsulation constructs.
1. Need for organization beyond subprograms
2. Need for separate compilation (Shiwlee Rahman)

27. What problems can occur using C to define abstract data types?
Using C to define abstract data types can cause problems because data and operations are not fully encapsulated, so the data can be accessed or modified directly. (Adeel Asaf)

28. What is a C++ namespace, and what is its purpose?
Have code that doesn't have conflicting names (Jeremy Haft)

29. What is a Java package, and what is its purpose?

A Java package is a group of related classes and interfaces organized together under a common name. Its purpose is to organize code, avoid name conflicts, and control access to classes. (Jahid Hasan)

30. Describe a .NET assembly.
A .NET assembly is the basic unit of deployment and execution in the .NET framework. It contains compiled code (IL), metadata describing types and references, and optionally resources, all packaged into a single file such as a .dll or .exe. (Jahid Hasan)

31. What elements can appear in a Ruby module?
Constants, Methods, Other modules, and Classes (Ahmed Jawhar)

Chapter 12: Support for Object-Oriented Programming

Summary

Object-oriented programming is based on three fundamental concepts: abstract data types, inheritance, and dynamic binding. Object-oriented programming languages support the paradigm with classes, methods, objects, and message passing.

The discussion of object-oriented programming languages in this chapter revolves around seven design issues: exclusivity of objects, subclasses and subtypes, type checking and polymorphism, single and multiple inheritance, dynamic binding, explicit or implicit deallocation of objects, and nested classes.

Smalltalk is a pure object-oriented language—everything is an object and all computation is accomplished through message passing. All type checking and binding of messages to methods is dynamic, and all inheritance is single.
Smalltalk has no explicit deallocation operation.

C++ provides support for data abstraction, inheritance, and optional dynamic binding of messages to methods, along with all of the conventional features of C. This means that it has two distinct type systems. C++ provides multiple inheritance and explicit object deallocation. It includes a variety of access controls for the entities in classes, some of which prevent subclasses from being subtypes. Both constructor and destructor methods can be included in classes; both are usually implicitly called.

While Smalltalk's dynamic type binding provides somewhat more programming flexibility than the hybrid language C++, it is far less efficient. Unlike C++, Java is not a hybrid language; it is meant to support only objectoriented programming. Java has both primitive scalar types and classes. All objects are allocated from the heap and are accessed through reference variables. There is no explicit object deallocation operation—garbage collection is used. The only subprograms are methods, and they can be called only through objects or classes. Only single inheritance is directly supported, although a kind of multiple inheritance is possible using interfaces. All binding of messages to methods is dynamic, except in the case of methods that cannot be overridden. In addition to classes, Java includes packages as a second encapsulation construct.

C#, which is based on C++ and Java, supports object-oriented programming.

Objects can be instantiated from either classes or structs. The struct objects are stack dynamic and do not support inheritance. Methods in a derived class can call the hidden methods of the parent class by including base on the method name. Methods that can be overridden must be marked virtual, and the overriding methods must be marked with override. All classes (and all primitives) are derived from Object.
Ruby is an object-oriented scripting language in which all data are objects. As with Smalltalk, all objects are heap allocated and all variables are typeless references to objects. All constructors are named initialize. All instance data are

private, but getter and setter methods can be easily included. The collection of all instance variables for which access methods have been provided forms the public interface to the class. Such instance data are called attributes. Ruby classes are dynamic in the sense that they are executable and can be changed at any time. Ruby supports only single inheritance.

The instance variables of a class are stored in a CIR, the structure of which is static. Subclasses have their own CIRs, as well as the CIR of their parent class. Dynamic binding is supported with a virtual method table, which stores pointers to specific methods. Multiple inheritance greatly complicates the implementation of CIRs and virtual method tables.

Reflection is a process by which a program can access its classes and types and possibly dynamically change them to affect program behavior. One of the primary uses of reflection is in the construction of software tools, such as visual program construction tools, debuggers, and test systems. The class and type information, called metadata, is collected by the compiler or interpreter for the language. In Java, the class information, such as the methods of the class, are available in the Class object of a class. Support for reflection in C#, which is similar to that of Java, is available in the System.Reflection namespace.

Review Questions

1. Describe the three characteristic features of object-oriented languages.
1. Encapsulation (combining methods and data)
2. Inheritance (creating new class from existing)
3. Polymorphism (allowing the same operation to behave differently for different objects)
(Adeel Asaf)

2. What is the difference between a class variable and an instance variable?
The difference between a class variable and an instance variable is that a class variable is shared by all instances of a class whereas an instance variable belongs to and has a separate copy for each individual object. (Sonu Khadgi)

A class variable is static and shared while instance variables are specific to individual object. (Jeaneth Lliguicota)

3. What is multiple inheritance?
Multiple inheritance is a feature of some programming languages where a class can inherit from more than one parent (base) class. (Arsalan Ansari)

4. What is a polymorphic variable?
It points/reference to objects of the and objects of its descendants(Jackie Lin)

5. What is an overriding method?
is when a child class has its own implementation of a method that's already in the parent class. (thasneem mohamed)

6. Describe a situation where dynamic binding is a great advantage over static binding.
Dynamic binding allows polymorphism, where if it had to search through a list of objects that share the same base called shape, but each shape can have a different specific shape like circle, square, triangle, etc. dynamic binding allows it to be iterated through in a simple loop. - (Vegus Tao)

7. What is a virtual method?
A class method that is designed to be optionally redefined to allow for runtime polymorphism. - (Dariel Urena)

8. What is an abstract method? What is an abstract class?
Abstract method is the one that does not include definitions, only names and rules, while the abstract class is the one that cannot be used to make objects and includes at least one virtual method. -->(thasneem mohamed)

An abstract method is a method that is declared but does not contain a body. It must be overridden with an implementation from a subclass. An abstract class is one which cannot be initialized and must be extended and can contain both abstracted and implemented classes. (Joseph Pasqualicchio)

9. Describe briefly the seven design issues used in this chapter for object oriented languages.
1. Object creation and destruction (decides how objects are made and how memory is freed)
2. Class definition (Decides how classes are written)
3. Inheritance (decides one class can reuse or extend class)
4. Method binding (Decides when method call is linked)
5. Access control (Decides how data and methods are hidden)
6. Multiple inheritance (decided whether class can inherit from more than one class)
7. Polymorphism. (decided how one interface can work with different object behaviors) (Adeel Asaf)

10. What is a nesting class?
A nested class is a class within another class. It increases encapsulation. C++ and Java allow you to use it. (Jeremy Haft)

11. What is the message protocol of an object?
SOAP - XML based standard for object communication (Jeremy Haft)
An object's message protocol is the collection of messages it understands and the actions associated with them. (Sonu Khadgi)

12. From where are Smalltalk objects allocated?
Smalltalk objects are allocated from the heap and managed by garbage collection.(Kaiwen Liu)

13. Explain how Smalltalk messages are bound to methods. When does this take place?
All computation is through message. Replies to messages have the form of objects and are used to return requested or computed information or only to confirm that the requested service has been computed. (Andrew Laboy)

14. What type checking is done in Smalltalk? When does it take place?
The only type of checking in Smalltalk is dynamic, and the only type error occurs when a message is sent to an object that has no matching method., either locally or through inheritance. (Andrew Laboy)

15. What kind of inheritance, single or multiple, does Smalltalk support?
Single inheritance only (Shiwlee Rahman)

16. What are the two most important effects that Smalltalk has had on computing?
1.Advancement of object-oriented programming
2. Introduction of the graphical user interface (GUI) (Shiwlee Rahman)

17. In essence, all Smalltalk variables are of a single type. What is that type?
Object (Shiwlee Rahman)

18. From where can C++ objects be allocated?
Answer: Stack, Heap, Static storage (Shiwlee Rahman)

19. How are C++ heap-allocated objects deallocated?
Delete keyword (Jeremy Haft)

20. Are all C++ subclasses subtypes? If so, explain. If not, why not?
No. Private inheritance can prevent a subclass from being a subtype. (Shiwlee)

21. Under what circumstances is a C++ method call statically bound to a method?
Circumstances in which a C++ method call is statically bound to a method:

1. Method is not virtual, so binding depends only on static type
2. Call is made on an object, not through a pointer or reference
3. The method is declared as final, which prevents overrides or dynamic dispatch
4. The method is static, so no runtime dispatch mechanism can apply
5. The call occurs inside a constructor or destructor, therefore disabling virtual dispatch.

Dispatch is how a language selects which method implementation to execute, a.k.a the actual code executed when a method is called. There are five different types of dispatch: 1. Static dispatch, which is when method binding occurs at compile time using the declared types, 2. Dynamic dispatch which occurs at runtime and is based on an object's actual type 3. Single dispatch which makes selection depend on the object receiving only. 4. Multiple dispatch, where selection depends on runtime types. 5. Manual dispatch, which is where the programmer explicitly selects which method implementation to execute.
(Vincent Comaianni)

22. What drawback is there to allowing designers to specify which methods can be statically bound?
Allowing designers to specify static binding (binding methods to the class, not an object) creates tight coupling, hinders testability (especially mocking), breaks polymorphism, creates hidden global state, and complicates maintenance by making dependencies implicit, leading to harder-to-understand, less flexible, and bug-prone code. (Weitong Chen)

23. What are the differences between private and public derivations in C++?
The public and protected members of a base class are also public and protected respectively, in a public derived class. In a private derived class both the public and protected members of the base class are private. (Andrew Laboy)

24. What is a friend function in C++?
Friends in c++ allow other classes to access the private/protected primitives/objects in a class.(Jeremy Haft)

25. What is a pure virtual function in C++?
A pure virtual function or pure virtual method is a virtual function that is required to be implemented by a derived class if the derived class is not abstract. (Andrew Laboy)

26. How are parameters sent to a superclass's constructor in C++?
A constructor initialization list will execute the base class constructor before the derived class'. E.g

```
class Derived : public base {
public:
Derived(int x, int y) : Base(x){     }};
```

-Kristopher Garcia

27. What is the single most important practical difference between Smalltalk and C++?
C++ is multi-paradigm. Smalltalk isn't. Smalltalk is purely object oriented. (Jeremy Haft)

28. How is the type system of Java different from that of C++?

29. From where can Java objects be allocated?
Java objects are allocated from the heap memory. (Joseph Pasqualicchio)

30. What is boxing?
Boxing is a pattern used to convert a variable from its primitive type into a class wrapper of that primitive. (Joseph Pasqualicchio)
An example of this can be:

```
int x = 1;
Integer y = x;
```

31. How are Java objects deallocated?
In java, objects are automatically deallocated by the garbage collector - you do not need to allocate and deallocate memory like you do in other languages like C++. (Joseph Pasqualicchio)

32. Are all Java subclasses subtypes?
Yes, every subclass is a subtype of its superclass. (Joseph Pasqualicchio)

33. How are superclass constructors called in Java?
In the constructor of the subclass, super(input parameters found within the superclass constructor). (Joseph Pasqualicchio)

34. Under what circumstances is a Java method call statically bound to a method?
A java method call is statically bound to a method when it cannot be overridden. This is determined at compile time when looking for keyword modifiers like static, final, and private where the access scope is known ahead of time. (Joseph Pasqualicchio)

35. In what way do overriding methods in C# syntactically differ from their counterparts in C++?
In C# ,a method must be explicitly declared with override whereas in C++, the overriding occurs implicitly when a derived-class method has the same signatures as a virtual based -class method. (Sonu Khadgi)

36. How can the parent version of an inherited method that is overridden in a subclass be called in that subclass in C#?
A subclass in C# calls the overridden parent method using base.methodName(). (Sonu Khadgi)

37. How does Ruby implement primitive types, such as those for integer and floating-point data?
In Ruby, "primitive" data types like integers and floating-point numbers are not true primitives in the C/C++/Java sense, but are all objects with methods and an object ID. However, internally, Ruby uses performance optimizations for integers by implementing them as immediates, while floating-point numbers are typically heap-allocated objects corresponding to the native architecture's double-precision format. (Weitong Chen)

38. How are getter methods defined in a Ruby class?
The getter methods are defined with attr_reader or by explicitly returning the instance variable in a Ruby class. (Sonu khadgi)

39. What access controls does Ruby support for instance variables?
Ruby does not support access control of instance variables; they are always private and can be only accessed directly within the object's methods.(Sonu Khadgi)

40. What access controls does Ruby support for methods? → (Thasneem Mohamed) can be either private, public, or protected. And it's private for all data by default.

41. Are all Ruby subclasses subtypes?
A subclass often acts as a subtype, but not always. - (Dariel Urena)

42. Does Ruby support multiple inheritance?
No, Ruby does not support multiple inheritances. However, by using a mechanism called mixin, you can achieve something similar. - (Dariel Urena)

43. What does reflection allow a program to do?
It allows the programmer to modify the program's structure and behavior during runtime. - (Dariel Urena)

44. In the context of reflection, what is metadata?

Types and structure of a program (Jeremy Haft).

45. What is introspection?
In Object Oriented Programming, introspection means an object can examine its own type and structure at runtime (Nicholas Roberts)

46. What is intercession?
In Object Oriented Programming, intercession is the ability to modify an object's structure at runtime. If introspection allows the object to examine its type, then intercession is what allows it to actually change.
(Nicholas Roberts)

47.
java.lang.Class (Jeremy Haft)

48. For what is the Java name extension .class used?

- The Java name extension .class is used for compiled Java bytecode. (Oscar Yang)

49. What does the Java getMethods method do? Returns a collection of all available methods for a specific class. (Tony Chen)

50. For what is the C# namespace System.Reflection.Emit used?
System.Reflection.Emit in C# is used for runtime code generation. (Jude Pierre)

Chapter 13: Concurrency

Summary

Concurrent execution can be at the instruction, statement, or subprogram level. We use the phrase physical concurrency when multiple processors are actually used to execute concurrent units. If concurrent units are executed on a single processor, we use the term logical concurrency. The underlying conceptual model of all concurrency can be referred to as logical concurrency.
Most multiprocessor computers fall into one of two broad categories—SIMD or MIMD. MIMD computers can be distributed.
Languages that support subprogram-level concurrency must provide two fundamental capabilities: mutually exclusive access to shared data structures (competition synchronization) and cooperation among tasks (cooperation synchronization).
Tasks can be in any one of five different states: new, ready, running, blocked, or dead.
Rather than designing language constructs for supporting concurrency, sometimes libraries, such as OpenMP, are used.
The design issues for language support for concurrency are how competition and cooperation synchronization are provided, how an application can influence task scheduling, how and when tasks start and end their executions,
and how and when they are created.
A semaphore is a data structure consisting of an integer and a task description queue. Semaphores can be used to provide both competition and cooperation
synchronization among concurrent tasks. It is easy to use semaphores incorrectly, resulting in errors that cannot be detected by the compiler, linker, or run-time system.
Monitors are data abstractions that provide a natural way of providing mutually exclusive access to data shared among tasks. They are supported by several programming languages, among them Ada, Java, and C#.
Cooperation synchronization in languages with monitors must be provided with some form of semaphores.

The underlying concept of the message-passing model of concurrency is that tasks send each other messages to synchronize their execution.
Ada provides complex but effective constructs, based on the message-passing model, for concurrency. Ada's tasks are heavyweight tasks. Tasks communicate with each other through the rendezvous mechanism, which is synchronous message passing. A rendezvous is the action of a task accepting a message sent by another task. Ada includes both simple and complicated methods of controlling the occurrences of rendezvous among tasks.

Ada 95+ includes additional capabilities for the support of concurrency, primarily protected objects. Ada 95+ supports monitors in two ways, with tasks and with protected objects.

Java supports lightweight concurrent units in a relatively simple but effective way. Any class that either inherits from Thread or implements Runnable can override a method named run and have that method's code executed concurrently with other such methods and with the main program.

Competition synchronization is specified by defining methods that access shared data to be implicitly synchronized. Small sections of code can also be implicitly synchronized. A class whose methods are all synchronized is a monitor.

Cooperation synchronization is implemented with the methods wait, notify, and notifyAll. The Thread class also provides the sleep, yield, join, and interrupt methods.

Java has direct support for counting semaphores through its Semaphore class and its acquire and release methods. It also had some classes for providing nonblocking atomic operations, such as addition, increment, and decrement operations for integers. Java also provides explicit locks with the Lock interface and ReentrantLock class and its lock and unlock methods. In addition to implicit synchronization using synchronized, Java provides implicit nonblocking synchronization of int, long, and boolean type variables, as well as references and arrays. In these cases, atomic getters, setters, add, increment, and decrement operations are provided.

C#'s support for concurrency is based on that of Java but is slightly more sophisticated. Any method can be run in a thread. Both actor and server threads are supported. All threads are controlled through associated delegates.
Server threads can be synchronously called with Invoke or asynchronously called with BeginInvoke. A callback method address can be sent to the called thread. Three kinds of thread synchronization are supported with the Interlocked class, which provides atomic increment and decrement operations, the Monitor class, and the lock statement.

All .NET languages have the use of the generic concurrent data structures for stacks, queues, and bags, for which competition synchronization is implicit.

Multi-LISP extends Scheme slightly to allow the programmer to inform the implementation about program parts that can be executed concurrently

Concurrent ML extends ML to support a form of threads and a form of synchronous message passing among those threads. This message passing is designed with channels. F# programs have access to all of the .NET support classes for concurrency. Data shared among threads that is mutable can have access synchronized.

High-Performance Fortran includes statements for specifying how data is to be distributed over the memory units connected to multiple processors. Also included are statements for specifying collections of statements that can be executed concurrently.

Review Questions

1. What are the three possible levels of concurrency in programs?
-Instruction level concurrency
-Task level concurrency

- Program level concurrency
(Ahmed Jawhar)

2. Describe the logical architecture of an SIMD computer.
In an SIMD computer, each processor has its own local memory. One processor controls the operation of the other processors. Because all of the processors, except the controller, execute the same instruction at the same time, no synchronization is required in the software. (Andrew Laboy)

3. Describe the logical architecture of an MIMD computer.
Each processor in an MIMD computer executes its own instruction stream. MIMD computers can appear in two distinct configurations: distributed and shared memory systems. The distributed MIMD machines, in which each processor has its own memory, can be either built in a single chassis or distributed, perhaps over a large area. The shared memory MIMD machines obviously must provide some means of synchronization to prevent memory access clashes. (Andrew Laboy)

4. What level of program concurrency is best supported by SIMD computers?
Instruction concurrency level (Andrew Laboy)
5. What level of program concurrency is best supported by MIMD computers?
Unit concurrency level (Andrew Laboy)
6. Describe the logical architecture of a vector processor.
A vector processor's logical architecture centers on Single Instruction, Multiple Data (SIMD) principles, featuring large, deep vector registers to hold arrays of data, specialized vector functional units (ALUs) that operate in deep pipelines, and a Vector Control Unit that orchestrates the same instruction across all data elements in parallel, enabling high-performance data-level parallelism for scientific and data-intensive tasks. (Weitong Chen)

7. What is the difference between physical and logical concurrency?
Physical concurrency have multiple independent processors, while logical concurrency is time sharing on one processor(Jackie Lin)

8. What is a thread of control in a program?
A thread of control in a program is a sequence of program points that reaches as control flow through program.
(Jeaneth Lliguicota)

9. Why are coroutines called quasi-concurrent?
Coroutines are called quasi-concurrent because it can provide the illusion of concurrency by switching back and forth, making it seem as though it is working on multiple processes. - (Dariel Urena)

10. What is a multithreaded program?
A multithreaded program is a program that allows multiple threads to execute concurrently within the same process, sharing memory but executing independently.(Kaiwen Liu)

11. What are four reasons for studying language support for concurrency?
A. Understanding how different languages take advantage of the hardware/middleware to manage such tasks.
B. Understanding the differences between operating systems and how your code running on them might impact performance
C. Understanding which language would be best handled to use for a program that needs to run multiple tasks at once
D. Understanding how a program can avoid issues like race conditions. Etc…
(Jeremy Haft)
12. What is a heavyweight task? What is a lightweight task?
Heavyweight task: A heavyweight task is a task that executes in their own address space
Lightweight task: a Lightweight task is a task that runs in the same address space. This is typically more efficient.
A task is disjoint if it does not communicate with or affect the execution of any other task in the program in any way.
(Nicholas Roberts)

13. Define task, synchronization, competition and cooperation synchronization, liveness, race condition, and deadlock.
Task: A task (Also called a process or thread) is a program unit that can be in concurrent execution with other program units.
Synchronization: A mechanism that controls the order in which tasks execute.
Competition Synchronization: Two or more tasks need to use a resource that cannot be shared
Cooperation Synchronization: Two or more tasks work together throughout their process and must depend on each other's execution to continue their own execution.
Race condition: a Race condition happens in concurrency when two or more tasks access shared data at the same time, and the programs result depends on who runs first
Liveness: Liveness is a characteristic of a program that means the unit will eventually complete its execution. If a program loses its Liveness, that means that its execution is being hindered
Deadlock: if all the tasks in a concurrent environment lose their Liveness, it is called a deadlock
(Nicholas Roberts)

14. What kind of tasks do not require any kind of synchronization?
If you have two processes which have no overlapping dependencies (I/O requests), two processes that are completely mutually exclusive and do not share any data. Might also require each independent process to run on its own CPU core to not require that the CPU manage and allocate time to each task on the same CPU core. (Jeremy Haft)

15. Describe the five different states in which a task can be.
Blocked (Stopped/Waiting for I/O), Running, New, Ready, Dead (Finished) (Jeremy Haft)

16. What is a task descriptor?
A task descriptor is a data structure maintained by the operating system that stores all information needed to manage and execute a task (process or thread). (Arsalan Ansari)

17. In the context of language support for concurrency, what is a guard?
A guard is a boolean condition that protects the accessing of shared data by delaying or blocking execution until a specified condition is true. -Kristopher Garcia

18. What is the purpose of a task-ready queue?
A task ready queue is meant to hold tasks that are ready to execute but are missing cpu time. -Kristopher Garcia

19. What are the two primary design issues for language support for concurrency?
- The two primary design issues for language support for concurrency are synchronization and scheduling. (Oscar Yang)

20. Describe the actions of the wait and release operations for semaphores.
The wait action blocks a process and puts it in the wait queue.
The release action unblocks a process from the wait queue and puts it in the ready queue. - (Dariel Urena)

21. What is a binary semaphore? What is a counting semaphore?
A binary semaphore has states 0/1, and it is used for mutual exclusion. A counting semaphore has value >1 and is used to allow N threads access concurrently. (Oscar Yang)

22. What are the primary problems with using semaphores to provide synchronization?
The primary problems with using semaphores to provide synchronization are:
Semaphores are low-level synchronization mechanisms that are hard to use correctly and can cause deadlock and starvation.(Sonu Khadgi)

23. What advantage do monitors have over semaphores?

Monitors are easier and safe than semaphores because synchronization is built in and handled automatically, reducing the chance of programming errors (Adeel Asaf)

24. In what three common languages can monitors be implemented?
The common languages that monitors can be implemented are Java, ADA and C#. Because they provide built-in support for monitors. (Sonu Khadgi)

25. Define rendezvous, accept clause, entry clause, actor task, server task, extended accept clause, open accept clause, closed accept clause, and completed task.
Rendezvous - A synchronization mechanism where a task calling another task waits until the called task accepts the request
Accept clause: Code in a task that specifies how it handles an incoming request (entry call).
Entry clause: A declaration that defines a task's entry point that other tasks can call.
Actor task: A task that initiates communication by making entry calls.
Server task: A task that responds to requests from other tasks using accept clauses.
Extended accept clause: An accept clause that includes extra code to be executed after the rendezvous.
Open accept clause: An accept clause that allows other tasks to queue while it is executing.
Closed accept clause: An accept clause that blocks other tasks from entering while it is executing.
Completed task: A task that has finished execution and will not run again.
(Unsa Chaudhry)

26. Which is more general, concurrency through monitors or concurrency through message passing?
Concurrency through message passing - (Vegus Tao)

27. Are Ada tasks created statically or dynamically?
Statically (when the program begins execution) - (Vegus Tao)

28. What purpose does an extended accept clause serve?
It allows additional code to execute concurrently with the calling task. (Shiwlee Rahman)

29. How is cooperation synchronization provided for Ada tasks?
Through guarded accept clauses. (Shiwlee Rahman)

30. What is the advantage of protected objects in Ada 95 over tasks for providing access to shared data objects?
They provide efficient, safe shared-data access without rendezvous. (Shiwlee Rahman)

31. Specifically, what Java program unit can run concurrently with the main method in an application program?
A thread executing a run() method. (Shiwlee Rahman)

32. Are Java threads lightweight or heavyweight tasks?
Lightweight threads (Shiwlee Rahman)

33. What does the Java sleep method do?
The sleep method in java stalls the current thread for a specified amount of time in milliseconds. So doing Thread.sleep(1000) will pause the current thread for 1 second. (Joseph Pasqualicchio)

34. What does the Java yield method do?
The yield method pauses the current thread and allows for other threads which are the same priority to execute, and puts the current thread at the back of the runtime queue scheduled to run again. (Joseph Pasqualicchio)

35. What does the Java join method do?
The join method pauses the current thread to wait for another thread to finish executing. (Joseph Pasqualicchio)

Example:
Thread t = new Thread();
t.start();
t.join();

36. What does the Java interrupt method do?
The java interrupt method marks a thread to terminate by setting a boolean flag for the thread to handle marking it as interrupted. (Joseph Pasqualicchio)

37. What are the two Java constructs that can be declared to be synchronized?
synchronized methods and blocks. (Tony Chen)

38. How can the priority of a thread be set in Java?
by default it's norm_priority and can later be defined to max_priority and min_priority. The priority of a thread can be changed using the method setPriority. → (thasneem mohamed)

The priority of a thread can be set in Java by calling a setter on a thread object (Vincent Comaianni)

39. Can Java threads be actor threads, server threads, or either?
Java threads are flexible and can be implemented as either actor threads or server threads, depending on the program design. (Sonu Khadgi)

40. Describe the actions of the three Java methods that are used to support cooperation synchronization.
The actions of the Java methods that are used to support cooperation synchronization are:
wait: causes the calling thread to release and monitor lock and execute suspension until it is notified by another thread.
notify: wakes up one waiting thread on the same monitor so it may complete to reacquire the lock and continue execution.
notifyAll: wakes up all threads waiting on the same monitor , allowing them to compete for the lock. (Sonu Khadgi)

41. What kind of Java object is a monitor?
Java is well known for its object monitors. All objects can be monitors???? "every object carries a monitor lock that synchronized, wait(), and notify() operate on." (Jeremy Haft)

42. Explain why Java includes the Runnable interface.
A Runnable interface allows a programmer to define the task a thread will take without the need to extend a Thread itself. (Tony Chen)

43. What are the two methods used with Java Semaphore objects?
acquire() and release(). Acquire attempts to get the resources it needs and release, releases the resource it has when done. - (Dariel Urena)

44. What is the advantage of the nonblocking synchronization in Java?
The advantage of nonblocking synchronization in Java is that it avoids thread blocking and context switches. It also improves throughput and prevents deadlocks. (Oscar Yang)

45. What are the methods of the Java AtomicInteger class and what is the purpose of this class?
An Atomic thread safe variable. (Jeremy Haft)

46. How are explicit locks supported in Java?
Java supports explicit locks through the Lock interface and the ReentrantLock class in the java.util.concurrent.locks package, allowing for more flexible synchronization features like non-blocking attempts and timeouts. (Cheuk Yiu Sung)

47. What kinds of methods can be run in a C# thread?
Static methods and instance methods - (Vegus Tao)

48. Can C# threads be actor threads, server threads, or either?
C# threads can be used to implement both actor threads and server threads, or either, as they are a general-purpose mechanism for concurrency rather than being inherently tied to a specific architectural pattern. (Weitong Chen)

49. What are the two ways a C# thread can be called synchronously?
1 - normal function call, calling the threads method directly or 2 - calling thread.join () which makes the calling thread wait until the thread finishes. (Unsa Chaudhry)

50. How can a C# thread be called asynchronously?
Starting it with Start() or using asyn constructs like async and await. (Unsa Chaudhry)

51. How is the returned value from an asynchronously called thread retrieved in C#?
Await keyword, But also possibly A callback function? (Jeremy Haft)

52. What is different about C#'s Sleep method, relative to Java's sleep?
Answer:

a) C#
    i) Encouraged for asynch situations
    ii) Clearer syntax
b) Java
    i) Also blocking call
        1) But offers non-blocking features (Monika Luchowska)

53. What exactly does C#'s Abort method do?
C#'s Abort method forcibly terminates a thread by raising a ThreadAbortException in that thread. This causes the thread to stop execution (after running any finally blocks), which can leave shared data in an inconsistent state, so its use is discouraged. (Jahid Hasan)

54. What is the purpose of C#'s Interlocked class?
Provides atomic operations on shared variables (Shiwlee Rahman)

55. What does the C# lock statement do?
Ensures mutual exclusion for a critical section.(Shiwlee Rahman)

56. On what language is Multi-LISP based?
Lisp. (Shiwlee Rahman)

57. What is the semantics of Multi-LISP's pcall construct?
It executes function calls in parallel. (Shiwlee Rahman)

58. How is a thread created in CML?
Using functional constructs like spawn. (Shiwlee Rahman)

59. What is the type of an F# heap-allocated mutable variable?
A reference type. (Shiwlee Rahman)

60. Why don't F# immutable variables require synchronized access in a multithreaded program?

Because they cannot change, eliminating race conditions. (Shiwlee Rahman)

61. What is the objective of the specification statements of HPF?
To help the compiler optimize execution on multiprocessors. (Shiwlee Rahman)

62. What is the purpose of the FORALL statement of HPF and Fortran?
Specifies statements that may execute concurrently.(Shiwlee Rahman)

Chapter 14: Exception Handling and Event Handling

Summary

C++ includes no predefined exceptions (except those defined in the standard library). C++ exceptions are objects of a primitive type, a predefined class, or a user-defined class. Exceptions are bound to handlers by connecting the type of the expression in the throw statement to that of the formal parameter of the handler. Handlers all have the same name—catch. The C++ throw clause of a method lists the types of exceptions that the method could throw.

Java exceptions are objects whose ancestry must trace back to a class that descends from the Throwable class. There are two categories of exceptions— checked and unchecked. Checked exceptions are a concern for the user program and the compiler. Unchecked exceptions can occur anywhere and are often ignored by user programs.

The Java throws clause of a method lists the checked exceptions that it could throw and does not handle. It must include exceptions that methods it calls could raise and propagate back to its caller.
The Java finally clause provides a mechanism for guaranteeing that some code will be executed regardless of how the execution of a try compound terminates.
Java now includes an assert statement, which facilitates defensive programming.
Python's exception handling is similar to that of Java, although it adds the else clause to the try construct. Also, it uses except clauses rather than catch clauses to define handlers and raise instead of throw. Access to the data of an exception object is gained by assigning the object to a variable with an as clause. Python's assert statement is a conditional raise.
Exception handling in Ruby is similar to that of Python. Every exception class has two methods, message and backtrace. Exceptions are often raised with a raise statement with a single string parameter. This creates a new RuntimeError object with the string as its message. The raise statement can be made conditional by adding a conditional expression to it. The scope of exception handlers usually is specified with a begin-end block. Handlers are defined in rescue clauses. A begin-end block can include an else clause and an ensure clause, which is like the finally clause of Python and Java.
An event is a notification that something has occurred that requires special processing. Events are often created by user interactions with a program through a graphical user interface. Java event han
h
dlers are called through event listeners. An event listener must be registered for an event if it is to be notified when the event occurs. Two of the most commonly used event listeners interfaces are actionPerformed and itemStateChanged.
Windows Forms is the original approach to building GUI components and handling events in .NET languages. A C# application builds a GUI in this approach by subclassing the Form class. All .NET event handlers use the same protocol. Event handlers are registered by creating an EventHandler object and assigning it to the predefined delegate associated with the GUI object that can raise the event.

Review Questions

1. Define exception, exception handler, raising an exception, continuation, finalization, and built-in exception.

Exceptions are errors that disrupt normal program flow, an exception handler is a block of code designed to handle exceptions and prevent them from crashing. Raising an exception means deliberately triggering an exception. Continuation is when the program returns to normal program flow after an exception has been handled. Finalization is a block of code that will always execute regardless of exceptions. Built-in exceptions are predefined exceptions provided by the programming library for common exceptions. -Kristopher Garcia

2. What are the two alternatives for designing continuation?
Resuming execution where the exception was raised or skip the code block and continue after it. (Kristopher Garcia)

3. What are the advantages of having support for exception handling built in to a language?
Being able to easily handle errors in code and act on it. It can also help the code manage errors that the programmer may have not foreseen being an issue. Example: Have a default action upon getting an error. Additionally it might allow a program to exit neatly without the whole system coming crashing down. (Jeremy Haft) It also helps avoid having to decrypt error codes from a list of errors for example when writing code in C you may have many different negative return values that correspond with some chart of predefined errors.

4. What are the design issues for exception handling?
The design issue for exception handling is:
How and where are exception handlers defined (Syntax and Placement)
How is an exception occurrence bound to an exception handler (Search process static or dynamic scoping)
Where is execution continued, if at all , after an exception handler completes its execution (continuation issue such as termination or resumption)
How are user-defined exceptions specified (can you create your own exception or are you limited)
Should there be a default exception handlers for programs (what happens when nothing catched)
(Jeaneth Lliguicota)

5. What does it mean for an exception to be bound to an exception handler?
When an exception is found in a program, the language binds that exception to the specific catch/handler block that will handle the exception
(Nicholas Roberts)

6. What is the name of all C++ exception handlers?
--> (thasneem mohamed) They are try catch blocks .

7. How can exceptions be explicitly raised in C++?
→ (Thasneem mohamed) by using the “Throw” statement.

8. How are exceptions bound to handlers in C++?
In C++ the exceptions can be explicitly thrown by throw. The general form of it is: throw[expression]. If the function header has a throw clause and raises an exception which is not listed in the clause, the compiler will call the default handler. (Andrew Laboy)

9. How can an exception handler be written in C++ so that it handles any exception?
In C++, an exception handler can be written to handle any exception by using catch -all handler with catch (....). (Sonu Khadgi)

10. Where does execution control go when a C++ exception handler has completed its execution?
Once the catch block is executed, the control jumps to the first statement after the catch block. (Andrew laboy)

11. Does C++ include built-in exceptions?

Although C++ had no predefined exceptions, the standard libraries define and throw exceptions. (Andrew Laboy)

12. Why is the raising of an exception in C++ not called raise?
The raising of an exception is not called raise because raise is already a library function used for signal handling, so C++ uses throw to avoid ambiguity.(Kaiwen Liu)

13. What is the root class of all Java exception classes?
The Throwable class (java.lang.Throwable) - (Vegus Tao)

14. What is the parent class of most Java user-defined exception classes?
java.lang.Exception - (Dariel Urena)

15. How can an exception handler be written in Java so that it handles any exception? Implement a try/catch for the general Exception type. (Tony Chen)

16. What are the differences between a C++ throw specification and a Java throws clause?
(Kaiwen Liu)
C++ : void f() throw (std::runtime_error){
throw std::runtime_error("error");}
Java: void f() throws IOException {
throw new IOException("error");}
A C++ throw specification declares exceptions a function may throw (checked at runtime, now deprecated), while Java throws clause declares checked exceptions a method must handle or declare, enforced at compile time.

17. What is the difference between checked and unchecked exceptions in Java?
The difference between checked and unchecked exceptions in Java is that checked exceptions are checked by the compiler whereas the unchecked exceptions are not checked at the compile time. (Sonu Khadgi)

18. How can an exception handler be written in Java so that it handles any exception?
(thasneem mohamed) using catch(Exception e)

19. What is the purpose of the Java finally clause?
The purpose of the Java finally clause is to guarantee execution of cleanup code, regardless of whether an exception occurs or not. (Arsalan Ansari)e

20. What advantage do language-defined assertions have over simple if-then constructs?
The advantage of do language-defined assertions is that they can be automatically enabled or disabled and are treated separately from normal program logic; unlike simple if-then constructs. (Sonu Khadgi)

21. Explain what an else block in Python does.
    The else block in Python defines what happens when the main block completes normally. When it's in an if-else block, it runs if the if statement is False. (Oscar Yang)

22. What is the purpose of an as clause in Python?
The purpose of an as clause in Python is to bind the results of a statement to a name for later use (Vincent Comaianni)
The as clause in Python assigns an alias to an object, allowing it to be referenced by a simpler or more convenient name. It is commonly used in imports and exception handling to improve readability and access. (Jahid Hasan)

23. Explain what an assert statement in Python does.
Raises an exception if a Boolean condition is false.(Shiwlee Rahman)

24. What does the message method of Ruby's StandardError class do?
Returns the exception's message string.(Shiwlee Rahman)

25. What exactly happens when a raise statement with a string parameter is executed?
A RuntimeError object is created with that string.(Shiwlee Rahman)

26. What exactly does a Ruby ensure clause do?
Executes code regardless of exceptions (like finally).(Shiwlee Rahman)

27. In what ways are exception handling and event handling related?
They both consume a signal to execute something else, one is triggered by an error and another is triggered by an intentional functionality. (Joseph Pasqualicchio)

28. Define event and event handler.
An event is an occurrence or action and an event handler is a consumer of that action where you can implement how you'd like to consume it for your usecase. (Joseph Pasqualicchio)
An event is a notification that something specific occurred, while an event handler is a response to an event.(Jackie Lin)

29. What is event-driven programming? →(Thasneem Mohamed) program where the flow is driven by events and not the sequence of code.

30. What is the purpose of a Java JFrame? A main application window that provides title bar, borders, close, and minimize button and acts as a root container for other Swing (Java GUI) components. (Tony Chen)

31. What is the purpose of a Java JPanel?
Groups other components like buttons, labels, and text fields within a window. - (Vegus Tao)
32. What object is often used as the event listener in Java GUI applications?
Answer: Defined by Java interfaces.(Shiwlee Rahman)

33. What is the origin of the protocol for an event handler in Java?
Ans - The protocol for an event handler in Java comes from the delegation-based event model, where events are handled by listener objects that receive and respond to events. (Adeel Asaf)

34. What method is used to register an event handler in Java?
In Java, an event handler is registered using the method addXListener of the event source, where X corresponds to the type of event (e.g. AddActionListener for action events) (Kaiwen Liu)

35. Using .NET's Windows Forms, what namespace is required to build a GUI for a C# application?
To build a GUI for a C# application, you must include the namespace System.Windows.Forms .(Kaiwen Liu)

36. How is a component positioned in a form using Windows Forms?
By setting its Location property. (Shiwlee Rahman)

37. What is the protocol of a .NET event handler?
A .NET event handler is a method with a specific signature that returns void and takes two parameters: the event source and event data. This standard format lets the runtime know how to call the method when the event occurs. (Jahid Hasan)

38. What class of object must be created to register a .NET event handler?
a delegate object - (Dariel Urena)

39. What role do delegates play in the process of registering event handlers?
Delegates act as type-safe method pointers, defining the signature for event handlers, and serve as the backbone for event registration by creating a list (invocation list) of methods to call when an event occurs, enabling loose coupling where an event publisher doesn't need to know about specific subscribers, just the delegate contract. (Weitong Chen)

Chapter 15: Functional Programming Languages

Summary

Mathematical functions are named or unnamed mappings that use only conditional expressions and recursion to control their evaluations. Complex functions can be defined using higher-order functions or functional forms, in which functions are used as parameters, returned values, or both.
Functional programming languages are modeled on mathematical functions.
In their pure form, they do not use variables or assignment statements to handling, a variety of data structures, and abstract data types.
ML does not do any type coercions and does not allow function overloading.
Multiple definitions of functions can be defined using pattern matching of the actual parameter form. Currying is the process of replacing a function that takes multiple parameters with one that takes a single parameter and returns a function that takes the other parameters. ML, as well as several other functional languages, supports currying.
Haskell is similar to ML, except that all expressions in Haskell are evaluated using a lazy method, which allows programs to deal with infinite lists.
Haskell also supports list comprehensions, which provide a convenient and familiar syntax for describing sets. Unlike ML and Scheme, Haskell is a pure functional language.
F# is a .NET programming language that supports functional and imperative programming, including object-oriented programming. Its functional programming core is based on OCaml, a descendent of ML and Haskell. F# is supported by an elaborate and widely used IDE. It also interoperates with other .NET languages and has access to the .NET class library.

Review Questions

1. Define functional form, simple list, bound variable, and referential transparency.
A functional form is a predefined function or operator used to build expressions, and a simple list is a list that contains only elements and no nested lists. A bound variable is a variable whose value is defined within an expression, and referential transparency means an expression can be replaced by its value without changing the program's result. (Jahid Hasan)

2. What does a lambda expression specify?
It describe nameless functions(Jackie Lin)
An anonymous function that has no name to be referenced by. -Kristopher Garcia

3. What data types were parts of the original Lisp?
Lisp data structures were composed by atoms and lists, where an Atom is symbols and numbers, and a List, which is built from cons cells.

4. In what common data structure are Lisp lists normally stored?
They are normally stored as singly linked lists built from cons cells (Ahmed Jawhar)

5. Explain why QUOTE is needed for a parameter that is a data list.
QUOTE is needed for a parameter that is a data list because without it, Lisp would attempt to evaluate the list as a function call rather than treat it as a literal data. (Sonu Khadgi)

6. What is a simple list?
A simple list is a linear data structure consisting of an ordered sequence of elements (Vincent Comaianni)

7. What does the abbreviation REPL stand for?
Read-eval-print loop - (Dariel Urena)

8. What are the three parameters to IF?

An IF statement has three parameters: a control expression, a then clause, and an optional else clause. (Arsalan Ansari)

9. What are the differences between =, EQ?, EQV?, and EQUAL?

- EQ? Takes two expressions as parameters and returns true if both parameters have the same pointer value
- EQV? Takes two expressions as parameters like EQ? But instead evaluates the numeric or symbolic value
- EQUAL? Takes two lists as parameters and returns true if the two lists are equal

(Nicholas Roberts)

10. What are the differences between the evaluation method used for the Scheme special form DEFINE and that used for its primitive functions?

The core difference is that DEFINE is a special form, meaning it controls its own evaluation (doesn't evaluate all args, binds names), while primitive functions follow standard procedure application: evaluate all arguments first, then apply the function; DEFINE bypasses standard evaluation to create bindings, allowing it to associate a name with an expression without immediately evaluating the expression's result. (Weitong Chen)

11. What are the two forms of DEFINE?

Defining a variable and binding it to an expression's value:

(define <variable> <expression>)

This form evaluates the <expression> and binds the resulting value to the <variable> identifier in the current environment.

Defining a procedure (function):

(define (<variable> <formals>) <body>)

This is syntactic sugar that creates a procedure (using lambda) and binds it to the <variable>. It is semantically equivalent to the first form:

(define <variable> (lambda (<formals>) <body>))

Here, <formals> are the procedure's parameters, and <body> is the code the procedure executes when called. (Weitong Chen)

12. Describe the syntax and semantics of COND.

The general syntax of COND in Scheme/Lisp is a list that begins with the symbol cond, followed by a series of clauses:

```
(COND
        (Predicate1 expression1 expression2 …)
        (Predicate2 expression1 expression2 …)
        …
        (PredicateN expressionM)
        [(Else expressionY expressionZ…)]

)
```

Each clause is a list itself, typically containing a predicate and one or more expressions. COND evaluates clauses top to bottom, executes all expressions in the first true clause, returns the value of its last expression, and uses ELSE as a default if no condition is true.

(Kaiwen Liu)

13. Why are CAR and CDR so named?

CAR: Contents of the Address part of Register number (stores the pointer to the first element).

CDR: Contents of the Decrement part or Register number (stores the pointer to the rest of the list/the tail).

It is named like this because it is short and deeply embedded into Lisp tradition - (Vegus Tao).

14. If CONS is called with two atoms, say 'A and 'B, what is the returned?

(A.B) (Unsa Chaudhry)

15. Describe the syntax and semantics of LET in Scheme.

The let expression in Scheme creates local variable bindings with a specific scope. It serves as a fundamental construct for introducing local variables to structure code and prevent namespace collisions. The syntax can also support a "named let" variant for iteration. (Weitong Chen)

16. What are the differences between CONS, LIST, and APPEND?

- CONS adds one element to the front of a list
- LIST creates a list by directly listing the elements
- and APPEND joins two lists together into one bigger list

(Nicholas Roberts)

17. Describe the syntax and semantics of mapcar in Scheme.
Syntax of mapcar in Scheme: (mapcar function list) (map function list)
Semantics of mapcar: applies each function to each element list and returns a new list. (Sonu Khadgi)

18. What is tail recursion? Why is it important to define functions that use recursion to specify repetition to be tail recursive?

A function that is rail recursive if its recursive call is the last operation of its function. This means that the return value of the recursive call is the return value of the non recursive call to the function. It is important to specify repetition to be tail recursive because it is more efficient(increase the efficiency). (Andrew Laboy)

19. Why were imperative features added to most dialects of Lisp?
Imperative features were added to most dialects of Lisp because they increased efficiency and ease of use
(Jeaneth Lliguicota)

20. In what ways are Common Lisp and Scheme opposites?
Ans - Common Lisp is large, complex, and feature-rich, while Scheme is small, simple, and minimal, focusing on a clean and consistent design. (Adeel Asaf)

21. What scoping rule is used in Scheme? In Common Lisp? In ML? In Haskell? In F#?
All use static (lexical) scoping.(Shiwlee Rahman)

22. What happens during the reader phase of a Common Lisp language processor?
Source code is converted into internal representation and reader macros are expanded. (Shiwlee Rahman)

23. What are two ways that ML is fundamentally different from Scheme?
ML is strongly typed, ML uses type inference. (Shiwlee Rahman)

24. What is stored in an ML evaluation environment?
Identifier names with their types and values. (Shiwlee Rahman)

25. What is the difference between an ML val statement and an assignment statement in C?
val binds a name to a value; assignment changes a variable's value.(Shiwlee Rahman)

26. What is type inferencing, as used in ML?
Type inferencing means the compiler can automatically figure out the type of a variable without you explicitly writing it. (Nicholas Roberts)

27. What is the use of the fn reserved word in ML?
Answer: To define anonymous (lambda) functions. (Shiwlee Rahman)

28. Can ML functions that deal with scalar numerics be generic?
Answer: **No**, due to strong typing.(Shiwlee Rahman)

29. What is a curried function?

- A curried function takes in multiple arguments one at a time and returns a new function for the remaining arguments. (Oscar Yang)

30. What does partial evaluation mean?

Partial evaluation is a program transformation technique that optimizes code by execution parts of a program that depend on known input data (static data) before runtime. (Andrew Laboy)

31. Describe the actions of the ML filter function.
In ML, the filter function takes a Boolean function and a list as input, and returns a new list containing only the elements that make the Boolean true.
(Nicholas Roberts)

32. What operator does ML use for Scheme's CAR?

The operator used to achieve the same result as schemes CAR is hd. (Andrew Laboy)

33. What operator does ML use for functional composition?
Functional composition is when you combine two functions into one new function, where the output of one becomes the input of the next

in ML the symbol o is used for functional composition
EX: val h = f o g
meaning h(x) = f(g(x))
(Nicholas Roberts)

34. What are the three characteristics of Haskell that make it different from ML?
The three characteristics of Haskell that make it different from ML are its lazy evaluation, pure functional language, and list comprehensions. (Unsa Chaudhry)

35. What does lazy evaluation mean?
Expressions are not evaluated until the results are needed. (Tony Chen)

36. What is a strict programming language?
A Strict Programming Language is one where the arguments of a function are evaluated before the function is executed (Nicholas Roberts)
Arguments are calculated first before being passed into the function - (Vegus Tao)

37. What programming paradigms are supported by F#?
Functional, Imperative, and OOP - (Vegus Tao)
38. With what other programming languages can F# interoperate?

39. What does F#'s let do? → F# let binds a name to a value or a function. (Thasneem Mohamed)

40. How is the scope of a F# let construct terminated?
The scope of a F# let construct is terminated when the block or expression it belongs ends, typically where the indentation ends. (Unsa Chaudhry)

41. What is the underlying difference between a sequence and a list in F#?
Ans - The underlying difference is that a sequence is evaluated lazily (values are generated when needed), while a list is evaluated eagerly (all values are stored in memory). (Adeel Asaf)

42. What is the difference between the let of ML and that of F#, in terms of extent?
Answer: "let" binding example:

a) Top Level event

    i) ML
        1) Persist for
            (a) lifetime of program execution
    ii) F#
        1) Persist for
            (a) Duration of f# interactive session

b) Class Level event

    i) ML
        1) Doesn't persist!!
            (a) ML doesn't have classes….
    ii) F#
        1) Persist for
            (a) lifetime of the module

c) Local binding event

    i) ML, F#...
        1) extent for
            (a) Body of the "in" expression (Monika Luchowska)

43. What is the syntax of a lambda expression in F#?
In F#, the syntax of a lambda expression is: fun <parameter(s)> -> <expression>, where fun introduces and anonymous function.(Kaiwen Liu)
44. Does F# coerce numeric values in expressions? Argue in support of the design choice.
45. What support does Python provide for functional programming?-->(Thasneem Mohamed) Python provides first-class functions, list comprehensions, and lambdas.
46. What function in Ruby is used to create a curried function?
In Ruby, the 'curry' method is used to create a curried function from Proc or Lambda.(Kaiwen Liu)
47. Is the use of functional programming expanding or shrinking?
Functional programming is expanding because its emphasis on immutability and pure functions is essential for writing safe, concurrent code in a multi-core and big data world. Most mainstream imperative languages are also evolving by integrating functional features like lambdas and map/reduce to manage modern software complexity. (Cheuk Yiu Sung)
48. What is one characteristic of functional programming languages that makes their semantics simpler than that of imperative languages?
Functional programming languages have no variable assignment or state changes. (Unsa Chaudhry)
49. What is the flaw in using lines of code to compare the productivity of functional languages and that of imperative languages?
Measuring productivity by lines of code is flawed because functional languages are inherently more concise, allowing a single line of functional code to perform tasks that require many lines of imperative code. This creates a "conciseness bias" where an imperative programmer appears more productive simply by writing more verbose, lower-level instructions to achieve the same result. (Cheuk Yiu Sung)
50. Why can concurrency be easier with functional languages than imperative languages?
Concurrency can be easier in functional languages because they emphasize immutability and pure functions, which reduces shared mutable state. With fewer side effects, there is less need for locks and synchronization, making concurrent programs simpler and safer. (Jahid Hasan)

Chapter 16: Logic Programming Languages

Summary

Symbolic logic provides the basis for logic programming and logic programming languages. The approach of logic programming is to use as a database a collection of facts and rules that state relationships between facts and to use an automatic inferencing process to check the validity of new propositions, assuming the facts and rules of the database are true. This approach is the one developed for automatic theorem proving.
Prolog is the most widely used logic programming language. The origins of logic programming lie in Robinson's development of the resolution rule for logical inference. Prolog was developed primarily by Colmeraeur and Roussel at Marseille, with some help from Kowalski at Edinburgh.
Logic programs are nonprocedural, which means that the characteristics of the solution are given but the process of getting the solution is not.
Prolog statements are facts, rules, or goals. Most are made up of structures, which are atomic propositions, and logic operators, although arithmetic expressions are also allowed.
Resolution is the primary activity of a Prolog interpreter. This process, which uses backtracking extensively, involves mainly pattern matching among propositions. When variables are involved, they can be instantiated to values to provide matches. This instantiation process is called unification.
There are a number of problems with the current state of logic programming.
For reasons of efficiency, and even to avoid infinite loops, programmers must sometimes state control flow information in their programs. Also, there are the problems of the closed-world assumption and negation.
Logic programming has been used in a number of different areas, primarily in relational database systems, expert systems, and natural-language processing.

Review Questions

1. What are the three primary uses of symbolic logic in formal logic?
    1. To express propositions (using symbol to state logical facts or claims about the state)
    2. To express relationship between propositions
    3. To describe how new propositions can be inferred

(Jeaneth Lliguicota)

2. What are the two parts of a compound term?-->(Thasneem Mohamed) functors- function symbol that names the relationship and the ordered list of parameters(tuples)
A function and a list of arguments it acts on. -Kristopher Garcia

3. What are the two modes in which a proposition can be stated?
Either as a fact or as a goal. - Kristopher Garcia

4. What is the general form of a proposition in clausal form?
the general form of a proposition in clausal form is the following

$C_1 \wedge C_2 \wedge \cdots \wedge C_n \rightarrow E$

In simple English this means if these conditions hold, then the effect E is true
(Nicholas Roberts)

5. What are antecedents? Consequences?
Antecedent is the left side of the clausal form, and consequent is the right side of the clausal form(Jackie Lin)

6. Give general (not rigorous) definitions of resolution and unification.

Resolution draws conclusions from facts that contain the same pieces of information. Unification is when two expressions are to be made the same by making logical inferences. (Lily Zheng)

7. What are the forms of Horn clauses?
A proposition in clausal form is a disjunction (OR) of literals, where each literal is a variable or its negation. (Adeel Asaf)

8. What is the basic concept of declarative semantics?
Declarative semantics defines the meaning of a program in terms of what it computes, independent of the order or method of execution.(Kaiwen Liu)

9. What does it mean for a language to be nonprocedural?
The programmer specifies what needs to be done rather then how to do it (Ahmed Jawhar)

10. What are the three forms of a Prolog term?-->(thasneem mohamed) Atom, numbers, and compound terms.

11. What is an uninstantiated variable?
An uninstantiated variable is one that is defined but not yet given a value. An example of this can just be: int variable; This is not actually assigned to anything yet but is defined in the scope. (Joseph Pasqualicchio)

12. What are the syntactic forms and usage of fact and rule statements in Prolog?
The syntactic forms and usage of fact and rules in Prolog are defined as so: a fact is written as a predicate followed by a period, and states something unconditionally true, a rule is written as a head and a body separated by a ":-", and defines a condition under which the head is true. (Vincent Comaianni)

13. What is a conjunction?
A conjunction is a logical operation that combines two Boolean expressions and is true only if both expressions are true. (Arsalan Ansari)

14. Explain the two approaches to matching goals to facts in a database.
Bottom-up resolution, or forward chaining: When a system begins with the facts and rules of the data base and attempts to find a sequence of matches that lead to the goal.
Top-down resolution, or backward chaining: beginning with the goal and attempting to find a sequence of machining proportions that lead to some set of original facts in the database. (Andrew Laboy)
15. Explain the difference between a depth-first and a breadth-first search when discussing how multiple goals are satisfied.

16. Explain how backtracking works in Prolog.
- In Prolog, backtracking is used to systemically explore all possible solutions to a query. If it fails at any point, it will go back to the previous choice point and try alternative possibilities until all possibilities exhausted or a solution is found. (Oscar Yang)

17. Explain what is wrong with the Prolog statement K is K + 1.
Prolog variables are not mutable (If K is already instantiated, it will just compare K to K + 1, which will always return false, or if K is not assigned a value, it will just return an error). In order to increment properly, you have to do K1 is K + 1 - (Vegus Tao)

18. What are the two ways a Prolog programmer can control the order of pattern matching during resolution?
Answer: the two ways are to
a) clause order in the database
    i) depth-first
    ii) more specific rules earlier in the code
b) cut operator (!)
    i) acts as control predicate

ii) freeing up search space
iii)Forcing specific outcome (Monika Luchowska)

19. Explain the generate-and-test programming strategy in Prolog.

- The generate-and-test programming strategy in Prolog is a trial and error approach automated by Prolog. First it generates possible candidates for the solution based off facts, rules, or domains of values. Then it tests each candidate against the problem constraints and if it passes it is a solution, if not then Prolog backtracks to generate the next candidate. (Oscar Yang)

20. Explain the closed-world assumption used by Prolog. Why is this a limitation?
In Prolog, the closed-world assumption means that anything not stated as a fact or rule is assumed to be false. This is a limitation because missing or incomplete information can lead Prolog to make incorrect conclusions. (Jahid Hasan)

21. Explain the negation problem with Prolog. Why is this a limitation?
In Prolog, negation works by assuming something is false if it cannot be proven true. This is a limitation because failure to prove a fact does not always mean the fact is actually false. (Jahid Hasan)

22. Explain the connection between automatic theorem proving and Prolog's inferencing process.
Prolog's inferencing process works like automatic theorem proving by using rules and facts to logically derive conclusions. It is limited because Prolog only searches for proofs within its given knowledge base and rules. (Jahid Hasan)

23. Explain the difference between procedural and nonprocedural languages.--> procedural: how to do the task, nonprocedural: what is true and desired

24. Explain why Prolog systems must do backtracking.
Prolog systems are declarative and nondeterministic, hence there is guaranteed a solution so if a certain step fails, the program must backtrack to the last choice point. (Tony Chen)

25. What is the relationship between resolution and unification in Prolog?
In Prolog, unification is the process of matching goals with facts or rules by finding variable substitutions that make them identical. Resolution uses unification to repeatedly apply rules and facts to derive new goals and ultimately find a solution. (Jahid Hasan)