The questions at the back of the chapters are also an excellent source for studying for this exam.

1. What are the advantages of allowing programmers to define enumerated types?

2. What are the primary design issues related to defining arrays in programming languages?

3. Give an example of an associative array.

4. What are the primary design issues related to pointers?

5. What are the two biggest problems created with pointers? Give examples in code.

6. There are two ways of implementing type equivalence. What are they? Give examples. In particular, indicate how type equivalence is affected by a programming language’s choice of one type or the other, using records as a point of discussion.

7. What is the concept of “visibility”? How is visibility controlled in a typical language?

8. What are the two mechanisms by which the order of evaluation of expressions is controlled by a programming language? Describe them.

9. The & operator in C++ is overloaded. Give examples of each use, and explain why this is a good example of the dangers of overloading.

10. What are the four kinds of sequence control provided by most programming languages?

11. There are four basic models for repetition. Why? Describe them (give an example).

12. Describe the “dangling else” issue and give an example. What are two ways this issue can be resolved?

13. What are the primary design issues for multiple selection statements?

14. What are the two mechanisms by which the order of evaluation of expressions is controlled by a programming language? Describe them.

15. What are the four kinds of sequence control provided by most programming languages?

16. There are four basic models for repetition. Why? Describe them (give an example).

17. Describe the “dangling else” issue and give an example. What are two ways this issue can be resolved?

18. What are the primary design issues for multiple selection statements?
19. Describe the contents of a typical activation record.

20. What are the differences between an activation stack’s static and dynamic links? When does a programming language need a static link in its activation record?

21. When local referencing environments are deleted between subprogram activations using a central stack as in C/C++, it sometimes appears as if values are retained. For example, if procedure Sub has a local variable X and Sub assigns the value 5 to X on the first call, then on a second call, if X is (inadvertently) referenced before it is assigned a new value, sometimes X still has its old value 5. However, in the same program, a third call on Sub may find X has not retained its old value from the second call. Explain this apparent anomaly. In what circumstances could an activation that references an uninitialized variable find that that variable still had a value assigned on a previous call? In what circumstances would it not have its previously assigned value (but may have some other value entirely unrelated to that variable)?

22. Give an example in code (either Java or C++) of causing the program stack to run out of space.

23. Give an example in code (either Java or C++) of causing the program heap to run out of space.

24. Given a program in a language that allows recursion and uses static scope, show the activation record at a given point in the program’s execution.

25. Given a program in a language that allows recursion and uses dynamic scope, show the activation record at a given point in the program’s execution.

26. Given a program in a language that allows nested subprograms, show the activation record at a given point in the program’s execution.

27. What are the two most important principles in the construction of abstract data types? Describe each.

28. What are the differences between the support for abstract data types in C++ and Java?

29. What are the information hiding mechanisms of Java? How are they different from similar ones in C++?

30. Define the term exception. Give an example of code (in Java or C++) that would cause an exception. Give an example of code (in Java or C++) that would handle that exception.

31. What are the reasons some languages support an exception handling mechanism?

32. What are some of the design issues with respect to including exceptions in a language?
33. Why do some languages include a mechanism for testing assertions?

34. Explain event handling, and how it is essential to a modern computing environment.

35. Explain what widgets are. Give an example in code.