

B.E.

Fourth Semester Examination, Dec.-2009

Programming Language (CSE-204-E)

Note : Attempt any *five* questions. All questions carry equal marks.

Q. 1. (a) Define programming language. Explain the syntactic & semantic rules of a programming Language.

Ans. Machine to design high level languages must still be perfected. Each language has shortcoming but each is also successful in comparison with the many hundreds of other languages that have been designed, implemented, used for a period of time and then allowed to fall into disuse.

Syntax and Semantics : The syntax of a programming language is what the program looks like. To give the rules of syntax for a programming language means to tell how statements, declaration and other language constructs are written.

The semantics of programming language is the meaning given to the various syntactic construct. For example, in C to declare a 10 element vector V of integers you would give a declaration such as,

in C: `V[10];`

in Pascal: `V : array[0, 9] of integer.`

Although both creates similar data objects at run time, their syntax is very different. To understand the meaning of the declaration you need to know the semantics of both Pascal and C for such array declaration.

Q. 1. (b) Explain the specification & implementation of scalar data types.

Ans. Scalar Data Type : We first consider a class of elementary data object known as scalar data objects. These are objects that have a single attribute for its data object. For example, an integer object has an integral value and no other information can be denoted from the object.

Next we consider composite data where an object may have multiple attribute values. For example a character string. Contains a sequence of character as its data value, but also may have other attributes such as a string size as an additional attribute value.

In general the scalar object follow the hardware architecture of a computer integers floating point numbers, characters. Composite data are usually complex structures created by the compiler that are not primitive hardware implementation object.

Q. 2. (a) Write short note on variable size data structures.

Ans. A data type is a class of data object together with a set of operations for creating and manipulating them. Although a program deals with particular data object such as an array A, the integer variable X or the file F a program language necessarily deals more commonly with data type such as the class of arrays, integers or files and the operations provide for manipulating arrays, integers or files.

A data structure is a data object that contains other data object as its elements. A data object that is construct as an agreements of other data objects called component is termed a data structure data object or data structure. A component may be elementary or it may be another data structure many of the issues and concepts surrounding data structures in programming languages are the same as elementary data objects and have been treated. As with elementary data objects.

Some of programmer defined and others are system define during program execution.

Q. 2. (b) Define "arrays" & their implementation in a programming language.

Ans. A matrix is conveniently implemented by considering it as a vector of a vectors, a three dimension array is a vector whose elements are vectors of vector and so on.

Whether a matrix is viewed as column of rows or a row of columns is important in some contexts most common is the column of rows structure in which the matrix is considered as a vector in which each element is a subvector representing one row of the original matrix. The representation is known as row-major order.

An array of any number of dimension is organized in row-major order when the array is first divided into vector of subvectors for each element in the range of subscript then each of these subvectors is subdivided into subvectors for each element in the range of the second sub-script and so on. Column-major order is the representation in which the matrix is treated as a single row-column.

The stage representation for a multidimensional array follows directly from that for a vector we store the data objects in the first row followed by the data object in the second row and so on.

Q. 3. (a) Define Abstract Data Types. What do you mean by data abstraction, discuss with example.

Ans. Abstract Data Types : Every language such as Fortran and COBOL limit the creation of new data types to subprogram definition. As the concept of data type is evolved new language designs have provided better facilities for specifying and implementing entire abstract data type such as the ada package and C++ or Java class.

Data Abstraction : In order to extend this encapsulation concept to programmer-defined data type, we define abstract data type as :

- (i) A set of data object ordinarily using one or more type definitions.
- (ii) A set of abstract operation on those data object.
- (iii) Encapsulation of the whole in such a way the user of the new type cannot manipulate data objects of the type except by the use of operations defined.

The entire definition should be encapsulated in such a way that the user of the type needs to know only the type name and semantics of the available operations such programmer created abstract data type often appear as special such program libraries in languages such as C, FORTRAN and Pascal, Ada, Java and C++ are among the few widely used language with data abstraction features.

Q. 3. (b) Discuss 'Names & referencing' environment for data control in a programming language.

Ans. Data control is concerned in large part with the binding of identifiers to particular data objects and subprograms such as binding in termed an association and may be represented as a pair consisting of the identifier and its associated data object or subprogram.

Referencing Environment : The referencing environment of a subprogram or ordinarily invariant during its execution. The referencing environment of a subprogram may have several components.

(i) Local Referencing Environment : The set of association created an entry to a subprogram that represent formal parameter local variables and subprograms defined within that subprogram forms the local referencing environment of that activation of the subprogram.

(ii) Non-Local Referencing Environment : The set of association for the identifiers that may be used within subprogram but that are not created on entry to its is termed to non-local referencing environment of the subprogram.

(iii) Global Referencing Environment : If the association created at the start of execution of the main program are available to be used in subprogram then these association from the global referencing environment of the subprogram.

Q. 4. (a) Define sequence control. What do you 'mean by sequence control with in expressions.

Ans. Sequence control structures may be conveniently categorized into four groups :

- (i) Expression form the basic building blocks of statement and express how data are manipulated and changed by a program. Properties such as precedence rules and parenthesis determine how expressions become evaluated.
- (ii) Statements or groups of statements such as conditional and iteration statements determine how control flows from one segment of program to another.
- (iii) Declarative programming is a execution model that does not depend on statement but nevertheless causes execution to proceed through a program.
- (iv) Subprograms such as subprograms calls and co-routines, form a way to transfer control from one segment of program to another.

Sequence Control with Arithmetic Expression : Expression are a powerful natural device for representing sequences of operations yet they raise new problems. Although it may be tedious to write out long sequence of instruction in machine languages, at least the programmer has a clear understanding of exactly the order in which the instruction are executed.

Q. 4. (b) Write short note on :

(a) Recursive subprogram,

(b) Implicit & Explicit sequence control.

Ans. (a) Recursive Subprogram : One of our subprogram assumption, no recursion and investigate language design that permits this feature. Recursion in the form of recursive sub-program calls, is one of the most important sequence control structure in programming many algorithm are most naturally represented using recursion. In LISP, where list structure are the primary-data structure available recursion is the primary control mechanism for repeating sequences of statements, replacing the iteration of most other languages. If we allow recursion sub-program calls a subprogram A may calls any other subprograms including A, a subprogram B that calls A and so on.

(b) Implicit & Explicit Sequence Control : A common environment set of explicitly for the sharing of data objects is the most straight forward method for data sharing. A set of data objects that are to be shared among a set of sub-programs is allocated storage in separate named block. The data object within the block are then visible within the sub-program and may be referenced by name in the usual way such as shared block is known by various name.

Q. 5. (a) Differentiate between static & dynamic scope with example.

Ans. Dynamic Scope : The dynamic scope of association for an identifier as defined in the preceding section is that set of sub-program activation in which the association is visible during execution. The dynamic scope of an association always includes a sub-program activation in which that association is created as part of local environment.

A Dynamic scoping rule define the dynamic scope of each association in terms of the dynamic course of program execution. For example, a typical dynamic scope of each association in terms of the dynamic course of program execution. For example, a typical dynamic scope the rule state that the scope of an association created during an activation of sub-program P includes not only that activation but also any activation of sub-program.

Static Scope : Each declaration or other definition of an identifier within the program text has a certain scope called its static scope. The static scope of declaration is that part of the program text where a use of the identifier is a reference to that particular declaration of a identifier. A static scope rule for determining the static

scope of declaration. In Pascal for example, A static rule is used to specify that reference to variable X in a sub program P refers to declarations of X at the beginning of P or if not declaration there then to the declaration X at the beginning of the subprogram C.

Q. 5. (b) Discuss the various parameter transmission schemes with examples.

Ans. When subprogram transfer control another subprogram these must be association of the actual parameter of the calling subprogram with the formal parameter of the called program. Two approaches are often used. The actual parameter may be evaluated and that value passed to the formal parameter or actual data object may be passed to formal parameter. Several methods describe for transmission method :

(i) Call by Name : This method of parameter transmission views a subprogram call as substitution for the entire body of the sub-program.

(ii) Call by Reference : To transmit a data object as a call by reference parameter means that a pointer to the location of the data object is made available to the subprogram. The data object does not change the position in memory. Call by reference occurs in two stages :

- (i) In the calling subprogram each actual parameter expression is evaluated to give a pointer to the actual parameter data object.
- (ii) In the called subprogram the cost of pointer to actual parameters is accessed to retrieve the appropriate r-value for the actual parameter.

(iii) Call by Value : If a parameter is transmitted by value the value of the actual parameter is the passed to called formal parameter.

(iv) Call by Value Result : If a parameter is transmitted by value result the formal parameter is local variable of the same data type as the actual parameter.

Q. 6. (a) Discuss Heap storage management with example.

Ans. Heap Storage Management : A heap is a block of storage within which pieces are allocated and freed in some relatively unstructured manner. Here the problems of storage allocation recovery compaction and reuse may be serve. There is no single heap storage management technique but rather a collection of techniques for handling various aspect of managing this memory.

The need of heap storage arises when a language permits storage to be allocated and freed at arbitrary points during program execution as when a language allow creation destruction of extension of programmer data structures at arbitrary program print. For example, in ML two list may be connected to create a new list at any arbitrary point during execution or the parameter may dynamically define a new type.

A heap storage management technique up into two categories depending on whether the element allocated are always of the same fixed size or of a variable size. Where fixed size element are used, management techniques may be considerably simplified.

Q. 6. (b) Differentiate between system controlled and programmer controlled storage management with suitable example.

Ans. The programmer be allowed to directly control storage management on one handed has become very popular because it allows extensive programmer control over storage via malloc and free, which allocate and free storage for programmer defined data structures. Many high level language allow to programmer no direct control storage management is affected only implicitly through the use of various language features.

The difficulty with programmer controlled storage management is two fold. It may place a large and often undesirable burden on the programmer and it way also interfere with the necessary system controlled storage management. No high level language can allow the programmer to shoulder the entire storage management burden. For example, the programmer can hardly be expected to be concerned with storage for temporaries

subprogram return points or other system data. At best a programmer might control storage management for local data. The advantage of allowing programmer control of storage management is that it is often extremely difficult for the system to determine when storage may be most effectively allocated and freed. The programmer often knows quite precisely when a particular data structure is needed or when it is no longer needed and may be freed.

Q. 7. (a) Compare procedural languages with object oriented programming languages using suitable example.

Ans. Procedure Language : It is structural language. It is known as a middle level language such as C language is a procedural language. The procedural language does not provide modularity. It cannot provide polymorphism, provide, encapsulation. Inheritance that type of feature are not in procedural language so we cannot provide modular programming.

While object oriented language such as C++ and Java. It is provided class and object. The object and data can be inherited from one class to another class. The property known as inheritance. Encapsulation property wrapping the data. It is a property of object oriented programming while procedural language does not provide polymorphism. Object oriented programming also provides multithreading of program while procedural language does not provide.

Q. 7. (b) Write short note on declaration and type checking of data structure.

Ans. The basic concept and concerns surrounding declarations and type checking for data structures are similar to those discussed for elementary data objects. However, structures are ordinarily more complex because there are more attributes to specify. For example, the C declaration,

float A[20];

At the beginning of a subprogram p specifies the following attribute of array A :

- (i) Data type is an array.
- (ii) Number of dimensions is one.
- (iii) Number of components is 20.
- (iv) Subscripts naming the rows are integers from 0 to 19.
- (v) Data type each component is float.

Declaration of those attributes allow a sequential storage representation for A and the appropriate accessing formula for selecting component A [i] of A to be determined at compile time.

Type checking is somewhat more complex for data structures because component selection operations must be taken into account. There are two main problems :

(i) Existence of Selected Component : The arguments to selection operation may be the right types but the component designated may not exist in the data structure.

(ii) Type Selected Component : A selection sequence may define a complex path through a data structure to the desired component. For example, the C :

A[2][3] : link → item.

Q. 8. Write short notes on the following :

- (a) Information Hiding,
- (b) Co-routines.
- (c) Programming language translators,
- (d) Exception & Exception Handlers.

Ans. (a) Information Hiding :

It is the term used for the central principal in the design of programme defined abstraction. Each such program component should hide as much information as possible from the component users. Thus, the language provide square root function is successful abstract operation because it hides the detail of the number representation and the square root computation algorithm from the user similarly programmer defined data type is successful abstraction.

- (i) Does not need to know the hidden information in order to use the abstraction and
- (ii) Is not permitted to directly use or manipulate the hidden information even if desiring to do so.

(b) Co-routines : To allow subprograms to return to their calling program before completion of execution such programs are termed co-routines when co-routines receives control from another subprogram. It executes partially and then is suspended when it return control. At latter point the calling program may resume execution of the co-routine from the point at which execution was previously suspend. Co-routine are not concurrently a common control structure is programming language outside of discrete simulation language. However they provide a control structure is many algorithm that is more natural than the ordinary subprogram hierarchy. Moreover co-routine struct may be readily simulated in many language using the goto and resume point.

(c) Programming Language Translator : Translation into two major parts the analysis of the input source program and synthesis of the executable object program. In most translators these logical stage are not clearly separate but instead are mixed so that analysis and synthesis alternate often on a statement by statement basis.

Translators are crudely grouped according to the number of passes they make over the source program. A simple computer typically uses two phases. The first analysis and second pass typically generates an object program from this collected information.

(d) Exception & Exception Handlers :

During execution of a program event or contains often occur that might be considered exceptional. Rather than continue with normal program execution a subprogram needs to be called to perform some special processing including being able to handle the following situations :

Error condition to process an error such as an arithmetic operation overflow or reference to any array element with a subscript out of bounds.

Unpredictable Condition : That arise during normal program execution such as the production of special output headings at the end of printer page or end of the indicator of an input files.

Training and monitoring during program testing such as printing trace output during program testing when the value of variable changes.