SCOC-05

Assignment - 1

PAGE No.
DATE / /

Q.1] List the programming paradigms. For any three states which programming languages are based on them and how?

→ Programming paradigm is fundamental style of computer programming.

There are four main types of Programming Paradigms:

1. Procedural.
2. Object-oriented.
3. Functional.
4. Logic & rule based.

1. Procedural:- It is also called imperative programming.
- It is based on concept of procedure calls, in which statements are structured into procedures.
- They are list of instructions to tell computer what to do step by step, procedural programming languages are known as top-down languages.
- FORTRAN was first major programming language to remove through abstraction the obstacles presented by machine code in creation of complex programms.
- In late 1950s & 1960s, ALGOL was developed in order to allow mathematical algorithms to be more easily expressed.

2. Object-oriented:- It uses "objects" - data structures encapsuling data fields and procedures together with their interactions - to design application & computer programe.

- Though it was invented with creation simula language in 1965, and further developed in Smalltalk in 1970s. It was commonly used in mainstream software application develo-

pement until the until 1990s.

3. Functional:-
- Functional programming is programming paradigm in which we try to bind everything in pure mathematical function style.
- It avoid states changes & mutable data.
- Functional programming has its roots in the lambda calcu a formal systems developed in 1930s, to investigate functi definition, function application & recursion.
- LISP was first operational Functional Programing language
- Functional Programing consist only PURE functions.

4. logic & rule-based:-
- It is also known as declarative programming.
- A declarative programe does not have code, instead it defi two pieces of knowledge.
a] Facts:- statements that are true.
b] Rules:- If-then statements that are truth preserving.

- Rules are written as logical clausses with head & body.
- PROLOG follows the logical paradigm and is probably the most famous language in logical programming fami
- PROLOG like SQL has two main aspects, one to expres data and another to query it. The basic constructs of logical programming terms, and statements and inhe ted from logic.

DATE / /

Q2] What are attributes of good language?

→ The most general attributes of good language are discussed below:

1 clarity, simplicity & unity:-

- A programming language provides a conceptual framework of thinking about the algorithm.
- It should provide clear, simple and unified set of concepts that can be used as primitives in developing algorithms.
- This attribute is called conceptual integrity.

2. orthogonality:-

- It is one of the most important feature of PL: orthogonality is property that means "changing A does not change B".
- When the feature of the language are orthogonal, language is easier to learn and programs are easier to write, because only few exceptions and special case to be remembered.

3. Naturalness:-

- The syntax of language should be such that, it should follow logical structure of algorithm.
- Programming structure reflects logical structure of algorithm.
- Various algorithms like sequential, concurrent, logic algorithm have different natural structures, which should be represented by the programming language.
- language should provide appropriate data structure, operations & control structure for the problem to be solved.

4. Support for Abstraction :-

- By this attribute programmer can concentrate only on abstract properties without bothering for their implementation details.

5. Ease of programme verification :-

- A programme is checked by various testing technique like formal verification method : Desk checking, input output testing checking.
- We verify the programming by many more techniques. A language that makes Programme verification difficult maybe far more troublesome to use.
- Simplicity for semantic & syntatic structure is a primary aspect that tends to simplyfy programme verification.

6. Programming Environment :-

- An appropriate programming environment adds an extra ability and make language to be implemented easily be like The availaibility of - Reliable - Efficient - well documentation.
- speeding up creation & testing by special editors - testing packages.
- Facility - maintaining & modyfying - Malti version of Program software product.

7. Portability :-

- Programming language should be portable means it should be easy to transfer a programe from which they are developed to the other computer.
- A programe whose definition is independent of feature of particular machine forms can only support portability.

Ex. Ada, FORTRAN, C, C++, Java.

8. Cost of use:-

a] Cost of execution:- Large production programme that will b executed repeatedly.
- The programe execution most cost greatly reduced due to optimizing compilers.

b] Cost of Translation:- For compiling large programme, compiler takes too much time, it's increase overall cost.

c] Cost of creation, testing & use:- Involved in programing design, coding, testing & modyfying.

Ex.] smalltalk, Pearl.

d] Cost of Maintaince:- The maintainance cost can be 4 time more than of deployment cost.
- It depend upon readibility.

Q.3] What is programming Environment? Explain the effect of pro gramming environment on language Design.

→ Programming Environment:- It is the collection of tools used in the developement of software.

Effects on Language Design:-

The programming environment influence on langu- age in two major areas namely:
1. Separate compilation.
2. Testing / Debugging.

1. Separate compilation:-
- There are varying no. of programming working on design,

- This requires the language to be structured so that individual sub-programms or other parts can be separate compiled & execute without requirement of part.
- It is difficult because compiling one sub-programme may need some information which is present in some other sub-programms.
- shared data is also used in the programme due to which separately compilation is not possible.
- To solve this problem, some lang. have adopted certain following features that aid separate compilation.

1] Extern:- To indicate that particular data is used from other sub-programme.

2] Scoping rules:- used to hide names. One sub-programme can be contained within another sub-programm so that the name of the outer sub-programms are known to separately compiled sub-programms.

3] Inheritance:- To used the shared data.

Ex.] OOP.

4. Testing/Debugging:-

Many lang. contains some features that help in testing & debugging like

1] Breakpoint feature:- When the breakpoint is reached, the execution of programme is interrupted and control is given to the programmer.

2] Execution trace feature:- Particular variable or statement can be tagged for tracing. (PROLOG, LISP).

3] Assertions:- It's conditional expression which is inserted as a separate statement in programme.

Q.4]. Explain the impact of Machine Architecture.

→ In developing programming lang. architecture of software influences the design of language in two ways.

1. The underlying computer on which programms written in language will execute.
2. The execution model or virtual computer that support language on actual hardware.

A]. The operation on computer:-

- Computer is an integrated set of software ~~influences~~ algorithms and data strutures capable of storing & executing programms.
- It may constructed physical device using wires, IC, circuit boards, & like it is called actual or hardware computer.
- Working of any computer is based on softwared present in it.
- Major components of computer are discussed below.

1. Data:- computer must provides various kinds of elementry data items & data structure to be manipulate.
- Major storage components are like, main-memory, high-speed registers, & external files.

2. Primitive Operations:- A computer must provide set of primitive operations

- It includes:-

a) Primitives for arithmetic on data type.
b) primitives for testing various properties of data items
c) primitives for accessing & modyfying various parts of data item
d) primitives to control input/output.

3. Sequence control:- A computer must provide mechanism for controlling sequence in which data primitive operations to be executed.

4. Data Access:- A computer must provide mechanism for control the data supplied to each execution of programs instruction.

5. Storage Management:-

- A computer must provide mechanism to control the allocation of storage for programme & data.
- To keep all resources of computer operating as much as possible.

6. Operating Environments:-

- A computer must provide mechanism for communication in external environment containing programs and data to be processed

Firmware computers:-

- A set of machine language instructions implemented programs, called micro-programs, stored in programmable read-only memory in computer.
- It provides flexibility & less cost of Hardware.

Transistors and virtual Architecture

High-level language -----> Machine language

1] Translation:-

Input: High level language programe.

Output: Machine language code.

Advantage: Efficient Executable.

2] Interpretation:-

- The programe is not translated to machine learning code instead, it is executed by another programe.

Advantages:-

- Easy to be implemented.
- Easy to debug.
- Portability.

D] Virtual computers:-

- It is completely reconfigurable in every respect.
- computing machines based on reconfigurable logic are hyper-scalable.

E] Binding Time:-

- Binding time is the moment in program's life cycle when its association occurs.
- Many properties of programing language during it's creation
- Binding occurs at

① Language Def^n ② Language Implementation.
③ Translation ④ Execution.

Q.5] What is the difference between compiler & interpretter.

→	Interpreter	compiler
1.	Interpreter translate just one statement of programe at a time into machine code.	compiler scans entire Programe and translate the whole of it into machine code at once.
2.	An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
3.	An interpreter does not generate any intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence, more memory is needed.

Ref:- Buisnessinsider.com.

4.	Keeps translating programs continously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scan the complete programe & hence rela tively debugging is harder while working with compiler.
5.	Interpreters are used by Progra- mming languages like Ruby & Python	compilers are used by programming languages like C and C++.

Q.6] Explain various classes of Binding time.

→

Binding time :- It is the moment in the programe's life cycle when this association occurs.

- Many properties of Programming language are defined duri it's creating.

1. Language Definition :-
   - Available data types and language structures.
   - e.g] in C++ the assignment statement is =, while in Pascal it is :=
2. At Language implementation :- concerns representation of numbers & arithmetic operations.
3. Binding at Translation :-
   3.1) chosen by programmer :- variable types & Arguments.
   3.2) chosen by compiler :- Relative location of variable & arrays.
   3.3) chosen by loader :- Absolute location.

4. At execution:-
- Memory containts
- Recursive programs
- Dynamic libraries.

5. At arbitrary point during execution:-
- Some binding can appear at any point during the execution of programme.
- An example is the basic binding of variable to values through assignment, whereas some languages LISP, smalltalk & ML allocate binding of names to storage locations it also appear at arbitrary points in the programe.