SP

Unit - 6

Q7 What are the fundamental steps in prg. development? Explain in detail.

→ Steps:

1. Program design, coding & documentation.
2. Preparation of prog. to machine readable form.
3. Prog. translation, linking & loading.
4. Prog. testing & debugging.
5. Performance Tuning.
6. Reformatting data &/or results of a prog. to suit other prog.

① Program design, coding & documentation
In this two tools are used:
- (a) Program Generators
- (b) Programming Environment

(a) Program Generators:-
- Generate a prog. which perform a set of f^n described in its specification.
- Saves design efforts.
- Coding efforts are saved
- Program is generated instead of hand coding.

(b) Programming environment including
- Supports program coding by incorporating awareness of programming lang. syntax & semantics & lang. editors.

Prog. Documentatn :-

- Project suffers from lack of up-to-date documentation.
- so, automatic documentatn tools are motivated & develop documents for source prog.

② Preparatn of prog. to machine readable.

- Ultimately, the source code of every human readable programming lang. must be translated to machine readable lang. (i.e binary lang.) by a compiler, because code is only lang. that computer hardware understand.

③ Prog. translation, linking & loading

Translatn - Prog., written in source code, is translated by compiler to produce ~~machine readable lang.~~ a ~~linking~~ prog. in target lang., that may or not necessarily be machine lang.

Linking

- Then the prog. is linked with other object prog. to build an executable prog, which is saved in specified locatn.

Loading

- When it is needed to be executed, the executable file is loaded into main memory before its execution.

Full Preparatn Also Individual questn for SM

(4) Prog. testing & debugging
- steps int'd
  1] selection of test data for prog.
  2] Analysis of test results to detect errors.
  3] Debugging.

- Tools assisting for testing & debugging:-
  1] Test Data Generators
  2] Automated Test Drivers
  3] Debug Monitors
  4] Source Code Control system.

(5) Performance Tuning :-
- It depends on
  - Efficiency of Algorithm
  - Efficiency of coding

2] State & explain types of editors. Also explain design of editors.

→ (a) Following are types of editors:
1. Line
2. Stream
3. Screen
4. Word Processor
5. Structure Editor.

(1) Line Editor:-
- Scope of edit operatn in line editor is limited to line of text.
- Line is designated either
  - Positional or
  - Contextually

② Stream Editor
- Views entire text as a stream of characters.
- Permit edit operat^n to cross line boundaries.
- Support character, line & context oriented commands.
- Indicated by
  - Position of text pointer or
  - Search commands

e.g. Dos File

③ Screen Editors
- Doesn't display text in manner in which it would appear for printing.
- Uses 'What you see is what you get" principle.
- Display screen full of text at a time.

e.g. Notepad, Wordpad.

④ Word Processors
- Basically document editors.
- Produce well formatted hard copy output.
- Features:
  - Copy/Paste/Cut
  - Find/Replace
  - Spell check

e.g. Word Start, MS Word.

5] Structure Editor :-
- Useful in browsing through document.
- Creation & Modification is very easy
- It is also called as syntax directed editor.

e.g. EditPlus, Netbeans.

(b) Design of Editor :-
- Fundamental fn of Editing are :-
  1) Travelling
  2) Editing
  3) Viewing
  4) Display

1) Travelling :- implies movement of the editing context to a new position within the text.

~~2) Editing :-~~

2) Viewing :- implies formating of text in a manner desired by user.
It is abstract view.

3) Display : components maps this view into the physical characteristics of display device being used.

Q3] Explain Debug Monitors:-

→ Provide following facilities:-

- Setting breakpoints in prg.
- Initializing debug conversation when control reaches breakpoint.
- Displaying values of variables.
- Testing user defined assertions & predicts involving prg. variables.

• Debug monitor funct^n can be easily implemented in an interpreter.

• But interpreters incur ~~condi~~ considerable execution time penalties.

• Hence, a debug monitor rely on instrumentation of prg.

• User must compile prg. under debug option.

• To display bug option for particular statements: no-op<statement number>

• Compiler generates table (var nm, address).

• Set break points with <si-Instruction><code>

• Prg. executes on CPU until si-instruction is reached.

• Code produces interrupt code.

• Instead of <si-instruction> We can use
  • BC ANY
  • DEBUG_MON

• Now debug monitor gains control & opens a debug conversation.

Q4] Write notes on –

a) Programming Environments :-

- It is a system software.
- Provides integral facilities for:-
  - Program creation.
  - Editing
  - Execution
  - Testing
  - Debugging.

- Components:-
  1] Syntax directed editor.
  2] Language Processor
     - Compiler
     - Interpreter
     - Both
  3] Debug monitor
  4] Dialog monitor.

- Components are accessed through dialog monitor.
- Syntax directed editor incorporate front end.
- Editor performs syntax analysis and construct IR giving abstract syntax tree.
- Compiler & debug monitor share IR.
- Thus, program execution starts then.
- In between, any time during execution, programmer can interrupt execution, resume program & restart execution.

- Debug monitors provide easy accessibility to all functions.
- Also allow,
  - Generation of prog.
  - Testing funct^n
  - Partial compilat^n & prg. execution.

- Errors are indicated if encountered.
- Also permits
  - reversible execution &
    irreversible ——— .

e.g. Cornell Prog. Syntheizer (CPS)
- It is a syntax directed programming environment. for PL/CS, which is subset of PL/I.

Contains:
- Syntax directed editor
- Compilation & Interpretation schemadic for execution.
- Collection of debuggig tools.

b) Design of Software tools.

- It has four tools:

① Program Pre-processing
② Program Instrumentation.
③ Program Interpretation
④ Program Generation.

① Program Pre-processor:-

- Used to support static analysis of program.

- Generate:-
  - Reference listing
  - List of un-referenced symbol.

- Uses:-
  - Test data generators.
  - Documentation aids.

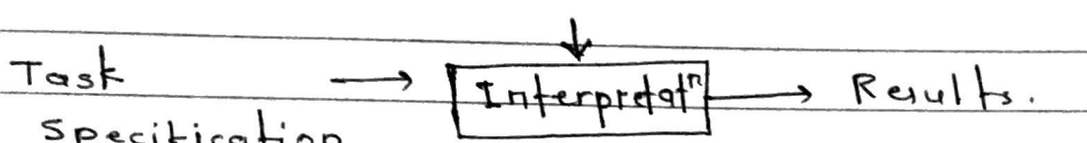
② Program Instrumentation:-

- Implies insertion of statement in prog.
- The instrumented prog. is translated by standard translators.
- Used by:-
  - Execution Profiles
    i.e. How may time statement is executed.
  - Debug Monitors
    i.e. Where execution has reached

③ Program Interpretation :-

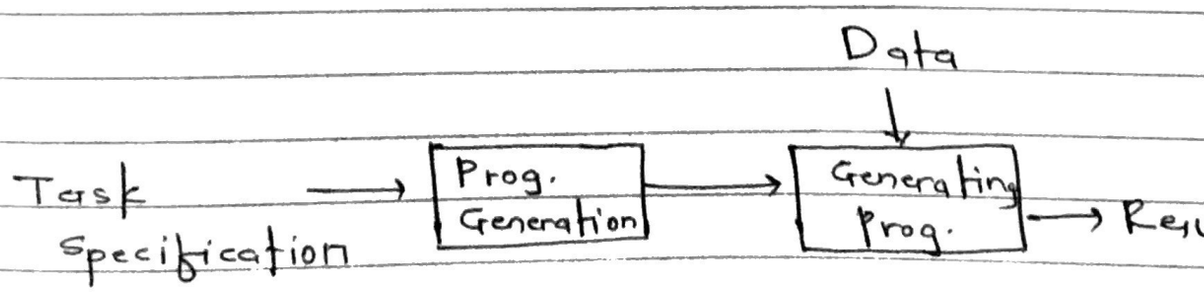
- Motivated by absence of translation phase in processing program.
- Most of the requirements which are met by software tools are ad-hoc.
- This helps eliminate translation phase from program execution.
- These tools suffers from
  - Poor efficiency
  - Poor portability
- Then, too, programs that is generated is efficient & can be made portable with few efforts.

Interpretation :-



④ Program Generation :-

- Need input as scree specification.
- Require less memory than interpretation.
- Also speedy in comparison of interpreter
- Permit dynamic changes in specificatn.
- Provide execution efficiency.
- Also provides code space efficiency.



c] User Interface:- (UI)

- Role : Simplifying interaction of a user with an application.
- Two Aspects:-
  1) Issuing Commands
  2) Exchange of Data

Requirement of UI:

- Before requirement was small & hence small applications.
- But now, application have become so large that one programmer's team don't know anything of other team working on same application.
- UI keep track of information of functionalities & educating users for those application.

- Two Components:-
  1) Dialog Manager:
     - Manages conversation betⁿ user & application.
     - Transmits command to applicatⁿ.
  2) Presentation Manager:
     - Displays data produced by the application in appropriate manner on screen.

UI cover following topic:-

1] Command Dialogs
   - Principle of command dialog design.
2] Presentation of Data
3] Online Help
   - Hypertext
4] Structure of UI
5] UIMs - User Interface Management Systems
   - MenuLay
   - HyperCard.

Q5] How command dialogs are implemented?

→ - Commands are issued to an application through a command dialog.
- Three ways to implement them:-
  1) Command Languages
  2) Command Menus
  3) Direct Manipulation.

1) Command Languages
- Similar to command languages for OS.
- Primitive Command languages support:-
  - Imperative Commands
  - Syntax: <action><parameters>
- Sophisticated Command language support:
  - Imperative Commands (e.g. instruction)
  - Declarative commands (e.g. algorithm)
- Have their own syntax & semantics.

Problem: - Need to learn syntax before using application.

Soln - Online we get help & reduce effort of memorizing.

2) Command Menus:
- Basic functionalities of application are reflected.
- Provide advantage.
- Hierarchy of menus explain functionality of applications.
  e.g. file, edit, view.
- Use can be simplified by "pull down" menu utilies. e.g. c, toolbar.

3) ~~Command~~

3) Direct Manipulation:
- provides user with a visual display of universe of application.
  i.e. C Help, MSDN.
- Display shows important objects in universe.
- Actions or operations over objects are indicated using pointing device e.g. Cursor.
- E.g Spread Sheet.

Q6] Explain structure of user interface

→ Two components:-

1) Presentation Manager: Responsible for
   - Managing user's screen.
   - Accepting data.
   - Presenting results.

2) Dialog Manager: Responsible for
   - Interpreting user command
   - Implementing them by invoking related modules
   - Error messages
   - Online Functions
   - Organizing changes in View.