Module – IV

Coding Techniques.

→ **Coding** :

Goal of Coding is to implemnt the design in the bst possible manner.

There are many diffrnt criteria for judging a prgm including readability, size of the prgm, execution time & required mly.

→ **Structured Coding Techniques** =

1) **Imperative** :

Tell prgm 'how' to do smething or solve prbm exactly.

2) **Declarative** :

Tell prgm 'wht' to do instead of worrying about hw exactly smthing needs to do done (നമ്മ് rslt ആണ് കിട്ടേണ്ടത്)

3) **Structured** :

code will execute the instru by instru 1 after the othr (eg → c).

4) **procedural** :

prgm code is ÷ into procedures & are discrete blks of code that carry out a single task.

5) Functional :
Style of progming that emphasizes the evaluation of exp rathr than execution of cmnds.

6) obj oriented :
prgming by defining obj that snd msgs to each othr.

7) Evnt drivn :
programing with emitters & listeners of asynchronous actions.

8) Flow based :
programing processes communicating with each othr ovr predefined channels.

9) Logic :
programing by specifying a set of facts & rules.

10) Array :
progrming with powerful array operators.

→ Coding Styles =

* Refers to technique used in writing the source code for a comp prgm.

* Are designed to hlp prgmrs quickly read & undrstands the prgm as well as avoid making errors.
* A gd c. style can overcome the many deficiencies of a bt prgming lang, while poor style can defeat the intnt of an excellent lang.
* The goal of gd prgming style is to provide understandable, & elegnt code.
* General rules in respect to p. style —
1) clarity & simplicity of exp:
prgms should be simplr & clear.
2) Naming :
care should be takn that naming style should not be cryptic & non-representative.
eg → instead of giving a = 3.14 * r * r
you can give area of circle = 3.14 * radius * radius.
3) control constructs :
use single entry & exit constructs as possible.
4) Info hiding :
can decrease the coupling b/s modules.

5) Nesting :
Becomes difficult to undrstnd the prgm logic, so avoid deep nesting.

6) user-defined types :
makes yr prgm code easy to write & easy to undrstand.

7) module size :
Should be uniform.

⇒ Coding Standards & Guidelines =

* Coding Standards refers to hw the developr writes code, ~~so there are with discuss standards~~
* An imp thing to remembr about coding Standards that many prgmrs seem to forgt is that C. Standards are not an abstract idea.
* 2 types → C.s written for a company
→ C.s written for a progming lang.

* coding Standards —

1) Indentation :
Propr Indutation is essential in producing easy to read & maintainable prgms.

used to emphasize the body of a conditional (s) & a new scope blk.

2) Inline cmnts :
Analyze the functioning of the Subroutine :
Subroutines is a Sequence of prgm instrn that performs a specific task.

3) Structured prgming :

4) Naming conventions for global, local var & const Identifiers :
Global var names always begin with a capital letr, local var names are made of small letrs & const names are always capital letters.

5) Error return conventions & exception handling systm :
All fns that encounting an error condin should either return a 0 / 1 for Simplifying the debugging.

→ Coding Guidelines :

* provide the prgmr with a set of the bst methods which can be used to make prgms more comfortable

to read & maintain.

1) **Line Length:**
Length of Source code lines at or below 80 char.
Sme printrs will truncate lines longr than 80 columns.

2) **Spacing:**
use of spaces within a line of code can improve readability.
eg → Bad ⇒ print("Hello world");
Better ⇒ print (" Hello world ");

3) **code should be well-documnted:**
As a rule of thumb, there must be at least 1 cmnt line on the avg for 3 source line.

4) Length of any () should not exceed 10 lines:
very lengthy () is very difficult to understand & leads to many bugs.

5) Do not use goto (S):
It makes a prgm unstructured & very tough to understand.

6) Limited use of globals.

7) **Error msgs:**
Error handling is essential aspect of comp prgming.

→ **Documentation standards** =

* Apart (S) develope source, executable files, several kinds of doc — SRS, design doc, test doc, etc are developed as a part (S) eng process.
* 2 types —

(a) Internal docmntation	(b) External docmn.
* It is the code comprehension featres provided in the source code itself.	These are the supporting doc like SRS doc, installation doc, design doc & test doc.
* eg: code indntation, module & () headrs, cmnts embedded in the source code.	Important feature of this docmn is consistncy with the code.

⇒ **Modern prgming Lang Features:**

* provide a variety of features to support development & maintaince of (S) products.

Py, JS, perl → dynamic PL.

* Main features are —

a) Type checking :

* process of ensuring that a prgm obeys the lang type compatibility rules.
* occur either at static time / dynamic time.

a) Static type :	(b) Dynamic Type :
It is done at compile time & It means that type of a var is known at compile time.	process of varifying the type safety of a prgm at runtime & type of var is allowed to change ovr its lifetime.

b) User defined datatype :

* It is the data type that derived frm an existing datatype.
* PL hve their own s/x & semantics in defining the user defined datatype
* 2 data type that used in C PL for improving logical of prgm —

1) Type definition (typedef) :
Allow user to define new datatypes that are equilent to existing datatype.
s/x → typedef type new-type ;

2) Enumration (enum) ;
used to define user defined type
s/x → enum identifier (mem1, mem2 ... membern);

c) Data Abstraction :

* Separates the interface & implementation by enforcing the interface to datatype.
* eg → Data type, info hiding Modules, abstract datatypes.

d) Exception Handling :

* Exception feature were added to progrming lang to provide the prgmr the capability to specify wht should hpn whn unusual execution coincidences.
* An Exception is an evnt that suspends normal execution of a prgm

* Below is the list of imp built in exception in java — Arithmetic, class not found, null pointr, interrupted, etc.

e) Concurrncy mechanism :
* It is ~~the~~ collection of techniqus & mechanisms that enable a comp prgm to perform several diffrnt task simultaneously.
* 3 fundamntal approachs to concrnt programing —
1) Shared variable : multiple processes hve access to a commony region of a mfy
2) Asynchronous msg passing : It involves association of buffers with concrnt tasks.
Allows a sndr to continue doing othr things as soon as the msg has been snt
3) Synchronous msg passing : The msg is passed directly b/w the sndr & receivr without being buffered in-b/w.
This requires the sndr to block until the receivr has received the msg, bfre continuing doing othr things.