

## Module - IV

CLASSMATE

Date \_\_\_\_\_

Page \_\_\_\_\_

### Coding Techniques.

⇒ Coding :

Goal of Coding is to implement the design in the best possible manner.

There are many different criteria for judging a program including readability, size of the program, execution time etc required only.

→ Structured Coding Techniques =

1) Imperative:

Tell program 'how' to do something or solve problem exactly.

2) Declarative:

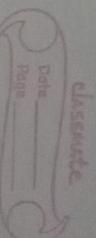
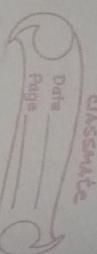
Tell program 'what' to do instead of worrying about how exactly something needs to be done.

3) Structured:

Code will execute the instruction by instruction after the other (e.g. → C).

4) Procedural:

Program code is divided into procedures. There are discrete blocks of code that carry out a single task.



### 5) Functional :

Style of programming that emphasizes the evaluation of exp rather than execution of commands.

### 6) Obj oriented :

Programming by defining obj that send msg to each other.

### 7) Event driven :

Programming with emitters & listeners of asynchronous actions.

### 8) Flow based:

Programming processes communicating with each other over predefined channels.

### 9) Logic :

Programming by specifying a set of facts & rules.

### 10) Array :

Programming with powerful array operators.

### → Cooling Styles =

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

### 1) Naming :

Style should be taken that naming style should not be cryptic & non-representative.

→ Instead of giving  $\pi = 3.14 + 8.4 \times$  you can give area of circle =  $3.14 \pi \times \text{radius} * \text{radius}$ .

### 2) Control constructs :

→ we single entry & exit constructs as possible.

### 3) Info hiding :

→ can decrease the coupling b/w modules.

\* Are designed to help programs quickly read & understand the prgm as well as avoid making errors.

\* A gd c. style can overcome the many deficiencies of a ft. programming lang.

\* The goal of gd programming style is to provide understandable, elegant code.

\* General rules in respect to prg-style—

### 1) Clarity & Simplicity of exp:

Programs should be simple & clear.

### 2) Naming:

care should be taken that naming

style should not be cryptic & non-

- 5) Nesting: becomes difficult to understand the logic, so avoid deep nesting.
  - 6) user-defined types: makes the code easy to write & easy to understand.
  - module size: Should be uniform.
- ⇒ Coding Standards & Guidelines:
- \* Coding Standards refers to how the developer writes code, ~~for better one~~ ~~with better standards~~
  - \* An important thing to remember about coding standards is that many programs seem to forget is that C standards are not an abstract idea.
  - \* 2 types → c.s. written for a company or a programming lang.

- Coding Guidelines:
- \* Coding Standards →
    - 1) Implementation: proper indentation is essential in producing easy to read & maintainable progs.
  - \* provide the progs with a set of the best methods which can be used to make progs more comfortable.

to read & maintain.

### 1) Line length:

length of source code lines can  
or below 80 char.

Some printers will truncate lines  
longer than 80 columns.

### 2) Spacing:

use of spaces within a line of  
code can improve readability

e.g. → bad  $\Rightarrow$  print("Hello world")  
better  $\Rightarrow$  print("Hello world")

### 3) code should be well-documented:

As a rule of thumb, there must  
be at least 1 comment line on the  
avg for 3 source line.  
4) length of any file should not exceed  
10 lines:  
Very lengthy file is very difficult  
to understand & leads to many bugs.  
5) Do not use got o (5):

It makes a program unstructured!

It's very tough to understand.

6) Limited use of global.

7) Error msg:

Error handling is essential aspect  
of comp. prgming.

### → Documentation Standards =

\* 63 of 65 developed software, executable files in  
Bjarne Stroustrup kind of doc -  
SRS, design doc, test doc, etc are  
developed as a part of eng process.

2 types -

#### (a) Internal documentation

#### (b) External docm n.

\* It is the code  
comprehension features  
provided in the  
source code itself.

These are the  
supporting doc

like SRS doc,

installation doc,

design doc &  
test doc.

e.g. code indentation,

module & connectors,

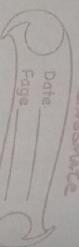
counts enclosed

in the source code.

Important feature  
of this docm is  
consistency with  
the code.

### ⇒ Modern programming lang Features:

\* Provide a variety of features to  
support development & maintenance  
of 65 products.



\* Main features are —

### a) Type checking :

- \* process of ensuring that a program obeys the lang type compatibility rules.
- \* occurs either at static time /
- dynamic time.

### b) Static type :

It is done at compile time. Eg

It means that type of a var known at

compile time.

type of var is allowed to change

over its lifetime.

### c) Dynamic Type:

process of verifying

the type safety

of a program at

runtime. Eg

type of var is

allowed to change

over its lifetime.

### b) User Defined datatype :

\* It is the data type that derived from

an existing datatype.

\* PL has their own set of semantics in defining the user defined datatype.

\* 2 data type that used in a PL for improving logical of program

### 1) Type definition (typedef):

Allow user to define new datatype that are equivalent to existing datatype.

Eg → typedef type new-type;

### 2) Enumeration (enum):

Used to define user defined type

Eg → enum Identifier (member, member, member);

### 3) Data Abstraction:

\* Separates the interface & implementation by enforcing the interface to datatype.

Eg → datatype, interface

modules, abstract datatypes.

### d) Exception Handling :

\* exception feature were added to programming lang to provide the program the capability to specify what should happen when unusual execution conditions.

An exception is an event that

suspends normal execution of a program

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

## ② Concurrency mechanism :

- \* It is ~~the~~ the collection of techniques mechanisms that enable a program to perform several different tasks simultaneously.

- \* 3 fundamental approaches to concurrent programming —

- 1) Shared variable : multiple processes have access to a common region of memory.
- 2) Asynchronous msg passing : It involves associations of buffers with concurrent tasks.

Allows a sender to continue doing other things as soon as the msg has been sent.

- 3) Synchronous msg passing : The msg is passed directly by the sender to receiver without being buffered in-b/w. This requires the sender to block until the receiver has received the msg, before continuing doing other things.