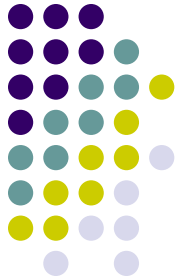


Managing Software Development

Day 5 Software Implementation

Source code with readability and understandability

Charoen Vongchumyen

Email : charoen.vo@kmitl.ac.th

02/2021

Thank to Tsuneo Yamaura

Software Implementation

- Implementation phase (Coding Phase)
  - The process to convert the detailed design to source code
    - → Do no change the contents of the detailed design.

Understand by human being → Understand by Machine

Func. Spec.	Design	Coding Debug	Test	Maintenance

Software Implementation

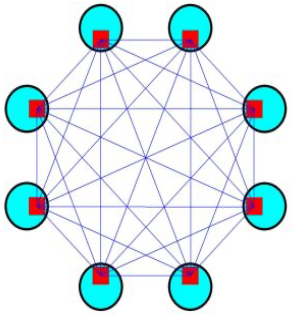
- Implementation phase (Coding phase)
  - Theoretically, coding by a single person is ideal.

    → A single person will be able to code 20KLOC/Person-year

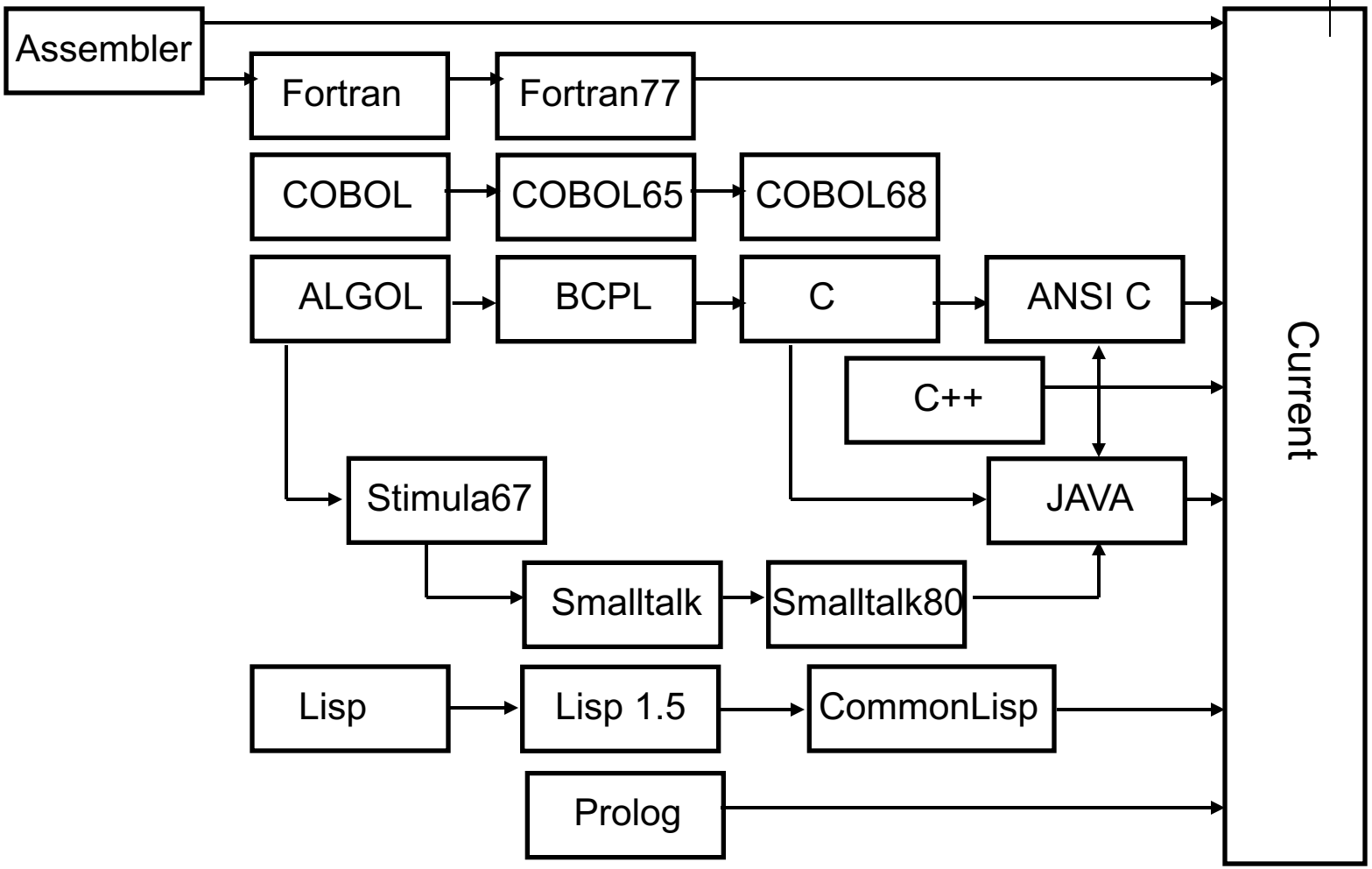
    → In a project, his coding will be reduce to 10KLOC/person-year due to overhead including communication among members.

Ideal prj

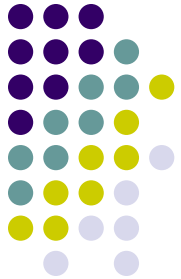
Communication
Path = n(n-1)/2



History of programming language

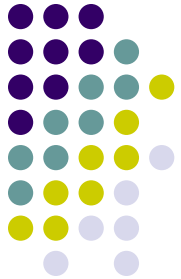


Programming manner (guide lines)



- The basic rules and styles for better coding like dining manners.
  (1) Easy to understand by the person who wrote the code.
    - When coding, easy to write and read.
    - When coding, not to make bugs.
    - When debugging, easy to detect bugs in the code.

Programming manner (guide lines)



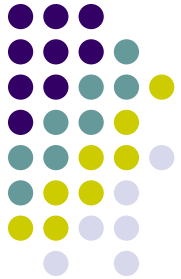
- The basic rules and styles for better coding like dining manners.

(2) Easy to understand by others

- When maintaining (by others), easy to pinpoint the code that should be changed.

  (Most likely, the person who maintains the code is not the person developed the source)

Basic ideas of programming style



- Four basic rules for better understanding

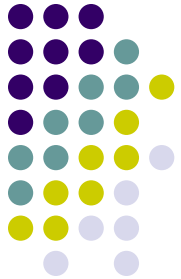
  (1) KISS (Keep It Simple and Stupid) coding.

  (2) Use a simple control structure.

  (3) Coding that does not target a particular language.

  (4) Add comments for better understandability

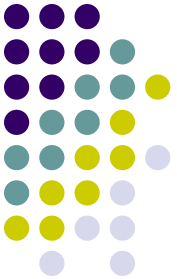
Basic ideas of programming style



(4) Add comments for better understandability

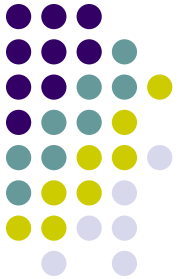
- As a header, add comments that describe:
  - Function of the module
  - Input data
  - Output data
- If you use complex algorithm, use formula and charts for better understanding.

Structured programming



- Basic of the programming style
- The most famous and used programming methods.
- History of ‘Structured programming’
  - (1) 1966 : Mathematical Theory
  - (2) 1968 : Letter from Dijkstra
  - (3) 1974 : Paper by Knuth

Structured programming

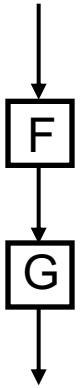


- History of the Structured Programming

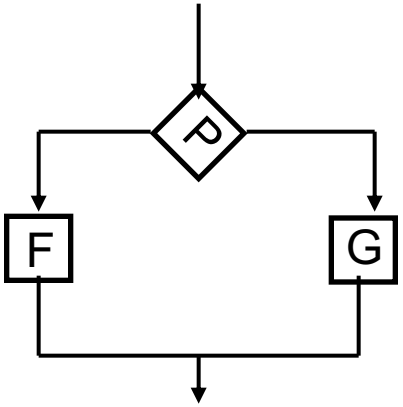
  (1) 1966 : Mathematical Theory (by Barry Boehem)

  → All the programs can be described with Concatenation, Selection and Iteration.

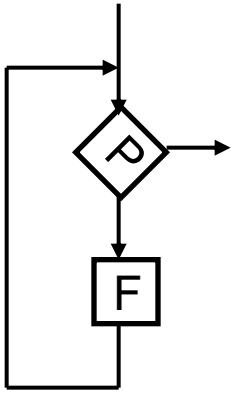
Structured programming



Concatenation
F follow by G

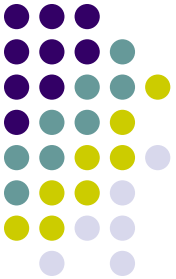


Selection
If P, then F else G



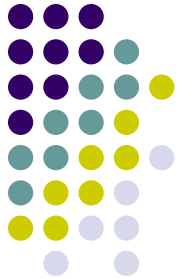
Iteration
While P do F

Structured programming



- Forty years ago, COBOL, Fortran and assembler were the major computer languages, which provided unconditional GOTO statement (they only had GOTO statement)
- The mathematical theory of programming without GOTO statement did not draw attention of computer engineers and scientists

History of the Structured programming



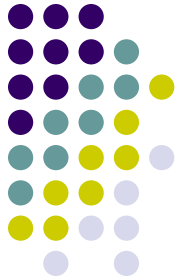
(2) 1968 : A letter from Dijkstra to ACM (Association for Computing Machinery)

'GOTO statement considered harmful'

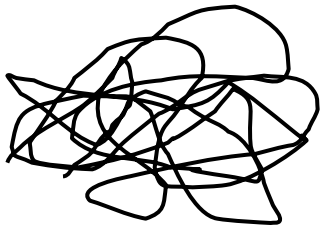
- Forty year ago, There were no computer language that provided Concatenation, Selection and Iteration.

  → Structured COBOL and Fortran came to use long afterword.

History of the Structured programming

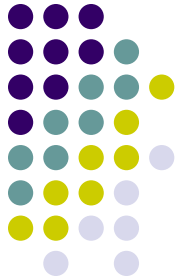


- There was a heating argument if ‘GOTO statement considered harmful’ was true or not.
- Some people said ‘We cannot program without GOTO statement’



GOTO statement increases spaghetti structure?

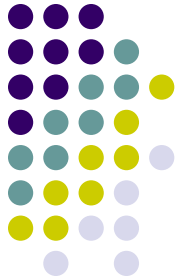
History of the Structured programming



(3) 1974 : A paper written by Knuth completed the argument.

- 'Structured programming is not 'Programming without GOTO statement', but is programming of easy to write and understand.
- The GOTO statement may make the programming structure simple (for ex. Error handlings)
- The argument came to conclusion, and cooled down.

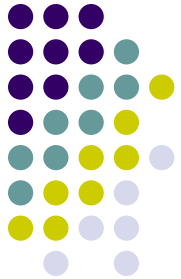
History of the Structured programming



(4) 1974 : A paper written by Knuth completed the argument.

- 'Structured programming is the one that does not use GOTO statement if possible.
- Structured programming is the one that is easy to write and understand.
- The GOTO statement should be limited only for error handling, and should be 'Forward GOTO'
  - When you enter a house, use doors not windows.
  - In the case of fire, evacuate from the nearby window.

History of the Structured programming

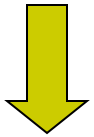


- The definition of Structured programming

The source code is called Structured programming if

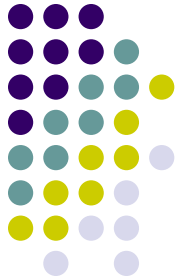
(1) The code only has concatenation, selection and iteration.

(2) Each module has only a single entry and exit.



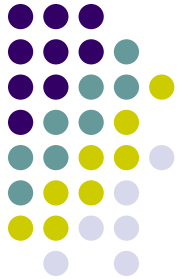
The modern Structured programming has on more structure : 'Call'

Programming style guide lines



1. Variable names
   - Easy and simple to understand for Developer and the person who maintains the code.
   - Use English names : English is a universal for SW engineers.

Basic rules of naming variables



AVERAGE_FREQ

FREQUENCY_MAXIMUM_01 → Do not use number

MINIMUM_FRQNCY

MIN_FR → Do not change orders

FRQNCY_TOTL

↓ Different var should use clearly different names

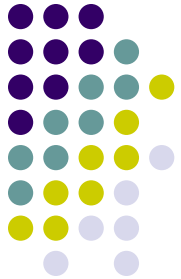
FREQ

FREQUENCY

FRQNCY

Use same word

Programming style guide lines

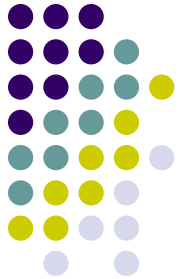


- ‘Self-documenting coding’ is very detailed descripting naming, That does not require comments, For ex.

  AVERAGE_POINTS_OF_MATH_IN_CLASS_2-A

  (1) Advantages
  - You can understand the meaning without reading documents.
  - Looks like source code is written with natural language.

Programming style guide lines

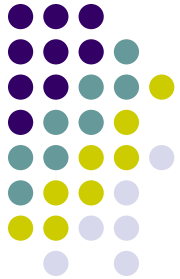


- Naming variables : self-documenting code??

  AVERAGE_POINTS_OF_MATH_IN_CLASS_2-A

  (1) Disadvantages
  - You have to be an expert programmer to come out sophisticated self-documenting code.
  - All the variable names cannot be sophisticated like that
  - The names are too long.
  - Too much depend on self-documenting code, Not be encouraged to write actual documents. (Bad maintainability)

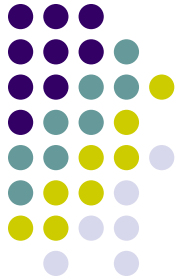
Programming style guide lines



2. Prologue comments
   - Programmers do not read source code to understand the function.
   - For easier understanding, Write following comments as a module header:
     - Module name
     - Brief description of module functions
     - Name of programmer, make date
     - Variable names, Formats and usage
     - Callers and callees
     - Error handling
     - Change history

Keep update the comments!

Programming style guide lines



3. Maintenance comments

- When source code is changed (due to error correction or maintenance), add comments that describe the change

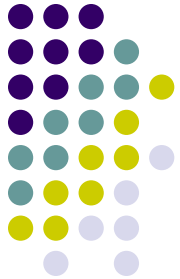
Source code

```
If item 01 > 100
    then
       price05 = 200
    else
       price05 = 300   #0123
End
```

change information #0123

- Why changed (if error describe the bug)
- Who changed and when
- Code before change
- Code after changed
- Test items

Programming style guide lines



4. Indentation

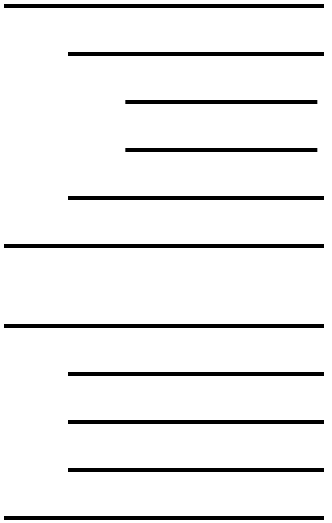
- Indentation is a powerful method for better readability.
- Basic rules for indentation are:

  (1) Write one statement in a single line.

  (2) Give same spacing to the same level.

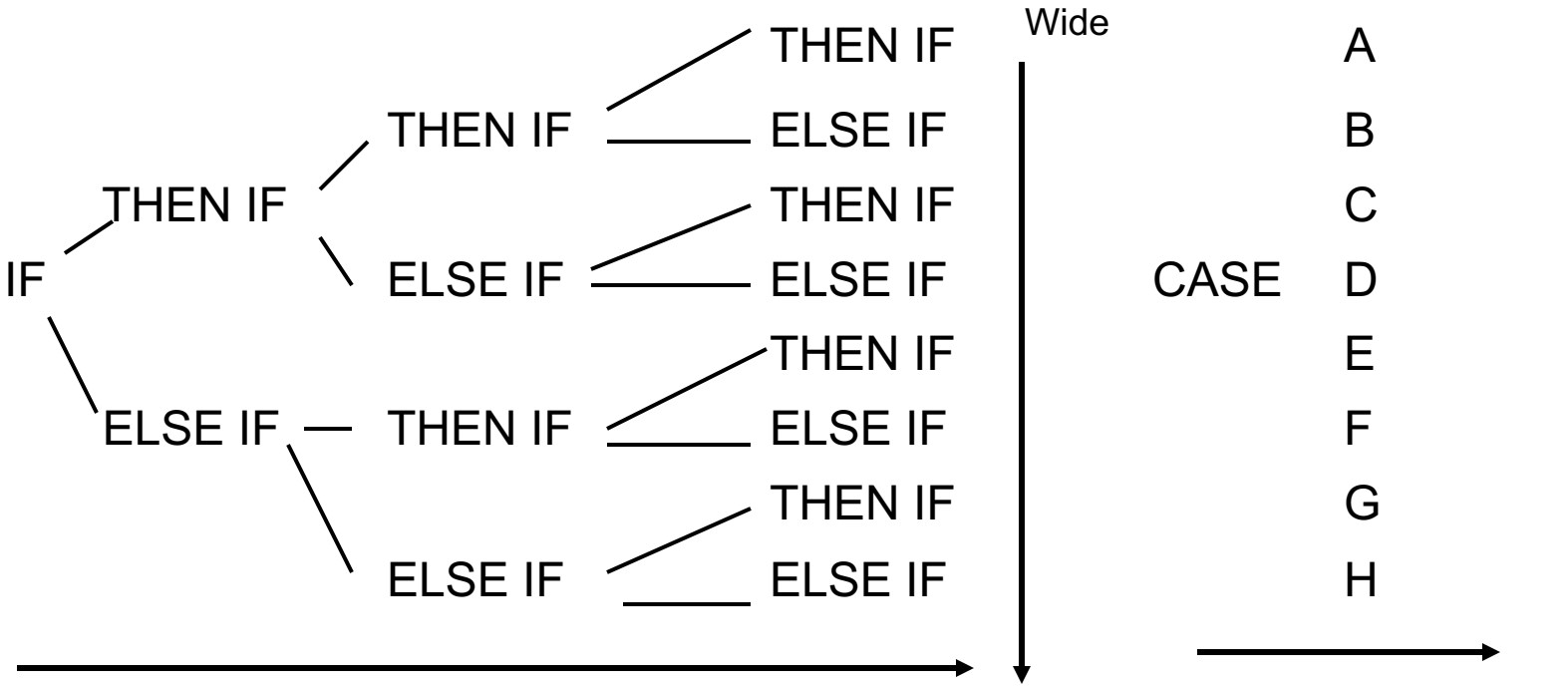
  (3) Use a text editor for automatic indentation.

  (4) Use a blank line for separation blocks.



Programming style guide lines

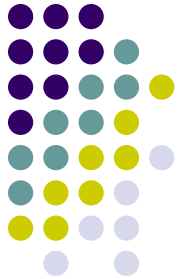
5. Nesting of IF statement



Nesting is too deep, Not > 4 levels

Nesting is not deep

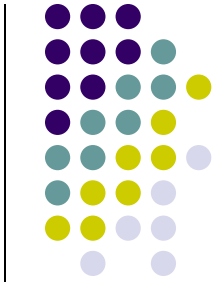
Programming style guide lines



6. Coding standards

- Coding standards are the rules that are applies to all the projects, eg. LOC of each module < 50
- The enforcement of standards will brings better readability.
- If applied too strict, you may have disadvantages
  - Needs time to learn the standards and check the code
  - Programmer unintentionally follow the standard.
  - So just guide line not law.

Practice



Implement the Triangle program from your perverse specification and design.

With any language.