Structured Programming

A VERY important Philosophy for Writing Programs

This is supposed to be a refresher only. You are expected to hone your skills by writing programs and reading books on structured programming if you have not mastered structured programming
Bad Programming Habits

- Foggy idea about what is to be done
- Write program with no planning; Start from the beginning and write to the end
- No systematic debugging; Considered it finished if it works on one example

Undesirable Results!
- Do not know how to program
- Programs with numerous bugs that take extremely long time to debug, or even failure to complete
- Any change of requirement invites rewriting of the entire program again
Benefits of Structured Programming

- Programs that meet the needs of the customer
- Though initially take longer time to generate code, often result in code with runs with no bugs the first time it’s run
- Easy to handle change in program specifications in the future
Structured Programming

- A tool that becomes popular since the 70’s
- Should have been learnt by student that have taken any programming course
- Absolutely essential for handling large programs that involve a team of programmers and huge number of man hours.
- The other popular philosophy is “object oriented programming”, but many programmers prefers structured programming
Seven Important Concepts of Structured Programming: 1. Structured Walkthrough

- Before writing any program, the programming team must sit down with the customer and find out the requirement
- Extremely important
- Customer’s requirement is often imprecise
- Iterative: several rounds of talks
- Must result in a specifications that is
  - very precise
  - Understandable by programmer in programming terms
2. Stepwise Refinement

- A “DIVIDE and CONQUER” strategy
- When given a large job, divide it into smaller jobs.
- Given any job, it is useful to divide it into
  - Input
  - Process
  - Output
- Draw a tree
- Refine each job level by level (Breadth first)
- Use pseudo code to describe each job
- Decision on data structure is delayed as much as possible
Extremely Complex Job

Input
- Input from user
- Initialize security settings

Process
- Initialize

Output
- ...
- ...

Level 1
Level 2
Level 3
3. Modular Design

- Each ellipse is a module
- A module is a self contained block:
  - It **only** receives inputs from its immediate ancestor
  - It **only** outputs to its immediate ancestor
  - Its computation should **only** require calling functions that are its immediate children and them only
- The input variables and output variables of each module should be specified when defining the module
- Each module must be “programmable” – no majic block should exist
4. Bottom Up Coding

- When the refinement has reached a simple function, code the simple function
- You can test the simple function independently of the rest of the program
- This gives you achievement and satisfaction, sustaining you through the long project
- Project Manager exercises division of labour here, ask a member to be responsible solely for that function
5. Testing Using Stubs

- A structured Programming project can be field tested before everything finishes.
- Stubs - for unfinished modules, use a human being to emulate it, act on the test inputs, she fits in the correct output data by hand.
- Then other programmers can test their work.
- Meanwhile she continues to program her own module (according to MS Project timelines).
6. White Box and Black Box Testing

- For each module and whole program

- White Box
  - Input something for which you know the desired result, it should give your expected output

- Black Box
  - Treat it as a black box, input some data, is the result reasonable?
A structured programming document is generated along with the program.

When requirement of customer changes, go to the document.

Does not need to rewrite the whole program, just find which modules need to rewrite and rewrite the module and the sub-tree under it.

Programmer usually forgets their code in 2 months; the structured document helps her to refresh her work quickly.
Advice

- YOU MUST TRY IT TO LEARN IT
References

- General Philosophy
  - Numerous books about Structured Programming in the library

- A very good structured programming example
  - W. Findlay and D.A. Watt, Pascal: An Introduction to Methodical Programming 1987, Ch. 7 illustrates how to use stepwise refinement to program a complicated task. **Try it YOURSELF once, then you would get it**