C++ Completing a Program

Chapter 7 pages 219 – 252 (first ed.)

Note: chronological according to book

Blue font: principles (about UI, clean code & testing)

Green font: repeated fragments about commenting

Introduction

- Considerations of a professional programmer
- Improve calculator from chapter 6
- Gradually improve a program
  - Note: This chapter does not teach you on «completing a program» despite its title

Input and Output

- Considerations about UI
  - Layout: Prompting for input(s)
  - Layout: Presentation of results

Error handling

- Invalid entries exist
  - Program will fail
- Keep window open
  - e.g. with «tilde» character. (~ not found on key board)
  - e.g. «q» to exit
- Robust error handling
  - With «catch»

Negative Numbers

- (0-1)/2 for «-1/2» to avoid fail
  - Improve by adding negative number as a Primary
  - Add to grammar
- Very easy

Remainder: % (modulo operation)

- Only working for integers
  - % as a token
  - Convert doubles before and after % to int
    - int i1 = int(left); or int i1 = narrow_cast<int>(left);
    - int i2 = int(d); or int i2 = narrow_cast<int>(d);
  - Add % to calculator (as a case)
  - Add error for doubles except 5.0
  - Add error for division by zero
    - if (i2 == 0) error («divison by zero»)

Cleaning up the code

- Measures for clean code:
  - Short
  - Clear
  - Good comments
- Actions
  - Symbolic constants (replace «8» for number)
  - Symbolic name (instead of comment)
  - Delete unncessary comments
  - Use global constants (like «e» for exit)

Cleaning up the code (cont.)

- Commenting (see Bob Martin)
  - Comment still valid?
  - Comment needed / adequate?
  - Short and concise?
  - Comments should express intent
    - for example in grammar

Cleaning up the code (cont.)

- Use of functions
  - Functions to reflect structure
  - Name should identify purpose
  - Each function only for single logical action
    - Main function only for «start», «end» and «error handling» (not for calculations)
- Code layout
  - Fit on one screen (each function)
  - Easy to read (else it is error prone)
  - Test early and often

Recovering from errors

- Typing errors
- Catch exceptions
- Clean up
- Main function handles errors
- Add «clean mess» function
  - For example using «ignore»

Variables

- Adding variables (to improve calcualtor)
- Examples: e and pi as input
- Define variables with «name» and «value» pairs
- Old tradition: «let»
- Add function called «declaration»
- Consider error handling (var and val)
- Keep track of variables in «map» (table)

Variables

- Adding names
  - Insert «= » - token in code
  - Use «isalpha»- function (in lib)
- Predefined names
  - Common names: pi and e
  - Put definition in main function
- Are we there?
  - Need to test
  - Review comments
  - Improve calculator (implement assignment)

EXERCISES

- Vary, improve and test calculator code from Chapter 6
- Write a manual / help text for the calculator