Chapter 11: The Power of Variable Names

Considerations in Choosing Good Names

Shouldn't be cute or sound good

- The variable and its name should be the same thing

The Most Important Naming convention

- The name fully and accurately describe the entity the variable represents

Problem Orientation

- Speak to the problem, not the solution
- Express the WHAT, not the how

Optimum Name Length

- Names that are too short don't convey anything
- Looks like 2-3 words w/o connectors are about right

Computed- Value Qualifiers in Variable Names

- Total, sum, average, max, min, record, string, pointer → put at end of name
    - revenueTotal, expenseAverage
- Num can go in front tho, but num itself not great
    - Count, total, and index are better options

Common Opposites in Variable Names

- Begin/end
- First/last
- Locked/unlocked
- Min/max
- Next/previous
- Old/new
- Opened/closed
- Visible/invisible
- Source/target
- Source/destination
- Up/down

Naming Specific Types of Data

Naming Loop Indexes

- If variable is used outside of the loop, should get name other than “i, j, k”
- If loop is bigger than a few lines, easy to forget what the variable means

Naming Status Variables

- Think of a better name than “flag” for status variables
- Flags themselves should be thought of as status variables
    - 0x1, 0x80
    - True, CONTROL_CHARACTER
        - → How to do this
        - Const int CONTROL_CHARACTER = 0x80
        - statusFlag = CONTROL_CHARACTER
- When you find yourself “figuring out” a section of code, consider renaming variables

Naming Temporary Variables

- Because temporary, usually treated more casually which leads to errors
- Just name descriptively lmao

Naming Boolean Variables

- Done
    - Use done to indicate whether something is done
    - Set to false before, then set to true after
- Error
    - Set to false, then true after error occurs
- Found
    - Use to indicate whether a value has been found
    - Set to false, then true after value found
    - Use for searching array, a file, list, etc
- Success or OK
    - Use to indicate a successful operation
- Don’t use names like status or sourceFile
    - These are ambigious and don’t scream True or False
- Can also use “is”- prefix
    - isDone
    - isFound
    - Must note that the second word must be useful

Naming Constants

- CONSTANTS
- Basically name what the constant represents, not the number the constant refers to

The Power of Naming Conventions

Why Have Conventions?

- They let you take more for granted. By making one global decision rather than many local ones, can concentrate on more important characteristics of code
- They help transfer knowledge across projects
  - Similarities in names give an easier and more confident understanding of what unfamiliar variables are supposed to do
- They help you learn code more quickly on a project
  - Rather than learning anitas looks like this, julias like that and kristens like something else, can work with a more consistent set of code
- They reduce naming proliferation
  - Without conventions, can easily call the same thing by two different names
    - pointsTotal, totalPoints
  - This is mega confusing
- They compensate for language weakness
  - You can use conventions to emulate named constants and stuff
- They emphasize relationships among related items
- **The key is that any convention is better than no convention, and conventions can be arbitrary**

When You should Have a Naming Convention

- General rules…
- When multiple programmers are working on a project
- When you plan to turn a program over to another programmer for modifications and maintenance (which is nearly always)
- When your programs are reviewed by other programmers in your organization
- When your program is so large you cant hold the whole thing in your brain at once and need to think about it in pieces
- When the program will be long lived enough that you might put it aside for a few weeks or months before working on it again
- When you have lots of unusual terms that are common on a project and want to have standard terms or abbreviations in coding

Informal Naming Conventions

Guidelines for a Language Independent Convention

- Differentiate between variable names and routine names
  - variableName vs RoutineName()
- Differentiate between classes and objects
  - Leading capitalization
    - Widget widget
    - Cant be applied in every language b/c some are case sensitive

    - All Caps
        - WIDGET widget
        - All caps are used for global variables so subpar
    - "t_" prefix for Types
        - t_Widget Widget
        - Not aesthetically pleasing
    - "a" prefix for variables
        - Widget aWidget
        - Have to change every variable name to accommodate
    - More specific names for the variables
        - Widget employeeWidget
        - Requires more thought on a variable by variable basis
        - This book uses this style
- Identify global variables
    - "g_" prefix can do that
- Identify member variables
    - "m_" prefix can do that
- Identify type definitions
    - Naming conventions for types serve two purposes
        - Explicitly identify a name as a type name
        - Avoid name clashes with variables
    - ALL CAPS or "t_" prefix
- Identify named constants
    - Named constants need to be identified so you can tell whether youre assigning a variable a value from another variable (whos value might change), or from a named constant
    - "c_" prefix, or ALL CAPS
- Identify elements of enumerated types
    - "e_" or "E_" or "Color_" or "Planet_" prefix
- Identify input only parameters
- Format names to enhance readability
- **Try not to mix techniques, that makes code hard to read**
-

Standardized Prefixes

User-Defined Type Abbreviations

- Ch = character in document → chCursorPosition
- Doc = document → docActive
- Pa = paragrapth → firstPaActiveDocument
- Scr = screen → scrUserWorkspace
- Sel = selection
- Wn = window → wnMain

Creating Short Names That are Readable

General Abbreviation Guidelines

- Some contradict each other so don't try to use all at the same time
- Use standard abbreviations (listed in a dictionary)
- Remove all non leading values
  - Computer → cmptr
  - Screen → scrn
- Remove articles
  - And, or, the
- Use the first letter or first few letters of each word
- Truncate consistently after the first, second or third (whichever is appropriate) letter of each word
- Keep the first and last letters of each word
- Use every significant word in the name, up to a maximum of three words
- Remove useless suffixes
  - Ing, ed, etc
- Keep the most noticeable sound in each syllable
- Be sure not to change the meaning of the variable
- Iterate through these techniques until you abbreviate each variable name to between 8-20 characters

Phonetic Abbreviations

- Skating → sk8ing
- Don't use lmao

Comments on abbreviations

- Don't abbreviate by removing one character from a word
  - Not worth the hassle or remembering if you removed char or not when using var
- Abbreviate consistently
  - Always use Num or always use No
- Create names that you can pronounce
- Avoid combinations of words that might be misread
  - End of B
    - BEND
    - ENDB
    - → B_end
- Use thesaurus to resolve naming collisions
- Document extremely short names in translation tables
  - XPOS = x-coordinate position (in meters)
- **Names matter more to the reader of the code than the writer**

Kinds of Names to Avoid

- Avoid misleading names or abbreviations
- Avoid names with similar meanings
    - If you can switch the names of two variables without hurting the program, both need to be renamed
- Avoid variables with different meanings but similar names
    - clientRecs vs clientReps → look similar but very different values
- Avoid names that sound similar
    - Rap and wrap
- Avoid numbers in names
    - If the numbers really are important, use array instead of separate values
    - File1, file2 etc
- Avoid misspelled words in names
- Avoid words that are commonly misspelled in English
- Don’t differentiate variable names solely by capitalization
- Avoid the names of standard (built in) types, variables, and routines
- Don’t use names unrelated to what the variables represent