Communicating in Code: Naming

315 Programming Studio

What's the Purpose of Coding?

What's the Purpose of Coding?

- To give the computer instructions?

What's the Purpose of Coding?

- To give the computer instructions

What’s the Purpose of Coding?

- To give the computer instructions
- To demonstrate your skill?

What’s the Purpose of Coding?

- To give the computer instructions
- To demonstrate your skill

What’s the Purpose of Coding?

- To give the computer instructions
- To demonstrate your skill
- An effective way to express ideas of what you want the computer to do

What’s the Purpose of Coding?

- An effective way to express ideas of what you want the computer to do
- Communication!
  - To the computer
  - To yourself (later on)
  - To others

What about Documentation?

- External documentation is very useful, but has its own problems
  - Can be out of date/inconsistent with program
  - Maintained separately (multiple files)
  - Often for a different audience
    - developer vs. user
- Clearly written code can be more important than well-written documentation of that code

Communicating in Code

- Choosing good names
- Including appropriate comments
- Following good layout and style

- These are all critical to documentation, and with good naming, commenting, and layout, other documentation may be unnecessary!

Names

- We assign names throughout a program
- Give identity
- Imply behavior/purpose
- Provide recognition

What gets named?

- Variables
- Functions
- Types/classes
- Namespaces
- Macros
- Source Files

Choosing Names

- Sometimes there are naming conventions
  - If you work at a company that has an agreed convention, follow it!
- But, there are several “wise” ideas to consider when choosing names.

Naming Considerations

Be sure it’s not a reserved name (Duh!)

Sometimes it’s easy to forget…

Make it informative

Keep it concise

Make it memorable

Make it pronounceable

Informative Names

- The amount of information a name needs depends on its scope – understand it when seen
- Use descriptive names for globals, short names for locals
- Large routines/loops need more descriptive names

```
s = 0;
for (WhichGroup=0; WhichGroup<num; WhichGroup+
   +) {
   s += G[WhichGroup].n();
}
```

Informative Names

- The amount of information a name needs depends on its scope – understand it when seen
- Use descriptive names for globals, short names for locals
- Large routines/loops need more descriptive names

```
nAnimals = 0;
for (i=0; i<NumAnimalGroups; i++) {
   nAnimals += AnimalGroup[i].NumberInGroup();
}
```

Descriptive Names

- Names should convey what it represents or does, unless obvious from context
- Describe everything a routine does
  - Print() vs. PrintAndCloseFile()
- Avoid meaningless or vague names
  - HandleData(), PerformAction(), etc.

Descriptive Names

- Procedures: Active names
  - Verb followed by noun
  - AnotherStudent(s) vs. AddStudent(s)
- Functions different: give return value
  - GetNumStudents() vs. numStudents()
- Booleans: Be clear what is returned
  - checkEOF vs. isEOF

Consistent Names

- Key: Be Consistent!
  - nKids, numKids, num_kids, NumKids, nkids, Number_Kids, numberofkids
  - Write1stObject(), WriteSecondObject(), write_third_object()
  - averageSalary vs. salaryMinimum
- Use related names for related operations
  - OpenFile(): CloseFile() vs. fclose()
  - open/close, first/last, old/new, min/max, etc.

Name Length

- Tradeoff between description and visual space
- Moderate-length names tend to be best
  - 8-20 characters
- If a glance at the code seems like it has lots of short or lots of long names, use caution!
- Scope plays a role
- Rarely-used functions might be longer

Other Random Naming Considerations

- Beware of “temp” variables
- Be careful of reusing variable names
- Be careful of overloading names
- Avoid intentional misspellings
- Consider pronunciation

Conventions

- Lots of conventions out there
- Conventions help convey information away from its definition
- Very useful for larger groups/programs
- Examples:
  - Globals have initial capital letters
  - Constants are in ALL CAPS
  - Etc.

Common Naming Conventions

- Beginning/ending with a p if a pointer
- Starting with n for a number
- i, j are integer indices
- s is a string, c or ch are characters

Example: Hungarian Naming Convention

- Base types:
  - wn Window
  - scr Screen Region
  - fon Font
  - ch Character
  - pa Paragraph
- Eg: wnMain, scrUserWorkspace

Example: Hungarian Naming Convention

- Prefixes

a	array
c	count
d	difference between two variables
e	element of array
g	global variable
h	handle
i	index into array

- e.g. iwnUserView = index into array of windows giving user views