Code Style & Documentation

Overview

- The style of your code and the program documentation both have a significant affect on the readability, maintainability and testability of your programs
- Today we will look at:
  - appropriate code style for C++ (and Java)
  - program documentation guidelines for C++ (and Java)

Program Style

Indentation & White Space

- Can be used to help group lines of code
- Some general rules include:
  1. Insert a blank line between **distinct sections** of statements
  2. **Indent** statements that are within another statement (e.g., inside an if statement or a while statement)
  3. Place closing braces on a distinct line. Place opening braces on a distinct line or at the end of the preceding line.

Program Style

Line Length

- In C++ it is usually advised to **avoid** lines that are longer than **80 characters**.
- **Why?**
  - Improved readability in terminals
  - Typically avoids bad wrapping by **printers**
  - …

What happens if our line is more than 80 characters? Need to wrap text!

Program Style

Line Length

- The following Java wrapping guidelines also work for C++:
  - *“Break after a comma.*
  - *Align the new line with the beginning of the expression at the same level on the previous line.*
  - *If the above rules lead to confusing code or to code that's squished up against the right margin, just indent 8 spaces instead.”* [1]
  - *…*

[1]Source of guidelines is http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

Program Style

Comments

- Comments are used in a program to increase the **readability** of the code
- Reasons for using comments include:
  - Explanations of statements
  - Description of usage of statements
  - Etc.
- (We will learn more about comments when we talk about documentation)

Program Style

Comments

– In C++ comments can occur in two ways:

```
// everything from start symbol to the end of line
// is a comment
```

or

```
/* Everything between the start and end symbols
is a comment. */
```

Program Style

Naming Conventions

- In C++ there are general **naming conventions** for variables, constants and functions
  - also for namespaces, etc.
- A **company** or a project may also have specific naming conventions

Program Style

C++ Naming Conventions

Identifier Type	Rules for Naming	Examples
File Names	*"…should be all lowercase and can include underscores (_) or dashes (-)"*	a_file.cpp
Type Names	*"…start with a capital letter and have a capital letter for each new word, with no underscores…"* Types should be nouns.	MyClass
Variable Names	*"…all lowercase, with underscores between words. Class member variables have trailing underscores."* Variables should be nouns.	local_variable member_variable_
Constant Names	*"Use a k followed by mixed case"*	kMyConstant
Function Names	*"Regular functions have mixed case"*	MyFunction();

Program Style

Java Naming Conventions

Identifier Type	Rules for Naming	Examples
Packages	*"The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries … Subsequent components of the package name vary…"*	com.blah
Classes, Interfaces	*"…should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive."*	Class MyClass; Interface MyInterface;

Methods	*"Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized."*	myMethod();
		Method();
		myLongMethod();

Program Style

Java Naming Conventions

Identifier Type	Rules for Naming	Examples
Variables	*"Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters…Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters."*	int I; int size; int mySize;
Constants	*"…should be all uppercase with words separated by underscores ("_")"*	static final int A_CNST=1;

Program Style

File Structure

- Most programs will have many source files
- The organization of these files is also part of the program style
- In Java, the use of packages defines directory structure.
- In C++, the programmer has to decide on the appropriate directory structure (usually based on the conventions of the company or project)

Documentation

- Comments are a form of in-program documentation
- When to comment and when *not* to comment comes with experience – but there are some **general rules**!

Documentation

- **Rule 1:** Comment at the beginning of the source file with the main method/function. These comments should include:
  - the purpose of the program,
  - how it should be executed,
  - the expected input/output,
  - the version number
  - Any additional information that would be appropriate

Documentation

- **Rule 2:** At the beginning of all files/classes provide comment about
  - The copyright notice
  - the author,
  - revision history and
  - a description of the class.
  - **Rule 2A:** If functions or methods within a class have dependencies regarding their usage you should document these as well.

Documentation

- **Rule 3 (C++):** For the function declaration include comments regarding the **usage of the function**. For the function definition include comments such as an explanation of **how the function works**, any parameters and return values.
- ***Rule 3 (Java):*** *For each method include comments detailing the purpose of the method, how it works, any parameters and return values.*

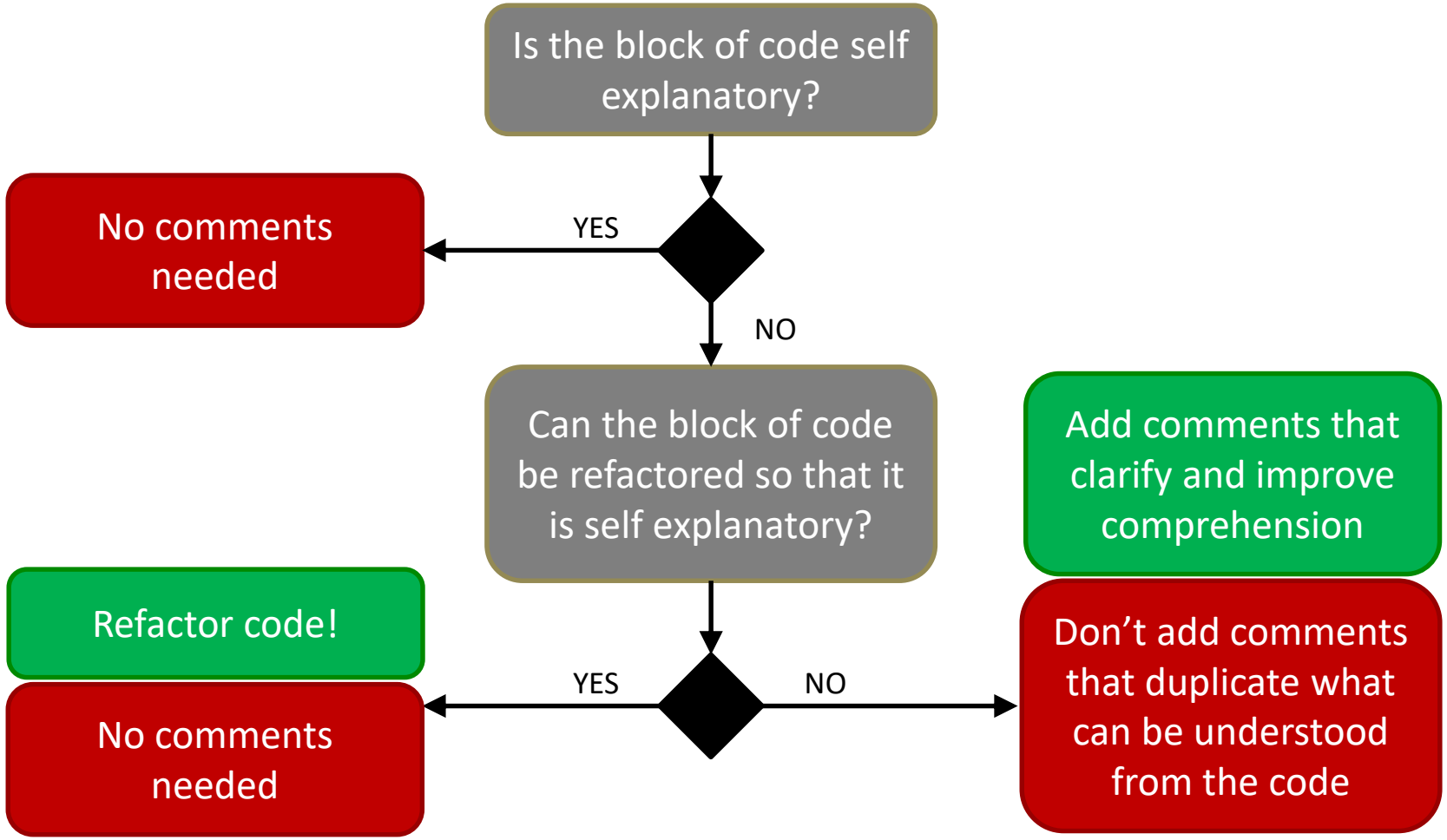
Documentation

- In-program documentation contained in the comments can also be used to generate other forms of documentation

Example: API documentation

- Often documentation generation tools are used
- **Why?**
  - Rewriting the document could **lead** to inconsistency between external and in-program documents
  - Rewriting documents takes more time and resources

Should I add a /*comment*/ to my block of code?



Code Refactoring

What is Refactoring?

- Refactoring is *"improving the design of existing code"* without affecting its external behavior
  - that is, it is simpler and better after refactoring, but it does exactly the same thing as before
- Uses a small number of *"rules"* characterizing better designs, and a catalog of code *"refactorings"* – patterns of change for transforming code from one design to another
- **A Catalog of Refactorings** – https://refactoring.com/

A Refactoring Example

Extract Method Refactoring

Before

```
void f () {
          ...
          //Compute score
          score = a + b + c;
          score -= discount;
          ...
}
```

After

```
void f () {
          ...
          computeScore ();
          ...
}

void computeScore () {
          //Compute score
          score = a + b + c;
          score -= discount;
          ...
}
```

Code Style & Documentation

Summary

- We've discussed guidelines for appropriate code style and documentation
- We've also introduced the concept of code refactoring and how it can improve the readability of source code

Code Style & Documentation

References

- Mozilla Developer Network Coding Style https://developer.mozilla.org/En/Mozilla_Coding_Style_Guide
- Google C++ Style Guide https://google.github.io/styleguide/cppguide.html
- Code Conventions for the Java Programming Language http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

Feedback

Please let me know about your thoughts: the course, the instructor (me), TAs, etc.

http://freesuggestionbox.com/pub/inqcjsz