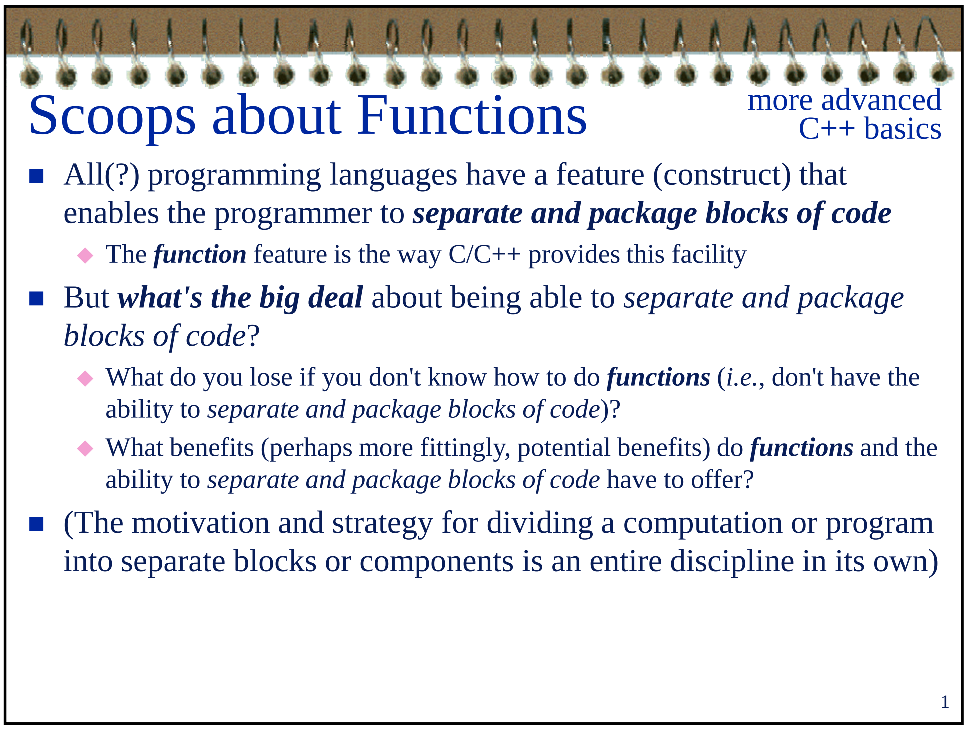
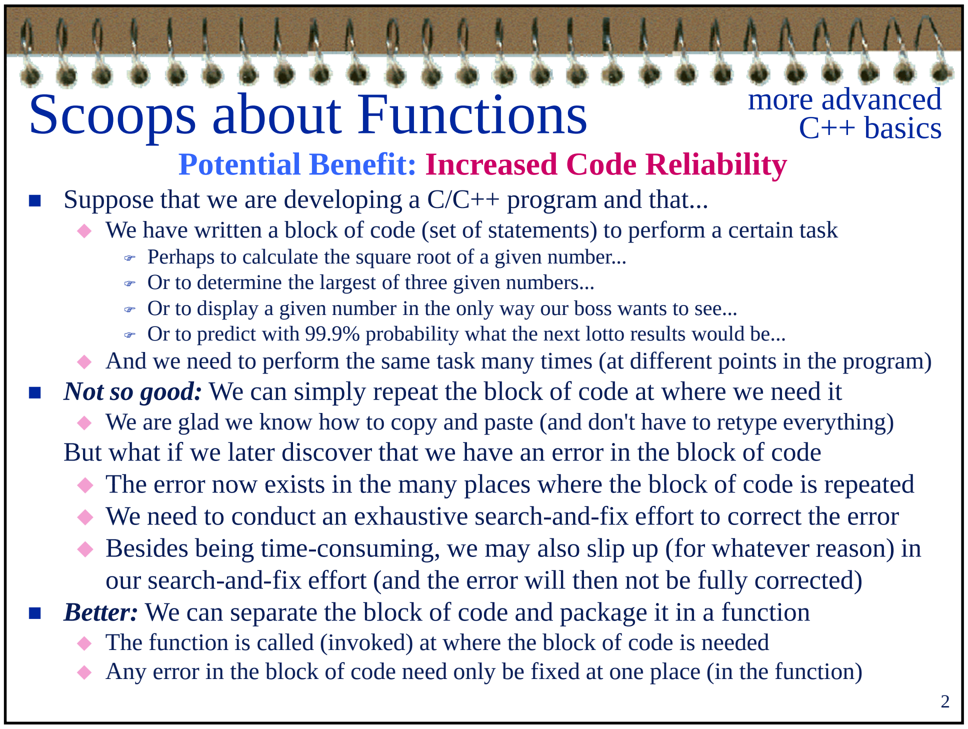


Scoops about Functions

more advanced
C++ basics

- All(?) programming languages have a feature (construct) that enables the programmer to ***separate and package blocks of code***
  - The ***function*** feature is the way C/C++ provides this facility
- But ***what's the big deal*** about being able to *separate and package blocks of code*?
  - What do you lose if you don't know how to do ***functions*** (*i.e.*, don't have the ability to *separate and package blocks of code*)?
  - What benefits (perhaps more fittingly, potential benefits) do ***functions*** and the ability to *separate and package blocks of code* have to offer?
- (The motivation and strategy for dividing a computation or program into separate blocks or components is an entire discipline in its own)



Scoops about Functions

more advanced
C++ basics

Potential Benefit: Increased Code Reliability

- Suppose that we are developing a C/C++ program and that...
  - We have written a block of code (set of statements) to perform a certain task
    - Perhaps to calculate the square root of a given number...
    - Or to determine the largest of three given numbers...
    - Or to display a given number in the only way our boss wants to see...
    - Or to predict with 99.9% probability what the next lotto results would be...
  - And we need to perform the same task many times (at different points in the program)
- ***Not so good:*** We can simply repeat the block of code at where we need it
  - We are glad we know how to copy and paste (and don't have to retype everything)

  But what if we later discover that we have an error in the block of code
  - The error now exists in the many places where the block of code is repeated
  - We need to conduct an exhaustive search-and-fix effort to correct the error
  - Besides being time-consuming, we may also slip up (for whatever reason) in our search-and-fix effort (and the error will then not be fully corrected)
- ***Better:*** We can separate the block of code and package it in a function
  - The function is called (invoked) at where the block of code is needed
  - Any error in the block of code need only be fixed at one place (in the function)

Scoops about Functions

more advanced
C++ basics

Potential Benefit: Increased Code Reusability

- Suppose that we are developing a C/C++ program and that...
  - We have written a block of code (set of statements) to perform a certain task
  - And the task is a common one ➔ we need to perform the task also in other programs
- ***Not so good:*** We can simply copy the block of code into the other programs
  - We are glad that in addition to copy and paste, we also know how to do find or search

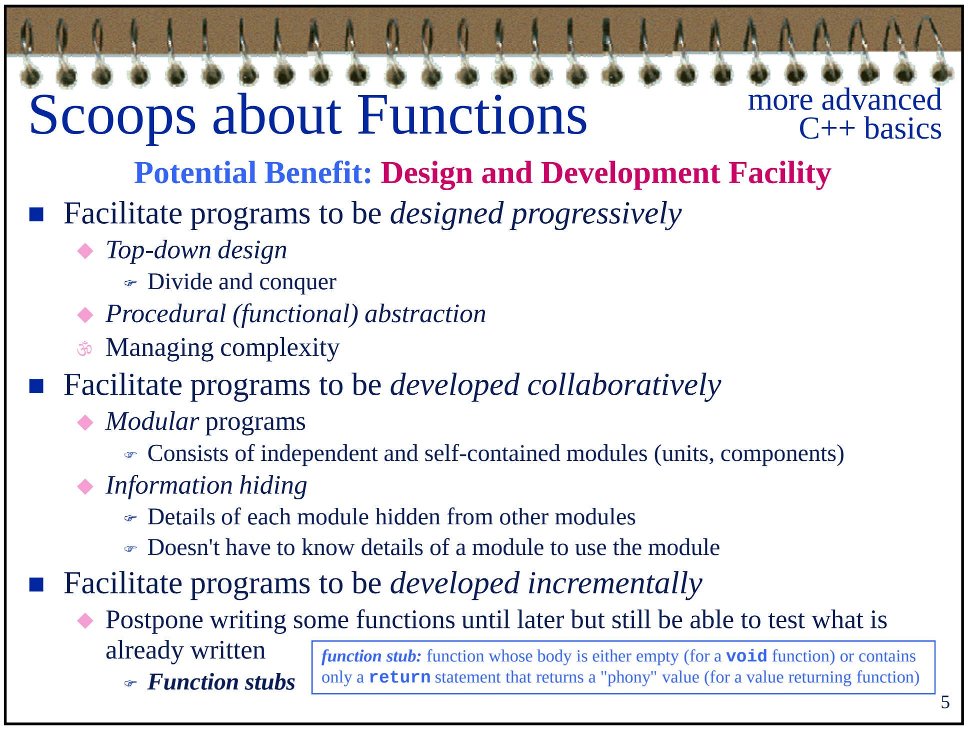
  But attempting to reuse code by simply copying a block of code that exists in the middle of another program is usually a long-shot endeavor at best
  - A block of code in the middle of a program is very likely *not self-contained and independent enough* to be easily reused in other programs
    - Most likely, the underlying algorithm is specialized to the variables and conditions specific to the originating program
- ***Better:*** We can separate the block of code and package it in a function
  - The block of code is copied as a function ➔ a more "wholesome" unit to deal with
    - It is also more hopeful to be reusable because it has been "separated and packaged"
  - ☯ Code reusability can be further increased by combining functions into ***libraries***
  - ☯ *Caveat:* Reusability (or at least the fuller extent of it) ***does not automatically result*** by simply collating blocks of code into functions ➔ the functions must be designed to be as *self-contained* and *independent* as possible for maximum reusability. In other words, functions ***don't bestow*** but only ***enable us to build*** reusability in our code.

Scoops about Functions

more advanced
C++ basics

Potential Benefit: Increased Code Updatability

- Suppose that we are developing a C/C++ program and that...
  - We have written a block of code (set of statements) to perform a certain task
  - And we are happy with it until someone told us about a much better algorithm
- ***Not so good:*** Because we simply embed the block of code in the program, we have no choice but to "mess" with the entire program in order to incorporate and take advantage of the better algorithm for performing the task
  - If we are lucky and careful, we may be able to update our program with not too much difficulty and without causing undue negative side effects ➔ we will have to thoroughly re-test the entire program again to ensure that it is in fact so
  - But we may not be so lucky... needless to say, all kinds of nightmares befall us
- ***Better:*** Because we separate the block of code and package it in a function, we may only have to update the affected function and not the rest of the program ➔ we may have to thoroughly re-test only the affected function
  - A well-designed function is self-contained and independent so that changes to how it works will not affect the rest of the program (ideally)
  - ☯ *Caveat:* Like reusability, updatability ***does not automatically result*** by simply collating blocks of code into functions ➔ the functions must be designed to be as *self-contained* and *independent* as possible for maximum updatability. In other words, functions ***don't bestow*** but only ***enable us to build*** updatability in our code.



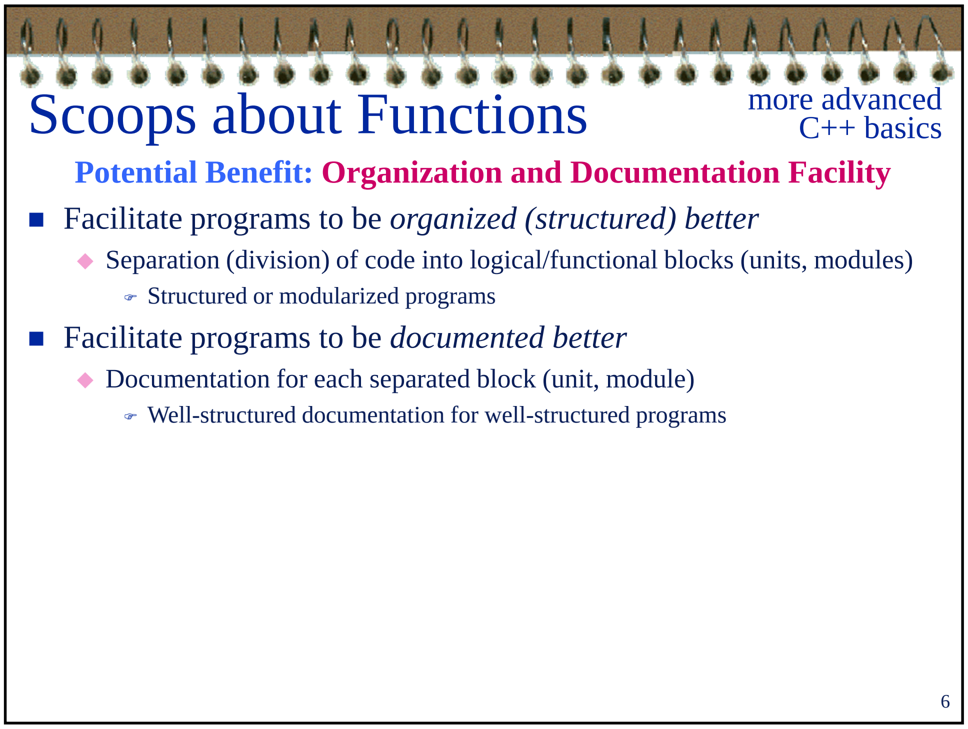
Scoops about Functions

more advanced C++ basics

Potential Benefit: Design and Development Facility

- Facilitate programs to be *designed progressively*
  - *Top-down design*
    - Divide and conquer
  - *Procedural (functional) abstraction*
  - Managing complexity
- Facilitate programs to be *developed collaboratively*
  - *Modular* programs
    - Consists of independent and self-contained modules (units, components)
  - *Information hiding*
    - Details of each module hidden from other modules
    - Doesn't have to know details of a module to use the module
- Facilitate programs to be *developed incrementally*
  - Postpone writing some functions until later but still be able to test what is already written
    - ***Function stubs***

function stub: function whose body is either empty (for a `void` function) or contains only a `return` statement that returns a "phony" value (for a value returning function)



Scoops about Functions

more advanced C++ basics

Potential Benefit: Organization and Documentation Facility

- Facilitate programs to be *organized (structured) better*
  - Separation (division) of code into logical/functional blocks (units, modules)
    - Structured or modularized programs
- Facilitate programs to be *documented better*
  - Documentation for each separated block (unit, module)
    - Well-structured documentation for well-structured programs