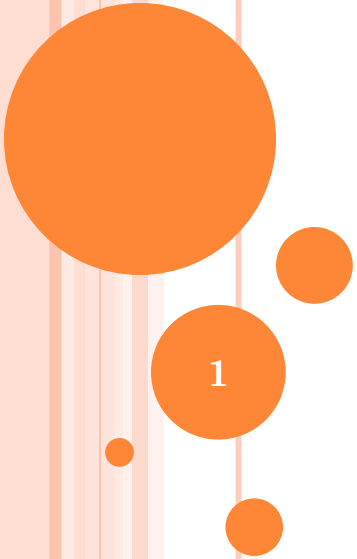

Parameters

CS A150 - C++ Programming 1

Parameter Passing

- Parameters can be passed to a function as
  - **Call-by-value**
    - "copy" of value is passed
  - **Call-by-reference**
    - "address of" actual parameter is passed
  - **Mixed** parameter list

Call-by-Value Parameters

- ***Copy*** of actual argument passed
- Considered "local variable" inside function
- If modified, only "local copy" changes
  - Function has **no** access to "actual argument" from caller
- This is the ***default*** method
  - Used in all examples so far

Book Example

- Example 1: Formal Parameter Used as a Local Variable

Call-by-Value Pitfall

- Common Mistake:
  - Declaring parameter "again" inside function:

```
void totalInches(int feet, int inches)
{
   int feet;    //NO!
   int inches;  //NO!

   inches = 12 * feet + inches;
   cout << inches << endl;
}
```

  - Compiler error results in a "*Redefinition error…*"
- ***Value arguments*** **ARE** like "local variables"
  - But function gets them "automatically"

Call-by-Reference Parameters

- Used to provide access to caller's actual argument
- Caller's data can be modified by called function
- Typically used for ***input*** function
  - To retrieve data for caller
  - Data is then "given" to caller
- Specified by ampersand **&**,

  after type in formal parameter list

Call-by-Reference Details

- What is really passed in?
- A **reference** back to caller's actual argument!
  - Refers to ***memory location*** of actual argument
  - Called "**address**", which is a unique number referring to distinct place in memory

Book Example

- Example 2: Call-by-Reference Parameter

When To Pass By Reference

- When you need to ***change the original value*** in the calling function
- When your function is asking the user to enter **more than one value** to store
  - **Remember:** You **cannot** return more than one value!
- When you are **passing large items**
  - Class objects
  - Vectors
  - (we will see them later)

Mixed Parameter Lists

- Can combine passing mechanisms
- Parameter lists can include pass-by-value and pass-by-reference parameters
- ***Order of arguments*** in list is critical:

```
void mixedCall(int&, int, double&);
```

  - Function call:

```
mixedCall(arg1, arg2, arg3);
```

    - **arg1** must be integer type, is passed by **reference**
    - **arg2** must be integer type, is passed by **value**
    - **arg3** must be double type, is passed by **reference**

Parameters and Arguments

- Often used interchangeably
- True meanings:
  - **Formal parameters**
    - In function declaration and function definition
  - **Arguments**
    - In function call (argument list)
    - Used to "fill in" a formal parameter

Choosing Formal Parameter Names

- Same rule as naming any identifier:
  - Choose a *meaningful name*
- Functions are "self-contained modules"
  - Designed separately from rest of program
  - Assigned to teams of programmers
  - All must "understand" proper function use
  - OK if formal parameter names are ***same*** as argument names
- Choose function names with same rules

Book Examples

- Example 3: Buying Pizza

Questions?

(Parameters)