

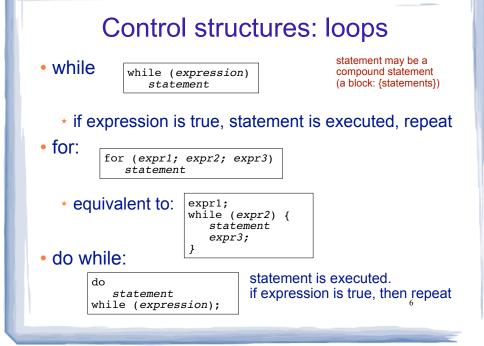
More assignment statements

Compound assignment

operator	usage	equivalent syntax:
+=	x += e;	x = x + e;
-=	x -= e;	x = x - e;
*=	x *= e;	x = x * e;
/=	x /= e;	x = x / e;

increment, decrement

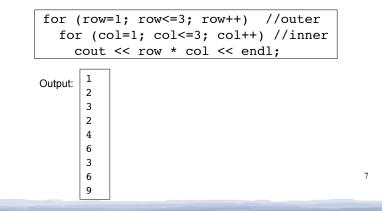
operator	usage	equivalent syntax:
++	x++; ++x;	x = x + 1;
	x;x;	x = x - 1;



Nested loops

5

- When one loop appears in the body of another
- For every iteration of the outer loop, we do all the iterations of the inner loop



continue and break Statements

- Use break to terminate execution of a loop
- When used in a nested loop, terminates the inner loop only.
- Use continue to go to end of **current** loop and prepare for next repetition
- while, do-while loops: go to test, repeat loop if test passes
- for loop: perform update step, then test, then repeat loop if test passes

Function Definitions Function definition pattern:

datatype identifier (parameter1, parameter2, ...) {
 statements . . .
}

Where a parameter is:

datatype identifier

- * datatype: the type of data returned by the function.
- * *identifier*: the name by which it is possible to call the function.
- * *parameters*: Like a regular variable declaration, act within the function as a regular local variable. Allow passing arguments to the function when it is called.
- * statements: the function's body, executed when called.

Function Call, Return Statement

• Function call expression

identifier (expression1, . . .)

- * Causes control flow to enter body of function named identifier.
- parameter1 is initialized to the value of expression1, and so on for each parameter
- * expression1 is called an argument.
- Return statement: return expression;
 - * inside a function, causes function to stop, return control to caller.
- The value of the return *expression* becomes the value of the function call

Example: Function // function example #include <iostream> using namespace std; int addition (int a, int b) { int result: result=a+b; return result; int main () { int z; z = addition (5,3);cout << "The result is " << z <<endl; int main () { What are the parameters? arguments? • What is the value of: addition (5,3)? 11 What is the output?

Void function

• A function that returns no value:

```
void printAddition (int a, int b) {
    int result;
    result=a+b;
    cout << "the answer is: " << result << endl;
}</pre>
```

- use void as the return type.
- the function call is now a statement (it does not have a value)

nt main () { printAddition (5,3);

Prototypes

- In a program, function definitions must occur before any calls to that function
- To override this requirement, place a prototype of the function before the call.
- The pattern for a prototype:

datatype identifier (type1, type2, ...);

* the function header without the body (parameter names are optional).

13

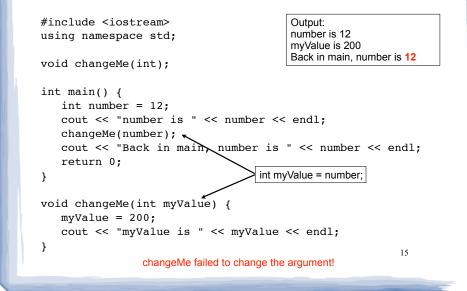
Arguments passed by value

- <u>Pass by value</u>: when an argument is passed to a function, its value is *copied* into the parameter.
- It is implemented using variable initialization (behind the scenes):

int param = argument;

- Changes to the parameter in the function body do **not** affect the value of the argument in the call
- The parameter and the argument are stored in separate variables; separate locations in memory.

Example: Pass by Value



Scope of variables

- For a given variable definition, in which part of the program can it be accessed?
 - * **Global variable** (defined outside of all functions): can be accessed anywhere, after its definition.
 - Local variable (defined inside of a function): can be accessed inside the block in which it is defined, after its definition.
 - * **Parameter**: can be accessed anywhere inside of its function body.
- Variables are destroyed at the end of their scope.

