

Why C?
 Low-level Direct access to memory WYSIWYG (more or less) Effectively no runtime system No garbage collector No other threads No "read" or "write barriers" Efficient Space & time C: effectively portable assembly code
UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science 2

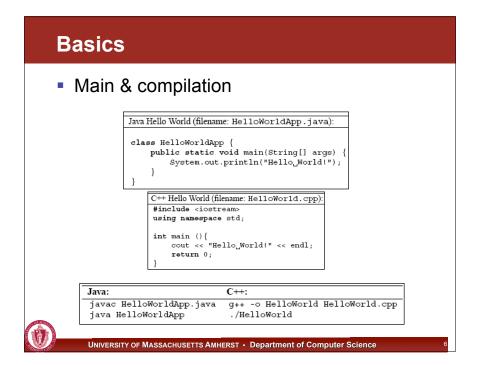
OK, Why C++?

- C++: extends C
 - Upwardly-compatible
- Adds significant software engineering benefits

UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science

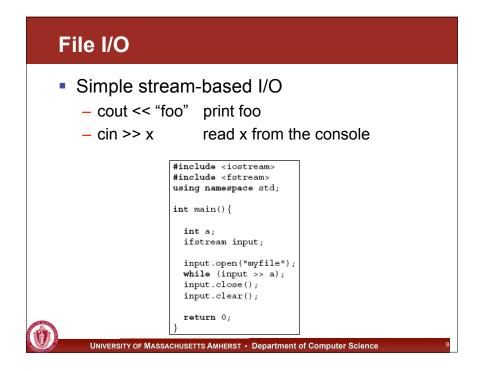
- Classes
- Encapsulation (private)
- Templates ("generics")
- Other modularity advantages
- Inlining instead of macros

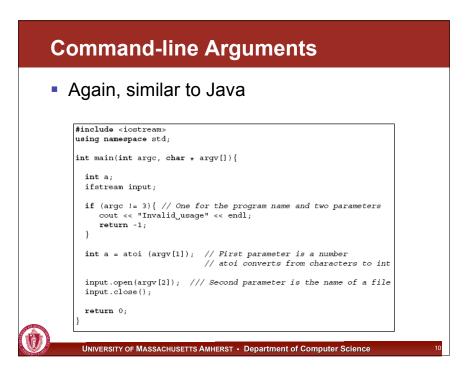
<section-header><section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item> <section-header>

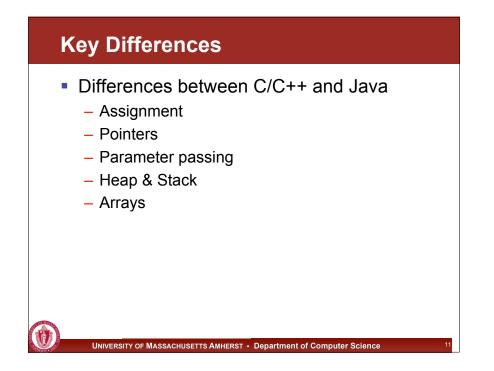


Intrinsic Types			
Ess	entially identical		
			1
	Java:	C++:	
	byte myByte;	char myByte;	
	short myShort;	<pre>short myShort;</pre>	
	<pre>int myInteger;</pre>	<pre>int myInteger;</pre>	
	long myLong;	long myLong;	
	<pre>float myFloat;</pre>	<pre>float myFloat;</pre>	
	double myDouble;	double myDouble;	
	char myChar;	char myChar;	
	boolean myBoolean;	bool myBoolean;	
]
A CONTRACTOR OF			
			7
UNIN	VERSITY OF MASSACHUSETTS AMHERST	Department of Computer Scier	ice /

 Mostly the same C/C++: nonzero int san 	ne as true
ava:	C++:
boolean temp = true; boolean temp2 = false;	<pre>bool temp = true; int i = 1;</pre>
if (temp) System.out.println("Hello_World!");	<pre>if (temp) cout << "Hello_World!" << endl;</pre>
<pre>if (temp == true) System.out.println("Hello_World!");</pre>	<pre>if (temp == true) cout << "Hello_World!" << endl;</pre>
<pre>if (temp = true) // Assigns temp to be true System.out.println("Hello_World!");</pre>	<pre>if (i) cout << "Hello_World!" << endl;</pre>







Assignment
 Java assignment: makes reference C++ assignment: makes copy
<pre>SomeClass x, y; SomeClass *a; x=y; // This copies object y to x. Modifying x does NOT modify y. a=&x // This copies a reference to x.</pre>
UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science 12

Pointers & Friends

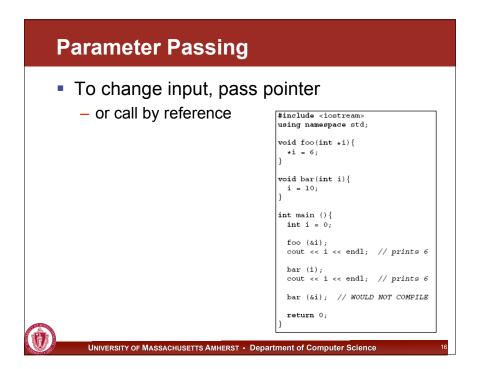
 "Pointers are like jumps, leading wildly from one part of the data structure to another. Their introduction into high-level languages has been a step backwards from which we may never recover."

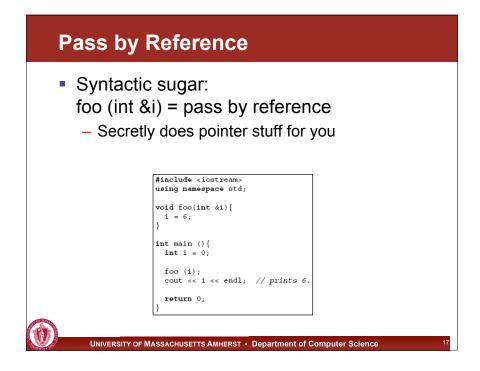
UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science

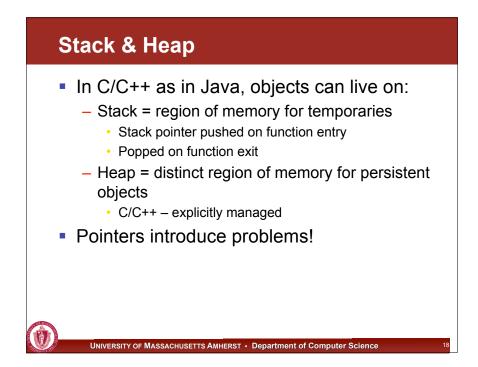
- C.A.R. Hoare

<text><code-block><code-block><code-block><code-block><code-block></code></code></code></code></code>

 C/C+- 	+ – all parameters copied by default	
	<pre>#include <iostream> using namespace std;</iostream></pre>	
	<pre>void foo(int i){ cout << i << endl; // Prints 1 }</pre>	
	<pre>int main () { foo (1);</pre>	
	return 0; }	



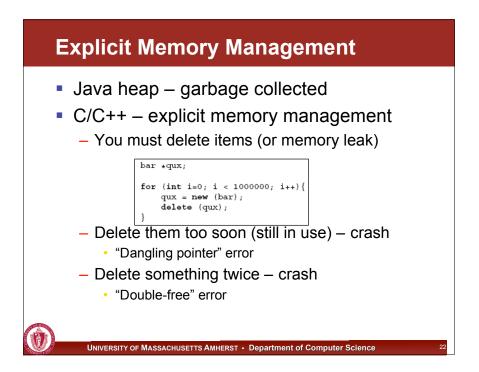


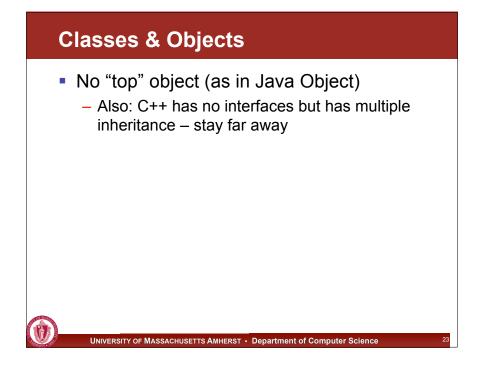


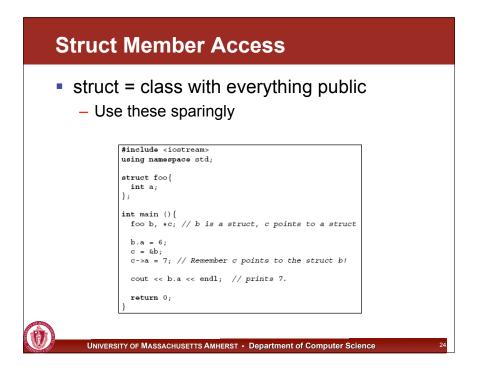
• Stack data: new every time int foo (int a) { int b = 10; return 0; }	The Stack
int b = 10;	Stack data: new every time
	int $b = 10;$

Big Stack Mistake		
Never re	eturn pointers to the stack!	
	<pre>#include <iostream> using namespace std;</iostream></pre>	
	// BAD BAD BAD	
	<pre>int* foo () { int b = 10;</pre>	
	return &b }	
	<pre>int main() { int *a;</pre>	
	<pre>a = foo(); cout << *a << endl; // Print out 10?</pre>	
	return 0;	
	}	
	MASSACHUSETTS AMHERST • Department of Computer Science	20

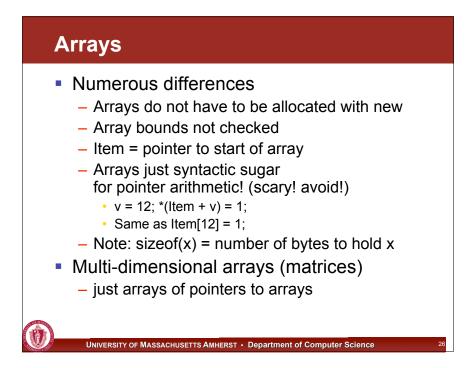
The Heap	
 Allocate persistent data on heap with new 	
<pre>#include <iostream> using namespace std; // GOOD GOOD GOOD int* foo () { int *b; b = new (int); *b = 10; return b; } int main() { int *a; a = foo(); cout << *a << endl; // Print out 10! return 0; } }</iostream></pre>	
UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science 21	







Class Declaration	
Pretty s	similar
	<pre>class IntCell { public: IntCell(int initialValue = 0) { storedValue = initialValue; } int getValue() { return storedValue; } int setValue(int val) { storedValue = val; } private: int storedValue; };</pre>
University	OF MASSACHUSETTS AMHERST • Department of Computer Science 25



Other Features

- Operator overloading
 - New meanings to existing operators
 - int operator+(MyType& a, MyType& b);
 - Controversial, but useful for things like complex math, matrix operations
 - int& operator()(int x, int y);
- Templates
 - A.k.a. generics in Java
 - template <class X> void foo (X arg);

UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science

Standard Template Library(STL) Implements useful data structures #include <iostream> #include <queue> using namespace std; queue<int> myQueue; int main(int argc, char * argv[]){ myQueue.push(10); myQueue.push(11); cout << myQueue.front() << endl; // 10</pre> myQueue.pop(); cout << myQueue.front() << endl; // 11</pre> myQueue.pop(); cout << myQueue.size() << endl; // Zero</pre> } UNIVERSITY OF MASSACHUSETTS AMHERST • Department of Computer Science

