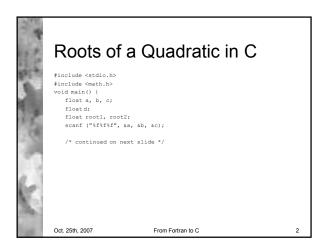
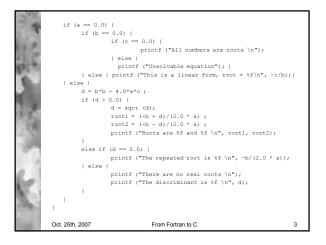
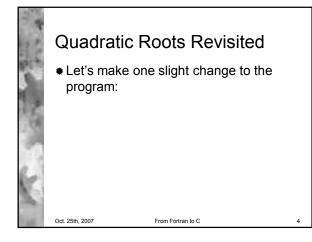
## Computers in Engineering COMP 208

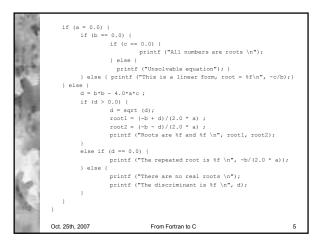
Moving From Fortran to C – Part 2 Michael A. Hawker

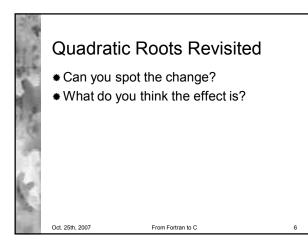


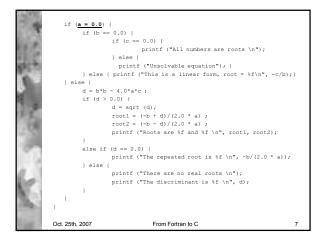














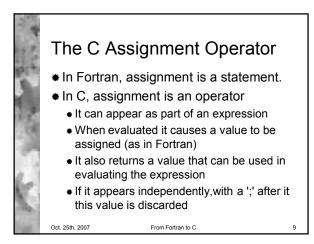
### What Happens Here?

- The equivalent statement in Fortran would cause a syntax error
- The expression (a = 0.0) is not of type logical

From Fortran to C

- \* But C does not have a type "logical"
- \* What happens in C?

Oct. 25th, 2007



### The C Assignment Operator

Oct. 25th, 2007

Syntax: variable = expression

#### Semantics

- 1. Evaluate the expression
- 2. Store the value in the expression in the variable

From Fortran to C

10

3. Return the value to the expression

Back to our example In C, (a=0.0) would assign 0 to a and then return 0 as the value of the expression The if condition, with value 0, would be taken as equivalent to "false" The else clause would be evaluated and when the root was calculated, there would be an attempt to divide by 0 Oct. 25th, 2007 From Fortran to C 11

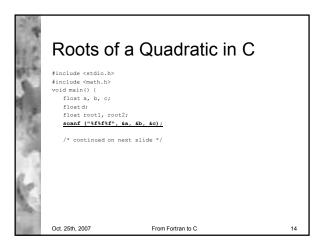
- Make	
C 164	if $(a = 0.0)$ {
10 C C	if (b == 0.0) {
and the second	if (c == 0.0) {
1000	printf ("All numbers are roots \n");
* S. S.	} else {
1207	printf ("Unsolvable equation"); }
	<pre>} else { printf ("This is a linear form, root = %f\n", -c/b);}</pre>
1000	} else {
1	d = b*b - 4.0*a*c;
100	if (d > 0.0) {
2.00	d = sqrt(d);
A Barris	root1 = (-b + d) / (2.0 * a);
1.000	root2 = (-b - d)/(2.0 * a);
m	printf ("Roots are %f and %f \n", root1, root2);
1000	}
	else if (d == 0.0) {
10.00	printf ("The repeated root is $f n", -b/(2.0 * a)$ );
A 18	} else {
Sec. 1	printf ("There are no real roots \n");
100	printf ("The discriminant is %f \n", d);
Sec. 1	)
1000	}
1. The second	
N. Law T	Oct. 25th, 2007 From Fortran to C 12
100	00.200, 2007 From on all 10 C 12

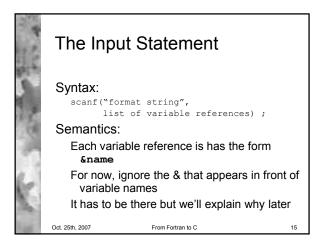


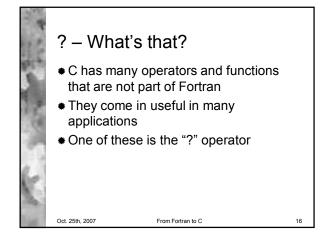
## Inputting Values in C • In the program for computing the roots of a quadratic, we began by reading the values for the coefficients a, b and c

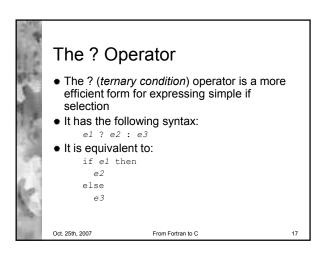
From Fortran to C

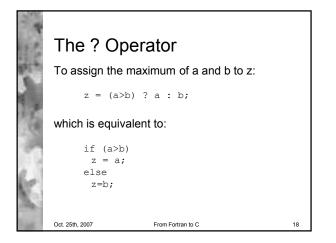
Oct. 25th, 2007





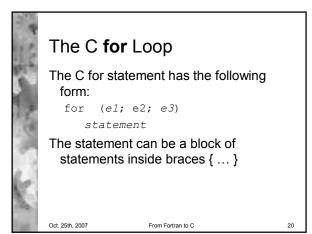




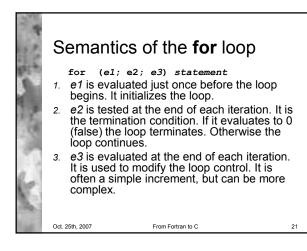


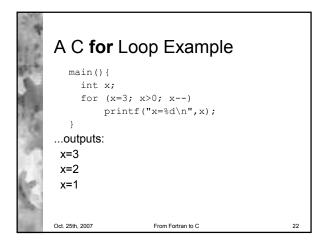
# Loops in C The basic looping construct in Fortran is the DO loop In C, there is a more complex looping command called the for loop

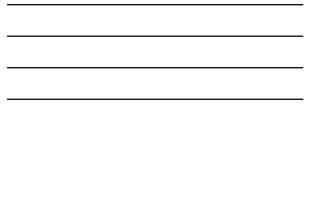
Oct. 25th, 2007

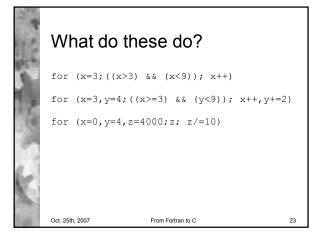


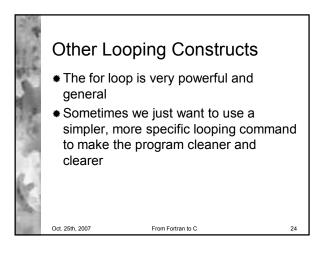
From Fortran to C

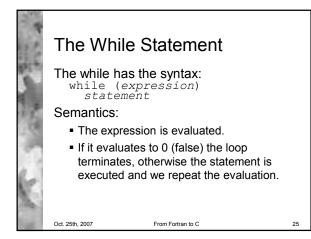


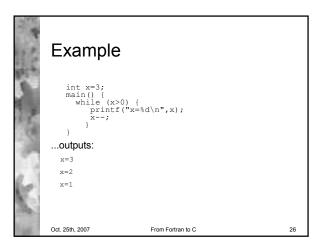


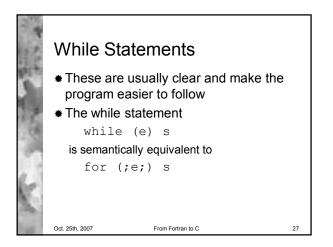


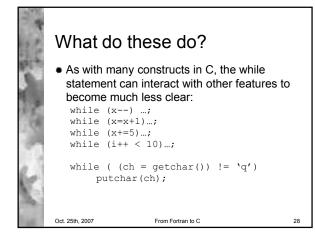


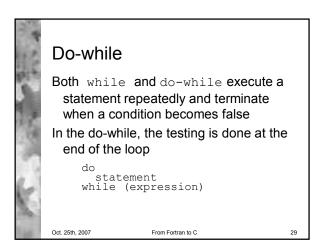












### Break and Continue

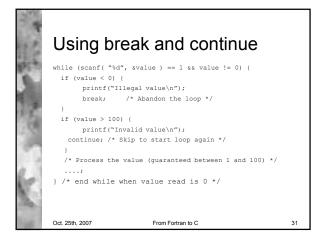
The break statement is similar to the Fortran EXIT and allows us to exit from a loop

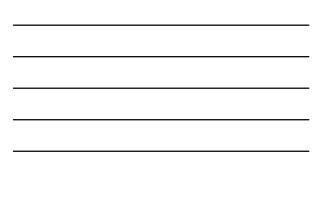
break; --exit from loop

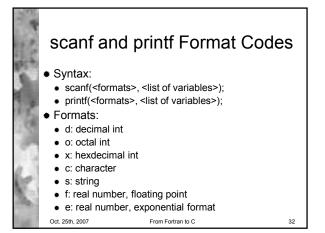
The continue statement forces control back to the top of the loop

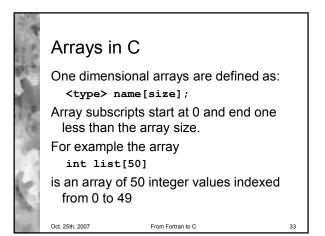
continue; --skip 1 iteration of loop

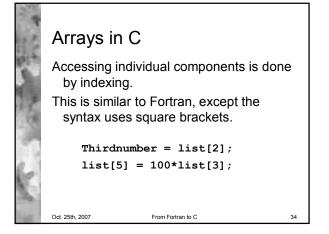
Oct. 25th, 2007

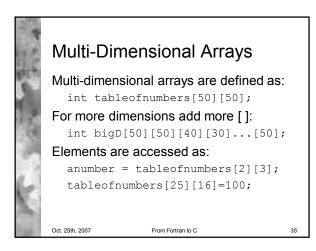


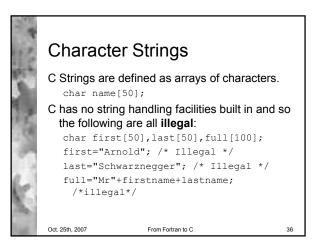


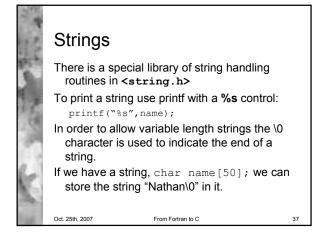


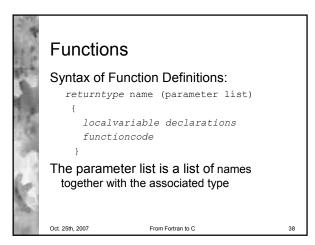


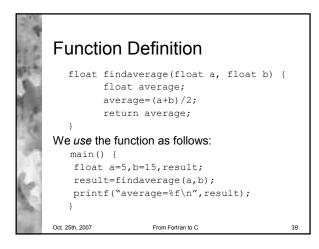


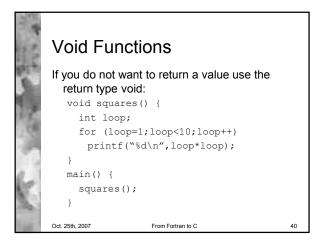


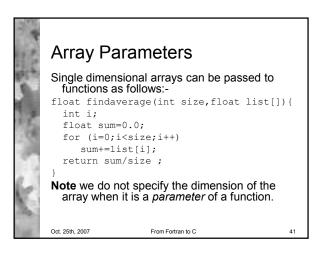


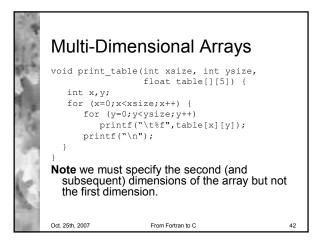


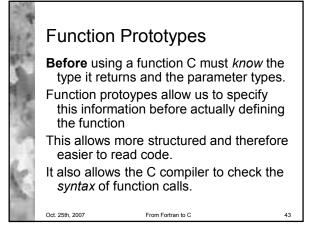


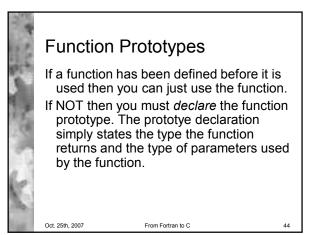


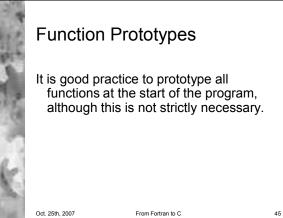


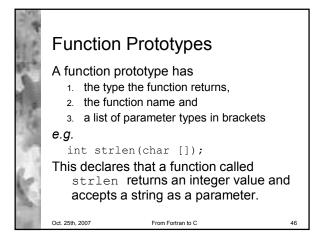


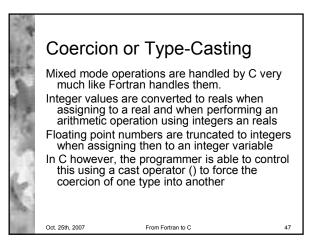


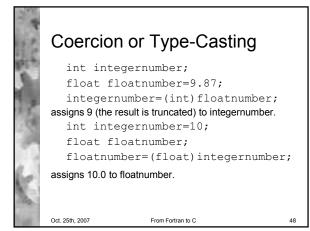




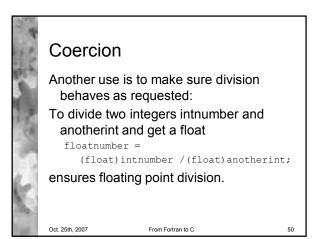


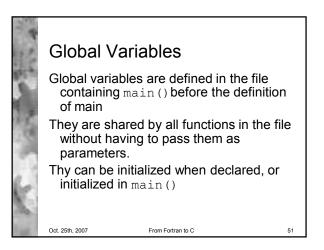






# Description Coercion can be used with any of the simple data types including char, so: int integernumber; char letter='A'; integernumber=(int)letter; assigns 65 (the ASCII code for 'A') to integernumber.





# Global Variable Example float sum=0.0; int bigsum=0; char letter='A'; funcl (int x, float y) { /\* uses sum, bigsum, letter, x, y, local vars \*/ } main() { /\* uses sum, bigsum, letter, funcl, local vars \*/ } Oct.25th,2007 From Fortran to C 52