Types during arithmetic operations

- See Expressions/binary_op.c . Notice how dividing different types leads to different results. Also, notice all the warnings that occur when you compile the program.
- Note that printing an integer as a float (i.e., printf("Dividing two integers x/y and format as float:%f\n", x / y);) results in undefined behavior, which is why you get the unexpected 0.000000

Assignment side effects

- Compile and run Expressions/chaining.c
- Observe how type conversion occurs during assignment
- Observe the side effect of chaining assignments
- Recall how the assignment operator is right associative so in this statement f = i = 33.3f; , the assignment i = 33.3f is evaluated first. Because i is an integer, the 33.3 is truncated and i now has the value 33, thus the result of the expression i = 33.3f is 33, which then gets assigned to the float variable f. This means that f's value is now 33.0 .

Reverse Digits

- See Expressions/reverse.c for complete program

post_pre.c

- See post_pre.c to run the example on the slide and see the result yourself

Extra Programs

These are sample programs not necessarily demoed in class.

- expr_eval.c -- shows the combination of postfix and prefix incremenet/decrement operations plus assignment chaining. Try to calculate the value of a on your own first then compile and run the program to see if you are right!
- undefined.c -- we cannot be sure of the value of k because we do not know which expression the compiler will evaluate first. What do you think the value of k should be and how can you be sure it matches your expectation?