Strings in C

- In C, strings are ordinary arrays whose elements are of type char
- Must include null character ‘\0’ at end
- C strings can be added to C++ strings, but not to other C strings
- Common operations implemented by functions: strcpy to assign, strcmp to compare, strcat to concatenate
- When modifying C strings, make sure enough memory is allocated!
Arithmetic Expressions

• Basic operations: +, −, *, /
• Remainder operator: %
• Applicable to integral and floating-point types, including bool and char
• Not applicable to variables of other types unless overloaded (like for strings)
• Unlike MATLAB and FORTRAN, exponentiation not supported (use pow)

Logical Expressions

• Logical AND operator: &&
• Logical OR operator: ||
• Logical NOT operator: !
• Operators act on expressions of type bool, and return values of type bool
• Connectors && and || use short-circuit evaluation, not evaluating expressions that won’t affect final result
Relational Expressions

- Relational operators:
  - equal: ==, not equal: !=
  - <, <=, >, >=
- Applicable to numeric types, but not other types unless overloaded
- Operators return values of type bool
- Common mistake: using = instead of ==, causing assignment instead of comparison!

Bitwise Operators

- Bitwise AND operator: &
- Bitwise OR operator: |
- Bitwise XOR operator: ^
- Left shift: x << y shifts bits of x to the left by y positions, multiplying by 2^y
- Right shift: x >> y shifts bits of x to the right by y positions, dividing by 2^y
- Applicable to integral types
Assignment Operators

- Basic assignment operator: =
- Accumulation operators:
  - Arithmetic: +=, -=, *=, /=, %=  
  - Bitwise: >>=, <<=, &=, |=, ^=  
- \( x \ op= y \) equivalent to \( x = x \ op y \)
- The left side is also the result, so assignments can be chained together
- Left side must be a modifiable lvalue, e.g. not a literal, array name, etc.

Other Operators

- Increment/decrement: ++, --  
  - Defined for numeric types  
  - For \( x++ \), result is old value of \( x \)  
  - For \( ++x \), result is new value of \( x \)  
- Ternary operator: \( x \ ? \ y \ : \ z \)  
  - Interpretation: if \( x \) is true (or nonzero), then the result of the operator is \( y \); otherwise \( z \)  
- Comma operator: used to allow multiple expressions in \texttt{for} statements
Precedence

- From highest to lowest precedence:
  - `++/--` (postfix), then `++/--` (prefix)
  - `!`, unary `-`, unary `+`, `*` (dereferencing), `&` (address)
  - `*` (multiplication), `/`, `%`, then `+`, `-`
  - `<`, `<=`, `>`, `>=`, then `==`, `!=`
  - `&` (bitwise AND), `^`, `|`
  - `&&`, `||`
  - `?` :
  - assignment
  - Comma

- Use parentheses to help order evaluations

Associativity

- An operator `op` is
  - left associative: `x op y op z = (x op y) op z`
  - right associative: `x op y op z = x op (y op z)`

- Left associative: arithmetic, shift, relational, bitwise, `&&` and `||`
- Right associative: `++/--`, `!`, unary `+-`, pointer operators, ternary, assignment, comma
Type Conversions

- An expression can be converted from one type to another using an explicit cast: `(type) expr` converts `expr` to indicated `type`
- Many operations perform automatic conversions, such as converting an `int` to a `double` for arithmetic operations involving `doubles`
- Expressions in conditions are converted to `bool`: nonzero is `true`, zero is `false`

Next Time

- `if` statements
- `for` statements
- `while` loops
- `switch` statements