Common Operators

- **Arithmetic operators:**
  - Unary negation: \(-x\)
  - Addition/subtraction: \(x+y\), \(x-y\)
  - Multiplication/division: \(x*y\), \(x/y\)
  - Division between integer types **truncates** to integer:
    - \(23/4 \rightarrow 5\)
  - \(x\%y\) returns the **remainder** of \(x\) divided by \(y\):
    - \(23\%4 \rightarrow 3\)
  - Division with real types yields a real result:
    - \(23.0/4.0 \rightarrow 5.75\)
  - Same rules as algebra for precedence and associativity

- **Comparison operators:**
  - Equality/inequality:
    - \(x == y\), \(x != y\)
  - Less than/greater than:
    - \(x < y\), \(x > y\)
  - Less than or equal/greater than or equal:
    - \(x <= y\), \(x >= y\)

(ex: operators.c, truncation.c and rounding.c)

The Assignment Operator

- **variable = value**
  - LHS must indicate space in memory
  - RHS must have value
    - should be of the same type
    - calculated before assignment
  - Both: A Statement and An Arithmetic Operator
    - changes the value of the space indicated by the LHS
    - returns the value that is assigned
    - right to left associative

(ex: assignments.c)
User Input

- stdin
  - input from keyboard
  - must be put into a variable
- scanf
  - like printf
  - it is a library function
  - defined in <stdio.h>
  - must tell it where the value is to be stored

More about variables

- declaration and (initialization or assignment)
  - int a,b = 5;
  - a = 8;
- space is associated with a word so it has a "name" (answers: who)
- variables given space in memory so it has an "address" (answers: where)
- that space is assigned an integer so it has a "value" (answers: what contents)
- that space is also of a specified size so it has a "type" (answers: what type)

Use the name

- to assign it a value use it on the LHS of an assignment statement
  - a = 10;
- to get to its value use its name in an expression
  - printf("%d",a);
  - b = a+100;
- to get at its address use its name with a & in front of the name
  - printf("%p",&a);
  - scanf("%d",&a);
- (ex: addresses.c and scanfdemo.c)
**scanf specifics**

- 1st argument = format specification string
- other arguments = list of places to put values of the indicated types
- format specifier – should not have any size indicator (just %d, %f or %c)
- If the input is the wrong type, it will usually cause a runtime error.

---

**Character input**

- not space delimited unless you put the spaces between
- int i1, i2, i3;
  char c1, c2, c3;
  scanf("%d%d%d", &i1, &i2, &i3);
  // reads space delimited
  scanf("%c%c%c", &c1, &c2, &c3);
  // reads the exact three consecutive values
  (ex: charinput.c)

---

**Additional Operators**

- Increment and Decrement
  - ++ and --
- unary operators
- prefix & postfix

  (ex: increment.c)
math library functions

- `#include <math.h>`
- `-lm` option when compiling
- list on page 115
- must be careful to watch the types

(ex: mathex.c)

<table>
<thead>
<tr>
<th>More formatted output</th>
<th>The printf format specifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What we've done so far</td>
</tr>
<tr>
<td></td>
<td>%d - integer in as much space as needed</td>
</tr>
<tr>
<td></td>
<td>%.2f – float in as much space as needed with two decimal places also shown</td>
</tr>
<tr>
<td></td>
<td>Other Variations</td>
</tr>
<tr>
<td></td>
<td>%nd – where n is any integer – leaves extra space before the number if there is any</td>
</tr>
<tr>
<td></td>
<td>%-nd – where n is any integer – leaves extra space on the right</td>
</tr>
<tr>
<td></td>
<td>%n.xf – where n and x are any integers – n indicates total width and x indicates decimal places</td>
</tr>
</tbody>
</table>

(ex: formatoutput.c)