This is CS50
Pass/Fail
CS50 Lunches
Fridays, 1:15pm
CS50 Queue
sectioning
starts later today, ends Mon at noon
problem set 1
standard edition
problem set I

Hacker Edition
walkthrough I
this Sun, 7pm
how to compile a program

make hello
how to run a program

./hello
functions

main
Standard Library

stdio.h

printf

...
CS50 Library

GetChar
GetDouble
GetFloat
GetInt
GetLongLong
GetString
primitive types

char  double  float  int  long  long  ...
CS50 types

bool string ...

escape sequences
\n \r \t \' \" \\ \0 ...
math

+  -  *  /  %
<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>Parentheses (grouping)</td>
<td>left-to-right</td>
</tr>
<tr>
<td>[]</td>
<td>Brackets (array subscript)</td>
<td>left-to-right</td>
</tr>
<tr>
<td>.</td>
<td>Member selection via object name</td>
<td>left-to-right</td>
</tr>
<tr>
<td>-&gt;</td>
<td>Member selection via pointer</td>
<td>left-to-right</td>
</tr>
<tr>
<td>++ --</td>
<td>Prefix increment/decrement (see Note 1)</td>
<td>right-to-left</td>
</tr>
<tr>
<td>++ --</td>
<td>Prefix increment/decrement</td>
<td>right-to-left</td>
</tr>
<tr>
<td>+ -</td>
<td>Unary plus/minus</td>
<td>left-to-right</td>
</tr>
<tr>
<td>! ~</td>
<td>Logical negation/bitwise complement</td>
<td>left-to-right</td>
</tr>
<tr>
<td>(type)</td>
<td>Cast (change type)</td>
<td>left-to-right</td>
</tr>
<tr>
<td>*</td>
<td>Dereference</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&amp;</td>
<td>Address</td>
<td>left-to-right</td>
</tr>
<tr>
<td>sizeof</td>
<td>Determine size in bytes</td>
<td>left-to-right</td>
</tr>
<tr>
<td>% / &amp;</td>
<td>Multiplication/division/modulus</td>
<td>left-to-right</td>
</tr>
<tr>
<td>+ -</td>
<td>Addition/subtraction</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&lt;&lt; &gt;&gt;</td>
<td>Bitwise shift left/Bitwise shift right</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; =</td>
<td>Relational less than/less than or equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&gt; &gt;=</td>
<td>Relational greater than/greater than or equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>== !=</td>
<td>Relational is equal to not equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&amp;</td>
<td>Bitwise AND</td>
<td>left-to-right</td>
</tr>
<tr>
<td>^=</td>
<td>Bitwise exclusive OR</td>
<td>left-to-right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bitwise inclusive OR</td>
</tr>
<tr>
<td>&amp; &amp;</td>
<td>Logical AND</td>
<td>left-to-right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>? :</td>
<td>Ternary conditional</td>
<td>right-to-left</td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>+= -=</td>
<td>Addition/subtraction assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>*= /=</td>
<td>Multiplication/division assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>%= &lt;=</td>
<td>Modulo/bitwise AND assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>^= l =</td>
<td>Bitwise exclusive/inclusive OR assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>&lt;&lt; &gt; &gt;=</td>
<td>Bitwise shift left/right assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separate expressions)</td>
<td>left-to-right</td>
</tr>
</tbody>
</table>
conditions

if (condition)
{
  // do this
}

conditions

if (condition)
{
    // do this
}
else (condition)
{
    // do that
}
conditions

if (condition)
{
    // do this
}
else if (condition)
{
    // do that
}
else
{
    // do this other thing
}
Boolean expressions

if (condition || condition)
{
    // do this
}

Boolean expressions

```java
if (condition && condition) {
    // do this
}
```
switches

```java
switch (expression) {
    case i:
        // do this
        break;

    case j:
        // do that
        break;

    default:
        // do this other thing
}
```
loops

for (initializations; condition; updates)
{
    // do this again and again
}
loops

while (condition)
{
    // do this again and again
}

loops

do
{
    // do this again and again
}
while (condition);
to be continued...