Lecture 16

Project A &
More on the workings of a computer

Project A: Minesweeper

00000
01234
0 . . .
1 . . .
2 . . .
3 . . .
4 . . .

Start of game with all the spaces covered

00000
01234
0 0001.
1 0012.
2 111..
3 ......
4 ......

User selected 0,0. The game uncovered the space and all spaces adjacent to any uncovered zero.

00000
01234
0 0001#
1 00122
2 1112
3 #1111
4 11000

User selected 4,0. This was a mine, so the game is over and the full board is displayed (# are mines)

So....

• We've supplied some code
  – The main
  – Display the board
  – Read from a file to load a board
• You get to fill in the rest.

Advice

• Read the specification carefully.
• Read the code very carefully
  – Think of it like HW2 where you had to answer questions about the code.
• Write the simpler functions first
  – Test each function as best you can.
  – Cout or the debugger can be very useful here.
  – Consider commenting out the displayMap() function so you can see your testing output.
Common errors

- I spent a fair amount of time writing this whole thing. I hit a number of really annoying bugs.
  - Most common was array bounds problems.
    - These are really hard to find.
  - Some “cut and paste” errors—mainly in my for loops.
    - for(i=0; j<100; i++) can create really annoying problems.

“tricks”

- Use –Wall a lot!
  - Saved me on a x==4 line when I wanted x=4
- If your program is crashing…
  - Use gdb (or ddd). Do the following:
    - gdb a.out
    - run
    - bt
  - The “bt” stands for backtrace and should tell you where the program crashed.
    - If it is someplace really wacky, look for array bounds problems.
      - Ask for help.

On to computer representations

Subtracting

\[
\begin{align*}
1001 - 0010 & = 0011 \\
1001 - 0011 & = 0011
\end{align*}
\]
Consider adding

\[
\begin{array}{c|c|c|c}
1001 & \text{abcd} \\
+ 0010 & \text{efgh} \\
\hline
1011 & \text{wxyz}
\end{array}
\]

- Notice that
  - \( z = (d \text{ and } h) \) or \((!d \text{ and } h)\)
  - \( c_1 = d \text{ and } h \)
- Who cares?
  - It turns out we can build an adder out of “and”, “or” and “not” operations.
  - We can also build subtrators, multipliers, etc.
- We can use these simple “ands”, “ors” and “nots” to build a computer. But we pretty much are stuck with base 2.

Gates

- We can draw logic gates

\[\text{AND} \quad \text{OR} \quad \text{NOT}\]

Gates example

\[
\begin{array}{c|c|c|c}
\text{0} & 1 & \text{c}_1 & \text{a} \\
+ 0 & + & \text{b} & \text{1} \\
\hline
\text{1} & \text{s}
\end{array}
\]

char

- A char is an 8-bit 2’s complement number.
  - So you can represent -128 to 127.
- But we usually use it to represent a character.
  - We map each of the 256 values to a different symbol.
Selected parts of the ASCII table

<table>
<thead>
<tr>
<th>B10</th>
<th>B08</th>
<th>B16</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
<td>00000000</td>
</tr>
<tr>
<td>007</td>
<td>007</td>
<td>007</td>
<td>00000111</td>
</tr>
<tr>
<td>008</td>
<td>010</td>
<td>008</td>
<td>00001000</td>
</tr>
<tr>
<td>009</td>
<td>011</td>
<td>009</td>
<td>00001001</td>
</tr>
<tr>
<td>010</td>
<td>012</td>
<td>00A</td>
<td>00001010</td>
</tr>
<tr>
<td>011</td>
<td>013</td>
<td>00B</td>
<td>00001011</td>
</tr>
<tr>
<td>012</td>
<td>014</td>
<td>00C</td>
<td>00001100</td>
</tr>
<tr>
<td>013</td>
<td>015</td>
<td>00D</td>
<td>00001101</td>
</tr>
<tr>
<td>014</td>
<td>016</td>
<td>00E</td>
<td>00001110</td>
</tr>
<tr>
<td>037</td>
<td>045</td>
<td>025</td>
<td>00100101</td>
</tr>
<tr>
<td>038</td>
<td>046</td>
<td>026</td>
<td>00100110</td>
</tr>
<tr>
<td>039</td>
<td>047</td>
<td>027</td>
<td>00100111</td>
</tr>
<tr>
<td>040</td>
<td>050</td>
<td>028</td>
<td>00101000</td>
</tr>
<tr>
<td>048</td>
<td>060</td>
<td>030</td>
<td>00110000</td>
</tr>
<tr>
<td>049</td>
<td>061</td>
<td>031</td>
<td>00110001</td>
</tr>
<tr>
<td>050</td>
<td>062</td>
<td>032</td>
<td>00110010</td>
</tr>
<tr>
<td>051</td>
<td>063</td>
<td>033</td>
<td>00110011</td>
</tr>
<tr>
<td>072</td>
<td>110</td>
<td>048</td>
<td>01001000</td>
</tr>
<tr>
<td>073</td>
<td>111</td>
<td>049</td>
<td>01001001</td>
</tr>
</tbody>
</table>

Some of the table is in your book (p958ish)
Easy to find on the web.