

CSC209H Worksheet: Binary I/O and Binary Files

Consider the following C statement: `char ch = 'A';` One byte of memory is allocated for the variable `ch` and in the first row of table below we see that the bits are set to 01000001. This is the 65, the ASCII code for the letter 'A'. (Aside: We don't expect you to memorize any ASCII codes. You can look them up in this table <http://www.ascii-code.com>.) **Hint for the table:** the ASCII code for '5' is 53, in decimal. Notice in the second row that when we use `fprintf` to print a string containing the uppercase A to a file, one byte is written to the file. Again it is the ASCII code for A.

Complete the rest of the table showing for each statement what is saved in memory and what is written to the file. (When you get to the point of storing integers don't stress about the actual 2's complement details that you might be learning in CSC258 – just make sure you know the difference between storing the `sizeof(int)` bytes in binary vs. the ASCII representation for each digit.)

What's written to the file? (show the bits)	Code	What's in the Memory? (show the bits)
N/A	<code>char ch = 'A';</code>	<div>01000001</div>
<div>01000001</div>	<code>fprintf(fp, "A");</code>	N/A
<div>0100 0001</div>	<code>char ch = 'A'; fprintf(fp, "%c", ch);</code>	<div>0100 0001</div>
<div>0011 0101</div>	<code>fprintf(fp, "5");</code>	<div>0011 0101</div>
<div>0011 0101</div>	<code>char ch = '5'; fprintf(fp, "%c", ch);</code>	<div>0011 0101</div>
	<code>int i = 5;</code>	<div>00000000 0000 0000 0000 0000 0000 0101</div>
<div>0011 0101</div>	<code>int i = 5; fprintf(fp, "%d", i);</code>	<div>←</div>
<div>0000 0000 0000 0000 0000 0000 0000 0101</div>	<code>int i = 5; fwrite(&amp;i, sizeof(int), 1, fp);</code>	<div>↓</div>
<div>0011 0101</div>	<code>char ch = '5'; fwrite(&amp;ch, sizeof(char), 1, fp);</code>	<div>0011 0101</div>

Flip over the page for another question.

## CSC209H Worksheet: Binary I/O and Binary Files

You should have noticed that more bytes are written for the int `i` than just a single character. Is this always the case? Identify (with justification) the range of integers `i` such that:

- (a) more bytes are required to store `i` in binary – using `fwrite(&i, sizeof(int), 1, fp)` – than in text – using `fprintf(fp, "%d", i)`.
- (b) the same number of bytes are required to store `i` in binary and in text.
- (c) fewer bytes are required to store `i` in binary than in text.

$> 10000$

$\curvearrowright$  0 – 999

$\rightarrow$  1000 – 9999

# Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

P2

28 28

0\_0\_0 255\_0\_0\_0

0	0	0	255
---	---	---	-----

>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder