Character Encodings

Representing numbers is great -- but what about words? Can we make sentences with binary data?
To begin to create words:

• A letter is __________ binary bits.
  __________ hex digits!

(We call this unit a __________.)
Organization

Global standard called the American Standard Code for Information Interchange (ASCII) is a _______________________ for translating numbers to characters.
### ASCII

<table>
<thead>
<tr>
<th>Column</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Pattern</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>ASCII</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

#### Fig. 14.12  ASCII, 1967 and 1968

- Change of name
- New character
- Moved character

**Coded Character Set, History, and Development (Charles MacKenzie)**

[https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf](https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf)
### ASCII

<table>
<thead>
<tr>
<th>Column</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pattern</td>
<td>SCK</td>
<td>STP</td>
<td>CTS</td>
<td>RTR</td>
<td>DSR</td>
<td>DCD</td>
<td>DSR</td>
<td>DCD</td>
</tr>
<tr>
<td>Row</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ASCII</td>
<td>RQ</td>
<td>ACK</td>
<td>NAK</td>
<td>BUSY</td>
<td>BAUD</td>
<td>XON</td>
<td>XOFF</td>
<td>DEL</td>
</tr>
</tbody>
</table>

**Fig. 14.12**  ASCII, 1967 and 1968
<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Column</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>b7 b6 b5</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b4 b3 b2 b1</td>
<td>Row</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>O</td>
<td>@</td>
<td>P</td>
<td>~</td>
<td>P</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>SOH</td>
<td>DC1</td>
<td>1</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>LTX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
0b 0100 0001 = 0x41 = A
0b 0100 0001 = 0x41 = A
0b 0100 0010 = 0x42 = B
\begin{align*}
0b\ 0100\ 0001 &= 0x41 = A \\
0b\ 0100\ 0010 &= 0x42 = B \\
&= 0x43 \\
&= 0x44
\end{align*}
0b 0100 0001 = 0x41 = A
0b 0110 0001 = 0x61 = a
Shortcomings with ASCII
Other Character Encodings
Character Encodings

There are many other character encodings beyond ASCII.
Character Encodings

One of the most common is the Unicode Transformation Format (8-bit), commonly called:
ISO/IEC 10646

ISO/IEC International Standard
ISO/IEC 10646
Final Committee Draft

Information technology – Universal Coded Character Set (UCS)
Technologie de l'information – Jeu universel de caractères codés (JUC

Second edition, 2010

https://www.iso.org/standard/76835.html
<table>
<thead>
<tr>
<th>Length</th>
<th>Byte #1</th>
<th>Byte #2</th>
<th>Byte #3</th>
<th>Byte #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-byte:</td>
<td>0___ _____</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-bytes:</td>
<td>110_ _____</td>
<td>10__ _____</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-bytes:</td>
<td>1110 ____</td>
<td>10__ _____</td>
<td>10__ _____</td>
<td></td>
</tr>
<tr>
<td>4-bytes:</td>
<td>1111 0___</td>
<td>10__ _____</td>
<td>10__ _____</td>
<td>10__ _____</td>
</tr>
</tbody>
</table>
Characters in UTF-8

a
Characters in UTF-8

ε

U+03b5
0100 1000 0110 1001 1111 0000

1001 1111 1000 1110 1000 1001

1001 1111 1000 1110 1000 1001
<table>
<thead>
<tr>
<th>Length</th>
<th>Byte #1</th>
<th>Byte #2</th>
<th>Byte #3</th>
<th>Byte #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-byte:</td>
<td>0___</td>
<td>_____</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-bytes:</td>
<td>110_</td>
<td>_____</td>
<td>10__</td>
<td>_____</td>
</tr>
<tr>
<td>3-bytes:</td>
<td>1110</td>
<td>_____</td>
<td>10__</td>
<td>10__</td>
</tr>
<tr>
<td>4-bytes:</td>
<td>1111</td>
<td>0___</td>
<td>10__</td>
<td>10__</td>
</tr>
</tbody>
</table>
Hi 🎉
Programming in C
You already know C++!
You already know C++!

Programming in C is a simplification of C++.
1. Program Starting Point:
2. Printing to `stdout`
```c
#include <stdio.h>

int main() {
    int i = 42;
    char *s = "Hello, world!";
    float f = 3.14;

    printf("%d  %s  %f\n", i, s, f);
    printf("%d\n", s[0]);
    printf("%d\n", s);
    printf("%d\n", f);

    return 0;
}
```
3. Pointers
4. Heap Memory Allocation
#include <stdlib.h>
#include <stdio.h>

int main() {
    char *s = malloc(10);
    int *num = malloc(sizeof(int));

    printf("%p %p\n", s, num);
    return 0;
}
5. Strings
5. Strings

There is no “data type” in C known as a string. Instead, we refer to “C Strings” as a sequence of characters:

- A “C string” is just a character pointer: __________.
- The string continues until it reaches a __________ byte.
char *s = malloc(6);
strcpy(s, "cs240");
printf("s[0]: 0x%x == %d == %c\n", s[0], s[0], s[0]);
printf("s[4]: 0x%x == %d == %c\n", s[4], s[4], s[4]);
printf("s[5]: 0x%x == %d == %c\n", s[5], s[5], s[5]);
printf("s == "%s", strlen(s): %ld\n\n", s, strlen(s));

char *s2 = s + 2;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == "%s", strlen(s2): %ld\n\n", s2, strlen(s2));

*s2 = 0;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == "%s", strlen(s2): %ld\n\n", s2, strlen(s2));

printf("s == "%s", strlen(s): %ld\n", s, strlen(s));
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
    char *s = malloc(5);
    s[0]=0xF0; s[1]=0x9F; s[2]=0x8E; s[3]=0x89; s[4]=0x00;

    char *s1 = "\xF0\x9F\x8E\x89";
    char *s2 = "🎉";
    char *s3 = "\U0001f389";  // \U - must be 8 bytes

    printf("%s %s %s %s\n", s, s1, s2, s3);
    printf("strlen(): %ld %ld %ld %ld\n", strlen(s),
        strlen(s1), strlen(s2), strlen(s3));
    return 0;
}
```
Some extremely useful built in string functions:

- **strcmp(char *s1, char *s2)** -- Compares two strings
- **strcat(char *dest, char *src)** -- Concatenate two strings
- **strcpy(char *dest, char *src)** -- Copies a string
- **strlen(char *s)** -- Returns the length of the string
Homework and MP1: Emojis