Today’s Goals

- Number Systems
- Pointers
  - Declaration
  - Assignment
  - Indirection/de-referencing

Decimal Number System

(Base-10 number system)

\[ 123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 \]

Section 1

Binary Number System

(Base-2 number system)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Binary Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

101_2

\[ = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]
\[ = 1 \times 4 + 0 \times 2 + 1 \]
\[ = 5_{10} \]

Capacity of Binary Numbers

- 1 bit can distinguish 2 states (0 or 1).
- An \( n \)-bit binary number can distinguish \( 2^n \) states.

Octal Number System

(Base-8 number system)

\[ 173_8 \]
\[ = 1 \times 8^2 + 7 \times 8^1 + 3 \times 8^0 \]
\[ = 1 \times 64 + 7 \times 8 + 3 \]
\[ = 123_{10} \]

Hexadecimal Number System

(Base-16 number system)

<table>
<thead>
<tr>
<th>Character correspondence:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
</tbody>
</table>

9AB_{16}

\[ = 9 \times 16^2 + 10 \times 16^1 + 11 \times 16^0 \]
\[ = 9 \times 256 + 10 \times 16 + 11 \]
\[ = 2475_{10} \]
### Byte

1 Byte
- 10001101
- 8 bits – can distinguish 256 ($2^8$) states
- Representable by 2 hexadecimal characters
- $10001101_2$ = $D_{16}$

### Kilobyte – KB

- Commonly denoted as KB, Kb, Kbyte or just K.
- Equal to 1000 or 1024 ($2^{10}$) bytes, depending on whom you ask.
- One Kb then can distinguish $2^{10} \times 2^8 = 2^{18} = 262,144$

### Section 2

#### Base-\(k\)-to-Decimal Conversion

- $101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5_{10}$
- $\sum_{i=0}^{n} a_i \times k^i$
- $9_{10} = 1001_2$

#### Decimal-to-Binary Conversion

- $5_{10} = 101_2$
- $2 \div 2 = 1 \ldots 0 \leftarrow$ the second bit from the right
- $\downarrow$ the leftmost bit (MSB)
- $MSB = \text{Most Significant Bit}$
- $LSB = \text{Least Significant Bit}$

### Decimal to hexadecimal

- $2475_{10} = 9AB_{16}$
- $16 \div 154 \ldots \text{11} \leftarrow$ the rightmost bit (LSB)
- $9 \ldots \text{10} \leftarrow$ second bit from the right
- $\uparrow$ the leftmost bit (MSB)

#### Decimal to Base-\(k\) conversions

- Decimal to Base-\(k\) conversions work the same way

### Section 3

#### Common C Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Largest value</th>
<th>Smallest value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>32 bit</td>
<td>$2 \times 10^9$</td>
<td>$-2 \times 10^9$</td>
</tr>
<tr>
<td>float</td>
<td>32 bit</td>
<td>$10^{\text{38}}$</td>
<td>$-10^{\text{38}}$</td>
</tr>
<tr>
<td>double</td>
<td>64 bit</td>
<td>$10^{\text{308}}$</td>
<td>$-10^{\text{308}}$</td>
</tr>
<tr>
<td>char</td>
<td>8</td>
<td>127</td>
<td>-128</td>
</tr>
</tbody>
</table>

- Based on 32-bit architecture
- Shaded values are approximate.
- Precision of float is 6 digits, double is 9-15 digits.
Variable and Address
- Variable = Storage in computer memory
  - Contains some value
  - Must reside at a specific location called *address*
  - Basic unit – byte
  - Imagine memory as a one-dimensional array with addresses as byte indices
  - A variable consists of one or more bytes, depending on its type (size)

Memory
- Each byte has a unique address
- A pointer stores a memory address, thus the size of a pointer is machine dependent
- With most data models it is the largest integer on the machine, size of **unsigned long**
- Defined in **inttypes.h**
  - **uintptr_t** and **uintmax_t**

Address Operations in C
- Declaration of pointer variables
  - The **pointer declarator** `*`
- Use of pointers
  - The **address of** operator `&`
  - The **indirection** operator `*` – also known as de-referencing a pointer

Pointer Declaration
- Syntax
  - `destinationType * varName;`
- Must be declared with its associated type.
- Examples
  - `int *ptr1;`  \(\Rightarrow ptr1\)
    - A pointer to an int variable
  - `char *ptr2;`  \(\Rightarrow ptr2\)
    - A pointer to a char variable

Pointers are NOT integers
- Although memory addresses are essentially very large integers, pointers and integers are not interchangeable.
- Pointers are not of the same type
- A pointer’s type **depends** on what it points to
  - `int *p1; // sizeof(int)`
  - `char *p2; // sizeof(char)`
- C allows free conversion btw different pointer types via casting (dangerous)
**Address of Operator**

- Syntax
  - `& expression`
  
  The expression must have an address. E.g., a constant such as “1” does not have an address.

- Example
  
  ```c
  int x = 1;
  f(&x);
  ```
  
  The address of `x` (i.e. where `x` is stored in memory), say, the memory location 567, (not 1) is passed to `f`.

**Pointer Assignment**

- A pointer `p` points to `x` if `x`’s address is stored in `p`

- Example
  
  ```c
  int x = 1;
  int *p;
  p = &x;
  ```
  
  Interpreted as:
  
  ![Pointer Assignment Example Diagram]

- Example
  
  ```c
  int x = 1;
  int *p, *q;
  p = &x;
  q = p;
  ```
  
  Interpreted as:
  
  ![Pointer Assignment Example Diagram]

**Pointer Diagram**

- Example
  
  ```c
  int i = 8;
  int *ip;
  ip = &i;
  ```

  ![Pointer Diagram Example Diagram]

**Indirection Operator**

- Syntax
  
  - `* pointerVar`

  - Allows access to value of memory being pointed to

  - Also called dereferencing

- Example
  
  ```c
  int x = 1, *p;
  p = &x;
  printf("%d\n", *p);
  ```
  
  `*p` refers to `x`; thus prints 1
Assignment Using Indirection Operator

• Allows access to a variable indirectly through a pointer pointed to it.
• Pointers and integers are not interchangeable
• Example
  - int x = 1, *p;
  - p = &x;
  - *p = 2;
  - printf("%d\n", x);
  - *p is equivalent to x

Summary

• Pointer and integers are not exchangeable
• Levels of addressing (i.e. layers of pointers) can be arbitrarily deep
• Remember the & that you MUST put in front of scanf variables?
• Failing to pass a pointer where one is expected or vise versa always leads to segmentation faults.