Week 3 Lecture

B&O Chapter 2 (continued)

## Integer Representations

1. C Integer Types

|  |  |  |  |
| --- | --- | --- | --- |
| Signed Integer Type | | Unsigned Integer Type | |
| **Integer Type** | **Format Specifier** | **Integer Type** | **Format Specifier** |
| char† | %c (%hhi for numerical) | unsigned char | %c (%hhu for numerical) |
| short | %hi | unsigned short | %hu |
| int | %i or %d | unsigned int | %u |
| long | %li | unsigned long | %lu |
| long long‡ | %lli | unsigned long long‡ | %llu |

† The type **char** can be either signed or unsigned. If you need a guarantee that it is signed, use type **signed char**.

‡ Introduced in C99 version of the standard.

* 1. Integer Type Ranges for Signed vs. Unsigned

|  |  |  |
| --- | --- | --- |
| Width | Encoding | Range |
| 8-bits | Unsigned | to |
| Signed | to |
| 16-bits | Unsigned | to |
| Signed | to |
| 32-bits | Unsigned | to |
| Signed | to |
| 64-bits | Unsigned | to |
| Signed | to |
| **-bits** | **Unsigned** | **to** |
| **Signed** | **to** |

1. **Unsigned Encoding**
   1. Recall how we converted binary to decimal during class last week.
   2. Let the binary to unsigned integer function .
      1. This is no different than what we did last time. This is just a new way of writing it.
2. **Signed Encodings**
   1. Sign and Magnitude Representation
      1. 1 bit for the sign
      2. All other bits for magnitude
      3. -bit binary strings can represent integers ranging from to
   2. One’s Complement Representation
      1. Let the one’s complement function be represented by .
      2. Two different ways to find one’s complement of an integer :
         1. Subtract the binary representation of from an equal-width binary string made up of all 1s
            1. Example: Finding one’s complement of using this first method.
         2. Flip the bits of x using bitwise not.
            1. Example: Finding one’s complement of using this second method.
   3. Two’s Complement Representation
      1. Let the two’s complement function be represented by .
         1. Example: Let . Find the binary representation of using two’s complement.
            1. Steps to solve:

Take the one’s complement of (i.e. flip each bit).

Add one.

So,

* + 1. Question: Does ?
       1. Let’s go back and look at the example we just used where .
          1. Recall our answer, .
          2. So, .
    2. Two’s Complement in C
       1. You could just return the negation of x.

|  |
| --- |
| int TC(int x) {  return -x;  } |

* + - 1. But what if we can’t use negation?

|  |
| --- |
| int TC(int x) {  return ~x + 1;  } |

* + 1. The C specification doesn’t say, but most implementations use two’s complement.

1. Casting from signed to unsigned
   1. Two’s complement to unsigned
      1. For such that
   2. Unsigned to two’s complement
      1. For u such that

# C Programming

## Pointers

* A pointer variable points to an address in memory.
* A pointee is the memory location to which the pointer points.
* The code below shows how to declare a pointer and how to use the & and \* operators (K&R, p. 94).

|  |
| --- |
| int x = 1, y = 2, z[10];  int \*p; /\* p is a pointer to an int \*/  p = &x; /\* p now points to x \*/  y = \*p; /\* y is now 1 \*/  \*p = 0; /\* x is now 0 \*/  p = &z[0]; /\* p now points to z[0] \*/ |

* + The address of operator, &, returns a reference to a variable’s address.
  + The dereferencing operator, \*, returns the value stored at the address to which p points.
* Example:
  + Let be an integer variable, such that and is stored at address . Now, let be an integer pointer, such that .
    - Can you fill in the values missing below?
* **Three rules**:
  1. Pointer and pointee are separate, so don’t forget to set up the pointee.
  2. Dereference a pointer to access its pointee.
  3. Assignment (=) between pointers makes them point to the same pointee.
* [Pointer Fun with Binky](https://youtu.be/5VnDaHBi8dM)