§8.0-8.3: Data Storage and Number Bases

- *Quiz ch6 today*

16 Oct 2006
CMPT14x
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review from §7.0-7.8

- Strings: manipulating text
  - Null-terminated strings
  - Comparing strings
- Application: cryptography (substitution cipher)
  - Creating a library for cryptography
  - Library-internal helper functions
- Application: pseudo-random number generator
  - Accessing global variables
  - Assessing randomness

Addendum on pseudorandom

- Python has a random number generator:

```
from random import random
random()
seed()
```

  - Random float in interval [0.0, 1.0)
- Histogram:
  - Split up interval [0.0, 1.0) into equal-size bins
  - Generate a list of random numbers
  - Count how many numbers fall in each bin
- HW07 (Py ch9 #5) due Fri

Quiz05: ch6 (10 min)

- Contrast a header file with an implementation file
- Why should we use accessor functions in an ADT?
- Name three out of the five possible modes in which one can open a file, and give example Python code for each
- A program needs to copy text one character at a time from one file into another file. Put the following building blocks in the correct order to do this:

read(1)	open(,'r')	open(,'w')
write()	close()	close()

Quiz05 answers: #1-2

- Contrast a header file with an implementation file [4]
  - Header: public interface, doesn't define bodies of functions
  - Implementation: contains bodies of functions
- Why should we use accessor functions in an ADT? [5]
  - Hide implementation details from user
  - Maintain the "illusion" of the ADT
  - Ease future upgrades of internal implementation

Quiz05 answers: #3

- Name three out of the five possible modes in which one can open a file [6]
  - Read: open('file.txt', 'r')
  - Write: open('file.txt', 'w')
  - Both read+write: open('file.txt', 'r+')
  - Append: open('file.txt', 'a')
  - Binary mode: open('file.txt', 'rb')

Quiz05 answers: #4

- A program needs to copy text one character at a time from one file into another file. Put the following building blocks in the correct order to do this: [5]

```
in = open('in.txt', 'r')
out = open('out.txt', 'w')
ch = in.read(1)
while ch != '':
    out.write(ch)
    ch = in.read(1)
in.close()
out.close()
```

Ch1-7 and ch8-12

- We've already covered one whole book:
  - You now know the basics of programming!
- CMPT145 (book 2, ch8-12) dives deeper into advanced techniques:
  - Data storage and how I/O works
  - Cool datatypes: sets and records
  - Scope/visibility, exceptions
  - Software development techniques
  - Pointers and dynamic ADTs

What's on for today (§8.0-8.3)

- Number bases:
  - Binary, hexadecimal, octal
- Units of measure of memory:
  - Bits, nibbles, bytes, words, pages
- Units of measure for hard disks:
  - Geometry, cylinders/heads/sectors
- SI units vs binary units

Ch8: Data storage and I/O

- As programmers, you're already expert users of various datatypes and file I/O
- Now we peek under the hood to see what the compiler and the OS are really doing to implement these
- Every variable we declare takes up space in memory (RAM):
  - How much space does each variable need?
  - How is our data stored?

Binary numbers

- At the lowest level, all computer data are stored using logical bits: each bit can be either 0 or 1
  - High voltage (1) vs. low voltage (0)
  - Most memory chips use a big bank of tiny capacitors: has charge (1) vs. no charge (0)
- We use groups of bits to represent data (numbers, characters, strings, etc.):
  - e.g., this pattern of eight bits: 0 1 0 0 0 0 1 1
    - Could represent the decimal number 35
    - Or it might represent the character "#"
    - Or something else – depends on how we interpret it

Number bases

- God gave us 10 fingers; so we often count in base 10:
  - "5927" interpreted as a decimal number:
    - 5 units of (10^3 = 1000)
    - 9 units of (10^2 = 100)
    - 2 units of (10^1 = 10)
    - 7 units of (10^0 = 1)
- Counting in binary is similar:
  - "0110" interpreted as a binary number:
    - 0 unit of (2^3 = 8)
    - 1 unit of (2^2 = 4)
    - 1 unit of (2^1 = 2)
    - 0 unit of (2^0 = 1)

Hexadecimal, octal

- Hexadecimal is base 16: we use 'A'..'F' to represent the "digits" ten, eleven, twelve, etc.
  - "BEEF" as a hexadecimal number:
    - B (11) units of (16^3 = 4096) => 45056
    - E (14) units of (16^2 = 256) => 3584
    - E (14) units of (16^1 = 16) => 224
    - F (15) units of (16^0 = 1) => 15
    - Total: BEEF (hex) => 48879 (dec)
- There's also octal, base 8:
  - only the digits 0..7 are used

Using bases in Python

- Python has special notation for expressing integer literals in hexadecimal and octal:
  - Hexadecimal: prefix "0x"

```
hexNum = 0xBEEF        # 48879
```

  - Octal: prefix "0"

```
octNum = 0115          # 1(8²) + 1(8¹) + 5(8⁰) = 77
```

- Convert into strings with hexadecimal/octal notation:

```
hexStr = hex(48879)    # '0xbeef'
octStr = oct(77)       # '0115'
```

Bits, bytes, nibbles, words

- One hexadecimal digit can be represented by four bits: one nibble
- Two nibbles (eight bits) is called a byte
  - One byte can be used to store one CHAR
- A group of bytes can be used to represent one datum: this is called a word
  - Pentium CPUs generally use 4-byte words (32 bits)
  - Newer CPUs can use 8-byte words (64 bits)
  - Word is the smallest unit of data the machine can store or retrieve

Accessing memory

- A computer's main memory (generally, RAM) stores everything it needs to do its current tasks
- A location within memory is uniquely identified by its address
  - Most modern CPUs use 32-bit words to store memory addresses
  - This means there is a maximum of 2^{32} unique memory addresses (the address space)
  - If each location stores one byte of data, then there is 2^{32} bytes = 4GB of addressable memory

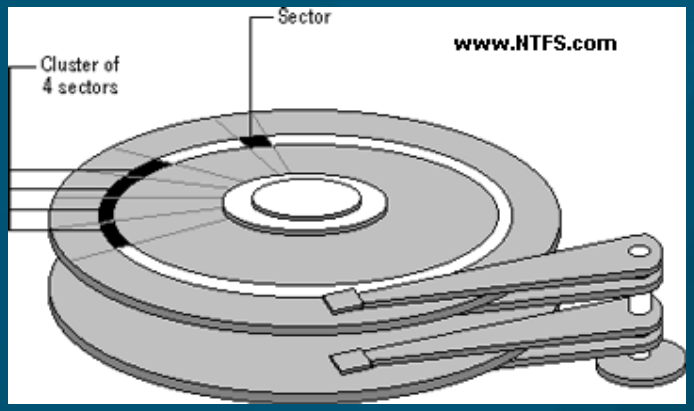
Units of measure

- SI abbreviations:
  - K = kilo = 1,000
  - M = mega = 1,000,000
  - G = giga = 1,000,000,000
- When working with binary data:
  - KB = kilobyte = 1,024 bytes = 2^{10} bytes
  - MB = megabyte = 1,024,576 = 2^{20} bytes
  - GB = gigabyte = 1,073,741,824 = 2^{30} bytes
  - But hard drive manufacturers use SI abbrevs

Units of measure, cont.

- Kilobytes vs. kilobits:
  - KB = kilobyte = 1,024 bytes = 8192 bits
  - Kb = kilobit = 1,024 bits
  - RAM chip manufacturers often use kilobits
- Also, in SI abbreviations,
  - M = mega = 10^6: e.g., megawatt = 10^6 watt
  - m = milli = 10^{-3}: e.g., milliwatt = 10^{-3} watt
- But not everyone is consistent, so be careful

Storage



- A page of memory is generally 256 bytes
- A sector is a unit of disk storage, also commonly 256 bytes (but sometimes 512 bytes)
- A block of disk storage is usually 512 bytes
- Hard disks are made up of platters, accessed by magnetic heads on movable arms
- The platters have concentric tracks that (across all heads) make up cylinders
- Hard drive geometry is often expressed in C/H/S: # cylinders / # heads / # sectors per track

Summary of today (§8.0-8.3)

- Number bases:
  - Binary
  - Hexadecimal (0xBEEF)
  - Octal (0115)
- Units of measure of memory:
  - Bits, nibbles, bytes, words, pages
- Units of measure for hard disks:
  - C/H/S geometry
- SI units vs binary units, KB vs. Kb, etc.

TODO items

- Lab05 due today/tomorrow/Wed: 6.11 #(33/35)
  - Two file: library and testbed program
- HW07 due Fri: Py ch9 #5
  - Also, write your own pseudorandom number generator, and
  - Create a histogram using your own pseudorandom, another histogram using the built-in random(), and compare
- Quiz06 (ch7) on Fri
- CMPT140 Final next week: W-Th 25-26Oct