CS 101: Computer Programming and Utilization

Jan-Apr 2017

Sharat
(piazza.com/iitb.ac.in/summer2017/cs101iitb/home)

Lecture 7: Numbers

About These Slides

- Based on Chapter 3 of the book
  An Introduction to Programming Through C++
  by Abhiram Ranade (Tata McGraw Hill, 2014)

- Original slides by Abhiram Ranade
  –First update by Varsha Apte
  –Second update by Uday Khedker
  –Third update by Sunita Sarawagi

Data Representation

What happens when you say
int x = 23;
char x= ‘a’;
float x = 23.2
long long int = 2345678

Model for Today’s Demo

1. We will “open up” the computer program
   - Compile using the “-g” flag
   - Run using the emacs debugger which allows step by step instruction
   - Example char letter = ‘A’;
2. We will use a calculator
   - Some steps will be ‘invisible’
   - Example real numbers
3. In both cases, we will need audience participation

Demo

How does the computer store c and d?

```
int c; char d; cin >> c >> d;
```

Using numeric codes

Define a numeric code for representing letters

- ASCII (American Standard Code for Information Interchange) is the commonly used code
- Letter ‘a’ = 97 in ASCII, ‘b’ = 98, …
- Uppercase letters, symbols, digits also have codes
- Code also for space character
- Words = sequences of ASCII codes of letters in the word

  ‘computer’ = 99, 111,109,112,117,116,101,114

- To write characters in, say, Devanagari, we need Unicode and a lot more concept

Representing Numbers

- Digital circuits can store 0's and 1's (using capacitors)
- How to represent numbers using this capability?
- Key idea : <u>Binary number system</u>
- Represent all data using only 1's and 0's

Number Systems

- Roman system
  - new symbols for larger numbers
  - could not represent larger numbers

Roman Numeral Table							
1	I	14	XIV	27	XXVII	150	CL
2	II	15	XV	28	XXVIII	200	CC
3	III	16	XVI	29	XXIX	300	CCC
4	IV	17	XVII	30	XXX	400	CD
5	V	18	XVIII	31	XXXI	500	D
6	VI	19	XIX	40	XL	600	DC
7	VII	20	XX	50	L	700	DCC
8	VIII	21	XXI	60	LX	800	DCCC
9	IX	22	XXII	70	LXX	900	CM
10	X	23	XXIII	80	LXXX	1000	M
11	XI	24	XXIV	90	XC	1600	MDC
12	XII	25	XXV	100	C	1700	MDCC
13	XIII	26	XXVI	101	CI	1900	MCM

MathATube.com

- Radix based number systems (e.g. Decimal)
- Revolutionary concept in number representation!

Radix-Based Number Systems

- Key idea: position of a symbol determines its value! PLACE VALUE
  - How do we determine its relative position in a list of symbols?
  - A Zero symbol needed to shift the position of a symbol

Decimal Number System

- RADIX is 10. Place-Values: 1, 10,100,1000...
- In the decimal system: 346
  - Value of "6" = 6
  - Value of "4" = 4 x 10
  - Value of "3" = 3 x 10 x 10
- Notice that we automatically decide to read either left to right, or vice versa based on convenience

Radix-Based Number Systems

- Key idea: position of a symbol determines its value! PLACE VALUE
  - How do we determine its relative position in a list of symbols?
  - A Zero symbol needed to shift the position of a symbol
- Number systems with radix *r* should have *r* symbols
  - The value of a symbol is multiplied by *r* for each left shift.
  - Multiply from right to left by: 1, r, r^2, r^3, ... and then add

Octal Number Systems

- RADIX is 8. Place Value: 1, 8, 64, 512,....
- 8 digits needed : 0,1,2,3,4,5,6,7
- 23 in octal
  - Value of 3 = 3
  - Value of 2 = 2 x 8
  - Value of 23 in octal = 19 in decimal
- 45171 in octal =
  - 1+8*7+8*8*1+8*8*8*5+8*8*8*8*4

    = 19065 in decimal

Binary System

- Radix= 2
- Needs ONLY TWO digits : 0 and 1
- Place-value: powers of two:

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

- 11 in binary:
  – Value of rightmost 1 = 1
  – Value of next 1 = 1 x2
  – 11 in binary = 3 in decimal
- 110011

128	64	32	16	8	4	2	1
		1	1	0	0	1	1

= 1x1 + 1 x2 + 0 x 4 + 0 x 8 + 1 x 16 + 1 x 32

= 1 + 2 + 16 + 32= 51 (in decimal)

Binary System: Representing Integers

- Decimal to binary conversion
  - Express it as a sum of powers of two
- Example: the number 154 in binary:
  - 154 = 128 + 16 + 8 + 2
  - $154 = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	0

  - Thus 154 in binary is 10011010

Binary System: Representing Numbers

- Decimal to binary conversion
  - Express it as a sum of powers of two
- Example: the number 154 in binary:
  - Repeatedly divided 154
    - Keep track of remainder
    - Keep track of quotient

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	0	1	1	0	1	0

Large Integers

- Number of bits decides how large the integers are
- But how many bits to use?
- The number of bits (capacitors/wires) used cannot be chosen arbitrarily
- Choices allowed: 8, 16, 32, 64
- Example: To store 25 using 32 bits:
  - 25 Decimal = 00000000000000000000000000011001
  - So store the following charge pattern (H=High, L=Low)
  - LLLLLLLLLLLLLLLLLLLLLLLLLLLHHLLH
- Range stored: 0 to $2^{32} - 1$. If your numbers are likely to be larger, then use 64 bits.
- Choose the number of bits depending upon how large you expect the number to be

Representing Negative Integers

- One of the bits is used to indicate sign
- Sign bit = 0 means positive, = 1 means negative number
- To store -25 use
  - 10000000000000000000000000011001, Leftmost bit = sign bit
- Range stored: $-(2^{31} – 1)$ to $2^{31} – 1$
- Notice the following though: How to add 2 and -1
  - 2 is `0010` -1 is `1001`
  - Cannot perform “usual addition”
- Two zeros 0000, and 1000: Every application will need to take extra steps to make sure that non-zero values are also not negative zero.

Two's complement

- If x is positive: (0 <= x <= 2^{n-1} – 1)
    - Binary form of x
- If x is negative (-2^{n-1} <= x < 0)
    - Binary form of 2^n - x
    - E.g. -25 in 2's complement: 11111111111111111111111111111100111 = (100000000000000000000000000000000 -00000000000000000000000000011001)
- In this representation, how to add 2 and -1?
    - `0010` and `1111`
- With two's complement, storing a 4-bit number in an 8-bit register is a matter of repeating its most significant bit: `0001` (1, in four bits). `00000001` (1, in eight bits) `1110` (-2), `11111110` (-2, in eight bits)

Demo