CS1007 lecture #5 notes,

thu 20 sep 2001

http://www.cs.columbia.edu/~sklar/cs1007

today:

- news
- data types and storage
- variables and assignment
- binary numbers and arithmetic
- ASCII
- homework #1
- reading: *ch 2.1-2.4*

news.

- *TA hours:*

Dali	Mon	9-10am	252 ET
Yajia	Mon	4-5pm	558 SCH
Olga	Mon	6-7pm	558 SCH
Peter	Mon	6-7pm	252 ET
Sage	Mon	7-8pm	252 ET
Steve	Mon	8-9pm	252 ET
Min	Tue	6-7pm	407 Math
Peter	Tue	7-8pm	407 Math
Steve	Tue	8-9pm	407 Math
Dali	Wed	9-10am	252 ET
Anupreet	Thu	9-10am	558 SCH
Gee	Thu	4-5pm	558 SCH
Genevieve	Fri	2-3pm	558 SCH

data and storage.

- last class we talked about output
- programs = objects + methods
- objects = data
- data must be *stored*
- all storage is numeric (0's and 1's)

memory.

- think of the computer's memory as a bunch of boxes
- inside each box, there is a number
- you give each box a name
  $\Rightarrow$ defining a *variable*
- example:

program code:
`int x;`

computer's memory:
x $\rightarrow$

variables.

- variables have:
  — name
  — type
  — value
- naming rules:
  — names may contain letters and/or numbers
  — but cannot begin with a number
  — names may also contain _ and $
  — can be of any length
  — cannot use Java keywords
  — Java is *case-sensitive!!*

primitive data types.

- numeric

byte	8 bits	-128 = $-2^8/2$	127 = $2^8/2 - 1$
short	16 bits	-32,768 = $-2^{16}/2$	32,767 = $-2^{16}/2 - 1$
int	32 bits	$-2^{32}/2$	$2^{32}/2 - 1$
long	64 bits	$-2^{64}/2$	$2^{64}/2 - 1$
float	32 bits	$\approx$ -3.4E+38, 7 sig dig	$\approx$ 3.4E+38, 7 sig dig
double	64 bits	$\approx$ -1.7E+308, 15 sig dig	$\approx$ 1.7E+308, 15 sig dig

- boolean

boolean	1 bit

- character

char	16 bits

  – 7 bits for ASCII
  – 8 bits for extended ASCII
  – 16 bits for Unicode

assignment.

- = is the assignment operator
- example:

program code:

```
int x; // declaration
x = 19; // assignment
or
int x = 19;
```

computer's memory:

x → 19

storage is binary.

x → 19

is really stored like this:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

this is base 2!

$19_{10} = 10011_2$

remember bases?

base 10:

$$\begin{aligned} 362 &= (2 * 1) + (6 * 10) + (3 * 100) \\ &= (2 * 10^0) + (6 * 10^1) + (3 * 10^2) \end{aligned}$$

base 2:

$$\begin{aligned} 1 &= 2^0 = 1 \\ 10 &= 2^1 = 2 \\ 100 &= 2^2 = 4 \\ 1000 &= 2^3 = 8 \\ 10000 &= 2^4 = 16 \\ &\ldots \end{aligned}$$

so

$$\begin{aligned} 10011_2 &= (1 * 2^0) + (1 * 2^1) + (0 * 2^2) + (0 * 2^3) + (1 * 2^4) \\ &= (1 * 1) + (1 * 2) + (0 * 4) + (0 * 8) + (1 * 16) \\ &= 19_{10} \end{aligned}$$

base conversion: 2 to 10.

1010100_2 =

	=	=
$(0 * 2^0)$	(0 * 1)	0
$+ (0 * 2^1)$	+ (0 * 2)	+ 0
$+ (1 * 2^2)$	+ (1 * 4)	+ 4
$+ (0 * 2^3)$	+ (0 * 8)	+ 0
$+ (1 * 2^4)$	+ (1 * 16)	+ 16
$+ (0 * 2^5)$	+ (0 * 32)	+ 0
$+ (1 * 2^6)$	+ (1 * 64)	+ 64

$= 84_{10}$

base conversion: 10 to 2.

84_{10} =

84 / 2	=	42	rem 0
42 / 2	=	21	rem 0
21 / 2	=	10	rem 1
10 / 2	=	5	rem 0
5 / 2	=	2	rem 1
2 / 2	=	1	rem 0
1 / 2	=	0	rem 1

$\Rightarrow 1010100_2$

two tricks.

base 8 (octal):

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

base 16 (hexadecimal, "hex"):

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A (10)
0011	3	1011	B (11)
0100	4	1100	C (12)
0101	5	1101	D (13)
0110	6	1110	E (14)
0111	7	1111	F (15)

- replace each octal/hex digit with the 3/4 digit binary
- replace every 3/4 binary digits with one octal/hex digit

back to storage.

$x \rightarrow \boxed{19}$

is really stored like this:

31	30	...	7	6	5	4	3	2	1	0
0	0	...	0	0	0	1	0	0	1	1

- bits are numbered, from right to left, starting with 0
- highest (rightmost, “most significant”) bit is *sign* bit

ASCII.

- ASCII = American Standard Code for Information Interchange
- characters are stored as numbers
- standard table defines 128 characters
- example:

```
char c = 'A';
```

‘A’ = 65_{10} = 01000001_2

$c \rightarrow$

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1