Command-line arguments

a.out
.... input thro' scanf()
rest of command line ignored.

a.out .. arguments ...

To operate with command-line arguments

int main (int argc, char *argv[])

argc = # of arguments.

argv[0] is the program name

eg. a.out

argv is an ARRAY (TBC) of CHARACTER STRINGS (TBC).

int
argc is the number of command-line arguments

char * argv [ ] is an array of character strings

This will be explained properly later.

Eg
```
#include <stdio.h>
int main (int argc, char *argv[])
{ int i;
  for (i = 0; i < argc ; ++i)
  { printf ("%s\n", argv[i]); }
}
```
↑ format item for character string

We'll need to convert command-line arguments to int/double.

```
#include <stdio.h>
#include <stdlib.h>  // atoi(), atof()
int main (int argc, char *argv[])
{ int i;
   int x;  // double x
   for( i=1; i<argc; ++i)
   { x = atoi(argv[i]);
                  // atof(argv[i])
      printf("%d\n", x);
             //%f
   }
}
```

Bases. Besides base 10, we encounter

base 2 binary 8 octal 16 hexadecimal.

A number in base (or radix) R is a sequence $a_{k-1} \cdots a_0$ of "digits,"

$0 \leq a_j < R, \quad 0 \leq j \leq k-1,$

a 'k-digit number' whose VALUE is

$$\sum_{j=0}^{k-1} a_j R^j \qquad (a_{k-1} \cdots a_0)_R$$

Bases. Besides base 10, we encounter

base 2 binary 8 octal 16 hexadecimal.

A number in base (or radix) R is a sequence $a_{k-1} \cdots a_0$ of "digits,"

$0 \leq a_j < R, \quad 0 \leq j \leq k-1,$

a 'k-digit number' whose VALUE is

$$\sum_{j=0}^{k-1} a_j R^j \qquad (a_{k-1} \cdots a_0)_R$$

Horner's method

$$(\cdots(a_{k-1}R + a_{k-2})R + a_{k-3})R + \cdots)R + a_0$$

$$(1100011)_2 =$$

$$(((1\times2+1)\times2+0)\times2+0)\times2+0)\times2+1)\times2+1$$

$$3\times2=6 \times2 = 12\times2 = 24\times2+1 = 49\times2+1 = 99$$

Hexadecimal is a better way to show binary numbers. BASE 16. hex digits

0–9	a	b	c	d	e	f
	10	11	12	13	14	15

$$(4d2)_{16} = (4\times16+13)\times16+2$$

$$= 77\times16+2 = 1234$$

Horner's method

$$(\cdots(a_{k-1}R + a_{k-2})R + a_{k-3})R + \cdots)R + a_0$$

$$(1100011)_2 =$$

$$(((1\times2+1)\times2+0)\times2+0)\times2+0)\times2+1)\times2+1$$

$$3\times2=6 \times2 = 12\times2 = 24\times2+1 = 49\times2+1 = 99$$

Hexadecimal is a better way to show binary numbers. BASE 16. hex digits

0–9	a	b	c	d	e	f
	10	11	12	13	14	15

$$(4d2)_{16} = (4\times16+13)\times16+2$$

$$= 77\times16+2 = 1234$$

$(63)_{16} = 6\times16+3 = 99$

$(ac)_{16} = 10\times16+12 = 172$

To convert from base 10 to base R, divide repeatedly + take remainder

EG 1234 in hex

$$16\overline{)1234}\quad 77 \qquad 112 \qquad 114 \qquad 112 \qquad 2$$

$$16\overline{)77}\quad 4 \qquad 64 \qquad 13 = d$$

$$16\overline{)4}\quad 0 \qquad 4$$

$(4d2)_{16}$

$4\times256 + 13\times16 + 2$

$1024 + 208 + 2 = 1234$

$(63)_{16} = 6\times16+3 = 99$

$(ac)_{16} = 10\times16+12 = 172$

To convert from base 10 to base R, divide repeatedly + take remainder

EG 1234 in hex

$$16\overline{)1234}\quad 77 \qquad 112 \qquad 114 \qquad 112 \qquad 2$$

$$16\overline{)77}\quad 4 \qquad 64 \qquad 13 = d$$

$$16\overline{)4}\quad 0 \qquad 4$$

$(4d2)_{16}$

$4\times256 + 13\times16 + 2$

$1024 + 208 + 2 = 1234$

ints are similar:

if $0 \leq x \leq 2^{31}-1$, face value 32-bit int

if $-2^{31} \leq x \leq -1$, represented by $2^{32} + x$

Calculations easier with short ints.

ints are similar:

if $0 \leq x \leq 2^{31}-1$, face value 32-bit int

if $-2^{31} \leq x \leq -1$, represented by $2^{32} + x$

Calculations easier with short ints.

Integers are 32 bit.

face value $0 \ldots 2^{32}-1$

BUT 'high order bit 1' are treated as negative.

There are also 16 bit 'short int'

high order bit 1 for negative.

So $0 \ldots 2^{15}-1$ nonnegative

$2^{15} \ldots 2^{16}-1$ negative.

---

if $0 \leq x \leq 32767$ then x is face value

if $-32768 \leq x < 0$ then x is represented by $65536 + x$.

Short int always 2 bytes.

EG 1234 as short int

$(4d2)_{16}$ previously

04d2 as 16 bits

EG −123 as short int

$123 = (7b)_{16}$

$$\begin{array}{r} 7 \\ 16\overline{)123} \\ \underline{112} \\ 11 \end{array} \qquad 11 \equiv b$$

Sub from 65536.

Trick: sub from ffff and add 1.

$$\begin{array}{r} ffff \\ 007b \\ \hline ff84 \end{array} +1$$

$-123 = ff85$ as short int.

Conversion to short int

If $0 \leq x < 2^{15}-1$, x is represented by 16-bit at face value.

if $-2^{15} \leq x \leq -1$, x represented by $2^{16} + x$.

So short ints in range $[2^{15}, 2^{16}-1]$ represent negative numbers.

To compute rep of x, $x < 0$, i.e., $2^{16} - x$

(i) get $|x|$ in 4 hex digits.
(ii) sub $|x|$ from ffff i.e, from $2^{16}-1$
(iii) add 1.

Adding in hex

```
  a b c d
  6 1 2 4
---------
    1 1
  1 0 c f
```

$d+4 = 17 = 16+1 \quad (11)_{16}$

$c+2+1 = 15 = f$

$b+1 = c$

$a+16 = 16 = (10)_{16}$

10cf1

$1234 = (04d2)_{16}$

$123 = (007b)_{16}$

$-123 \equiv (ff85)_{16}$

```
  f f f f
-   7 b
---------
  f f 8 4
+       1
---------
  f f 8 5
```

Addition of short ints.

drop the final carry.

EG

$$\begin{array}{r} 1234 \\ -123 \\ \hline 1111 \end{array} \qquad \begin{array}{r} 04d2 \\ +\ ff85 \\ \hline 10457 \end{array}$$

$(0457)_{16}$

$= (4 \times 16 + 5) \times 16 + 7$

$= 69 \times 16 + 7 = 1111$

Drop final carry

means addition mod 2^{16}

Correct

if $x, y,$ and $x+y$ are all in correct range.

EXERCISE convert

-21555 and 24868 to

short int

ANSWER a b c d

6 1 2 4

add them ~~1~~ 0 c f 1

$(0cf1)_{16} = \boxed{3313}$

$= -21555 + 24868$

Ex: 27182 − 8284 in short int

```
   1698          106           6          0
16)27182      16)1698      16)106     16)6
   16            16          96         0
   111           98          10         6
   96            96
   158           2
   144
   128
   14
```

(6a2e)$_X$ or 16

```
    517          32           2          0
16)8284      16)517       16)32      16)2
   80            48          32         0
   28            37          0          2
   16            32
   124           5
   112
   12
```

(205c)$_X$

ffff
− 205c
+dfa3
1
dfa4

6a2e
dfa4
1 4 9 d 2

Fractions in different bases

(see section 9)

$\frac{7}{13} = .d_1 d_2 \cdots$ in decimal $\times 10$

$\frac{70}{13} = d_1 . d_2 \cdots \quad d_1 = 5$

$\frac{5}{13} = .d_2 d_3 \cdots \quad \frac{50}{13} = d_2 . d_3 \cdots \quad d_2 = 3$

$\frac{11}{13} = .d_3 d_4 \cdots \quad \frac{110}{13} = d_3 . d_4 \cdots \quad d_3 = 8$

$\frac{6}{13} = .d_4 d_5 \cdots \quad \frac{60}{13} = d_4 . d_5 \cdots \quad d_4 = 4$

$\frac{8}{13} = .d_5 d_6 \cdots \quad \frac{80}{13} = d_5 . d_6 \cdots \quad d_5 = 6$

$\frac{2}{13} = .d_6 d_7 \cdots \quad \frac{20}{13} = d_6 . d_7 \cdots \quad d_6 = 1$

$\frac{7}{13} = .d_7 d_8 \cdots$ recurrence $.\dot{5}38461\dot{}$

Tabulating

	70/13	50/13	110/13	60/13	8%13	20/13
d_i	5	3	8	4	6	1

BINARY $.d_1 d_2 \ldots$

14/13	2/13	4/13	8/13	16/13	6/13	12/13	24/13	22/13	18/13	10/13	20/13
1	0	0	0	1	0	1	1	1	1	0	1

recurring.

hexadecimal.

$\frac{112}{13}$ $\frac{128}{13}$ $\frac{176}{13}$ $\frac{112}{13}$ ...

8 9 13 8 recurring $\quad .\dot{8}9\dot{d}$