Brief summary of C language

Colm Ó Dúnlaing

October 10, 2012

1 Simplest kind of program

```
#include <stdio.h>
        /* essential for input/output */

        /*
         * Comments as shown here.  Can be on several lines.
         * Asterisks on left unnecessary: for a neat appearance.
         * End comment with asterisk and forward slash.
         * Double slash for single-line comments, C++ style: //
         */

main()
{
  /* variables declared */
  /* statements: i/o, arithmetic */
  /* semicolons at end of each declaration and each statement */
  /* indent for readability: to show program structure */
}
```

2 Declaring variables

Here are some examples.

```
  char a;
  int b, c;
  double double_precision_variable_001;
        /* variable with a long name */

  double d[100];
        /* array of doubles */
```

```
int e[10][10];
      /* two-dimensional array of ints */

char * string;
      /* declares an address --- advanced. */
```

3 Input/output

- `printf( format, item, ... , item );`
- Formats: `%d, %s, %c, %f, %e, %g` integer, string, floating, scientific, general. **More about this later.**
- Special characters. `\n` newline (carriage return), `\t` tab, `%%` double percent means single percent, `\"` 'escaped' quotes means quotes,
- `\0` null character is used to mark the end of a character string.
- Input by command-line arguments
- Input by `scanf( format, address, ... , address );`
- `scanf()` *returns a value,* the number of items read. Useful for detecting end-of-data.
- Ctrl-D from the terminal signals end-of-data.
- Address of `x` is `&x`
- Related functions and routines: `fprintf, fscanf, fgets, snprintf`.

4 More about format

- 'Field width' can be included, such as `%4d`, `%10s`, etcetera.
- If the item is too short, it is **right-justified** to fit.
- A minus-sign causes **left justification.** It has nothing to do with the sign of numbers.
- With `int` items, a zero causes padding with zeroes rather than blanks. For example,

```
printf("%06d\n", 7)
```

  produces

  `000007`

- With formats `%f, %e, %g`, a decimal point gives precision. Thus

```
printf("%6.3f\n", 7)
```

produces 3 decimal places in a field width of 6:

```
 7.000
```

5 For loops

```
for ( i = low; i < too_high; ++ i )
{
  <statements>
}
```

NOTE Right curly bracket } marks end of group of statements. There **must** be a semicolon before } (unless empty), and there **should not** be a semicolon immediately after.

This is one of the peculiarities of C.

The above description shows a for-loop as it is most frequently used. In full generality, a for-loop looks like

```
for ( <do first>; <(while) still going>; <do between reps> )
{
  <statements>
}
```

Relation symbols

Mathematical form	C form
$<$	<
$\leq$	<=
$=$	==
$\geq$	>=
$>$	>
$\neq$	!=

6 Constants and character strings

- Number constants are as you would expect, such as `123` or `-123.456`

- A character constant is enclosed in single quotes, like

```
'a'    or   '\t'    or    '\n'    or   '\0'
```

meaning a, tab, newline, and null character.

- A character string is a sequence of characters stored in memory. It must be terminated with a null character. A string constant is enclosed in double quotes:

  `"hello\n", "123\t455\n", "\"hello\""`

- Notice `\"`: the backslash is an 'escape' character so the double quote is taken as a character, not enclosing the character string.

7 Command line input

- `main ( int argc, char * argv [] )`
- `argc` is the number of command-line arguments, beginning the count at 0.
- `argv[0]` is a character string, actually the name of the program (such as `a.out`).
- `argv[1], argv[2], ... , argv[ argc - 1 ]`

  are the first, second, ..., command-line arguments.

- Include

```
#include <stdlib.h>
```

  to gain access to the functions `atoi()` and `atof()` functions.

- Each argument is a character string, and `atoi()` can convert it to integer, as in

```
n = atoi ( argv[ 1 ] );
```

- `atof()` converts character strings to *double*.

8 Naming variables

Any mixed string of letters, digits, and underscores, not beginning with a digit, is an acceptable name. Case sensitive: one should use lowercase letters, reserving uppercase for other things.