2 For-loops

Here is a simple program.

```
#include <stdio.h>

main()
{
  int i;

  for ( i=0; i<5; i = i+1 )
    printf( "hello\n");
}
```

The output is

```
hello
hello
hello
hello
hello
```

- Every C program must contain one section `main(){ ...}`.
- This program uses one *variable,* an integer `i`.
- `printf()` prints to the terminal. It is essential, but it is not part of the C language proper. The line

  ```
  #include <stdio.h>
  ```

  tells **gcc** that there is a file (somewhere) called `stdio.h` which needs to be included. It helps explain the `printf()` statement.
- The text `"hello\n"` is called a *character-string constant.* It includes the *newline* (or carriage-return) character `\n`.
- The statement

  ```
  i = i+1;
  ```

  means *replace the variable* `i` (stored somewhere in central memory) by the new value `i+1`. There is a shorter way to write this:

  ```
  ++i;
  ```

  This abbreviation should be used with care: it is more complicated than it looks.

- The `for (...) ...` statement is called a **for-loop.** It operates as follows.
  - `i` is set to 0, then compared to 5.
  - $0 < 5$, so the statement `printf("hello\n");` is executed.
  - `i` is incremented to 1, and again compared to 5.
  - $1 < 5$, so the print statement is executed.
  - And so on, with $i = 0, 1, 2, 3, 4$. Then i is incremented to 5, 5 is not < 5, so the loop terminates and the program terminates.

TEMPLATE for a for-loop

```
for ( A; B; C )
{
  D;    // statement or group of statements.
}
```

```
Initialisation: A
Continuation condition: B
Action between steps: C

First do A (which can be blank)
if B holds, perform D then C (both can be blank)
if B still holds, perform D then C (both can be blank)
if B still holds, perform D then C (both can be blank)
and so on until B no longer holds.
```

Indentation. It is important that the code be readable. This is aided by indentation. A group of statements should be indented further than the curly braces, which should be level with the 'for.' A single statement should be indented further than the 'for.' (Indentation makes it easier to understand the program structure.)

We can have a single statement

```
printf("hello\n");
```

or a group of statements, each terminated by semicolon, and the group between braces — see below.

BEST PRACTICE. It is better to group statements between braces, even when there is only one.

Semicolons. There must be a semicolon after each statement, including the last in a group.

The symbol < means 'less than,' of course. Other relations include

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

Here is another example.

```
#include <stdio.h>

main()
{
  int i, j;

  for ( i=0; i<5; ++i )
  {
    for ( j=0; j<i; ++j )
      printf ( " " );

    printf( "hello\n");
  }
}
```

The output is

```
hello
 hello
  hello
   hello
    hello
```