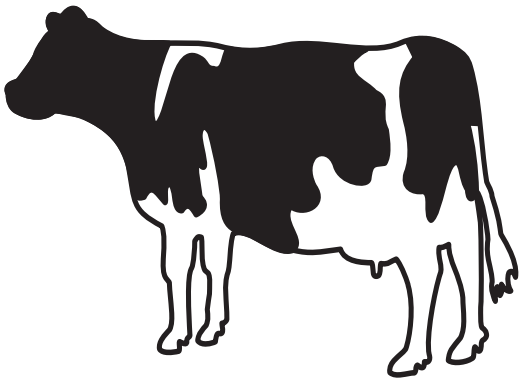
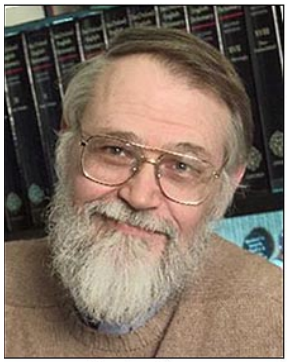
Week 2: Introduction to Programming in C
CS211: Programming and Operating Systems

Niall Madden

~~Wednesday and~~ Thursday, ~~22~~+23 Jan, 2020



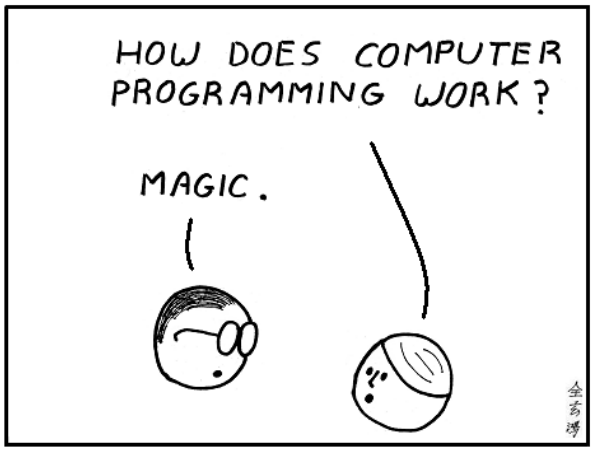


Brian Kernighan

Dennis Ritchie

This week's classes

1. Introduction
   - A little history
   - Books and Compilers
2. Course Content
3. Basic Structure
   - A simple example
   - A comment about comments
4. Variables
   - Variable names
   - Printing the value of a variable
5. Keywords
6. Operators
   - Arithmetic and Assignment
   - Relational and Logical Operators
7. Selection statements and loops
   - `if` statements
8. Exercises



Abstruse Goose: under the hood

To display the value stored in a variable, we use *printf*

02Variables.c

The k^{th} conversion character is matched with the k^{th} variable.

```
   int k=-101;
12 float f=1.23456;
   char c='a';
14 printf("Values of f, k, c are: %f, %d, %c\n",
           f, k ,c );
```

(1) (2) (3) — (1) (2) (3)

Explanation:

This displays the message

Values of f, k, c are: 1.23456, -101, a (newline)

%f, %d & %c are "conversion characters"

for, respectively, float, decimal integer, character

f d

In this example, we use an **array**

Example (Using `printf`)

```
#include <stdio.h>
int main(void )
{
  int Fib[3];
  Fib[0]=1; Fib[1]=1;
  Fib[2]=Fib[0]+Fib[1];
  printf("Fib[2] = %d\n", Fib[2]);
  return(0);
}
```

Declares an int array of length 3

Sets the values of the first 2 entries in the array.

Sets 3rd to be sum of first 2.

output

Explanation:

- To print a line of text: `printf("Hello world");`
- To print some text followed by a new line:
  `printf("Hello world\n");`
  Here `\n` is an example of an "escape character". Others include `\t` for a horizontal tab and `\a` for an "alert", i.e., a beep.
- `%d` is a ***conversion character***. It means "treat the next variable as an integer". Other important ones include: `%c` (a character), `%f` (a float), `%s` (a sting – i.e., and array of characters).

Keywords

In has a set of reserved keywords; they cannot be used as variable or function names:

`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

Some “new” ones, which may be supported by old compilers, include

`restrict` `_Bool` `_Complex` `_Imaginary`

Operators come in **four** flavours: ***Arithmetic***, ***assignment*** ***relational*** and ***logical***.

Arithmetic Operators available in C include:

Symbol	Definition	Example
+	addition	`c = a + b;`
-	subtraction	`c = a - b;`
*	multiplication	`c = a * b;`
/	division	`c = a / b;`
%	remainder	`c = a % b;`

Unlike Python, there isn't a built-in function for powers or truncating division.

Eg 13 % 10 Evaluates as 3.
So too does 13%5. 13%2 is 1

The Assignment and Arithmetic-Assignment Operators are:

Symbol	Definition	Example
=	assignment	`a=b;`
++	increment	`a++;`
--	decrement	`a--;`
+=	increment and assignment	`a+=2;`
-=	increment and assignment	`a-=2;`
*=	increment and assignment	`a*=2;`
/=	increment and assignment	`a/=2;`
%=	increment and assignment	`a%=2;`

a = a+1;

a = a-1;

a = a+2.

The following is legal, but not encouraged: `i=j=k=0` and is the same as `i = (j = (k = 0))`. Sets all of i, j, K to 0.

The operator `++` can be used in both ***prefix*** and `post-fix` form: in prefix form, the increment takes place before the value is used.

03Operators.c

```
int main(void)
{
  int i=1;
  printf("i++ = %d; ", i++);
  printf("++i = %d\n", ++i);
  i=1;
  printf("++i = %d; ", ++i);
  printf("i++ = %d\n", i++);
  return(0);
}
```

A **Relational Operator** tests if some relation holds between two quantities or variables, and evaluates as **true** or **false**.

Comparison.

Symbol	Definition
<	less than
<=	less than or equal to
>	
>=	
==	Equality
!=	inequality

Eq if x = 10 (assignment)

then x == 10 is true

(x%2) == 0 is also true.

These all evaluate as `0` for ***false*** or `1` for true.

04Logic.c

```
1  // 04Logic.c; For CS211, Jan 2019. NM
   #include <stdio.h>

   int main(void)
5  {
     int i=1, j=2;
7    printf("i=%d and j=%d\n", i, j);
     printf("i>j \t\t evaluates as %d\n", i>j);
9    printf("++i >= j \t evaluates as %d\n", ++i>=j);

11   return(0);
   }
```

7 outputs i=1 and j= 2.

8 outputs i>j evaluates as 0

9 outputs ++i " as 1 (since ++i sets i=2

Relational operators can be combined into more complex operators, as follows.

Symbol	Definition
!	not
&&	and
||	or.

See also Exercise on Slide 39

Eg, use as (a==5) && (a==c).

Selection statements and loops

To control the **flow** of a program, one uses

- **Selection Statements:** the main ones are `if` and `switch` – select a particular execution path. Also `?:`
- **Iteration statements:** `for`, `while` and `do`
- **jump statements:** `break`, `continue` and `goto`

Important: if, for

useful: while and/or do, switch

dangerous: goto

Finished here Thursday.

`if` statements are used to conditionally execute part of your code.

Structure:

```
if( exprn )
  {
    perform statements if exprn evaluates as
                  non-zero
  }
  else
  {
    statements if exprn evaluates as 0
  }
```

Exercises

Exercise (2.1)

*Suppose $x = 2$, $y = 3$ and $z = -5$. Write a C programme that check if the following statements are **true** or **false**.*

1. $(x > y) \vee (x < y)$.
2. $\big(x = (y - 1)\big) \wedge \big((y \leqslant x) \vee (y \leqslant z)\big)$.
3. $\neg(y \geqslant x - z) \vee (y \geqslant x + 1)$.

$\vee$ ~ "or"

$\wedge$ ~ "and"

$\neg$ ~ "not".