Programming in C

The lcc compiler – see handout

Functions in C – the main function

Functions are blocks of program that carry out a particular task – often useful to breakup task into parts.

All programs have 1 function called “main”

```
int main ( void )
{
        -
        -
    return 0;
}
```

Data variable types

see Tutorial p 16

Integers

short	0 - 255 (2^8)	±127
int *	0 - 65,535	±32767
long	0 - 4E9	± 2E9

Floating point, decimal

float	$\pm 10^{\pm 38}$	6 digits
double	$\pm 10^{\pm 308}$	15 digits

Text ?

* - for most desktop computers, the int type is the same as the long type

Array – group of variables with same name

Think of as a table with rows and columns

x[ ] – an array called x (column of numbers)

x[ ] [ ] – a 2D array (rows and columns)

x[0] – number in the first row of x
x[1] – number in the second row
x[2][3] – third row, fourth column

Storing text

char (stores a number from 0 – 255)
Words – strings of text – stored as an array of chars.

Declaring variables – before using a variable, C must know to set aside space.

```
int n;
unsigned int n;
float x;
float x[100];
char a[100];
```

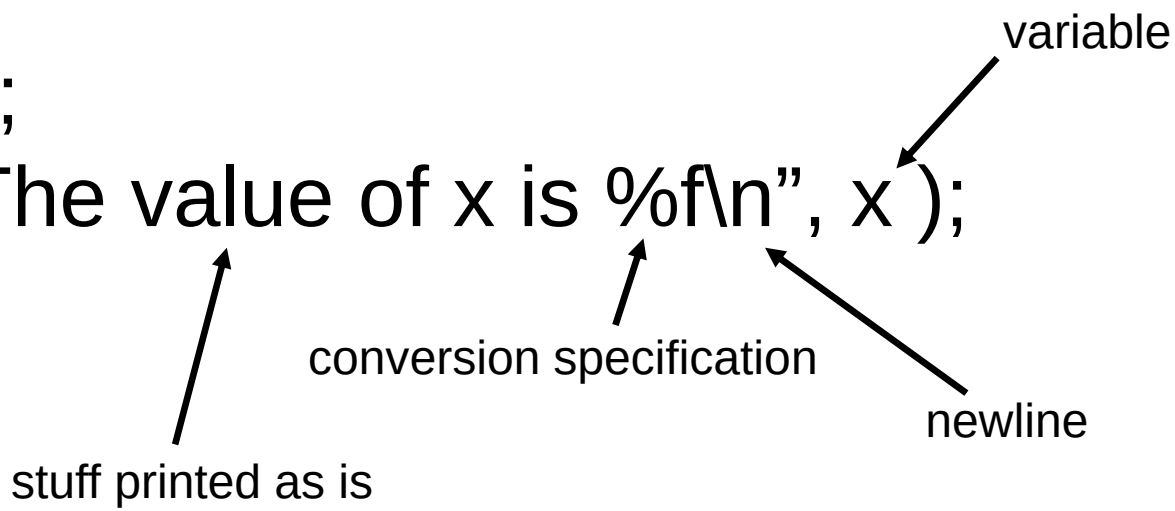
Program instructions (statements)

1) Output

printf

printf(format, variables, …);

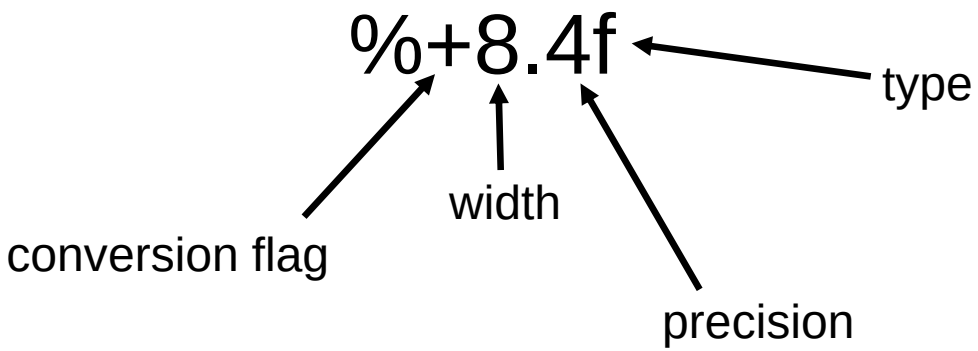
x = 1.23;
printf(“The value of x is %f\n”, x);



Conversion specifications

see tutorial.pdf
pp 57 - 60

float	%f
int	%i
string	%s



2) Input

```
scanf( format, variables, … );

printf( “\nEnter i : “ );
scanf( “%i”, &i );
printf( “Value entered = %i\n”, i );
```

3) Calculation (assignment)

```
x = 1.234;
```

Math operators : + - * /

```
x = x + 2;
```

Ok in programming !

x += 2;
x -= 2;
x *= 2;

x++; increment
x--; decrement

Operator precedence (priority)

$$\frac{x}{2y}$$

means x divided by product of 2 and y – order important!!

$$x/2*y \neq x/(2*y)$$

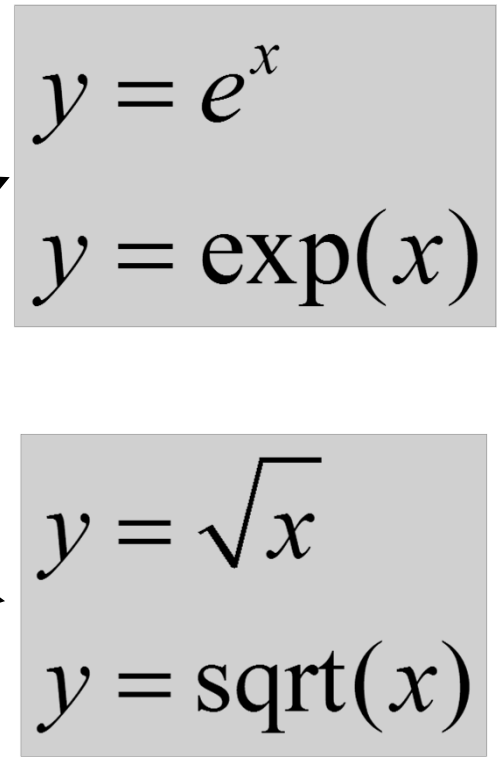
$$\frac{x}{2+y} = x/(2+y) \neq x/2+y$$

<u>precedence</u>:
( ) – do first
negation
* /
\+ -

see Tutorial p 56

Math functions

sin(x)
cos(x)
tan(x)
exp(x)
sqrt(x)

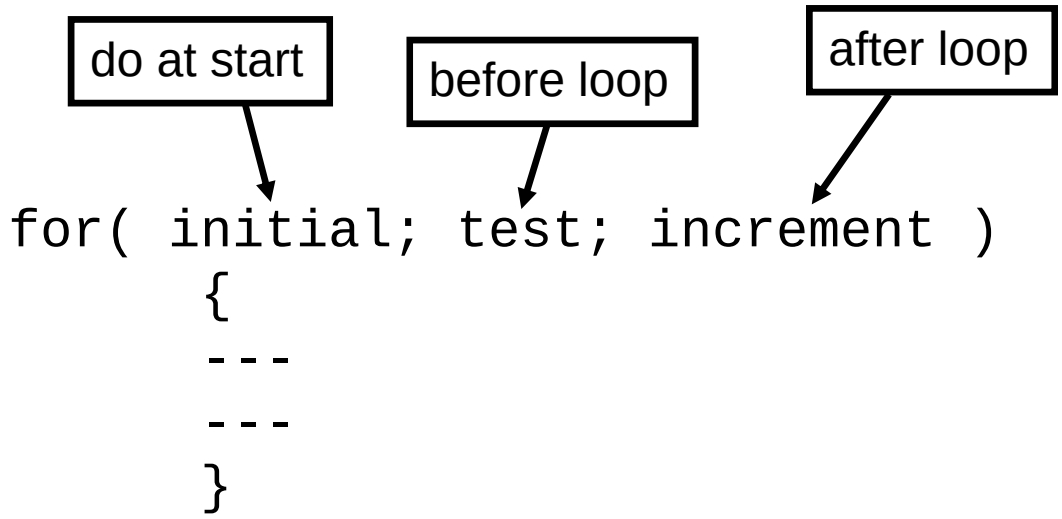


4) Program flow…

4) Program flow

see Tutorial.pdf
p 14

a) Loops (repeated sections)
for, do, while

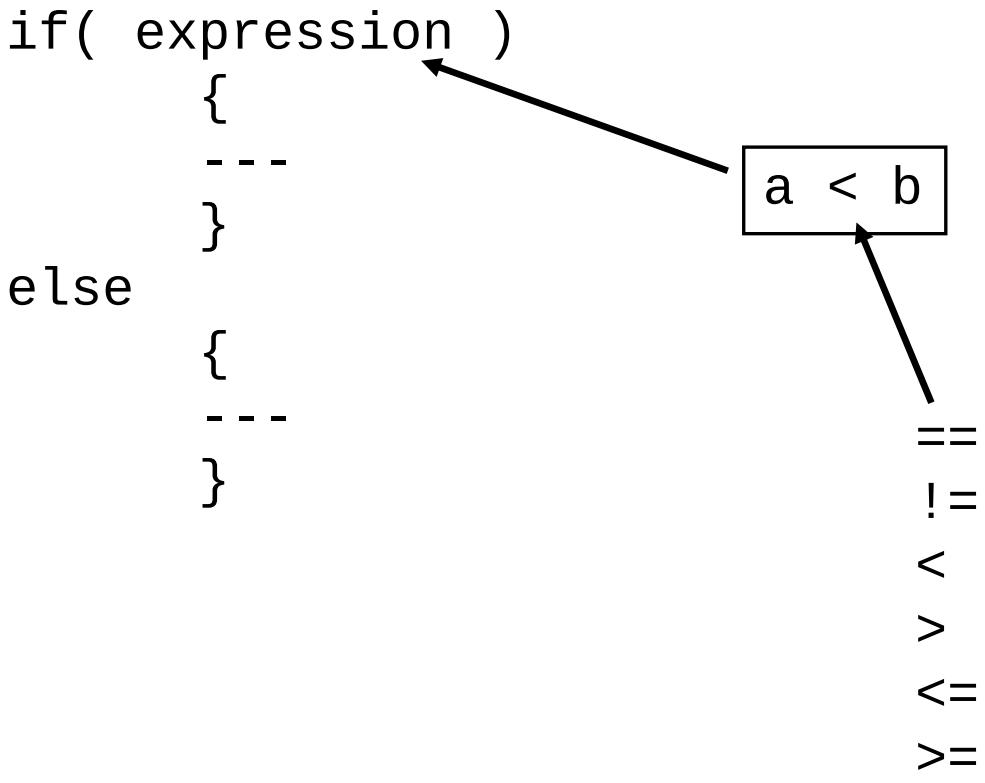


Want to set some values for x

```
for( i=0; i<5; i++ )
      {
      x[i] = 2.4;
      }
```

```
for( i=0; i<5; i++ )
      {
      x[i] = i;
      }
```

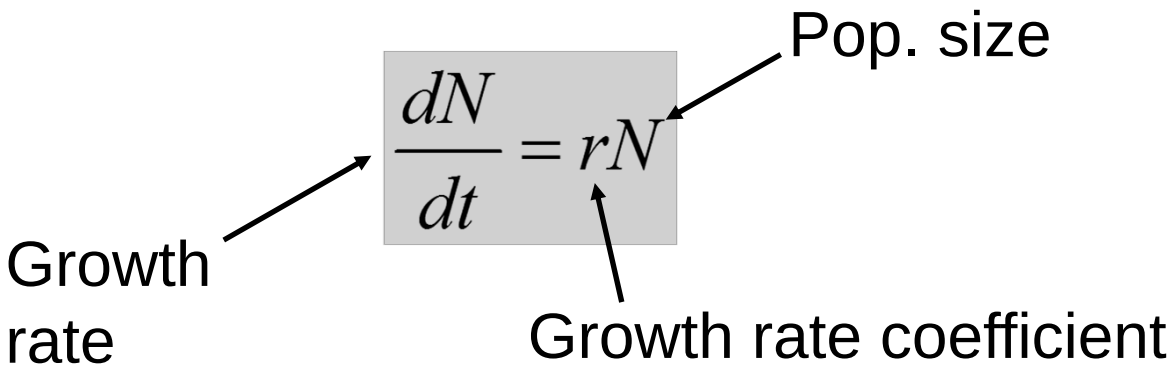
b) Conditional execution



Simple population growth model

Haefner
(Sec 2.3)

Simplest model says that growth rate of a populations is proportional to how large it is.



Like money!

Simplify by using a “finite difference” version

$$N_{t+1} = N_t + rN_t$$

How do we solve this model??

$$N_{t+1} = N_t + rN_t$$

Start at some initial time:

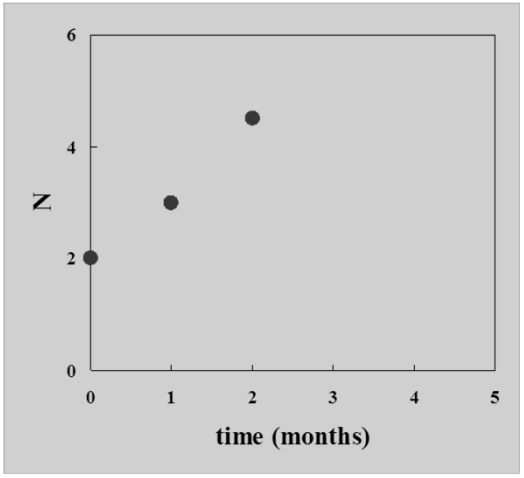
$t = 0, N = N_0$

Use the model equation to calculate N for future times…

$$N_0 = 2$$

$$N_1 = 2 + 0.5 \times 2 = 3.0$$

$$N_2 = 3.0 + 0.5 \times 3.0 = 4.50$$



Use computer program!

```
for( i=0; i<10; i++ )
     {
     N = N + r * N;
     t = t + 1;
     printf( “At time %i, N = %f\n, t, N );
     }
```

Before the loop:

```
t = 0;
N = 2;
r = 0.5;
```

Full program needs:

- header files
- main function
- declare all variables

```
/* First simple version of Exponential growth model */

#include <stdheaders.h>

int main( void )
{
        int i;
        float r, N, t;

        r = 0.5;
        t= 0;
        N = 2;

        for( i=0; i<10; i++ )
                {
                        N = N + r * N;
                        t = t + 1;
                        printf( "At time %f, N = %f\n", t, N );
                }
        return 0;
}
```

Alternate method using arrays

```
/* Exponential growth model
                - array method
*/

#include <stdheaders.h>

int main( void )
{
        int i;
        float r;
        float N[11], t[11];

        r = 0.5;
        t[0]= 0;
        N[0] = 2;

        for( i=0; i<10; i++ )     //This loop does the calculating
                {
                        N[i+1] = N[i] + r * N[i];
                        t[i+1] = t[i] + 1;
                }
        for( i=0; i<=10; i++ )    //This loop does the printing
                {
                        printf( "At time %4.1f, N = %7.4f\n", t[i], N[i] );
                }
        return 0;
}
```