CS23710 C Programming (and UNIX) Batch_Two

David Price
Computer Science

Decision or Conditional Statements

if (expression) statement

if (expression) statement1
else statement2

expression ≡ **FALSE** if zero
TRUE otherwise

else matches the closest elseless ***if***

Layout is purely conventional

Looping Control - the while statement

while (expression) statement

Evaluate the expression, if it is true
then execute the statement and loop back to
re-evaluate and reconsider the expression.
If the expression is false then give up.

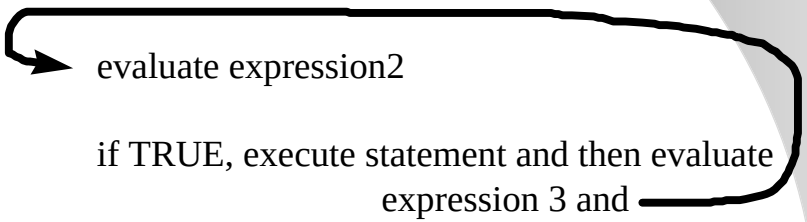
Looping Control - the do statement

do statement while (expression) ;

Execute the statement, evaluate the expression
and if it is TRUE then loop back and
execute the statement again etc..
If an evaluation of the expression yields a
FALSE result then give up.

Looping Control - the for statement

for (expression1 ; expression2 ; expression3) statement



evaluate expression1

evaluate expression2

if TRUE, execute statement and then evaluate expression 3 and

otherwise, give up

Note: can omit exp1, exp2 or exp3, exp2 assumed TRUE

while example

```
X = 1;
while ( X < 5 )
   { X = X + 1;
     printf ("X = %d\n", X);
   }
```

do example

```
X = 1;
do { X = X + 1 ;
     printf ("X = %d\n", X ) ;
   }
while ( X < 5 );
```

for example

```
for ( X = 1 ; X < 5 ; X ++ )

      printf("X = %d\n", X ) ;
```

Some Common Constructs

while (1) ……..

provides an endless loop

while (X != 0) ……

is the same as

while (X) ……

Input and Output

See: Kelley and Pohl Chapter 11

NOT `C', but library functions

Standard IO library

Often (normally) need to "include" various definitions from header files.

#include <stdio.h>

Simple Character I/O

putchar (ch) ;

ch = getchar () ;

Single character output or input (actually macros).
Note: ***ch*** is of type ***int*** in this example.
Note: at end of file, **getchar()** returns the value **EOF** which is defined in stdio.h

A common piece of code

```
while ( ( c = getchar() ) != EOF )
        {
        ……
        ……
        ……
        }
```

printf function

printf ("format string", arg1, arg2, arg3, ….);

The format string may contain normal characters which then get printed or conversion specifications

%d	next argument decimal integer
%o	octal
%x	hexadecimal
%c	single character
%s	string
%f	float or double
%%	an actual % character

modifying the output

You can insert various characters etc to gain more control over the output. These characters are inserted between the % and the format character.

- implies left aligned
+ implies insert plus or minus sign

field width and precision e.g.

%10.4f

h,l,d implies short, long or long double

scanf function

scanf ("format", arg1, arg2, arg3, …);

The arg1, arg2 and so on must refer to the addresses of places where the input acquired can be placed. “format” is similar to printf, see Kelley and Pohl page 499 for details or on-line man pages.

scanf example - Ammeraal p157

Example:

```
printf("Enter an integer:");
scanf("%d", &n);
printf("Enter a character:");
scanf(" %c", &ch);
```

Note: this space is to cause a skip of the newline pressed after the integer.