C Programming - Spring03

Course Web Page

http://www.cs.helsinki.fi/u/narayana/cprog03.html

Text book

The C programming Language , Second Edition,
Kernighan & Ritchie, Prentice Hall, 1988
(we refer to this book as K&R)

C - An Overview

C is a general purpose programming language which features economy of expression, modern control flow and data structures , and a rich set of operators.

C is not a 'very high level' language, nor a 'big' one and is not specialized to any particular area of application.

But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages.

- from K&R , Preface to the First Edition (1978)

C - Quotes

C is not a big language and is not well served by a big book

- from K&R , Preface to the Second Edition (1988)

C is quirky, flawed, and an enormous success

- Dennis Ritchie

C is a razor sharp tool, with which one can create an elegant and efficient program or a bloody mess

- Kernighan & Pike in 'The Practice of Programming'

Learn from the Masters

- Niel Henrik Abel

The First C Program - 'hello, world'

```
#include <stdio.h>

main()
{
    printf("hello, world\n ");

}
```

produces the famous output:

```
hello, world
```

- K&R Page 7

Concepts in 'hello, world'

- Structure of a C program
- Role of main() in a C program
- Preprocessor
- Standard Library for Input/Output
- Role of braces i.e. {}
- printf function
- passing arguments to functions
- String Constants
- Escape sequences , e.g. \n for newline character
- Semicolon (;) as statement terminator.

Explanation of 'hello, world'

A program is a collection of functions. The general form of a function definition is

```
type_name function_name (argument_list)
declaration_of_argument_types
function_body
```

This program defines the function main(). Here the function main() takes no arguments. When a program is run, main() is the first function called. Every C program should include a main() function that may or may not have arguments.

Here main() invokes the function printf which takes the argument "hello, world\n" , a string constant.

A string constant is written within double quotes (" ")

A Cousin of the First C Program

```
#include <stdio.h>

main()
{
    printf("hello,  ");
    printf("world");
    printf("\n ");
}
```

produces identical output:

```
    hello, world
```

- K&R Page 8

F-C Program - Version 1

```
 #include <stdio.h>
/* print Fahrenheit-Celsius table
     for fahr = 0, 20, ..., 300 */
 main()
{
        int fahr, celsius;
        int lower, upper, step;

        lower  =  0;              /* lower limit of the temp. table*/
        upper  =  300;   /* upper limit  */
        step  =  20;         /* step size */

        fahr  =  lower;
        while (fahr  <=  upper)  {
    celsius  =  5  *  (fahr-32)  /  9;
                printf("%d\t%d\n",  fahr,  celsius);
                fahr  =   fahr  +  step;
        }
}
```

- K&R Page 9

Concepts in F-C version 1

- Comment in a program
- Decalaration of variables
- Statements
- Assignment statement
- Compound statement
- Looping construct while
- Arithmetic expressions in C
- Integer division vs Real division
- formatting the output

Explanation of F-C version 1

Comments appear within /* comment */ and can run across many lines.

Statements are either simple or compound. Simple statements are composed of expressions including function calls and they must be terminated by a semicolon. The simplest statement is the null statement which is just a semicolon , i.e, ;

A compound statement is a sequence of variable declarations followed by statements, all surrounded by braces ({}). A compound statement can appear wherever a simple statement can appear in a C program.

Statements are executed in the sequence in which they appear, starting with the first statement in main()

Explanation of F-C version 1

Variables hold the values that are manipulated by a program.
All variables in a C program must be declared before their use.

Variables are user defined names that take values corresponding to their type.

A type specifies the set of values that a variable can take and the operations that can be performed on them.

A variable declaration associates the name of a variable to its type.

```
type variable_name
 int a;
```

In C one or more variables can be declared in the same declaration.

```
int  i, j, k;
```

Explanation of F-C version 1

Basic types in C are char , int, float.

char - 8 bits long (1 byte)
int - 16 bits / 32 bits
float - 32 bits

The sizes of these objects are machine dependent.

short , long , unsigned are variants of int
double is a variant of float

Explanation of F-C version 1

Assignment statement assigns a value to a variable.

```
i = 0;
i = i + 1;
i = j;
i = factorial(3);
i = j = k = 0;
```

the last one is one of the surprises of the C language which can be used to advantage in C programs

Assignments are expressions in C, an assignment yields a value corresponding to the variable on the left hand side of the assignment statement.

Explanation of F-C version 1

Conditional Statement

A conditional statement selects one of two alternative statements for execution. It has the form

```
if expression
    statement_1
else
    statement_2
```

A variant of this is

```
if  expression
    statement
```

with no else part

When conditionals are nested use the form

```
if  ...
else if ...
else if ...
```

Explanation of F-C version 1

A conditional statement selects one of two alternative statements for execution. It has the form

```
if expression
        statement_1
    else
        statement_2
```

```
if ( x <= 0 )
        x = -x;
    else
        x = x;
```

A variant of this is

```
    if expression
        statement        ( with no else part )
    if ( x <= 0 )
        x = -x;
```

Explanation of F-C version 1

Nested if

```
if  (condition_1)
        statement_1
else if (condition_2)
        statement_2
...
        ...
else
        statement_n
```

Any number of else if can appear between initial if and final else

Explanation of F-C version 1

while statement:

while condition

statement

```
i = 0;
while ( i <= 100 ) {
    printf ( "%d\n", i);
    i++;    /* increment i  i.e., i = i + 1*/
}
```

causes the numbers 0 to 100 to be printed one number in each line.

In while statement the condition is checked before the statement is executed. After each execution of the statement the loop condition is tested again. If the condition is true, then the statement is executed again. If the condition is false the execution of the while statement terminates and control goes to the next statement in the program.

Note that if the condition is false initially the statement is never executed. So the statement in the body of the while loop is executed zero, one or more times.

Explanation of F-C version 1

for statement:

```
for ( expression_1;  expression_2;  expression_3)
        statement
```

Here the expression_1 is the initialization of the loop, expression_2 is the loop condition and expression_3 is the loop increment.

```
for  ( i  =  0;  i  <=  100;  i++  ) {
        printf ( "%d\n", i);
}
```

causes the numbers 0 to 100 to be printed one number in each line.

The above for statement is equivalent to the while statement discussed before. Notice that the loop control variable (i in the example) is in one place.

'for' is preferable to 'while' when initialization and increment are single statements and are logically related. (K&R page 14)

F-C Program - Version1- Result

0	-17
20	-6
40	4
60	15
80	26
100	37
120	48
140	60
160	71
180	82
200	93
220	104
240	115
260	126
280	137
300	148

- K&R Page 8

F-C Program - Version 2

```
 #include <stdio.h>
/* print Fahrenheit-Celsius table
     for fahr = 0, 20, ..., 300 ;  floating point version */
 main()
{
        float fahr, celsius;
        int lower, upper, step;

        lower  =  0;              /* lower limit of the temp. table*/
        upper  =  300;   /* upper limit  */
        step  =  20;         /* step size */

        fahr  =  lower;
        while (fahr  <=  upper)  {
    celsius  =   (5.0/9.0)  *  (fahr-32.0) ;
                printf("%3.0f  %6.1f\n",  fahr,  celsius);
                fahr  =   fahr  +  step;
        }
}
```

- K&R Page 12

F-C Program - Version 2 - Result

```
  0       -17.8
 20        -6.7
 40         4.4
...
```

- K&R Page 13

F-C Program - Version 3

```
#include <stdio.h>
/*  print Fahrenheit-Celsius table  */
 main()
{
        int fahr;

        for (fahr = 0; fahr <= 300; fahr = fahr + 20)
                printf("%3d  %6.1f\n", fahr, (5.0/9.0) * (fahr-32));

}
```

produces same results as version 2

- K&R Page 13

File Copying - Version 2

```
#include <stdio.h>

/* copy input to output; 2nd version */

 main()
{
        int     c;

        while  ((c  =  getchar())  !=  EOF)
                putchar(c);
}
```

- K&R Page 17

Idiom

```
while ((c = getchar()) != EOF)
        ...
```

Character Count - Version 1

```
#include <stdio.h>

/*  count characters in input;  1st version  */
 main()
{
        long  nc;

        nc  =  0;
        while  ( getchar()  !=  EOF)
                ++nc;
        printf( "%ld\n",  nc);
}
```

- K&R Page 18

Character Count - Version 2

```
#include <stdio.h>

/*  count characters in input;  2nd version  */
 main()
{
        double      nc;

        for  ( nc  =  0;  getchar()  !=  EOF;  ++nc)
                ;
        printf( "%.0f\n",  nc);
}
```

- K&R Page 18

Line Count

```
#include <stdio.h>

/*  count lines in input  */
 main()
{
        int    c,  nl;

        nl  =  0;
        while  ((c  =  getchar())  !=  EOF)
                if  (c  ==   '\n' )
                        ++nl;
        printf( "%d\n",  nl);
}
```

- K&R Page 19

Word Count

```
#include <stdio.h>
#define     IN          1       /*  inside a word  */
#define     OUT         0       /*  outside a word  */
/*  count lines, words, and characters  in input  */
 main()
{
        int     c,  nl,  nw,  nc,  state;
        state  = OUT;
        nl  =  nw  =  nc  =  0;
        while  ((c  =  getchar())  !=  EOF) {
                ++nc;
                if  (c  ==   '\n' )
        ++nl;
                if  ( c  ==  ' '  ||  c  ==   '\n'  ||  c  ==  '\t' )
                        state  =  OUT;
                else  if  (state  ==  OUT)  {
                        state  =  IN;
                        ++nw;
                }
        }
        printf( "%d  %d  %d\n",  nl,  nw,  nc);
}
```

- K&R Page 20

Suggested Reading

Programming in C: A tutorial, B.W. Kernighan
http://www.lysator.liu.se/c/bwk-tutor.html
this vintage introduction to C dating back to 1974 has a timeless appeal; a must read

The Development of the C Language , D. Ritchie
http://cm.bell-labs.com/cm/cs/who/dmr/chist.html
get an inside view of C from its creator;enjoy!

Exercises

I hear and I forget;
I see and I remember;
I do and I understand.
- Confucius (551-479 BC)

The road to wisdom is plain and simple to express:
Err and err again, but less and less and less.
- Piet Hein (1905-1996)

Exercises

Each exercise will contain about 6 questions. You can find the answers by carefully reading K&R.

The questions fall into different categories. Some test your understanding of the concepts, some your ability to read programs with care and some your skills to apply the ideas to write simple programs.

Exercise-1 will be in the course webpage after today's lecture.