CS23710 / CSM2510
C Programming

David Price (and Friends)
Computer Science

C History

- Starts early 70's "for" Unix
- "The C Programming Language" by Kernighan and Richie - 1978
- 1983 - ANSI - forms X3J11 which leads towards establishment of ANSI C

'C' Program

- One or more functions
- One function called 'main'
- contained in one or multiple files
- powerful
- easy to write programs which are syntactically correct but **not** what the programmer intended
- But you can write **good** 'C'

```
/* This is a first C Program */

main ()
{
int a, b, c;   /* define three integer variables */

      a = 4;  b = 5;  /* give them values */

      c = a + b;      /* add them up */

      /* print out some answers */

printf("The sum of %d and %d is %d", a, b, c);
}
```

Using ‘C’

- Create source files using favourite editor
- Sources files called *.c
- compile and link by typing …..

```
gcc  fred.c
```

this creates an executable file called **a.out**

```
gcc  fred.c -o fred
```

this creates an executable file called **fred**

‘C’ Grammar Rules

- Identifiers - letters, digits & underscores, (first not digit)
- At least 31 characters significant for internal identifiers
- Reserved Keywords - auto, break, case, char, const, continue, default, …….. (see Ammeraal page 10)

Constants

- INTEGER
  - 527 decimal
  - 025 octal
  - 0xB3A hexadecimal
  - L suffix => long
  - U suffix => unsigned

Character Constants

- Character ---- enclosed in single quotes
- all have type ----

int

- **‘B’**

(value is the internal representation, likely to be ASCII => value is 66.)

Constants

- Character
- Special characters - escape sequences etc.

- **'\n'** **newline**
- **'\r'** **carriage return**
- **'\t'** **tabulate**
- **'\b'** **backspace**
- **'\f'** **formfeed**
- **'\0'** **null**
- **'\●●●'** **octal**
- **'\x●●'** **hexadecimal**

Floating Constants

Real Numbers ….

85.37

5E4 ==> $5 * 10^4 = 50000$

All of type **DOUBLE**

Suffix F ==> (single) float
Suffix L ==> long double

String Constants

"Hello Fred"

Note: "Z" is not the same as 'Z'

Strings have NULL ('\0') added to the end

You include " by using \"

"Hello \" whats this \""

Note:

"hello" " fred" is equivalent to

"hello fred" (new ANSI feature)

Comments

```
/*   This is all
        a comment
     now
*/
```

NOTE: cannot have nested comments

Expressions and Statements

- all normal arithmetic operators
-
- integer and real arithmetic

3 / 4 is not equal **3.0/4.0**

Integer Real

I.e. / is the symbol for two operators

Expressions

(3 * 4.0) + 7.5 / (3.1 - 2.7)

4.7

C * D - 5.7

X = Y * 3.9

Note = is an operator

Are ALL expressions …..

EXPRESSION ;
is a statement

Composite Operators

X = X + Y

can be written as

X += Y

–= *= /= %=

are all allowed too

Increment and Decrement

X++ is same as X = X + 1

Can Have

X++ Use X and then increment
X-- Use X and the decrement
++X Increment X and then use
--X Decrement X and then use

Types and typical sizes

● char		1 byte
● short	(short int)	2
● int		2 or 4
● enum	(new ansi type)	2 or 4
● long	(long int)	4
● float		4
● double		8
● long double		8 or 10

Other Types

Can prepend unsigned to char, short, int, long

sizeof allows us to get the size

Initialization

Can define a variable and specify an inital value

char b = 'g';

int k=5, mon = 3*4;

Note: constant fixed at compile time

Enumeration types (ANSI)

```
enum fruit
        { APPLE, PEAR, GRAPE, ORANGE
        } breakfast_fruit, dinner_fruit;
```

Type name is **enum fruit**
Values **APPLE, PEAR, GRAPE, ORANGE**
Variables **breakfast_fruit, dinner_fruit**

New variable

```
enum fruit  fruit_piece;
```

Note: compatible with integer. Values as specfied above are
0 (APPLE), 1 (PEAR), 2 (GRAPE), 3 (ORANGE)
Can specify other values : see Ammeraal pages 23-24

Type Qualifiers

register int A1;

==> hold A1 in a hardware register
==> fast usage
==> cannot ask for the address of ….
Limited number of registers available

const char c = 'z' ;
==> variable cannot be changed

volatile int x;
==> hardware or operating system might change the value

Comparison and Logical Operators

< > <= >=

== != && || !

Note: == is not =

5 == 7 result FALSE that is **0**

(3 + 2) == 5 result TRUE that is **1**

Potential Pitfall !

Y = 7 ; /* set Y equal to 7 */

Y == 7; /* compare Y with 7 which will generate a result equal to 1 or 0 and then discard it ! */

More Statements

Null statement …. just a

;

Complex statement

```
{   expression;
    expression;
    expression;
}
```

Compound Statement;