CSSE2010

CSSE2010/CSSE7201
Lecture 10B

Introduction to Embedded C Programming

School of Information Technology and Electrical Engineering
The University of Queensland

C Programming Introduction

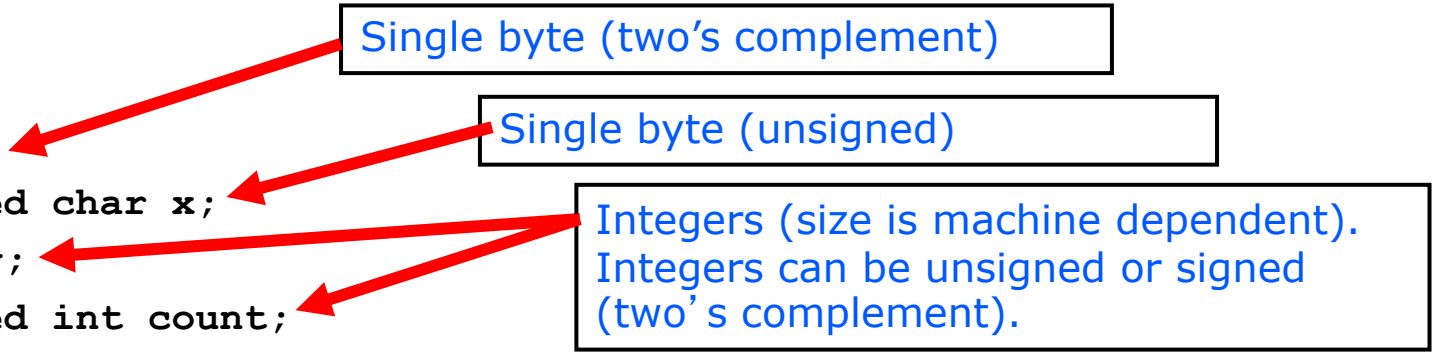
- In this course we expect you to…
  - be able to write small C programs and functions from scratch
  - understand the meaning of C programs
  - be able to modify C programs
- Lectures can’t teach programming
- You’ll need to practice
  - Learning Lab 9 continues the introduction to C programming, but you’ll need to practice!
  - C tutorials and other resources available on Blackboard

C Program – Basic Structure

- There must be a function called **`main`**
  - This function is executed when program starts
- Blocks of code enclosed by braces { }
- C statements must end with a semicolon ;
- C statements are case sensitive
  - **`variable`** is not the same as **`Variable`**
- Comments are
  - within /* … */ or
  - from // to end of line

C - Declaring Variables

- Declaration
  - `type-name variable-name, variable-name …;`
- Examples
  - `char c;`
  - `unsigned char x;`
  - `int day;`
  - `unsigned int count;`



- **`char`**, **`int`**, **`float`**, **`double`** are among data types supported by C
- Any value can be treated as a boolean
  - zero means false, non-zero means true

Function Example

```
/*
    Return the average of two integers
   (result will be rounded towards 0)
*/
int average(int a, int b) {
    int avge;

    avge = (a+b)/2;

    return avge;
}
```

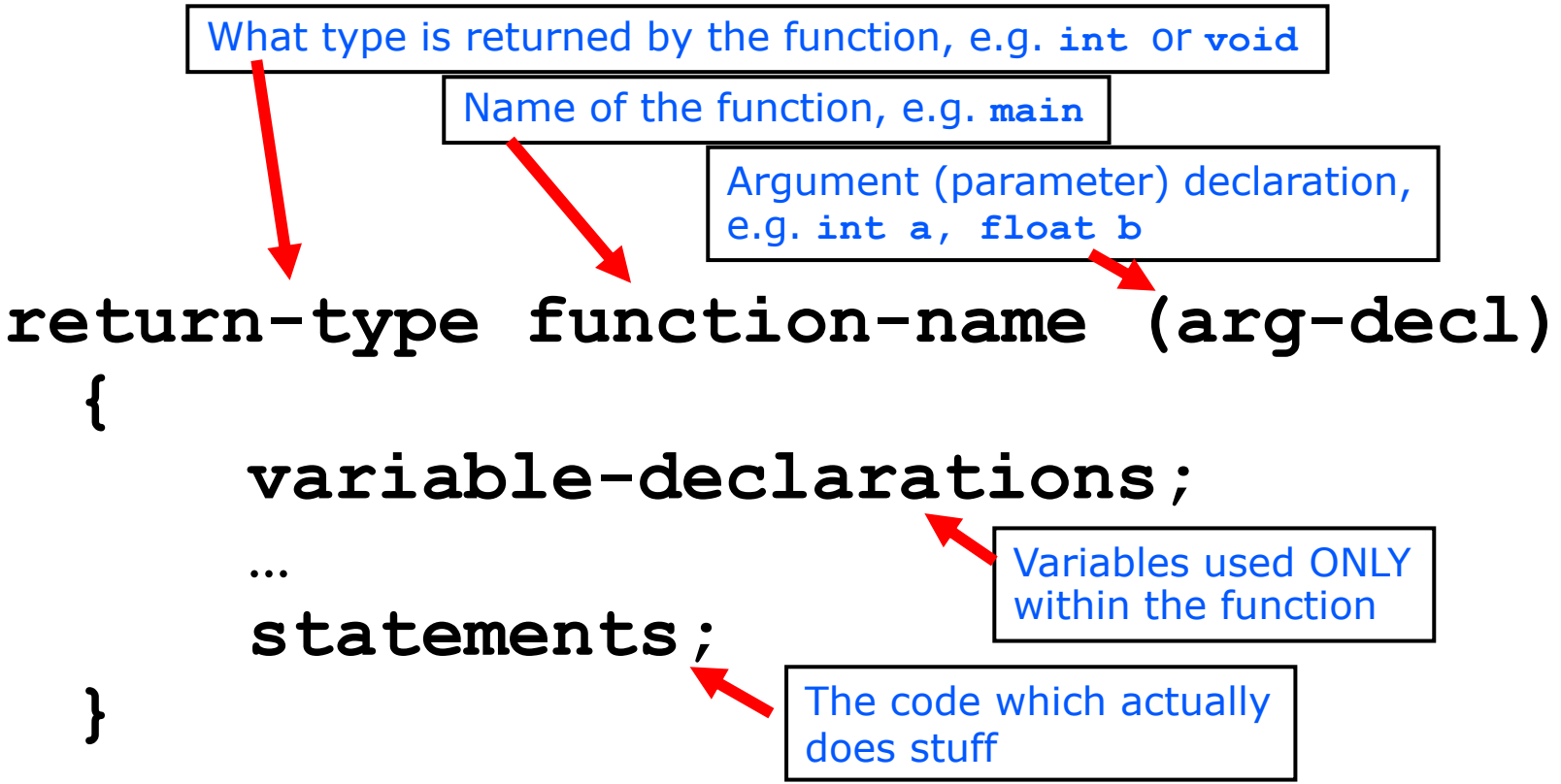
Statements

This is an **assignment** operator

These are **expressions.** Outer expression is a statement.

The function **returns** a result when finished

Function Definitions



C Constants

- **Character constants**
  - Use single quotes, e.g. 'a', 'b', '1' etc
  - Some special characters – **backslash escaped**
    - '\n' = newline, '\'' = single quote, '\t' = tab, '\\' = backslash
- **String constants**
  - Use double quotes (can include backslash escapes)
    e.g. "abc \n \" hello\t"
- **Integer constants**
  - Decimal – e.g. 3 , -27 , 65535 , +5
  - Hexadecimal (leading 0x), e.g. 0x5F , 0xFFFF , 0xDEADBEEF
  - Octal (leading 0), e.g. 0377 (= 255 decimal)
- **Floating point constants**
  - Include decimal point (.) and/or "e" for exponent
  - Examples: 3.1416 , -7. , 6.02e23 , -5.2e-2
  - Note 7 is an integer, 7. is floating point

Some C Operators

- Binary operators

+	addition
–	subtraction
*	multiplication
/	division
%	remainder (integer)
>	greater than
>=	greater than or equal
==	equal
!=	not equal
<	less than
<=	less than or equals
&	bitwise AND
|	bitwise OR
^	bitwise XOR
&&	logical AND
||	logical OR
=	assignment

- Unary operators

!	logical not
~	one's complement (invert)
–	two's complement (negate)
++	increment (prefix or postfix)
– –	decrement (prefix or postfix)

More Operators: Bit-shifting and assignment

- `a << b` means a shifted left by b bits
- `a >> b` means a shifted right by b bits
- `a = b` means a is assigned the value of b
- `a += b` is shorthand for `a=a+b`
- Similarly `-=, *=, /=, %=, &=, |=, ^=, <<=, >>=`
- Examples

  `1 << 5` is $1 * 2^5 = 32$

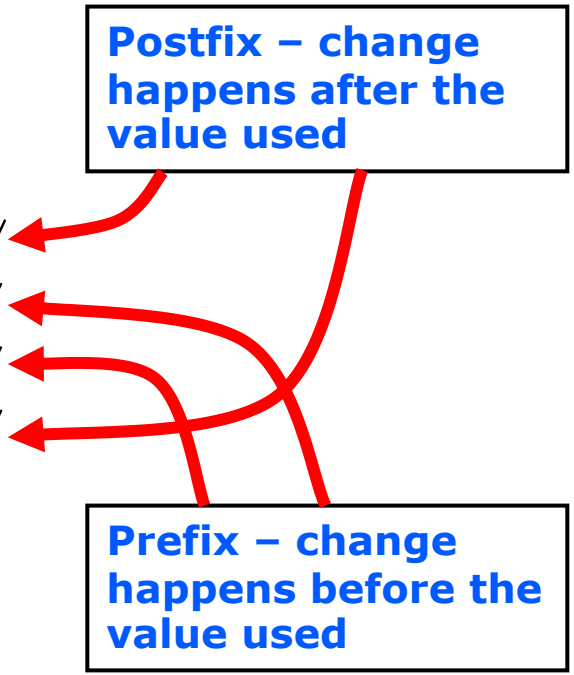
  `3 << 4` is $3 * 2^4 = 48$

  `a += 1` same as a++

Postfix/Prefix Increment and Decrement

- Example:

```
int a,b,c,d,e;
a = 4;
b = a++;      /* b=a; a=a+1; */
c = --a;      /* a=a-1; c=a; */
d = ++a;      /* a=a+1; d=a; */
e = a--;      /* e=a; a=a-1; */
```



- After these statements, values are
  a=4, b=4, c=4, d=5, e=5

What value will b have after this C code is executed?

```
a = 1;
b = 3;
b += a++;
```

10

if Statement

- **`if (`***`expression`***`)`** *`stmt`* **`else`** *`stmt`*

- **`else`** clause is optional
- Note on expressions:
  - C interprets any 0 value as false, anything else as true
  - **`if(a)`** means **`if(a != 0)`**
  - **`if(!a)`** means **`if(a == 0)`**
- *stmt* can be replaced by multiple statements enclosed in braces { }