

Chapter 13 Control Structures

Based on slides © McGraw-Hill
Additional material © 2004/2005 Lewis/Martin
Modified by Diana Palsetia

CIT 593

1/19

Control Structures

Conditional

- Making decision about which code to execute, based on evaluated expression
- `if`
- `if-else`
- `switch`

Iteration

- Executing code multiple times, ending based on evaluated expression
- `while`
- `for`
- `do-while`

CIT 593

2/19

Example If Statements

```
if (x <= 10)
    y = x * x + 5;
```

Style: avoid singleton
if statements
(Often cause of errors)

```
if (x <= 10)
    y = x * x + 5;
z = (2 * y) / 3;
```

only first statement is conditional;
second statement is
always executed

```
if (x <= 10) {
    y = x * x + 5;
    z = (2 * y) / 3;
}
```

compound statement;
both executed if `x <= 10`

CIT 593

3/19

More If Examples

```
if (month == 4 || month == 6 ||
    month == 9 || month == 11) {
    printf("The month has 30 days.\n");
}
```

```
if (x = 2) {
    y = 5;
}
```

- This is a common programming error (`=` instead of `==`), not caught by compiler because it's syntactically correct.

CIT 593

4/19

Generating Code for If Statement

```

if (x == 2) {
    y = 5;
}

LDR R0, R5, #0 ; load x into R0
ADD R0, R0, #-2 ; subtract 2
BRnp NOT_TRUE ; if non-zero, x is not 2
AND R1, R1, #0
ADD R1, R1, #5
STR R1, R5, #1 ; store 5 to y

NOT_TRUE ... ; next statement
    
```

For now assume that addr of x is R5 and addr of y is R5 + 1, it will be clear in chapter 14, why R5 contains addr of x.

CIT 593

5/19

Generating Code for If-Else

```

if (x) {
    y++;
    z--;
}
else {
    y--;
    z++;
}

LDR R0, R5, #0
BRz ELSE ; x is not zero
LDR R1, R5, #1
ADD R1, R1, #1 ; incr y
STR R1, R5, #1 ; store y++
LDR R1, R5, #2 ; decr z
ADD R1, R1, #-1
STR R1, R5, #2 ; store z--
BR DONE ; skip else code ; x is zero
ELSE LDR R1, R5, #1 ; decr y
ADD R1, R1, #-1
STR R1, R5, #1 ; store y--
LDR R1, R5, #2 ; incr z
ADD R1, R1, #1
STR R1, R5, #2 ; store z++
... ; next statement
    
```

R5 = addr of x
 R5 + 1 = addr of y
 R5 + 2 = addr of z

CIT 593

DONE

6/19

Matching Else with If

Else is always associated with closest unassociated if

```

if (x != 10)
    if (y > 3)
        z = z / 2;
    else
        z = z * 2;
    
```

is the same as...

```

if (x != 10) {
    if (y > 3){
        z = z / 2;
    }
    else{
        z = z * 2;
    }
}
    
```

is NOT the same as...

```

if (x != 10) {
    if (y > 3)
        z = z / 2;
}
else
    z = z * 2;
    
```

**Solution: always use braces
 (avoids the problem entirely)**

CIT 593

7/19

Chaining If's and Else's

```

if(month == 4 || month == 6 || month == 9 || month
== 11) {
    printf("Month has 30 days.\n");
}
else if (month == 1 || month == 3 ||
month == 5 || month == 7 ||
month == 8 || month == 10 ||
month == 12) {
    printf("Month has 31 days.\n");
}
else if (month == 2) {
    printf("Month has 28 or 29 days.\n");
}
else {
    printf("Don't know that month.\n");
}
    
```

CIT 593

8/19

Generating Code for While

```

x = 0;
...
while (x < 10) {
    printf("%d ", x);
    x = x + 1;
}

```

```

AND R0, R0, #0
STR R0, R5, #0 ;x=0
...
; test
LOOP LDR R0, R5, #0 ;load x
ADD R0, R0, #-10
BRzpb DONE
; loop body
...
<printf>
...
LDR R0, R5, #0 ;load x
ADD R0, R0, #1 ;incr x
STR R0, R5, #0
BR LOOP ;test again

DONE ; next statement

```

CIT 593

9/19

Infinite Loops

The following loop will never terminate:

```

x = 0;
while (x < 10) {
    printf("%d ", x);
}

```

Loop body does not change condition...

- ...so test is never false
- Common programming error that can be difficult to find

Sometimes you might start out with true case:

```

int x = 1;
while(x){
    //remember that there is no boolean type in C
    //make x to 0, so the loop terminates
}

```

CIT 593

10/19

Generating Code for For

```

for (i = 0; i < 10; i++) {
    printf("%d ", i);
}

```

```

; init
AND R0, R0, #0
STR R0, R5, #0 ;i=0
; test
LOOP LDR R0, R5, #0 ;load i
ADD R0, R0, #-10
BRzpb DONE
; loop body
LDR R0, R5, #0 ;load i
...
<printf>
...
; re-init
ADD R0, R0, #1 ;incr i
STR R0, R5, #0
BR LOOP ;test again

DONE ; next statement

```

This is the same
as the while example!

CIT 593

11/19

Example For Loops

```

/* -- what is the output of this loop? -- */
for (i = 0; i <= 10; i++) {
    printf("%d\n ", i);
}
/* -- what does this one output? -- */
char letter = 'a'; int c;
for (c = 0; c < 26; c++) {
    printf("%c\n", letter+c);
}
/* -- what does this loop do? -- */
int numberOfOnes = 0; int bitNum;
for (bitNum = 0; bitNum < 16; bitNum++) {
    if (inputValue & (1 << bitNum)) {
        numberOfOnes++;
    }
}

```

CIT 593

12/19

For vs. While

In general:

For loop is preferred for **counter**-based loops

- Explicit counter variable
- Easy to see how counter is modified each loop

While loop is preferred for **sentinel**-based loops

- E.g. eof = end of file or \n = newline character

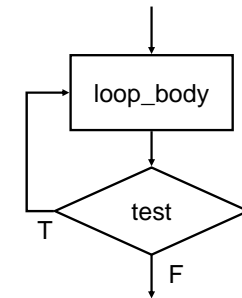
Note: Either kind of loop can be expressed as other, so its really a matter of style and readability

CIT 593

13/19

Do-While

```
do
    loop_body;
while (test);
```



Executes loop body as long as test evaluates to TRUE (non-zero).

Note: Test is evaluated **after** executing loop body

CIT 593

14/19

Break and Continue

break

- used *only* in switch statement or iteration statement
- passes control out of the "nearest" (loop or switch) statement containing it to the statement immediately following
- usually used to **exit a loop before terminating condition occurs** (or to exit switch statement when case is done)

continue;

- used *only* in iteration statement
- terminates the execution of the loop body for that iteration
- loop expression is evaluated to see whether another iteration should be performed
- if for loop, also executes the re-initializer

CIT 593

15/19

Example

What does the following loop do?

```
for (i = 0; i <= 20; i++) {
    if (i%2 == 0) {
        continue;
    }
    printf("%d ", i);
}
```

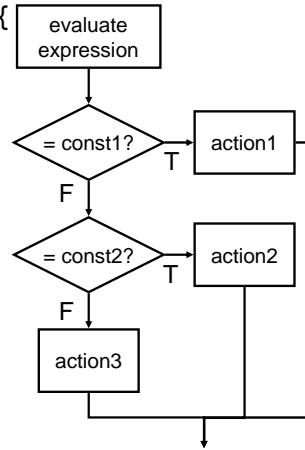
- What would be an easier way to write this?
- What happens if **break** instead of **continue**?

CIT 593

16/19

Switch

```
switch (integer_expression) {  
  case const1:  
    action1;  
    break;  
  case const2:  
    action2;  
    break;  
  default:  
    action3;  
}
```



Alternative to long if-else chain.
If break is not used, then
case "falls through" to the next.

CIT 593

17/19

Switch Example

```
/* same as month example from cascaded if-else */  
switch (month) {  
  case 4: case 6: case 9: case 11:  
    printf("Month has 30 days.\n");  
    break;  
  
  case 1: case 3:  
    /* some cases omitted for brevity... */  
    printf("Month has 31 days.\n");  
    break;  
  
  case 2:  
    printf("Month has 28 or 29 days.\n");  
    break;  
  
  default:  
    printf("Don't know that month.\n");  
}
```

CIT 593

18/19

Enumerations

Keyword **enum** declares a new type that take on symbolic values

- enum colors { RED, GREEN, BLUE, YELLOW, MAUVE };
- RED is now 0, GREEN is 1, etc.
- Gives meaning to constants, groups constants

```
enum colors house_color; //Note: colors house-color is illegal in C  
house_color = get_color();  
switch (house_color) {  
  case RED:  
    /* code here */  
    break;  
    /* more here... */  
}
```

Enums are just ints, but can provide more type checking

- C++ adds even more checking support

CIT 593

19/19