Mc Graw Hill Education

Because learning changes everything.®

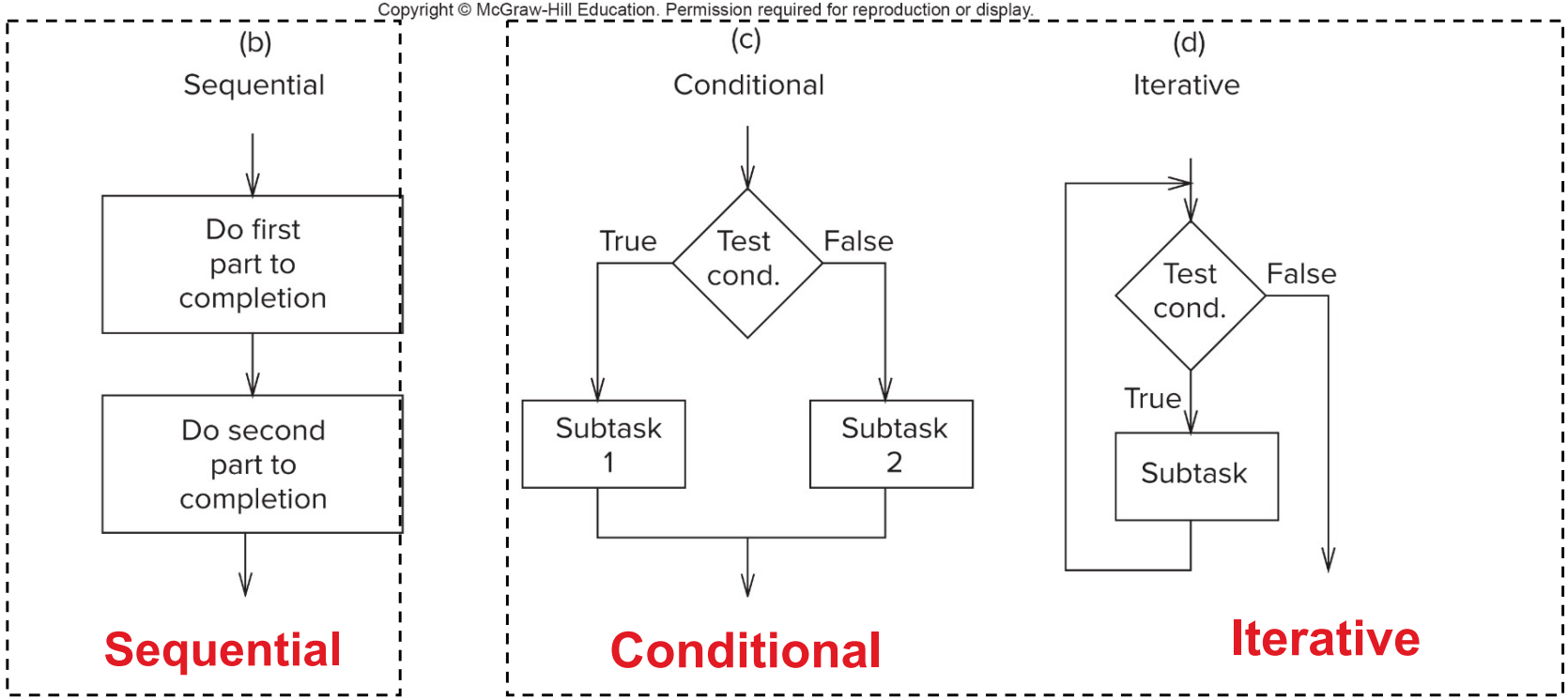
From Bits and Gates to C and Beyond

Control Structures

Chapter 13

Sequential, Conditional, Iterative

In Chapter 6, we discussed the three basic control structures. In this chapter, we will discuss the C language features for conditional and iterative code.



Chapter 12 *Chapter 13*

Access the text alternative for slide images.

`if` Examples 1

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

Style Guideline:
Put conditional statement on the next line and indent. Clear that statement is conditional.

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

Use of a compound statement. Both statements are executed ONLY if condition is true.

Style: Open brace on same line as if (...). Indent statements inside the block. Closing brace on its own line, aligned with if.

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

Bad use of indentation. Looks like both statements are conditional, but only the first is. z = ... should not be indented.

`if` Examples 2

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

Common mistake!

Using assignment (=) instead of equal-to (==).

Condition changes value of x and is ALWAYS true!

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

Corrected version of previous statement.

```
if (x == 3)
    if (y != 6) {
        z = z + 1;
        w = w + 2;
    }
```

Nested if statements. Note the indentation.

LC-3 Code for `if`

Code uses the same pattern discussed in Chapter 6.

```
        ; if (x == 2) y = 5;
        ; assume offset for x is 0, y is -1
        ; (1) code to test condition
        LDR R0, R5, #0    ; load x
        ADD R0, R0, #-2
        BRnp NEXT         ; (2) BR if condition not true
        AND R0, R0, #0    ; (3) code for action
        ADD R0, R0, #5
        STR R0, R5, #-1   ; store to y
NEXT    ...
```

LC-3 Code for `if-else`

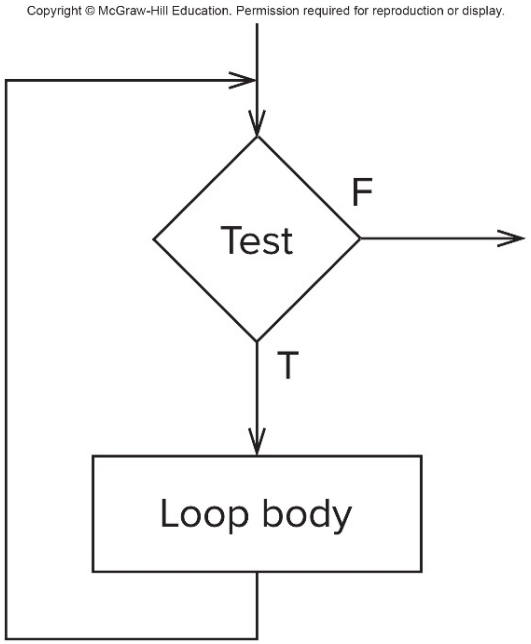
```
if (x) {
    y++;
    z--;
}
else {
    y--;
    z++;
}
```

Offsets:
x = 0, y = -1, z = -2

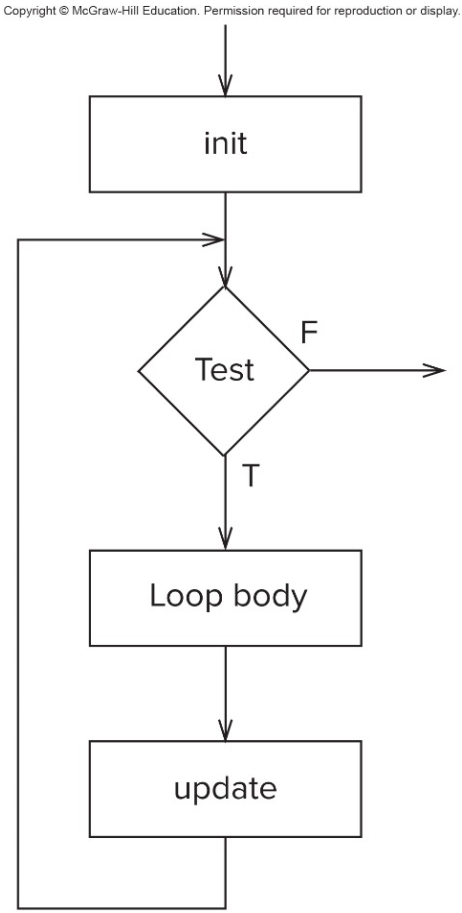
```
        LDR R0, R5, #0     ; condition (x)
        BRz ELSE           ; jump if false

        LDR R0, R5, #-1    ; y++
        ADD R0, R0, #1
        STR R0, R5, #-1
        LDR R0, R5, #-2    ; z--
        ADD R0, R0, #-1
        STR R0, R5, #-2
        BRnzp NEXT         ; skip else action

ELSE    LDR R0, R5, #-1    ; y--
        ADD R0, R0, #-1
        STR R0, R5, #-1
        LDR R0, R5, #-2    ; z++
        ADD R0, R0, #1
        STR R0, R5, #-2

NEXT    ...
```

Iteration

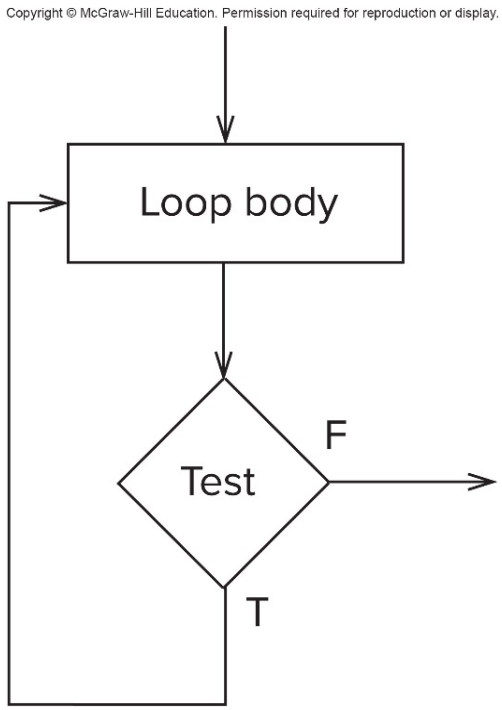
C provides three different iterative statements, representing three different ways to express a loop. Choose the one that best fits the problem.



while



for



do-while

Access the text alternative for slide images.

`while` Examples

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

Loop body can be simple or compound.
Indent the statements in the loop body, so that it's easier to tell what statements are inside the loop.

```
char echo;
while (echo != '\n') {
    scanf("%c", &echo);
    printf("%c", echo);
}
```

Common mistake #1: Forgetting to initialize.

Since we don't initialize echo in this code, we don't know whether it will be equal to linefeed ('\n') or not. We have no control over whether the loop will ever execute.

Solution: Make sure the loop test variable has a reasonable value before the first test.

```
while (x < 10)
    printf("%d ", x);
```

Common mistake #2: Infinite loop.

Assuming x is less than 10, this loop will execute forever. Why?

Solution: Make sure to change loop variable in body.

LC-3 Code for `while`

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

```
        AND R0, R0, #0    ; x = 0
        STR R0, R5, #0

LOOP    LDR R0, R5, #0    ; test x < 10
        ADD R0, R0, #-10
        BRzp NEXT         ; exit if false

        ; loop body
        ...code to call printf...
        LDR R0, R5, #0    ; x = x+1
        ADD R0, R0, #1
        STR R0, R5, #0
        BRnzp LOOP        ; test again

NEXT    ...
```

LC-3 Code for `for`

```
int x;
int sum = 0;
for (x = 0;
     x < 10;
     x++)
   sum += x;
```

```
          AND R0, R0, #0    ; sum = 0
          STR R0, R5, #-1
          ; for statement: init expr
          AND R0, R0, #0    ; x = 0
          STR R0, R5, #0
LOOP      LDR R0, R5, #0    ; test x < 10
          ADD R0, R0, #-10
          BRzp NEXT         ; exit if false

          ; loop body
          LDR R0, R5, #-1   ; sum
          LDR R1, R5, #0    ; x
          ADD R0, R0, R1
          STR R0, R5, #-1   ; sum += x

          ;update expression
          LDR R0, R5, #0    ; x = x+1
          ADD R0, R0, #1
          STR R0, R5, #0
          BRnzp LOOP        ; test again

NEXT      ...
```

LC-3 Code for do-while

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

```
          AND R0, R0, #0    ; x = 0
          STR R0, R5, #0

LOOP      ; loop body
          ...code to call printf...
          LDR R0, R5, #0    ; x = x+1
          ADD R0, R0, #1
          STR R0, R5, #0

          LDR R0, R5, #0    ; test x < 10
          ADD R0, R0, #-10
          BRn LOOP          ; repeat if true

NEXT      ...
```

`break` and `continue`

These two statements are used to alter the "normal" execution of a loop.

`break`:

Immediately **exits** the loop (or switch).

Unconditional branch to the next statement after the loop/switch.

`continue`:

Immediately goes to the **next iteration** of the loop, skipping the remainder of the loop body.

Unconditional branch to the "top" of the loop:

- `while` and `do-while` -- goes to the test expression.
- `for` -- goes to the update (re-init) expression.

Use these rarely for exceptional conditions. Use loop logic or conditionals in most cases: easier to understand.

Examples: `break` and `continue`

These two simplified examples illustrate the difference.

```
for (i=0; i<10; i++) {
   if (i == 5)
      break;
   printf("%d ", i);
}
```

Output:
```
0 1 2 3 4
```

```
for (i=0; i<10; i++) {
   if (i == 5)
      continue;
   printf("%d ", i);
}
```

Output:
```
0 1 2 3 4 6 7 8 9
```

Because learning changes everything.®

www.mheducation.com