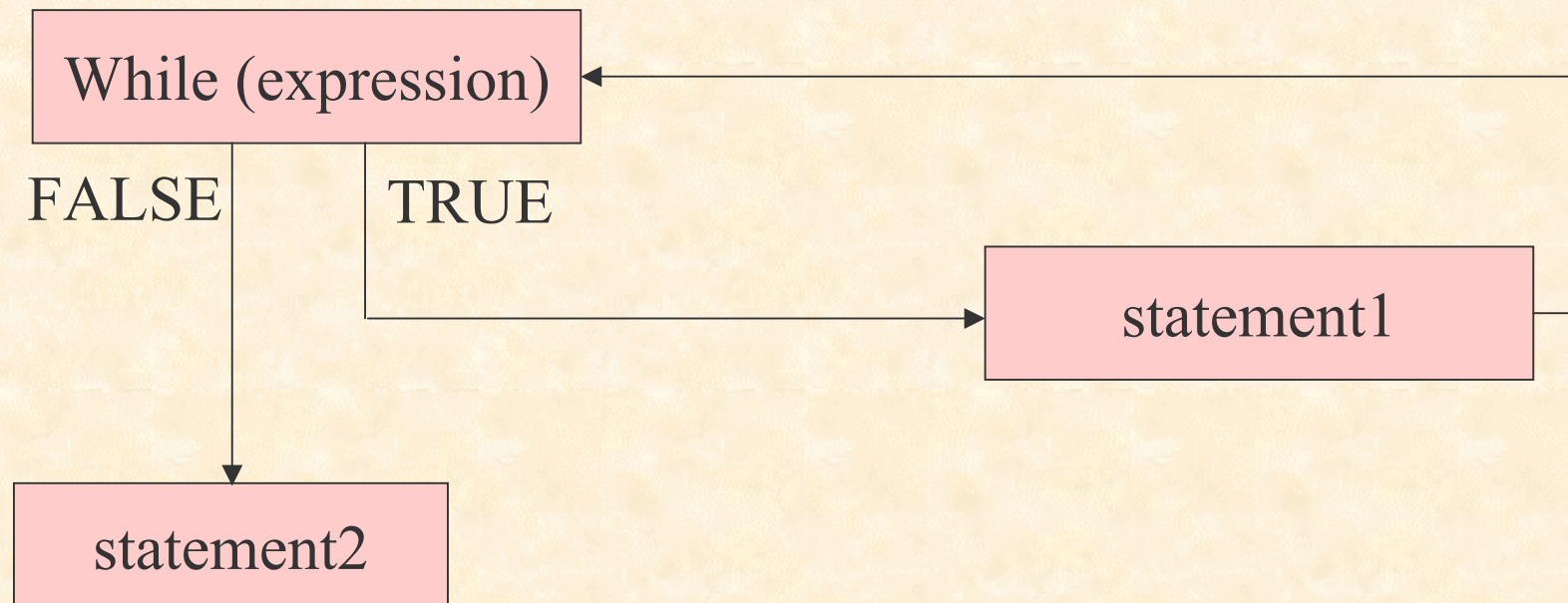


ITERATION STATEMENTS - INTRODUCTION

- Iteration allows a group of instructions to be repeated a certain number of times without having to rewrite each instruction



While Statement - Flow of Control

ITERATION - WHILE

- ❑ Repeatedly execute a statement sequence **while** some condition is **true**

- ❑ **While Loop Format**

while (control-expr)

statement

This means while the control-expr evaluates to true (non-zero) the statement will be repeatedly executed

- The statement may be a compound statement and must cause the value of the control-expr **to change to false** (zero) in order to terminate the loop
- If the control-expr **never becomes true** the loop will not be entered
- If the control-expr **does not change to false** the loop will never terminate (will go on forever)

AN EXAMPLE OF ITERATION USING WHILE

```
#include <iostream.h>
void main ()
{
    int counter = 1;    // loop control variable
    while (counter < 10) {
        // statements executed in loop
        cout << "in loop with counter as ";
        cout << counter << endl;
        // increment control variable
        counter++;    }
}
```

DANGEROUS EXAMPLE OF WHILE ITERATION

```
#include <iostream.h>
void main ()
{
    int counter = 1;    // loop control variable
    while (counter++ < 10)
        // statements executed in loop
        cout << "counter is " << counter << endl;
}
// remember to add braces {} if more statement are added to the loop
```

ITERATION - DO-WHILE LOOP

❑ A do-while loop is similar to a while loop except that the loop test comes after the loop body

❑ **do-while** Loop Format:

do

statement

while (control-expr)

The body of a do loop is executed at least once

EXAMPLE OF A DO-WHILE LOOP

```
#include <iostream.h>

int main()
{
    int counter = 1;
    do {
        // statements executed in loop
        cout << "counter is " << counter << endl;
        // increment loop counter
        counter++;
    }
    while (counter < 10);
}
```

ITERATION - FOR LOOP

- Imagine writing a program which draws the following



ITERATION - FOR LOOP

❑ Option A

1. Write function to draw a star called **star()**
2. Write down **star()** **80 times**

❑ Option B

1. Write function to draw a star called **star()**
2. Write function **row_of_stars()** to draw a row of 10 stars by writing **star()** **10 times**
3. Write function **star_array()** to draw 10 by 8 stars by writing **row_of_stars()** 8 times

ITERATION - FOR LOOP

□ Option C

1. Write function to draw a star called `star()`
2. Use for loop iteration in Option B (2) and (3)

To draw a row of 10 stars in option C

1. Create a variable to be used as a counter
2. Set it to zero
3. As long as the counter is less than 10
 - 3.1 Draw a star
 - 3.2 Move the pen to the right
 - 3.3 Increment the counter by one
 - 3.4 Repeat 3

ITERATION - FOR LOOP

□ For Loop Format:

```
for (expression1; expression2; expression3)
    statement
```

- `statement` may be compound statement
- `expression1` may be replaced by a declaration
- all three expressions may be omitted

The expressions translate to

```
for (init-expr; condition-expr; update-expr)
    statement
```

- `init-expr` is used to initialize loop variables, e.g. `int i = 1`
- `condition-expr` is used to test for the end of the loop, e.g. `i < 10`
- `update-expr` is used to update the loop variable, e.g. `i++`

FOR LOOP EXAMPLE

```
int I;  
for (I = 1; I <= 10; I++)  
    cout << I << endl;
```

This loop uses I as the loop variable

I = 1 initializes I before the loop is executed

I <= 10 is evaluated every time before loop body is executed

The loop body is only executed if I <= 10 is true

SO, I counts through 1,2,3,4,5,6,7,8,9,10

JUMP STATEMENTS

- Jump statements allow the early termination of loops

return, goto, break, continue, exit

These cause **unconditional** branches

return will return to the calling function

goto is bad practice and will not be dealt with

break will exit the inner most loop

continue will force the next iteration

exit will quit the program

THE BREAK STATEMENT

- If a break statement is executed within a loop body the loop will immediately terminate

```
while (exp1) {  
    // some statements  
    if (exp2)  
        break;           // Quit now  
    // more statements  
}
```

ITERATION WITH EARLY TERMINATION

```
#include <iostream.h>

int main ()
{
    int counter = 1;
    while (counter++ < 10) {
        if (counter == 6)
            break;
        cout << "counter is " << counter << endl;
    }
    return 0;
}
```

THE CONTINUE STATEMENT

- ❑ If a continue statement is executed in a loop body, control will immediately jump back to the start of the loop

```
while (exp) {  
    if (a == b) {  
        // jump back to start  
        continue;  
    }  
    // otherwise execute rest of loop body  
}
```

EXAMPLE OF CONTINUE STATEMENT

```
#include <iostream.h>

int main()
{
    int counter = 1;
    while (counter++ < 10) {
        if (counter == 6)
            continue;
        cout << "counter is" << counter << endl;
    }
    return 0;
}
```

NOTES ON ITERATION

- Iterations using **do-while** and **for** decompose to **while** loops

do-while loop

```
do
  statement
while (control-expr)
```

This is equivalent to

```
statement
while (control-expr)
  statement
```

for loop

```
for (expr1; expr2; expr3)
  statement
```

This is equivalent to

```
expr1;
while (expr2) {
  statement
  expr3;
}
```

TERMINATING AND NON TERMINATING LOOPS

□ The following loop statements will run forever

- `for (i i i) i`
- `for (int i = 0; ; i++);`
- `int j = 0;`
`for (int j = 0; j < 10; j--)`
`cout << "hello world\n";`
- `for (int j = 0; j < 10; j++) {`
`cout << "hello world\n";`
`j = 0;`
`}`
- `while (1)`
`cout << "hello world\n";`

TERMINATING AND NON TERMINATING LOOPS

- `while (int j = 1)`
 `cout << "hello world\n";`
- `int j = 1;`
 `do cout << "hello world\n"; j += 2;`
 `while (j != 10);`

❑ The following loops will never run

- `for (int j = 0; j; j++)`
 `cout << "hello world\n";`
- `while (0)`
 `cout << hello world\n";`

❑ A **do-while** statement will always execute at least once