for loops

Comp Sci 1570 Introduction to C++

Missouri S&T | Computer Science

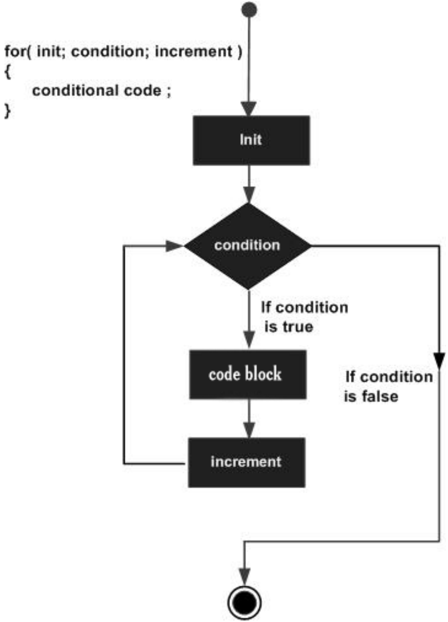
Outline

1. Introduction
    - Diagram
    - Syntax
2. Operation
    - Multi-variable
    - Sequence of operations
    - Optional fields
    - Scope matters

Outline

1. Introduction
   - Diagram
   - Syntax
2. Operation
   - Multi-variable
   - Sequence of operations
   - Optional fields
   - Scope matters

for loop



```
for( init; condition; increment )
{
    conditional code ;
}
```

Outline

1. Introduction
   - Diagram
   - Syntax
2. Operation
   - Multi-variable
   - Sequence of operations
   - Optional fields
   - Scope matters

Syntax

- For loop is used to execute a set of statement repeatedly until a particular condition is satisfied.
- Like the while-loop, this loop repeats statement while condition is true.
- It is especially useful to use counter variables as condition.

```
for(initialize(s); condition; increment(s)/decrement(s))
{
  statement(s);
}
```

Syntax

```
for(initialize(s); condition; increment(s)/decrement(s))
{
  statement(s);
}
```

- for is a reserved word.
- statement is a valid simple or compound C++ statement.
- field 1 initializes the LCV.
- field 2 checks the LCV.
- field 3 updates the LCV.
- exactly two semicolons, one after initialization and second after condition.

Outline

1 Introduction
   - Diagram
   - Syntax

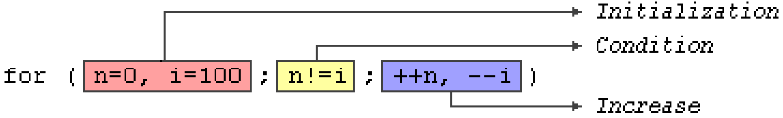
2 Operation
   - Multi-variable
   - Sequence of operations
   - Optional fields
   - Scope matters

Outline

1 Introduction
- Diagram
- Syntax

2 Operation
- Multi-variable
- Sequence of operations
- Optional fields
- Scope matters

Multi-variable



```
for ( n=0, i=100 ; n!=i ; ++n, --i )
```

- Can have more than one initialization or increment/decrement, separated using comma operator.
- Condition can be complex

Outline

1 Introduction
   - Diagram
   - Syntax

2 Operation
   - Multi-variable
   - Sequence of operations
   - Optional fields
   - Scope matters

A for statement is evaluated in 3 parts:

1. The init-statement is evaluated. Typically, the init-statement consists of variable definitions and initialization. This statement is only evaluated once, when the loop is first executed.
2. The condition-expression is evaluated. If this evaluates to false, the loop terminates immediately. If this evaluates to true, the statement is executed.
3. After the statement is executed, the end-expression is evaluated. Typically, this expression is used to increment or decrement the variables declared in the init-statement. After the end-expression has been evaluated, the loop returns to step 2.

Outline

1. Introduction
   - Diagram
   - Syntax
2. Operation
   - Multi-variable
   - Sequence of operations
   - Optional fields
   - Scope matters

Optional fields

- The three fields in a for-loop are optional.
- They can be left empty, but in all cases the semicolon signs between them are required.
- For example, for $(; n < 10;)$ is a loop without initialization or increase (equivalent to a while-loop);
- for $(; n < 10; + + n)$ is a loop with increase, but no initialization (maybe because the variable was already initialized before the loop).
- A loop with no condition is equivalent to a loop with true as condition (i.e., an infinite loop).

Outline

1 Introduction
- Diagram
- Syntax

2 Operation
- Multi-variable
- Sequence of operations
- Optional fields
- Scope matters

Scope matters

- Variables declared and initialized inside the loop or its expression go out of scope when the loop is finished.
- Variables declared and initialized before the loop persist beyond the loop