Repetition Control Structure

B. Ramamurthy
Chapter 3

Introduction

- Many application require certain operations to be carried out more than once. Such situations require repetition in control flow.
- In C++ repetition in execution can be realized using a “while”, “do-while” and a “for” statement.
Topics for discussion

- Repetition structure (loop) design
- **while** loop : syntax, semantics, example
- Loop control
- **for** loop : syntax, semantics, example
- **do-while** : syntax, semantics, example
- Case Study 1
- nested loops
- Case Study 2
- Summary

while loop : syntax

```
while ( condition )
    statement
```

- “condition” is a logical expression that evaluates to true or false. It could be a relational or Boolean expression.
- “statement” could a single statement or more than one statement bounded by { }. It is often referred to as the body of the loop.
while loop : semantics

1) The condition is evaluated.
2) If it is true, then the body of the loop is executed. Then the control is transferred back to the condition for re-evaluation.
3) If the logical expression is false, the while loop is exited and control is transferred to the statement after the while statement.

The while statement

while (condition)
statement;

while (condition)
{
    statement block
}
while loop: example

**Problem:** Write statements to determine the sum to 
natural numbers. Print out the computed sum.

```c
int sum = 0;
while (n > 0)
{
    sum = sum + n;
    n = n - 1;
}
cout << "Sum of first " << n << " natural numbers is " << sum
    << endl;
```

Loop Design

**Loop design should consider:**
- **Initialization of conditions** (sum = 0)
- **Termination of the loop** (when n == 0)
- **Testing** (at the top or bottom of the loop)
  - (at the top, n > 0)
- **Updating conditions**
  - (n = n - 1)
- **Of course, the body of the loop.**
  - (sum = sum + n; n = n - 1;)

**Body of the loop:** Statements representing the 
process to be repeated. These are statements within 
the scope of a loop.
More about “while”

- Initialize the conditions which are evaluated in the “while”.
- Conditions are evaluated at the “top” of while statement or before the execution of the body.
- “while” body is executed 0 or more times.
- Updating of the conditions are done inside the body of the “while”.
- Use “while” when the number of times a loop is executed is dependent on some condition set during execution of the body.
- Example: Input a list of positive number. The list is terminated by a negative number.

Example

- Write statements that takes as input n, the side of a square and print and square of size n with asterisks.
  
  //PRE: N is a value between 2 and 20
  //Output is
  ***
  ***
  ***
Do-while - syntax

- Use this control structure:
- When a loop needs to be executed at least once.
- When the testing of the conditions needs to be done at the bottom.

```
do
    statement
while ( condition );
```

do-while semantics

1) Execute the statement.
2) Evaluate the expression.
   If it is TRUE then proceed to step 1)
   else exit the loop.

NOTE: do-while is executed at least once.
The do/while statement

```
do
  statement;
while (condition)
do
  {
    statement block
  } while (condition)
```

Example for do-while

Usage: Prompt user to input “month” value, keep prompting until a correct value of moth is input.

```
do
  {
    cout <<“Please input month {1-12}“);
cin >> month;
  } while ((month < 1) || (month > 12));
```
For loop - syntax

for (initialize exp; test exp; update exp) 
  statement

initialize exp: done only once at the start

test exp: This is a condition that evaluates to TRUE or FALSE.

update exp: This specifies how to update condition.

Note: for loop is used when the number of times to be repeated is fixed/known apriori.

For loop - Semantics

1) Initialize exp is executed.
2) Test exp is evaluated.
   If it is TRUE, body of for is executed
   else exit for loop;
3) After the execution of the body of the loop, update exp is executed to update condition; go to Step 2 above.
The for statement

```
initialize
increment/decrement

statement(s)
```
For - Examples

Problem 1: Write a For statement that computes the sum of all odd numbers between 1000 and 2000.

Problem 2: Write a For statement that computes the sum of all numbers between 1000 and 10000 that are divisible by 17.

Problem 3: Printing square problem but this time make the square hollow.

Summary

- Loop design
- Repetition control structures: while, for, do-while
- Nested loops
- Beware of infinite looping problem
- Read chapter 3.