Introduction to C++ Programming

Its Applications in Finance

Thanh Hoang

Claremont Graduate University

September 19, 2012

Today Agenda

1. Random Number
2. Selection Statements
    - *if* statement
    - *switch* statement
3. Iteration Statements
    - *for* loop statement
    - *while* loop statement
    - *do-while* loop statement
4. Jump Statements
    - *break* statement
    - *continue* statement
    - *return* statement
    - *goto* statement
5. Summary

Generating Random Numbers

The *rand*() Function

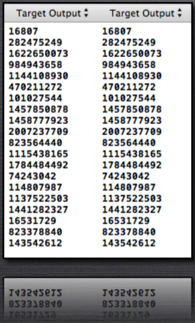
1. Belongs to the *cstdlib* library
2. Creates a random number between 0 and RAND_MAX
3. Generates a random integer number in the range $[a, b]$

$$\text{int num} = \text{rand}()\ \%\ (b - a + 1) + a;$$

Random Problem

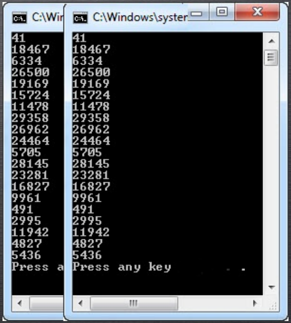
Problem of the *rand* () function

1. Does not really generate a random number
2. A really long built-in list of integers
3. A same list every time when we restart the *rand* () function



```
Target Output     Target Output
16807             16807
282475249         282475249
1622650073        1622650073
984943658         984943658
1144108930        1144108930
470211272         470211272
101027544         101027544
1457850878        1457850878
1458777923        1458777923
2007237709        2007237709
823564440         823564440
1115438165        1115438165
1784484492        1784484492
74243042          74243042
114807987         114807987
1137522503        1137522503
1441282327        1441282327
16531729          16531729
823378840         823378840
143542612         143542612
```

(a) Mac OS (64-bit)



```
C:\Wir
41
18467
6334
26500
19169
15724
11478
29358
26962
24464
5705
28145
23281
16827
9961
491
2995
11942
4827
5436
Press a
```

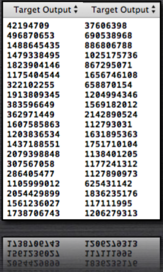
```
C:\Windows\syster
41
18467
6334
26500
19169
15724
11478
29358
26962
24464
5705
28145
23281
16827
9961
491
2995
11942
4827
5436
Press any key
```

(b) Windows OS (32-bit)

Random Problem (*cont.*)

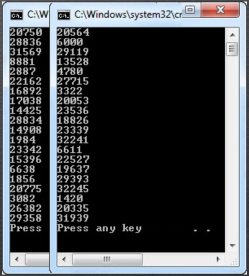
Solution

1. Changes the seed of the random number generator by the *srand* (int *a*) function
2. Does not want to change *a* manually every time – The easiest way is to use the current time as an initial point.
3. Declares a <*ctime*> library to use the *time* (int *t*) function
   - 3.1 *time* (0) returns the number of seconds from January 1, 1970 to the current time.
   - 3.2 (int) *time*(0) is required since time returns a special number of type time_t.



Target Output	Target Output
42194709	37606398
496870653	690538968
1488645435	886806788
1479338495	1025175736
1823904146	867295071
1175404544	1656746108
322102255	658870154
1913809345	1204994346
383596649	1569182012
362971449	2142890524
1607585863	112793031
1203836534	1631895363
1437188551	1751710104
2079398848	1138401205
307567058	1177241312
286405477	1127890973
1105999012	625431142
2054429899	1836235176
1561236027	117111995
1738706743	1206279313

(c) Mac OS (64-bit)

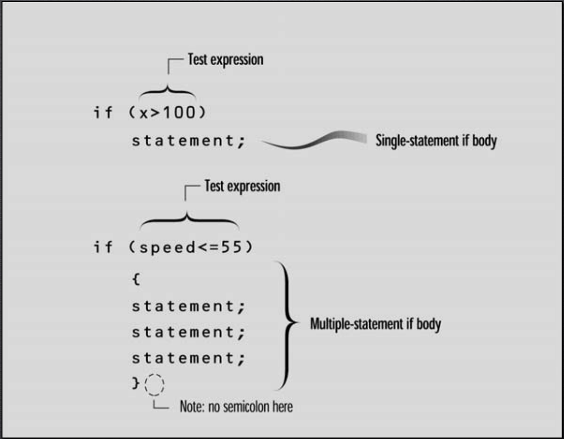


```
20750
28836
31569
8881
2887
22162
16892
17038
14425
28834
14908
1984
23342
15396
6638
1856
20775
3082
26382
29358
Press
```

```
20564
6000
29119
13528
4780
27715
3322
20053
23536
18826
23339
32241
6611
22527
19637
29393
32245
1420
20335
31939
Press any key           . .
```

(d) Windows OS (32-bit)

The *if* Statement



Test expression

```
if (x>100)
    statement;
```

Single-statement if body

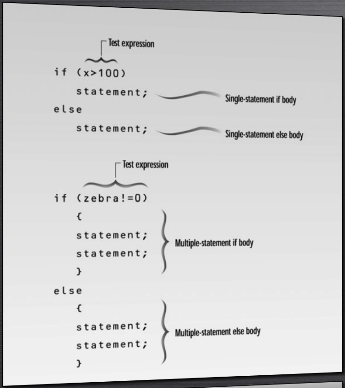
Test expression

```
if (speed<=55)
    {
    statement;
    statement;
    statement;
    }
```

Multiple-statement if body

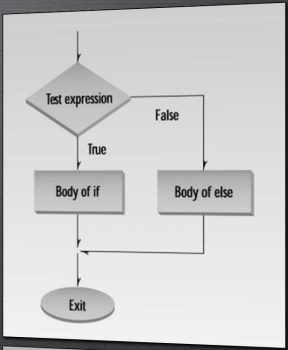
Note: no semicolon here

Operation of the *if* Statement



```
if (x>100)
   statement;
else
   statement;

if (zebra!=0)
   {
   statement;
   statement;
   }
else
   {
   statement;
   statement;
   }
```

Test expression
Single-statement if body
Single-statement else body
Test expression
Multiple-statement if body
Multiple-statement else body



Guess a Number

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    int mega_rand;  // a random number
    int guess; // guess from the user

    srand((int) time(0));

    // Generates a Mega random number in [1,46]
    mega_rand = rand() % 46 + 1; // rand()%(b-a+1)+a

    cout << "Enter your guess: ";
    cin >> guess;

    cout << "Let s see. Your guess number is " << guess
    << " and the Mega random number is " << mega_rand << "." << endl;

    if (guess == mega_rand)
        cout << "** Congratulation!!! **" << endl;
    else
        cout << "Good luck next time!" << endl;

    return 0;
}
```

Using an *int* value to Control the *if* Statement

```
#include <iostream>
using namespace std;

int main()
{
    int a, b;

    cout << "Enter numerator: ";
    cin >> a;
    cout << "Enter denominator: ";
    cin >> b;

    if (b)
        cout << "Result: " << (float) a/b << endl << endl;
    else
        cout << "Cannot divide by zero." << endl << endl;

    return 0;
}
```

An Example of Nested *if*

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    int mega_rand; // a random number
    int guess;     // user s guess

    srand((int) time(0));
    mega_rand = rand() % 46 + 1; // gives us a Mega random number in [1,46]

    cout << "Enter your guess: ";
    cin >> guess;

    cout << "Let s see. Your guess number is " << guess
    << " and the Mega random number is " << mega_rand << "." << endl;

    if (guess == mega_rand)
        cout << "** Congratulation!!! **" << endl;
    else {
        cout << "...Good luck next time!";
        if (guess > mega_rand)
            cout << " Your guess is too high." << endl;
        else
            cout << " Your guess is too low." << endl;
    }

    return 0;
}
```

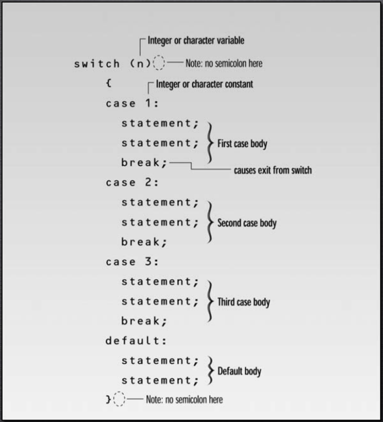
The *if-else-if* Ladder

if-else-if

```
if (condition)
        statements;
else if (condition)
        statements;
else if (condition)
        statements;
        .........
        .........
else
        statements;
```

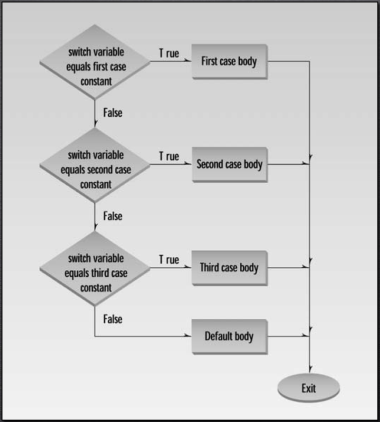
```
#include <iostream>
using namespace std;

int main()
{
    for (int a=0; a<6; a++) {
        if (a==1)
            cout << "a is one." << endl;
        else if (a==2)
            cout << "a is two." << endl;
        else if (a==3)
            cout << "a is three." << endl;
        else if (a==4)
            cout << "a is four." << endl;
        else
            cout << "a is not between one and four." << endl;
    }

    return 0;
}
```

The *switch* Statement



```
switch (n)
   {
   case 1:
     statement;
     statement;
     break;
   case 2:
     statement;
     statement;
     break;
   case 3:
     statement;
     statement;
     break;
   default:
     statement;
     statement;
   }
```

Operation of the *switch* Statement


switch variable equals first case constant
T rue
First case body
False
switch variable equals second case constant
T rue
Second case body
False
switch variable equals third case constant
T rue
Third case body
False
Default body
Exit

The *switch* Statement

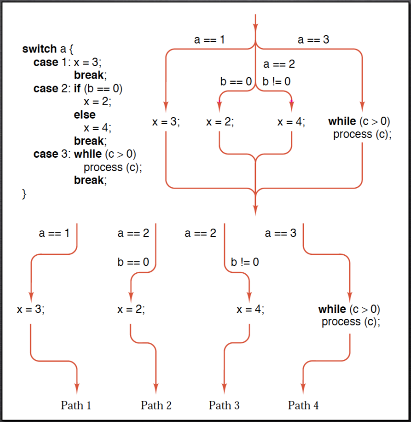
Some C++ rules:

1. The *switch* expression must be a character or an integer value.
2. The case constants must be also a character or an integer value.
3. Floating-point values are not allowed.

Four important aspects of the *switch* statement:

1. The *switch* can test only for equality.
2. No two case constants in the same *switch* can have identical values.
3. A switch statement is usually more efficient than nested *if*.
4. The statement sequences associated with each case are not blocks.

Operation Process of *switch* Statement



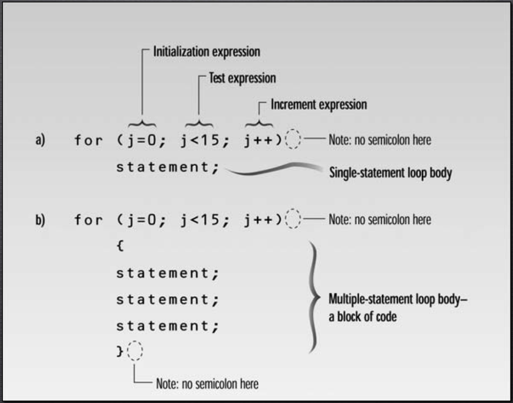
```
switch a {
   case 1: x = 3;
           break;
   case 2: if (b == 0)
             x = 2;
           else
             x = 4;
           break;
   case 3: while (c > 0)
             process (c);
           break;
}
```

An Example of the *switch* Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    for (int a=0; a<4; a++) {
        switch (a) {
            case 0:
                cout << "a is less than one." << endl;
                break;
            case 1:
                cout << "a is less than two." << endl;
                break;
            case 2:
                cout << "a is less than three." << endl;
                break;
            default:
                cout << "a is less than four." << endl;
                break;
    }

    cout << endl;
    }

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    for (int a=0; a<4; a++) {
        switch (a) {
            case 0:
                cout << "a is less than one." << endl;
            case 1:
                cout << "a is less than two." << endl;
            case 2:
                cout << "a is less than three." << endl;
            default:
                cout << "a is less than four." << endl;
        }

        cout << endl;
        }

        return 0;
}
```

The *for* Statement



Initialization expression

Test expression

Increment expression

```
a)   for (j=0; j<15; j++)
         statement;

b)   for (j=0; j<15; j++)
         {
         statement;
         statement;
         statement;
         }
```

Note: no semicolon here

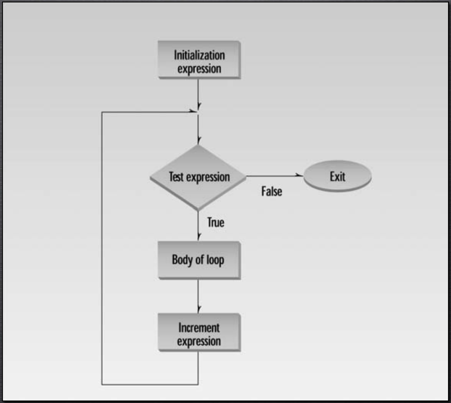
Single-statement loop body

Note: no semicolon here

Multiple-statement loop body–a block of code

Note: no semicolon here

Operation of the *for* Loop Statement



An Example of the *for* Statement

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    for (int a=1; a<=100; a++) {
        double sqroot = sqrt(a);
        cout << sqroot << " is a square root of " << a << endl;
    }

    return 0;
}
```

Another Example of the *for* Statement

```
#include <iostream>
using namespace std;

int main()
{
    for (int a=100; a>=-100; a -= 10)
        cout << a << "   ";

        cout << endl;

        return 0;
}
```

Use Numeric Test in *for* Loop Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    int i, limit;

    cout << "Enter the starting countdown value: ";
    cin >> limit;

    for (i = limit; i; i--) // quits when i is 0
        cout << "i = " << i << endl;

    cout << "Done now that i = " << i << endl;

    return 0;
}
```

Multiple *for* Loop Control Variables

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a, b;

    for (a=0, b=10; a <= b; a++, b--)
        cout << a << "  " << b << endl;

    return 0;
}
```

Output

```
0  10
1  9
2  8
3  7
4  6
5  5
```

Do Not Use The Loop Control Variable

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int a, rand_num=0;

    for (a=0; rand_num <= 20000; a++)
        rand_num = rand(); // creates a random number

    cout << "The number is " << rand_num
       << ". It was generated on try " << a << ".\n";

    return 0;
}
```

Missing Increment / Decrement Expression

```cpp
#include <iostream>
using namespace std;

int main()
{
    for (int a=0; a!=911; ) {
        cout << "Enter a number: ";
        cin >> a;
    }

    return 0;
}
```

Missing Both Initial Value and Increment Expression

```
#include <iostream>
using namespace std;

int main()
{
    int a=0;

    for ( ; a!=911; ) {
        cout << "Enter a number: ";
        cin >> a;
    }

    return 0;
}
```

Loops Without Body

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int a, sum=0;

    // Sum the numbers from 1 to 100
    for (a=1; a<=100; sum += a++) ;

    cout << "The total sum of the numbers from 1 to 100 is "
       << sum << endl;

    return 0;
}
```

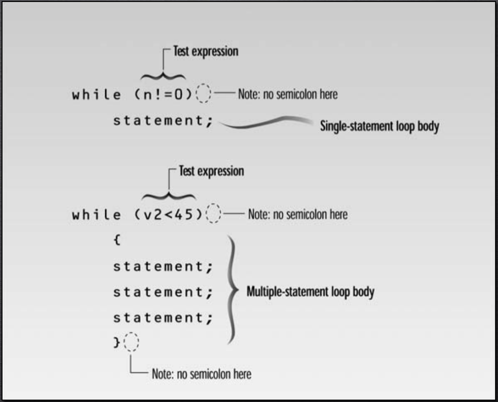
Loop Control Variables Inside The *for* Loop Statement

```
// Find the sum from 1 to a number and the number s factorial

#include <iostream>
using namespace std;

int main()
{
   int a, sum=0, factorial=1;

   cout << "Enter a number: ";
   cin >> a;

   // Compute the factorial of a numbers
   for (int i=1; i <= a; i++) {
      sum += i;
      factorial *= i;
   }

   cout << "The total sum from 1 to " << a << " is " << sum << endl;
   cout << a << "! is " << factorial << endl;

   return 0;
}
```

Nested *for* Loops

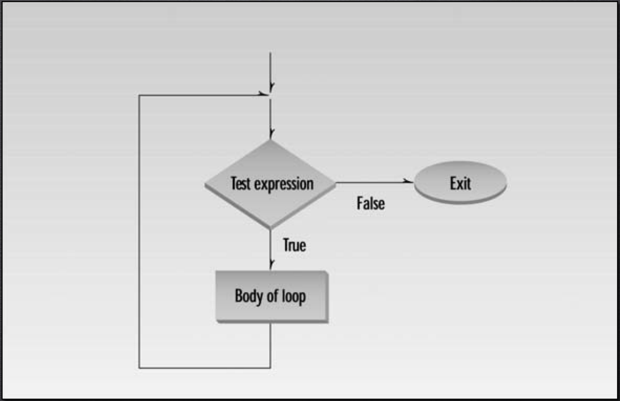
```
// Find factors of numbers between 2 and 25

#include <iostream>
using namespace std;

int main()
{
    int a, b;

    for (a=2; a<=25; a++) {
        cout << "The factors of " << a << ": ";

        for (b=2; b<a; b++)
            if (!(a % b)) cout << b <<     ;

        cout << endl;
    }

    return 0;
}
```

The *while* Loop Statement



```
while (n!=0)
    statement;

while (v2<45)
    {
    statement;
    statement;
    statement;
    }
```

Operation of the *while* loop statement



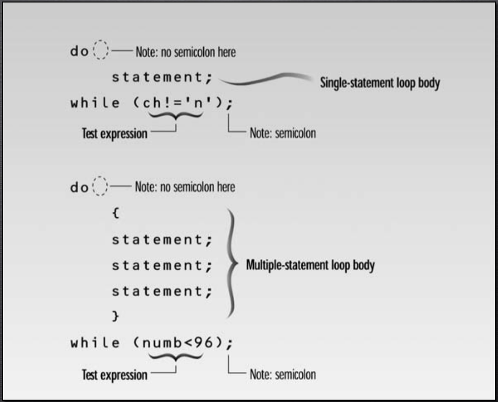
An Example of the *while* Loop Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n=1; // sets an initial value of n as non-zero

    while (n) {
        cout << "Enter a number (0 to exit): ";
        cin >> n;
    }

    cout << endl;

    return 0;
}
```

Another Example of the *while* Loop Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    int length;

    cout << "Enter the length (1-79): ";
    cin >> length;

    while (length>0 && length<80) {
        cout << "_";
        length--;
    }

    cout << endl;

    return 0;
}
```

The *do-while* Loop Statement



```
do
    statement;
while (ch!='n');
```

Note: no semicolon here

Single-statement loop body

Test expression

Note: semicolon

```
do
    {
    statement;
    statement;
    statement;
    }
while (numb<96);
```

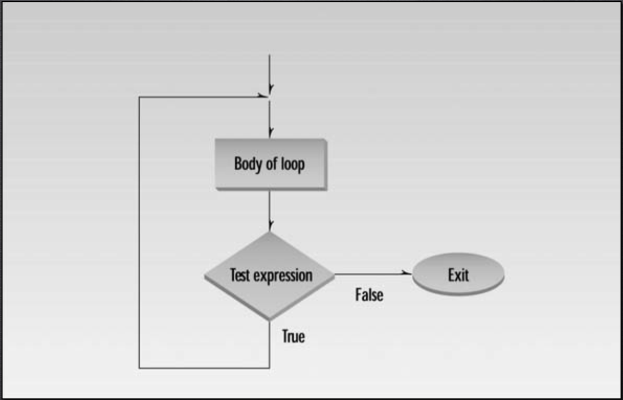
Note: no semicolon here

Multiple-statement loop body

Test expression

Note: semicolon

Operation of the *do-while* Loop Statement



An Example of the *rand()* Function

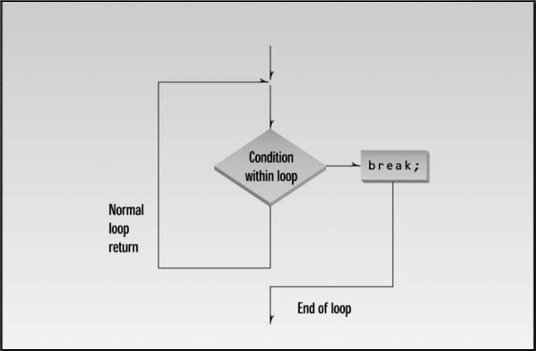
```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    char key;
    int a=1, b=6;

    do {
        int randDice = rand() % (b - a + 1) + a;
        cout << "A random dice rolled: " << randDice << endl;
        cout << "Do another (y/n): ";
        cin >> key;
    } while (key !="n");

    return 0;
}
```

An Example of the *while* Loop Statement

```
#include <iostream>
using namespace std;

int main()
{
    char keyword;
    int numerator, denominator;

    do {
        cout << "Enter a numerator: ";
        cin >> numerator;

        cout << "Enter a demominator: ";
        cin >> denominator;

        cout << numerator <<  /  << denominator
           << " is " << numerator / denominator
           << " and remainder is " << numerator % denominator;

        cout << "\nDo another? (y/n): ";
        cin >> keyword;
        cout << endl;
    } while (keyword !=  n );

    return 0;
}
```

Jump Statements

1. *break* statement

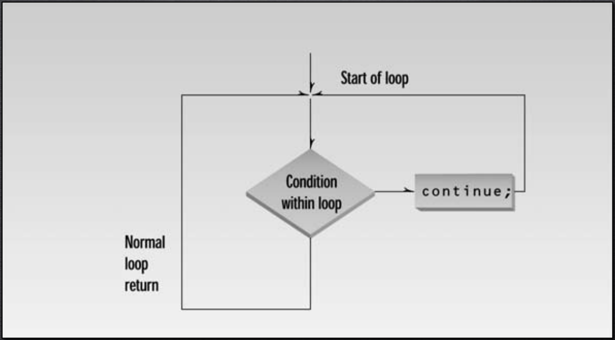


An Example of the *break* Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    // The loop shows the numbers from 0 to 9,
    // not from 0 to 100.
    for (int a=0; a<=100; a++) {
        if (a==10) break;

        cout << a << "  ";
    }

    cout << endl;

    return 0;
}
```

Jump Statements

1. *break* statement
2. *continue* statement



An Example of the *continue* Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    for (int a=0; a<=100; a++) {
        if(a % 2) continue;

        cout << a << "   ";
    }

    cout << endl;

    return 0;
}
```

Jump Statements

1. *break* statement
2. *continue* statement
3. *goto* statement

General Form

```
goto somewhere;
    somewhere:
```

An Example of the *goto* Statement

```cpp
#include <iostream>
using namespace std;

int main()
{
    int i;

    for(i=0; i <=100; i++) {
        add1: i++;

        if (i % 2) goto add1;

        cout << i << "  ";
    }

    return 0;
}
```

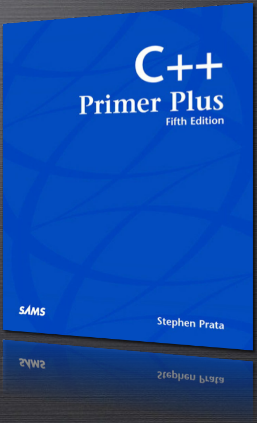
Summary

1. Random Number
2. Selection Statements
   - *if* statement
   - *switch* statement
3. Iteration Statements
   - *for* loop statement
   - *while* loop statement
   - *do-while* loop statement
4. Jump Statements
   - *break* statement
   - *continue* statement
   - *return* statement
   - *goto* statement

Reading

Stephen Prata
C++ Primer Plus, 5th Edition,
Chapter 5 and Chapter 6

SAMS Publishing, 2004.



Sum of Series

Write a C++ program to find the sum of the first *n* terms of the following series, where *n* is a number entered by the user:

$$-1 + \left(\frac{1}{3}\right)^2 - \left(\frac{1}{5}\right)^2 + \left(\frac{1}{7}\right)^2 \ldots$$

Internal Rate of Return

TVBH Corporation is considering an investment of $50 million in a capital project that will return after-tax cash flows of $16 million per year for the next four years plus another $20 million in Year 5. Calculate the Internal Rate of Return (IRR) that makes the Net Present Value (NPV) of all cash flows from the project equals to zero.

The decision rule for the IRR is as follow:

- Invest if $IRR > r$ (10%)
- Do not invest if $IRR < r$ (10%)

Simulating Standard Normal Random Variables

A standard normal variable has probability density function ($\mu = 0$ and $\sigma^2 = 1$)

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \qquad (x \in \mathbb{R})$$

Box-Muller Methodology

Step 1 Generates two random uniform numbers U_1 and U_2 distributed over $[0, 1]$

```
srand((int) time(NULL));
double runiform = rand() / (double) RAND_MAX;
```

Step 2 Sets $V_1 = 2U_1 - 1$; $V_2 = 2U_2 - 1$; and $S = V_1^2 + V_2^2$

Step 3 Returns to Step 1 if $S > 1$

Step 4 Obtains two standard normal random numbers:

$$X = V_1 * \sqrt{\frac{-2\ \log S}{S}} \qquad Y = V_2 * \sqrt{\frac{-2\ \log S}{S}}$$