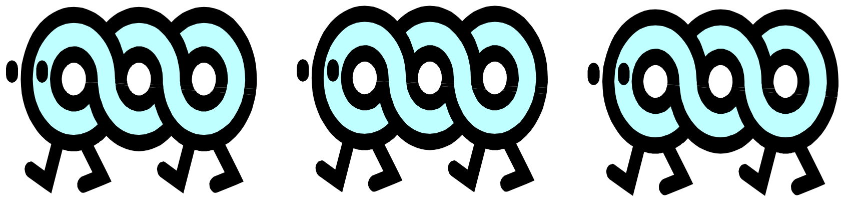
Lecture # 6

Repetition



Review

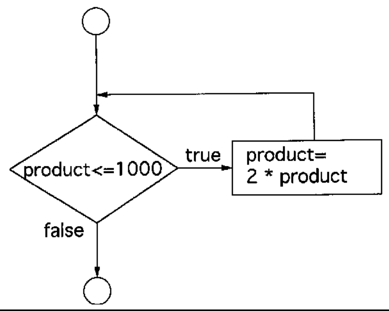
- If – Else Statements
- Comparison Operators
- Boolean Expressions
- Nested Ifs
  - Dangling Else
- Switch Statements

While loops

while

```
int product = 2;

while (product <= 1000)
  product = 2 * product;
```



Syntax for the While Statement
(Display 2.11 - page 76)

```
while (Boolean_Expression)
{
        while_statement1;
        while_statement2;
        while_statement3;
        …
}
```

Execution of While Statement

1. Boolean Expression is checked

2. Statement(s) in body is/are executed

Program Example of While Loop (Display 2.10 page 75)

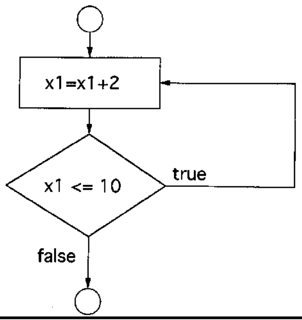
```
#include <iostream>
using namespace std;
int main( )
{
   int count_down;
   cout << "How many greetings do you want? ";
   cin >> count_down;
   while (count_down > 0)
   {
            cout << "Hello ";
            count_down = count_down - 1;
   }
   cout << endl;
   cout << "That's all!\n";
   return 0;
}
```

Do While Loops

do/while

```
int x1 = 2;

do
  x1 = x1+2;
while (x1 <= 10);
```



Syntax for Do While Statement
(Display 2.12 page 78)

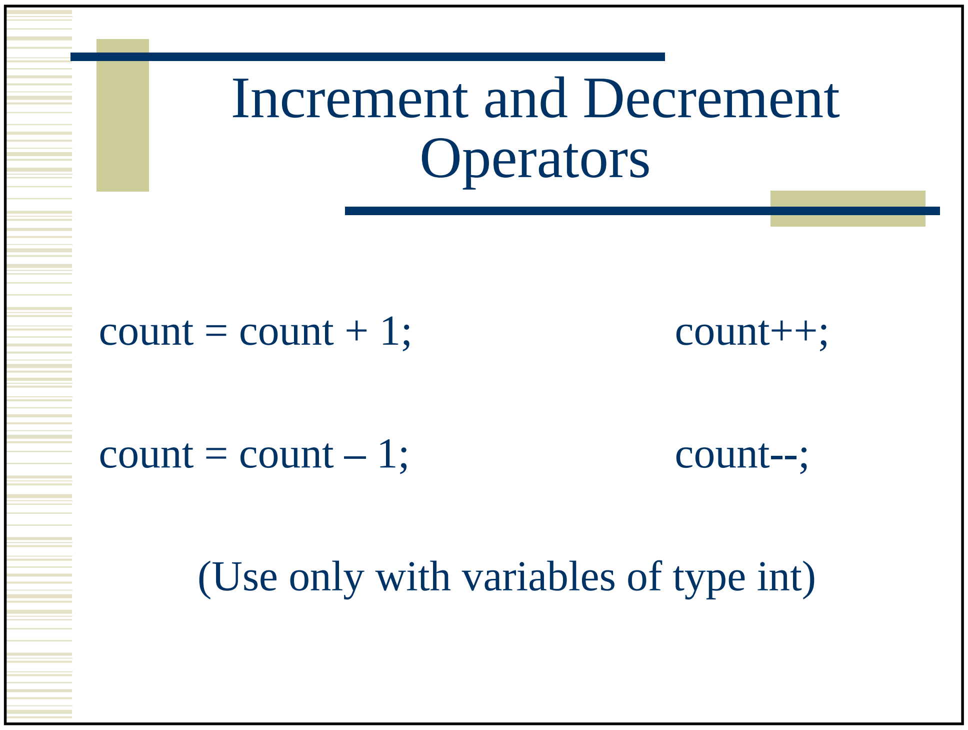
```
do
{
        DoWhile_Statement1;
        DoWhile_Statement2;
        DoWhile_Statement3;
        …
}
while (Boolean_Expression);
```

Execution of Do While Statement

1. Statement(s) in body is/are executed
2. Boolean Expression is checked

Program Example of Do While Loop (Display 2.13 page 79)

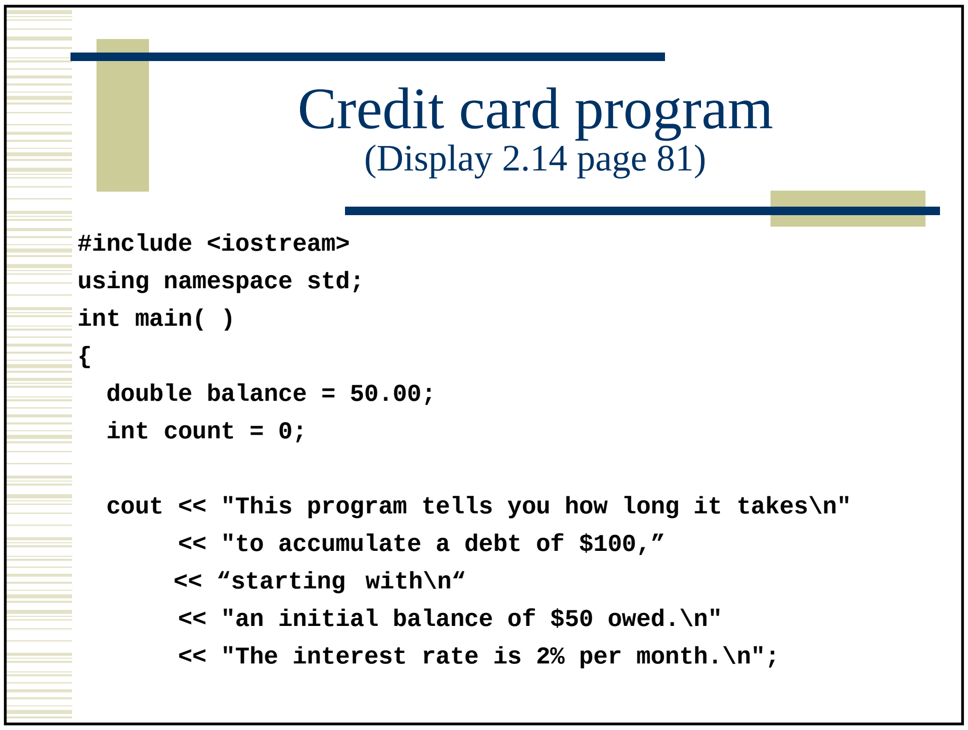
```
#include <iostream>
using namespace std;
int main( )
{
  char ans;
  do
  {
    cout << "Hello\n";
    cout << "Do you want another greeting?\n"
            << "Press y for yes, n for no,\n"
            << "and then press return: ";
    cin >> ans;
  } while (ans == 'y' || ans == 'Y');
  cout << "Good-Bye\n";
  return 0;
}
```



Increment and Decrement Operators

count = count + 1;	count++;
count = count – 1;	count--;

(Use only with variables of type int)



Credit card program

(Display 2.14 page 81)

```
#include <iostream>
using namespace std;
int main( )
{
  double balance = 50.00;
  int count = 0;

  cout << "This program tells you how long it takes\n"
       << "to accumulate a debt of $100,”
      << “starting with\n“
       << "an initial balance of $50 owed.\n"
       << "The interest rate is 2% per month.\n";
```

Credit card continued

```
while (balance < 100.00)
  {
    balance = balance + 0.02 * balance;
    count++;
  }

  cout << "After " << count << " months,\n";
  cout.setf(ios::fixed);
  cout.setf(ios::showpoint);
  cout.precision(2);
  cout << "your balance due will be $" << balance << endl;

  return 0;
}
```

Infinite Loops

```
x = 2;
while (x != 2)
{
   cout << x << endl;
   x = x + 2:
}
```

What is happening here?

How could you fix this?

Increment and decrement operators

- We’ve just seen how Number++ and Number – increment (or decrement) values of variables.
- Consider the following statement:

  number = number + count++;
- Is the value of count added to number before or after count is incremented?

New Increment and Decrement Operators

```
num = 5;
count = 1;

// num++ and Num-- increment or decrement after the expression
is evaluated
num = num + (count++);                    // num = 6

// ++num and --num increment or decrement before the expression
is evaluated
num = num + (++count);                    // num = 7
```

Increment Operator as an expression

```
while (count++ <= number_of_items)
  {
     cin >> calories_for_item;
     total_calories = total_calories
                  + calories_for_item;
  }
  cout << "Total calories eaten today = "
        << total_calories << endl;
  return 0;
```

Take a Break

for statement

Display 7.11 page 371

```
for (Initialization_Action, Boolean_Action, UpdateAction)
        Body_Statement;
```

Example:

```
        for (number = 100; number >= 0; number--)
            cout << number
                    << " Bottles of beer on the wall.\n";
```

For vs While

```
//for statement
for (number = 100; number >= 0; number--)
                Body_Statement;
```

```
// while equivalent
number = 100
while (number >= 0)
{
            Body_Statement;
            number--;
}
```

for Statement

Display 7.12 page 372

```
// Illustrates a for-loop.
#include <iostream>
using namespace std;

int main( )
{
    int sum = 0;

    for (int n = 1; n <= 10; n++)  //Note that the variable n is a local
        sum = sum + n;             //variable of the body of the for loop!

    cout << "The sum of the numbers 1 to 10 is "
         << sum << endl;
    return 0;
}
```

Extra Semicolon in a for statement

```
for (int count = 1; count < 10; count++);
         cout << “Hello\n”;
```

Causes a problem.

This is NOT an error.

The problem is that the null statement between the close parenthesis and the semicolon gets executed 10 times, then the cout << “Hello\n”; which was intended to be executed 10 times is executed only once.

What Kind of Loop to Use

The best answer to this is WAIT until design is completed.

Design using pseudocode, then as the pseudocode is translated into C++, this decision is easy.

A loop that involves a numeric control variable that changes by a fixed amount each iteration, then a for loop is indicated.

Most other situations a while loop is appropriate, unless you know that the loop is to be executed once regardless, then the do-while is appropriate.

Break statement

- We saw that the break statement exits a switch.
- The break statement also may be used in a loop to terminate the loop.
- In nested loops, a break statement in an inner loop ends only the inner loop.

Display 7.14: A break Statement - page 378

```
//Sums a list of 10 negative numbers.
    int number, sum = 0, count = 0;
    cout << "Enter 10 negative numbers:\n";
    while (++count <= 10)
    {
        cin >> number;
        if (number >= 0)
        {
            cout << "ERROR: positive number"
                 << " or zero was entered!\n"
            break;
        }
        sum = sum + number;
    }
    cout << sum << " is the sum of the first "
         << (count - 1) << " numbers.\n";
    return 0;
```

Designing loops

- Body
- Initialization statement
- Conditions to end the loop

Ending a loop

- List Headed by size (repeat “n” times)

for (i=1; i <= 10; i++)
while (i <= 10)

- Ask before iterating

while (response == ‘y’ || response == ‘Y’)

- List ended with a sentinel value

while (next != “\n”)

- Running out of input

while (in_stream >> next)

Nested loops

- **A loop body may contain statements of any kind, including another loop. A loop that contains another loop is called a nested loop.**
- **The next two sets of slides contain examples of nested loops.**
- **They do the same task, but the first is easier to understand than than the second.**

Display 7.15 Nicely Nested Loops (1 of 4) – page 385

```
//Determines the total number of green-necked vulture eggs
//counted by all conservationists in the conservation district.
#include <iostream>
using namespace std;
void instructions( );
void get_one_total(int& total);
//Precondition: User will enter a list of egg counts
//followed by a negative number.
//Postcondition: total is equal to the sum of all the egg counts.
int main( )
{
   instructions( );
   int number_of_reports;
   cout << "How many conservationist reports are there? ";
   cin >> number_of_reports;
```

29

Display 7.15 Nicely Nested Loops (2 of 4)

```
 int grand_total = 0, subtotal, count;
   for (count = 1; count <= number_of_reports; count++)
   {
      cout << endl << "Enter the report of "
            << "conservationist number " << count << endl;
      get_one_total(subtotal);
      cout << "Total egg count for conservationist "
          << " number " << count << " is "
          << subtotal << endl;
      grand_total = grand_total + subtotal;
   }

   cout << endl << "Total egg count for all reports = "
         << grand_total << endl;
   return 0;
}
```

30

Display 7.15 Nicely Nested Loops (3 of 4)

```
//Uses iostream:
void instructions( )
{
   cout << "This program tallies conservationist reports\n"
       << "on the green-necked vulture.\n"
       << "Each conservationist’s report consists of\n"
       << "a list of numbers. Each number is the count of\n"
       << "the eggs observed in one"
       << " green-necked vulture nest.\n"
       << "This program then tallies"
       << " the total number of eggs.\n";
}
```

Display 7.15 Nicely Nested Loops (4 of 4)

```
//Uses iostream:
void get_one_total(int& total)
{
   cout << "Enter the number of eggs in each nest.\n"
       << "Place a negative integer"
       << " at the end of your list.\n";
   total = 0;
   int next;
   cin >> next;
   while (next >= 0)
   {
      total = total + next;
      cin >> next;
   }
}
```

Display 7.16 Explicitly Nested Loops (1 of 3)

```
//Determines the total number of green-necked vulture eggs
//counted by all conservationists in the conservation district.
#include <iostream>
using namespace std;
void instructions( );
int main( )
{
   instructions( );
   int number_of_reports;
   cout << "How many conservationist reports are there? ";
   cin >> number_of_reports;
   int grand_total = 0, subtotal, count;
```

33

Display 7.16 Explicitly Nested Loops (2 of 3)

```
for (count = 1; count <= number_of_reports; count++)
  {
     cout << endl << "Enter the report of "
            << "conservationist number " << count << endl;
     cout << "Enter the number of eggs in each nest.\n"
            << "Place a negative integer"
            << " at the end of your list.\n";
     subtotal = 0;
     int next;
     cin >> next;
     while (next >= 0)
     {
        subtotal = subtotal + next;
        cin >> next;
     }
     cout << "Total egg count for conservationist "
         << " number " << count << " is "
         << subtotal << endl;
     grand_total = grand_total + subtotal;
  }
  cout << endl << "Total egg count for all reports = "
       << grand_total << endl;
  return 0;
}
```

34

Display 7.16 Explicitly Nested Loops (3 of 3)

```
//Uses iostream:
void instructions( )
{
   cout << "This program tallies conservationist reports\n"
       << "on the green-necked vulture.\n"
       << "Each conservationist’s report consists of\n"
       << "a list of numbers. Each number is the count of\n"
       << "the eggs observed in one"
       << " green-necked vulture nest.\n"
       << "This program then tallies"
       << " the total number of eggs.\n";
}
```

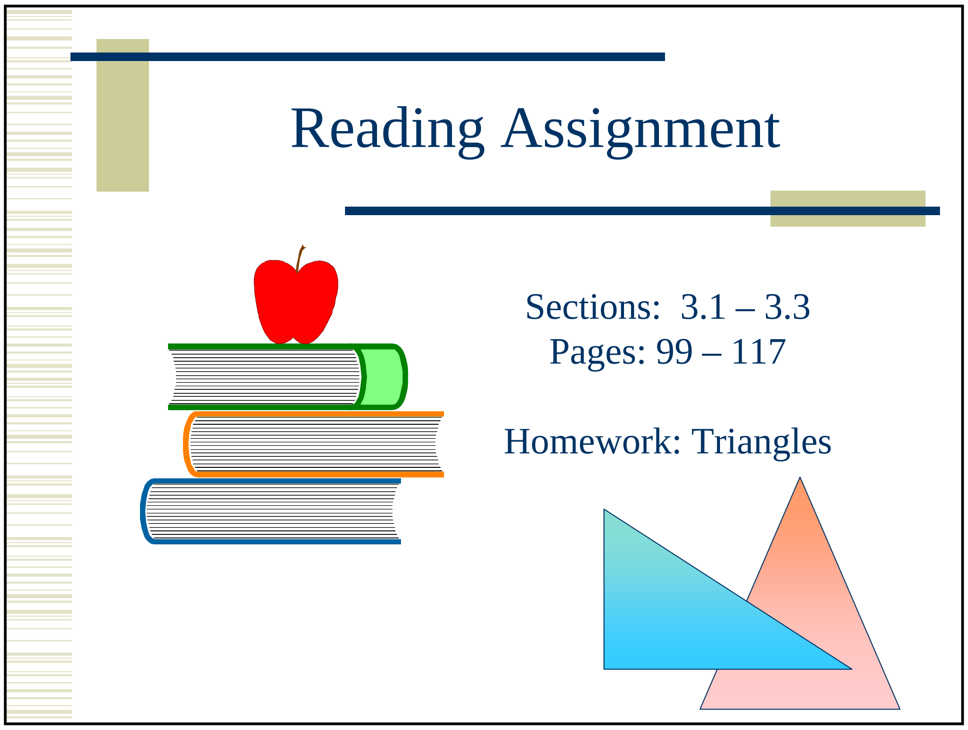
Debugging Loops

- A common loop error is the “off by one” error.
- The error is either running the loop one too many times or one too few times.
- A second error is an infinite loop.
- If your error isn’t one of the common errors more sophisticated testing will be required.
- You can use output statements to print out these variables.

Take a Break

Look at last week’s problem

Write a program that determines whether a meeting room is in violation of fire law regulations regarding the maximum room capacity. The program will read in the number of people that plan to attend the meeting. If the number of people is less than or equal to the maximum room capacity, the program announces that it is legal to hold the meeting and tells how many additional people may legally attend. If the number of people exceeds the maximum room capacity, the program announces that the meeting cannot be held as planned due to fire regulations and tells how many people must be excluded in order to meet the fire regulations. The room capacity is 75 and should be included in your program as a defined constant.



Reading Assignment

Sections: 3.1 – 3.3
Pages: 99 – 117

Homework: Triangles