

## + - Arithmetic Operators

- + Unary Do Not Change Sign
- Unary Change Sign

```
nValue = -5;
nValue = +(nValue - 1);
cout << nValue << endl;
nValue = -(nValue - 1);
cout << nValue << endl;
nValue = -(nValue - 1)+(-(+nValue));
cout << nValue << endl;
```

Copyright © 2003 R.M. Laurie 1

## + - Arithmetic Operators

- + Unary Do Not Change Sign
- Unary Change Sign

```
nValue = -5;
nValue = +(nValue - 1);
cout << nValue << endl;      // -6
nValue = -(nValue - 1);
cout << nValue << endl;      // 7
nValue = -(nValue - 1)+(-(+nValue));
cout << nValue << endl;      // -13
```

Copyright © 2003 R.M. Laurie 2

## ++ -- Arithmetic Operators

- ++ Increment (Postfix or Prefix)
- Decrement (Postfix or Prefix)

```
nCount++;
// Equivalent nCount=nCount+1;
nCount--;
// Equivalent nCount=nCount-1;
```

For now use Postfix increment.

CMIS 140 will cover in more detail.

Copyright © 2003 R.M. Laurie 3

## Increment Example

```
int nCount=0;
while(nCount<5)
{
    cout << nCount << ' ';
    nCount++;
}
cout << endl;
while(nCount < 10)
    cout << nCount++ << ' ';
cout << endl;
while(nCount < 15)
    cout << ++nCount << ' ';
```

Copyright © 2003 R.M. Laurie 4

## Increment Example

```
0 1 2 3 4
5 6 7 8 9
11 12 13 14 15
```

Copyright © 2003 R.M. Laurie | 5

## Decrement Example

```
int nCount=15;
while(nCount>=10)
{
    cout << nCount << ' ';
    nCount--;
}
cout << endl;
while(nCount > 5)
    cout << --nCount << ' ';
cout << endl;
while(nCount > 0)
    cout << nCount-- << ' ';
```

Copyright © 2003 R.M. Laurie | 6

## Decrement Example

```
15 14 13 12 11 10
8 7 6 5
5 4 3 2 1
```

Copyright © 2003 R.M. Laurie | 7

## Operators Precedence

```
( )
float() int() sizeof() ! ++ -- + -
* / %
+ -
<< >>
< <= > >=
== !=
&&
||
=
```

Copyright © 2003 R.M. Laurie | 8

## File Input and Output

- File Input allows a program to read text files
  - The text data can be assigned to variables
  - No prompting is necessary
  - Usually used for batch type processing
- File Output allows a program to write text data to a text file.
  - Text strings can be written to a text file
  - Variable contents can be written to a text file
  - Usually used to record results or batch output.
  - Use .txt or .dat file extension

Copyright © 2003 R.M. Laurie 9

## File Input and Output

- File Input or Output requires use of library file:
 

```
#include <fstream>
```
- Declare File Streams using file identifier:
 

```
ifstream ifileDataIn;
ofstream ofileResults;
```

  - *Note: Can not both read and write to same file*
- Open File:
 

```
ifileDataIn.open("Transactions.txt");
ofileResults.open("Results.txt");
```

Copyright © 2003 R.M. Laurie 10

## File Read and Write

- File Read
  - Use insertion extraction >> operator
  - Instead of standard input cin use file identifier
- Reading Data from File:
  - `ifileDataIn >> fPrice;`
  - `ifileDataIn >> nQuantity >> fCost >> nYear;`
- File Write
  - Use insertion insertion << operator
  - Instead of standard output cout use file identifier
- Writing Data to File:
  - `ofileResults << nBlackChips << ' '`
  - `ofileResults << nBlueChips << "Blue/n";`

Copyright © 2003 R.M. Laurie 11

## Naming Convention Additions

PREFIX	DATA TYPE
c	Char (8 bits)
n	Integer (16 bits)
f	Float (32 bits)
un	Unsigned integer
ln	Long Integer (32 bits)
df	Double Float (32 bits)
sz	String terminated with zero byte
ifile	input file
ofile	output file

Copyright © 2003 R.M. Laurie 12

## Writing to File Example 1

```
#include <iostream>
#include <fstream>

using namespace std;

int main(void)
{
    ofstream ofileResults;
    ofileResults.open("Results.txt");
    ofileResults << "This is a Simple Test";
    cout << "Done\n";
    return 0;
}
```

Copyright © 2003 R.M. Laurie 13

## Writing to File Example 2

```
#include <iostream>
#include <fstream>
using namespace std;
int main(void)
{
    int nNum = 0;
    ofstream ofileLoop;
    ofileLoop.open("Results.txt");
    ofileLoop << "This is a Simple Test\n";
    while(nNum <= 10)
        ofileLoop << ++nNum << ' ';
    cout << "Done\n";
    return 0;
}
```

```
This is a Simple Test
1 2 3 4 5 6 7 8 9 10 11
```

Copyright © 2003 R.M. Laurie 14

## Testing for File Open Error (pp. 175, 241)

- ofileResults object will return zero when operation failed and nonzero if succeeded.

```
ofstream ofileLoop;
ofileLoop.open("a:/Results.txt");
if (!ofileLoop)
{
    cout << "Can't Open File.\n";
    return 1;
}
ofileLoop << "This is a Simple Test\n";
return 0;
}
```

Copyright © 2003 R.M. Laurie 15

## Looping

- Loops classified as:
  - Count Controlled Loop  
Loop executes specified number of times  
while(nCount < 10)  
{ }
  - Event Controlled Loop  
Terminates when a test condition is met  
while(cQuestion == 'C' || cQuestion == 'c')  
{ }

Copyright © 2003 R.M. Laurie 16

## Counter Controlled Loop

- Requires Counter Initialization  
nCount = 10;
- Requires Counter Test  
while(nCount < 10)
- Requires Counter Increment  
nCount++;

Copyright © 2003 R.M. Laurie 17

## Event Controlled Loop

- Sentinel Controlled Loops
  - Often used to read lists of data until a sentinel (trailer mark) is encountered to signal the end of data.
- End-of-File Controlled Loops
  - Often used to read entire file contents
  - while(ofileResults){ }
- Flag Controlled Loops
  - Generally uses a Boolean Flag Value

Copyright © 2003 R.M. Laurie 18

## Designing Loops

See page 292

1. What condition will end the loop?
2. How should condition be initialized?
3. How should condition be updated?
4. What is the process being repeated?
5. How should the process be initialized?
6. How should the process be updated?
7. What is the state of the program on exiting the loop?

Copyright © 2003 R.M. Laurie 19

## Nested Loops

```
int main()
{
    int nI=0, nJ;
    cout << "MULTIPLICATION TABLE: 5X10\n\n";
    cout << " X |  1  2  3  4  5  6  7  8  9
10\n";
    cout << "-----";
    while(nI <= 5)
    {
        cout << endl << setw(2) << nI << " |";
        nJ=0;
        while(++nJ <= 10)
            cout << setw(4) << nI*nJ;

        nI++;
    }
    return 0;
}
```

Copyright © 2003 R.M. Laurie 20

## Nested Loops Output

MULTIPLICATION TABLE: 5X10

X	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50

Copyright © 2003 R.M. Laurie | 21

## Object Oriented Design - 1

- The difference between C and C++
- Software design methodology in which solution is expressed in terms of objects.
- Objects are self contained entities composed of data and operations.
- Class = Predefined type from which objects can be created.
- cin is object of the class istream
- cout is object of the class ostream
- iostream defines the classes istream and ostream
- fstream defines the classes ifstream and ofstream

Copyright © 2003 R.M. Laurie | 22

## Object Oriented Design - 2

- Operations that can be performed on an object are called member functions.
- Member functions operator denoted '.'  

```
cin.ignore(100, '\n');
```

```
cin.get(cYesNo);
```
- Lot's more to come in CMIS140!
  - Functions
  - Data Structures
  - Object Oriented Design
- That's all for now!

Copyright © 2003 R.M. Laurie | 23