

### Problem Solving Phase:

- Write Program Specifications Report
  - Analysis of requirements
  - Describe program specifications
    - Describe what the goals of the program
    - Describe appearance of input and output
  - Algorithm Design
    - Mathematical Analysis and Algorithm
    - Flow Chart to describe event sequencing
  - Verify algorithm
    - Test with known data
    - Solve manually

Copyright © 2001 R.M. Laurie 1

### Implementation Phase:

- Translate Algorithm into Code
  - Must follow syntax of programming language C++
  - Compiler will detect syntax errors
- Test Program
  - Test with known data
  - Detects program logic errors
  - Often requires several iterations
  - May require re-evaluation of specifications and algorithms

Copyright © 2001 R.M. Laurie 2

### Maintenance Phase:

- Program is used for its intended purpose.
- Program bugs may be found by users for untested input data cases or possibly in the results output.
- Requires a method of addressing problems as they arise.
- Program is often revised and improved, based on user feedback.

Copyright © 2001 R.M. Laurie 3

### Presumed Knowledge - 1

- Number Systems and Code Conversion
  - Binary, Decimal, Hexadecimal, ASCII
- Declarations
- Variable Initialization
- Constants
- Identifier Naming Requirements
- Identifier Naming Convention
  - Continue use of Hungarian Notation

Copyright © 2001 R.M. Laurie 4

### Naming Convention

1. CONSTANTS are UPPER CASE
2. UserFunctions are in TitleCase
3. Variables use prefix and remainder in TitleCase

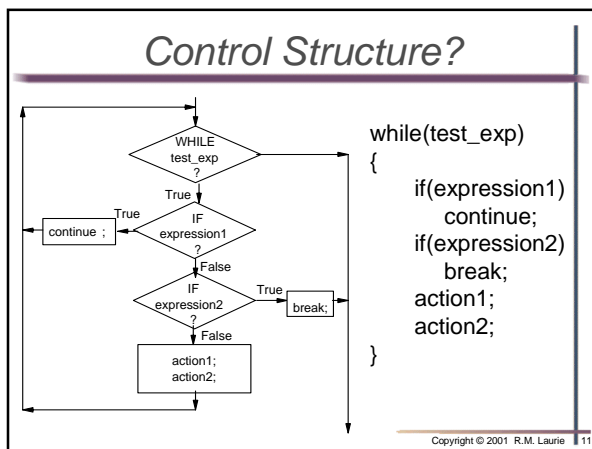
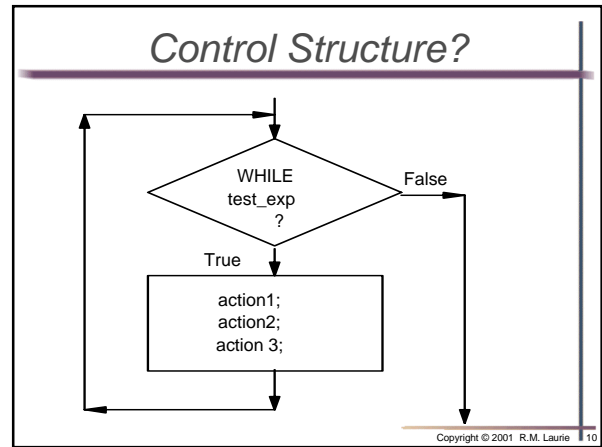
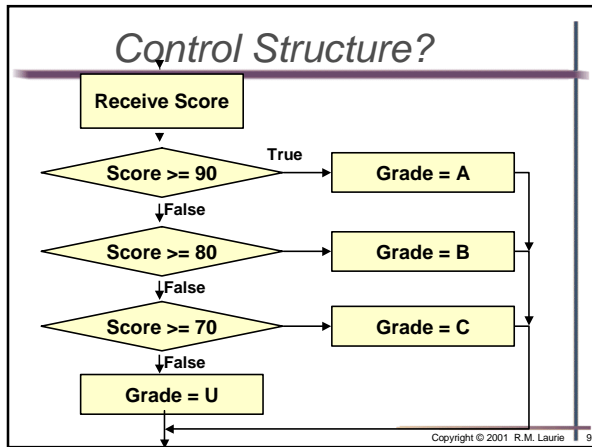
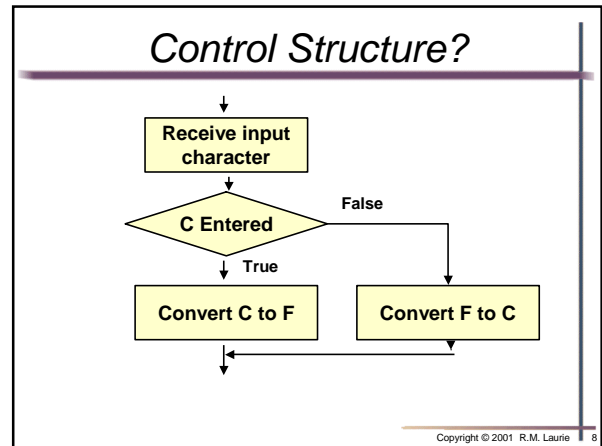
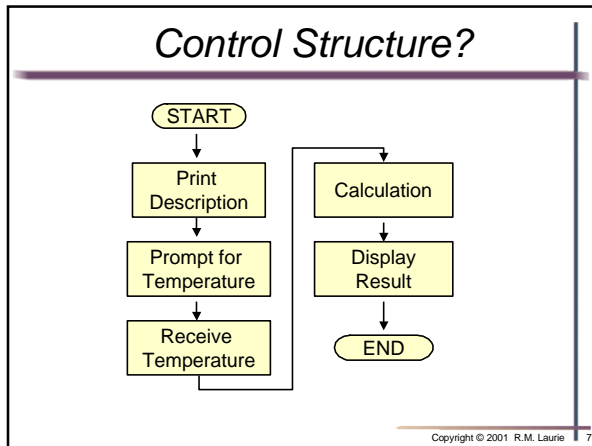
PREFIX	DATA TYPE
c	Char (8 bits)
n	Integer (32 or 16 bits)
f	Float (32 bits)
un	Unsigned integer
ln	Long Integer (32 bits)
df	Double Float (32 bits)
p	Pointer
sz	String terminated with zero byte
ifile	input file
ofile	output file

Copyright © 2001 R.M. Laurie 5

### Presumed Knowledge - 2

- Program Control Structures
  - Sequential Control Structure
  - Selection (Branching) Control Structure
    - Relational and Logical Operators
    - if( ) else if( ) else
  - Repetition (Loop) Control Structure
    - Relational and Logical Operators
    - while( ) continue; break;

Copyright © 2001 R.M. Laurie 6



### Presumed Knowledge - 3

#### Operators Precedence

```

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

Copyright © 2001 R.M. Laurie 12

### Presumed Knowledge - 4

#### Program I/O

- Stream I/O Functions are available in the `iostream` header file.
  - `cout` is used for information output
  - `cin` is used for data input
- Data format functions are available in the `iomanip` library file
  - `setw(#)` sets the field width # of the next data
  - `setprecision(#)` works only with floating point
- Text string special characters
  - `\n` = newline     `\r` = carriage return     `\t` = tab
  - `\a` = bell     `\"` = double quote     `\?` = question
  - `\\` = backslash     `\'` = single quote     `\x###` = hex

Copyright © 2001 R.M. Laurie 13

### Presumed Knowledge - 5

#### Reading / Writing to Files

Requires use of header file:

```
#include <fstream>
```

Declare File Streams:

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

Open File:

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

Copyright © 2001 R.M. Laurie 14

### Reading / Writing to Files

Reading Data from File:

```
ifileDataIn >> fPrice;
ifileDataIn >> nQuantity >> fCost >> nYear;
```

Writing Data to File:

```
ofileResults << nBlackChips << ' ';
ofileResults << nBlueChips << "Blue "
<< "Chips \n";
```

Copyright © 2001 R.M. Laurie 15

### Writing to File Example

```
#include <fstream>
#include <iostream>
int main()
{
    int nValue;
    ofstream ofileResults;
    ofileResults.open("Results.txt");

    ofileResults << "Hello This is a Test";

    cout << "Done";
    return 0;
}
```

Copyright © 2001 R.M. Laurie 16