

## Problem Solving Phase

---

- Write Program Specifications Report
  - Analysis of requirements
  - Program specifications description
    - 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

## Program Specifications Report

---

- Analysis
  - Temperature conversion from degrees Celsius to degrees Fahrenheit
- Program Specifications Description
  - The program will accept an entered Celsius temperature and convert it to Fahrenheit.
  - The program only needs to do this once before exiting.
  - Any real number may be entered.
  - The appearance of input prompting and the results will be displayed as follows.

Copyright © 2001 R.M. Laurie | 2

## Program Specifications (Cont'd)

---

### The appearance of input and output display

```

This program converts a temperature from
degrees Celsius to degrees Fahrenheit.

Enter the temperature in degrees Celsius

>25

Results:
25 C = 77 F
            
```

Copyright © 2001 R.M. Laurie | 3

## Algorithm Design - Mathematical

---

- Mathematical Description
  - Boiling point
    - F = 212
    - C = 100
  - Freezing point
    - F = 32
    - C = 0

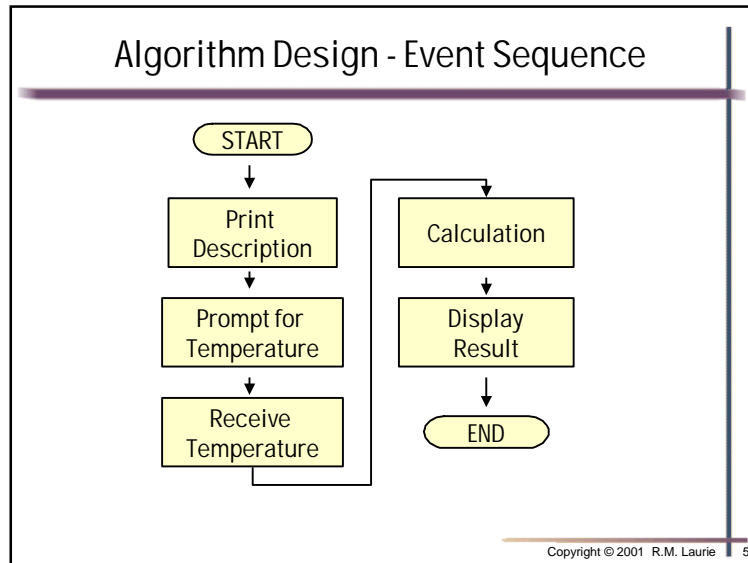
**Y = MX + B**

$$F = (180 / 100) C + 32$$

$$= (9/5) C + 32$$

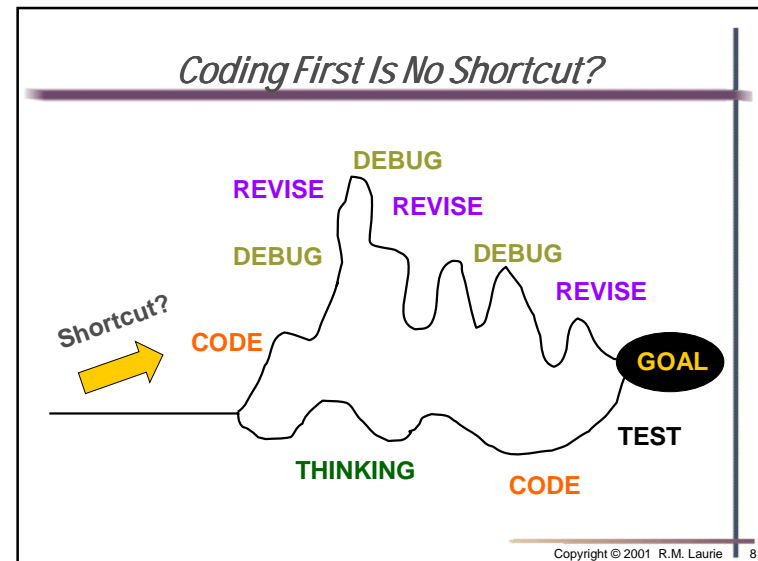
$$= 1.8 C + 32$$

Copyright © 2001 R.M. Laurie | 4



- ### Verify Algorithm
- Testing with known data
    - Boiling point  
F = 212      C = 100
    - Freezing point  
F = 32      C = 0
    - Collect Data
      - ◆ Bank thermometer
      - ◆ Radio weather report
  - Solve manually by hand using calculator
- Copyright © 2001 R.M. Laurie | 6

- ### Problem Solving Phase - Review
- Write Program Specifications Report
    - Analysis of requirements
    - Program specifications description
      - ◆ 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 | 7



## Implementation Phase

- Translate Algorithm into Code
  - Create source code file that follows syntax of C++ programming language
  - Compile to 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 9

## Build Processing

- Preprocessing
  - Removes all comments from source code
  - Handles all preprocessor directives
    - Begin with the # character
    - #include <iostream>
      - include file contents into build processing
      - < > Looks for file in standard include directory
      - file called iostream has library functions
- Compiling
  - Converts source code to machine code contained in object file *filename.obj*
- Linking
  - Combines machine code of all object files
  - Creates one executable file *filename.exe*

Copyright © 2001 R.M. Laurie 10

## Using <iostream> Library

- Standard C and Pre-Standard C++
 

```
#include <iostream.h>
int main(void)
```
- ISO/ANSI C++ syntax (1999 and later)
 

```
#include <iostream>
using namespace std;
int main()
```
- Defined in iostream are the functions:
 

```
cout, endl, cin
```

Copyright © 2001 R.M. Laurie 11

## Using <string> Library

- String is series of characters enclosed in double quotes
 

```
"Hello" "Year 2000" "1234"
```
- String is not a standard data type
- Provided in the C++ standard library <string>
 

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string szName = "Bob Laurie";

```
- Concatenation string operator +
 

```
szName = szFist + " " + szLast;
```

Copyright © 2001 R.M. Laurie 12

## Operators Precedence (Highest to Lowest)

- ( ) Parenthesis defines order of operation
- int( ) float( )** sizeof( )
- \* / % Multiplication, Division, Modulus
- + - Addition, Subtraction
- << >> Insertion, Extraction
- = Assignment

Copyright © 2001 R.M. Laurie 13

## Type Coercion/Casting

- Often computations require using variables of different data types.
- int float conversions
- Type Casting is the Explicit conversion of a value to a given type
- Type Coercion is the Implicit (automatic) type conversion of a value
- Use Casting if any doubt what type result is from expression

Copyright © 2001 R.M. Laurie 14

## Type Coercion Rules

- Whenever an integer value and floating point value are joined by operator, implicit coercion occurs:
  - Integer is coerced to floating point
  - Operation is performed.
  - Result is a floating point.

Copyright © 2001 R.M. Laurie 15

## Type Coercion Examples

- Result = Chips \* 5;
- Tax = Winnings \* 0.28;
- Result = Winnings \* 28 / 100;
  
- fResult = nChips \* 5;
- fTax = nWinnings \* 0.28;
- fResult = nWinnings \* 28 / 100;
- nTaxRate = fTax / fEarnings \* 100;

Copyright © 2001 R.M. Laurie 16

## Type Casting

- Definition: Explicit conversion from one data type to another.
- Example:
 

```
fCircum = float(2 * nRad) * fPI
fAvg = float(10 + 5) / 2 ==> fAvg = 7.50
fAvg = float(nVal1 + nVal2) / float(2)
nVal = int(fAvg + 33.33)
```

Copyright © 2001 R.M. Laurie 17

## Program Output

- C++ does not have any formal way to input data or output information for the user.
- However you may use library routines to communicate with the user.
- Stream I/O Functions are available in the `iostream.h` header file.
  - You must include this header file before `main()`
  - `cout` is used for information output
  - `cin` is used for data input
- Data format functions are available in the `iomanip` library file

Copyright © 2001 R.M. Laurie 18

## cout = Program Output - 2

- `endl`

```
cout << "123" << endl << "abc" << endl;
```
- Text string special characters
 

```
\n = newline    \r = carriage return  \t = tab
\a = bell      \" = double quote     \? = question
\\ = backslash \' = single quote     \x### = hex
```

  - Examples:
 

```
cout << "123" << endl << "abc\n";
cout << "Hello\t" << "I'm "
<< szFirst + ' ' + szLast + "Bob\n\a";
```

Copyright © 2001 R.M. Laurie 19

## Output Field Width = `setw()`

- `setw(#)` sets the field width # of the next data item inserted into `cout` and right justifies output
- Will work with both numbers and strings of characters, but not single characters
- `#include <iomanip>` // to use `setw`

```
cout << "=" << setw(5) << nPerc << "%\n";
Output:    =   132%
```

```
cout << "$" << setw(8) << fCashPaid;
Output:    $   34.55
```
- `cout << "Answer: " << setw(5) << "NO";`

```
Output: Answer:   NO
```
- See p. 125 to 128 for examples

Copyright © 2001 R.M. Laurie 20

## Output Field Width Example

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{
    int nA=1, nB=12, nC=123;
    cout << "$" << setw(4) << nA << endl;
    cout << "$" << setw(4) << nB << endl;
    cout << "$" << setw(4) << nC
        << "\n-----\n"
        << "$" << setw(4) << nA + nB + nC;
    return 0;
}
```

```
$  1
$ 12
$ 123
-----
$ 136
```

Copyright © 2001 R.M. Laurie 21

## Cout Flag Specifiers

- Used to configure output format flags
- #include <iostream> must be used
- cout << fixed << fCash; // fixed point notation
- cout << showpoint << fCash; // show decimal point
- cout << scientific << fDistance; // scientific notation
- cout << showpos << fProfit; // show positive sign
- cout << hex << nVal; // show as hexadecimal

Copyright © 2001 R.M. Laurie 22

## cout.setf Flag Example

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{
    float fA=1, fB=12, fC=123;
    cout << fixed << right << showpoint;
    cout << "$" << setw(12) << fA << endl;
    cout << "$" << setw(12) << fB << endl;
    cout << "$" << setw(12) << fC
        << "\n-----\n"
        << "$" << setw(12) << (fA + fB + fC);
    return 0;
}
```

```
$ 1.000000
$ 12.000000
$ 123.000000
-----
$ 136.000000
```

Copyright © 2001 R.M. Laurie 23

## Setting Floating Point Precision

- setprecision(#) works only with floating point values and sets the number of digits (#) to the right of decimal point.
- This precision setting remains until changed for all floating point values
- #include <iomanip> must be used
- Must include the following two statements before any floating point values are output. Otherwise it may display in scientific notation.  
cout.setf(ios::fixed);  
cout.setf(ios::showpoint);
- For Examples see program on pages 127 and 128

Copyright © 2001 R.M. Laurie 24

### Floating Point Precision Example

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{
    float fA=1, fB=12, fC=123;
    cout << fixed << right << showpoint;
    cout << "$" << setw(9) << setprecision(2)
        << fA << endl;
    cout << "$" << setw(9) << fB << endl;
    cout << "$" << setw(9) << fC
        << "\n-----\n"
        << "$" << setw(9) << fA + fB + fC;
    return 0;
}
```

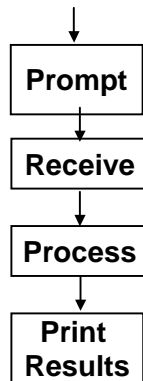
```
$    1.00
$   12.00
$  123.00
-----
$  136.00
```

### Numerical Program Input

- The cin function with the extraction operator (>>) receives ASCII text from the keyboard.
- This text is sent converted automatically to the variable type
- cin skips white space characters
- Examples:  
`cin >> nScore;`  
`cin >> fTemp;`

### Program Prompt and Receive

- To make your programs user friendly it is best to prompt for data entry by displaying a message.
- Receive a value from keyboard.
- Process data to create results.
- Finally display entered data with results.



### 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.