C++ Background

• Introduced by Bjarne Stroustrup of AT&T’s Bell Laboratories in mid-1980’s
• Based on C
• Supports object-oriented programming
• 1998 Standard
2.1 C++ Language Elements

- Comments
- Compiler directives
- Function `main`
- Declaration statements
- Executable statements
Comments

• // symbols indicate a line comment - apply to just the rest of the line
• Block comments start with /* and end with */ - apply to as many lines as you like
• Used to describe the code in English or provide non-code information
• E.g. to include the name of the program or the author’s name
// miles.cpp
// Converts distance in miles to kilometers.

#include <iostream>
using namespace std;

int main() // start of main function
{

    const float KM_PER_MILE = 1.609; // 1.609 km in a mile
    float miles, // input: distance in miles
           kms; // output: distance in kilometers

    // Get the distance in miles.
    cout << "Enter the distance in miles: ";
    cin >> miles;

    // Convert the distance to kilometers.
    kms = KM_PER_MILE * miles;

    // Display the distance in kilometers.
    cout << "The distance in kilometers is " << kms << endl;

    return 0; // Exit main function
}
#include <filename>

- Compiler directive
- Includes previously written code from a library into your program
- E.g.
  
  ```
  #include <iostream>
  ```
  
  has operators for performing input and output within the program
- Libraries allow for code reuse
using namespace std;

• Indicates to compiler that this program uses objects defined by a standard namespace called std.
• Ends with a semicolon
• Follows #include directives in the code
• Must appear in all programs
Function `main`

```c
int main ( )
{
    // function body
}
```
Function **main**

- Exactly one **main** function per program
- A function is a collection of related statements that perform a specific operation
- **int** indicates the return type of the function
- ( ) indicates no special information passed to the function by the operating system
Types of Statements

• Declaration statements - describe the data the function needs:
  
  const float KM_PER_MILE = 1.609;
  float miles,
  kms;

• Executable statements - specify the actions the program will take:
  
  cout << "Enter the distance in miles: ";
  cin >> miles;
2.2 Reserved Words (Keywords)

• Have special meaning in C++
• Cannot be used for other purposes
<table>
<thead>
<tr>
<th>Reserved Words</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const</code></td>
<td>Constant; indicates a data element whose value cannot change</td>
</tr>
<tr>
<td><code>float</code></td>
<td>Floating point; indicates that a data item is a real number</td>
</tr>
<tr>
<td><code>include</code></td>
<td>Preprocessor directive; used to insert a library file</td>
</tr>
<tr>
<td><code>int</code></td>
<td>Integer; indicates that the main function returns an integer value</td>
</tr>
<tr>
<td><code>namespace</code></td>
<td>Region where program elements are defined</td>
</tr>
<tr>
<td><code>return</code></td>
<td>Causes a return from a function to the unit that activates it</td>
</tr>
<tr>
<td><code>using</code></td>
<td>Indicates that a program is using elements from a particular namespace</td>
</tr>
</tbody>
</table>
Identifiers

• Names for data and objects to be manipulated by the program
• Must begin with a letter or underscore (not recommended)
• Consist only of letters, digits, and underscores
• Cannot be reserved word
• Upper and lower case significant
# Identifiers

## Identifier Use

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>cin</td>
<td>C++ name for standard input stream</td>
</tr>
<tr>
<td>cout</td>
<td>C++ name for standard output stream</td>
</tr>
<tr>
<td>km</td>
<td>Data element for storing distance in kilometers</td>
</tr>
<tr>
<td>KM_PER_MILE</td>
<td>Conversion constant</td>
</tr>
<tr>
<td>miles</td>
<td>Data element for storing distance in miles</td>
</tr>
<tr>
<td>std</td>
<td>C++ name for the standard namespace</td>
</tr>
</tbody>
</table>
2.3 Data Types

• Defines a set of values and operations that can be performed on those values

• integers
  – positive and negative whole numbers, e.g. 5, -52, 343222
  – short, int, long
  – represented internally in binary
  – predefined constants INT_MIN and INT_MAX
Data Types (con’t)

• Floating point (real)
  – number has two parts, integral and fractional
  – e.g. 2.5, 3.66666666, -.000034, 5.0
  – float, double, long double
  – stored internally in binary as mantissa and exponent
  – 10.0 and 10 are stored differently in memory
Data Types (con’t)

• Boolean
  – named for George Boole
  – represent conditional values
  – values: true and false
Data Types (con’t)

• Characters
  – represent individual character values
    E.g. ‘A’ ‘a’ ‘2’ ‘*’ ‘”’ ‘ ‘
  – stored in 1 byte of memory
  – special characters: escape sequences
    E.g. ‘\n’ ‘\b’ ‘\r’ ‘\t’ ‘\\’
**string Class**

- Strings not built-in, but come from library
- Classes extend C++
- `string` literal enclosed in double quotes
  
  E.g.: “Enter speed: “ “ABC” “B” “true” “1234”

- `#include <string>`
  - for using string identifiers, but not needed for literals
Variable Declarations

• Set aside memory with a specific name for the data and define its values and operations
• The value can change during execution
• `type identifier-list;`
• E.g.: `float x, y;`
  `int me, you;`
  `float week = 40.0;`
  `string flower = “rose”;`
Constant Declarations

- Memory cells whose values cannot change once set
- `const` type constant-identifier = value;
- E.g.:
  ```
  const float KM_PER_MILE = 1.609;
  ```
- Often identified by using all upper case name
Listing 2.2 Printing a welcoming message

```c++
// File: hello.cpp
// Displays a user’s name

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

int main()
{
    char letter1, letter2; // input and output: first two initials
    string lastName; // input and output: last name

    // Enter letters and print message.
    cout << "Enter 2 initials and a last name: ";
    cin >> letter1 >> letter2 >> lastName;
    cout << "Hello " << letter1 << ". " << letter2 << ". " << lastName << ";
    cout << "We hope you enjoy studying C++." << endl;

    return 0;
}
```

Enter 2 initials and a last name: EB Koffman
Hello E. B. Koffman! We hope you enjoy studying C++.
2.4 Executable Statements

• Assignment statements
• Input statements
• Program output
• The return statement
Figure 2.2  Memory (a) before and (b) after execution of a program

(a) 

memory

machine language
miles-to-kms
conversion program

miles

?

kms

?

(b) 

memory

machine language
miles-to-kms
conversion program

miles

10.00

kms

16.09
Assignment Statements

• `variable = expression;`

• E.g.:

```kms = KM_PER_MILE * miles;```
Input Statements

• Obtain data for program to use - different each time program executes
• Standard stream library \texttt{iostream}
• \texttt{cin} - name of stream associated with standard input device (keyboard by default)
• \textbf{Extraction} operator (\texttt{>>})
• E.g.:

  \begin{verbatim}
  cin >> miles;
  cin >> age >> firstInitial;
  \end{verbatim}
In Listing 2.2:

\[ \text{cin >> letter1 >> letter2 >> lastname;} \]

has the effect:
Figure 2.6  Effect of

\texttt{cin >> letter1 >> letter2 >> lastname;}

Characters entered:

\begin{align*}
\text{letter1} & \rightarrow \text{E} \\
\text{letter2} & \rightarrow \text{B} \\
\text{lastName} & \rightarrow \text{Koffman} \\
\text{EBKoffman} & \rightarrow \text{EBKoffman}
\end{align*}
Program Output

• Used to display results of program
• Also standard stream library **iostream**
• **cout** - name of stream associated with standard output device (monitor by default)
• **Insertion** operator (<<) for each element
  ```
  cout << data-element;
  ```
Program Output

• cout statement can be broken across lines
• Strings cannot be broken across lines
• Prompt messages used to inform program user to enter data
• Screen cursor is a moving marker indicating the position of where the next character will be displayed
• endl (or ‘\n’) causes a new line in output
Output Example

```cpp
cout << "The distance in kilometers is " << kms << endl;
```

If variable kms has value 16.09, the output is:

The distance in kilometers is 16.09
The **return** Statement

- Last line of main function is typically
  ```
  return 0;
  ```

- This transfers control from the program to the operating system, indicating that no error has occurred
Listing 2.3  General Form of a C++ Program

```cpp
compiler directives
using namespace std;

int main()
{
    declaration statements
    executable statements
}
```
Program Style

• Use of spacing
  – one statement per line
  – blanks after comma, around operators
  – in between some lines of code for readability

• Use of comments
  – header section
  – document algorithm steps
  – describe difficult code
Program Style

• Naming conventions for identifiers
  – Variables in all lower case, with initial capital letter for additional words. No underscore.
  – Constants in all upper case, with underscores between words.
2.6 Arithmetic Expressions

- **int** data type
  - +  -  *  /  %

- **Integer division examples - result is integer**
  - $15 / 3 = 5$
  - $15 / 2 = 7$
  - $0 / 15 = 0$
  - $15 / 0$ undefined
Modulus for Integers

- Used *only* with integers
- Yields remainder - the result is integer
- Examples:
  - $7 \% 2 = 1$
  - $299 \% 100 = 99$
  - $49 \% 5 = 4$
  - $15 \% 0$ undefined
  - $15 \% -7$ system dependent
Mixed-type Expressions

• E.g.: 4.6 / 2 evaluates to 2.3
• Rule: *when an integer and a floating point operand are combined by an operator, the integer gets converted to the floating point type*
• **Caveat:** this rule is dependent on operator precedence rules
Mixed-type Assignments

• If the variable on left side of assignment is of different type than the type of the evaluated expression on the right side of =, the result of the expression must be converted to the appropriate type
Mixed-type Assignment Examples

```c
float a, b, x;
int m, n;

a = 10; // result is 10.0 stored in a
b = 3.5;
m = 5;
n = 10;
x = m / n; // result is 0 assigned to x
m = b * 3; // result is 10 assigned to m
```
Order of Operator Precedence

Highest

( ) nested expressions evaluated inside out

unary +, -

*, /, %

binary +, -

Lowest

Associativity

Warning: watch out for the types of operands and the type of the result from evaluating each operand!
Step-by-Step Expression Evaluation

\[ \text{area} = \pi \times \text{radius} \times \text{radius} \]

\[ \begin{array}{c}
3.14159 \\
2.0 \\
2.0 \\
\hline
6.28318 \\
\hline
12.56636
\end{array} \]
Figure 2.10 Evaluation for
\[ z - (a + b / 2) + w \times -y; \]
Figure 2.11 Evaluation tree for \( m = x + k / 2 \):
Mathematical Formulas in C++

• $a = bc$  \textit{not valid C++ syntax}
  
  Must use $*$ operator
  
  $a = b * c$;

• $m = \frac{y - b}{x - a}$
  
  Must use $( )$ and $/$
  
  $m = (y - b) / (x - a)$;
Case Study: Coin Collection

• Problem statement
  Saving nickels and pennies and want to exchange these coins at the bank so need to know the value of coins in dollars and cents.

• Analysis
  – Need to count of nickels and pennies separately
  – Determine total value in cents
  – Use integer division by 100 to calculate dollars
  – Use modulus (%) to get remaining cents
Case Study: Data Requirements

• Problem input
  – string name
  – integer nickels
  – integer pennies

• Problem output
  – integer dollars
  – integer change

• Additional program variables
  – integer totalCents
Case Study: Formulas

- One dollar equals 100 pennies
- One nickel equals 5 pennies
Case Study: Design - Algorithm

1. Read in your niece’s first name
2. Read in the count of nickels and pennies
3. Compute the total value in cents
4. Find the value in dollars and loose change.
5. Display the value in dollars and loose change.
Case Study: Design - Algorithm

1. Read in your niece’s first name
2. Read in the count of nickels and pennies
3. Compute the total value in cents
   3.1 totalCents is 5 times nickels plus pennies
4. Find the value in dollars and loose change.
   4.1 dollars is integer quotient of totalCents and 100
   4.2 change is integer remainder of totalCents and 100
5. Display the value in dollars and loose change.
Case Study: Implementation

- `/` used to implement step 4.1
- `%` used to implement step 4.2
- Verify that correct data types are used
- Verify mixed-type operations and promotions
Case Study: Testing

• Test results using various input combinations
• Verify results by hand or with calculator
Listing 2.4  Value of a coin collection

// File: coins.cpp
// Determines the value of a coin collection
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;       // input: niece’s first name
    int pennies;       // input: count of pennies
    int nickels;       // input: count of nickels
    int dollars;       // output: value of coins in dollars
    int change;        // output: value of coins in cents
    int totalCents;    // total cents represented
Listing 2.4  Value of a coin collection (continued)

// Read in your niece’s first name.
cout << "Enter your first name: ";
cin >> name;

// Read in the count of nickels and pennies.
cout << "Enter the number of nickels: ";
cin >> nickels;
cout << "Enter the number of pennies: ";
cin >> pennies;
Listing 2.4  Value of a coin collection (continued)

// Compute the total value in cents.
    totalCents = 5 * nickels + pennies;

    // Find the value in dollars and change.
    dollars = totalCents / 100;     // integer division
    change = totalCents % 100;

    // Display the value in dollars and change.
    cout << "Good work " << name << '!' << endl;
    cout << "Your collection is worth "
         << dollars << " dollars and "
         << change << " cents."
    return 0;
}
coins.cpp Sample Execution

Enter your first name: Sally
Enter the number of nickels: 30
Enter the number of pennies: 77
Good work Sally!
Your collection is worth 2 dollars and 27 cents.
2.7 Interactive Mode, Batch Mode, and Data Files

• Interactive mode - input from user via keyboard
• Batch mode - input via a file, no user interaction
• Input/output redirection can be used for batch mode
Input Redirection

• Requires a file already containing all input data before the program is executed
• At the time the program is executed, the input file is specified
• E.g. in UNIX
  
  `metric < mydata`

• Executes the program metric using the file mydata for input
• Echo printing often used with batch input
Output Redirection

- Sends ALL output to a file instead of to the display monitor
- Not typically used with interactive input mode, since even prompt messages will be sent to the output file
- E.g. in UNIX (not typical)
  
  \[ \text{metric} > \text{myoutput} \]

- Both input AND output redirection
  
  \[ \text{metric} < \text{mydata} > \text{myoutput} \]
Batch version of miles-to-kms conversion program

// File: milesBatch.cpp
// Converts distance in miles to kilometers.

#include <iostream>
using namespace std;

int main()
{
    const float KM_PER_MILE = 1.609;  // 1.609 km in a mile
    float miles,                      // input: distance in miles
         kms;                        // output: distance in kilometers

    // Get the distance in miles.
    cin >> miles;
    cout << "The distance in miles is " << miles << endl;

    // Convert the distance to kilometers.
    kms = KM_PER_MILE * miles;

    // Display the distance in kilometers.
    cout << "The distance in kilometers is " << kms << endl;

    return 0;
}
2.8 Common Programming Errors

• Syntax
  – a grammatical error in the formation of a program statement
  – detected by the compiler
  – prevents translation of source code into object code, so no execution possible
  – messages are compiler dependent, so you must learn how your compiler identifies errors
Miles-to-kms program with syntax error – missing “

// Miles.cpp
// Converts distance in miles to kilometers.

#include <iostream>     // class for stream input/output
using namespace std;    // use the standard namespace

int main()               // start of main function
{
    const float km_per_mile = 1.609;  // 1.609 km in a mile
    float miles,       // input: distance in miles
          kms;    // output: distance in kilometers

    // Get the distance in miles.
    //cout << "Enter the distance in miles: ; // missing quote
    cout << "Enter the distance in miles: " << kms << endl;
    cin >> miles;

    // Convert the distance to kilometers.
    kms = km_per_mile * miles;

    // Display the distance in kilometers.
    cout << "The distance in kilometers is " << kms << endl;

    return 0;
}
Syntax Error display

[C++ Error] miles.cpp(15): E2380 Unterminated string or character constant.
[C++ Error] miles.cpp(16): E2379 Statement missing ;.
[C++ Error] miles.cpp(19): E2451 Undefined symbol 'kms'.
[C++ Warning] miles.cpp(25): W8080 'miles' is declared but never used.
[C++ Warning] miles.cpp(25): W8004 'KM_PER_MILE' is assigned a value that is never used.
Common Programming Errors

• Run-time errors
  – detected and displayed during execution of a program
  – usually fatal - halts execution of code
Common Programming Errors

• Undetected errors
  – program runs to completion, but results are incorrect
  – often the result of input of the wrong type being entered
Common Programming Errors

• Logic errors
  – caused by a faulty algorithm
  – often difficult to locate
  – can result in either a run-time or undetected error
  – system cannot identify - up to programmer to locate
  – vital to verify program output to ensure correct results