Learning Objectives

- Introduction to C++
  - Origins, Object-Oriented Programming, Terms
- Variables, Expressions and Assignment Statements
- Console Input & Output
- Style of C++ Programs
- Libraries and Namespaces

Introduction to C++

- C++ origins
  - Low-level languages
    - Ex: Machine, assembly
  - High-level languages
    - Ex: C, C++, ADA, COBOL, FORTRAN
  - Object-Oriented Programming in C++
- C++ Terminology
  - Programs and functions
  - Basic Input & Output (I/O) with cin and cout

Algorithms vs. Procedures

- Before writing a program, a programmer must clearly understand
  - What data is to be used
  - Desired result
  - Procedure needed to produce this result
- The procedure is referred to as an algorithm
- Algorithm: Step-by-step sequence of instructions describing how to perform a computation
Example of Algorithms (1/2)

- Assume that a program must calculate sum of all whole numbers from 1 through 100

- A computer can not respond to heuristic command: “Add the numbers from 1 - 100”

- A computer is algorithm-responding machine and not intuition-responding machine

- Several methods or algorithms can be used to find the required sum

Example of Algorithms (2/2)

\[
\text{sum} = \frac{n(a + b)}{2}
\]

where

- \( n \) = number of terms to be added (100)
- \( a \) = first number added (1)
- \( b \) = last number to be added (100)

\[
\text{sum} = 100(1 + 100)/2 = 5050
\]

Question: any other algorithm?

Flowchart Symbols (1/2)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Terminal</td>
<td>Indicates the beginning or end of an algorithm</td>
</tr>
<tr>
<td></td>
<td>Input/Output</td>
<td>Indicates an input or output operation</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>Indicates computation or data manipulation</td>
</tr>
<tr>
<td></td>
<td>Flow lines</td>
<td>Used to connect the flowchart symbols and indicate the logic flow</td>
</tr>
</tbody>
</table>

Flowchart Symbols (2/2)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decision</td>
<td>Indicates a decision point in the algorithm</td>
</tr>
<tr>
<td></td>
<td>Loop</td>
<td>Indicates the initial, final and increment values of a loop</td>
</tr>
<tr>
<td></td>
<td>Predefined process</td>
<td>Indicates a predefined process, as in calling a sorting process</td>
</tr>
<tr>
<td></td>
<td>Connector</td>
<td>Indicates an entry to, or exit from another part of the flowchart</td>
</tr>
</tbody>
</table>
Example of Flowchart

**Question:**
How do you design a flowchart to calculate the average of three numbers?

**Exercise:**
Design a flowchart to sort five numbers and output the biggest one.

Classes and Objects

- **Data Object:** Set of values packaged as single unit
- **Class:** Set of objects with similar attributes

- General concept of object-oriented programming is difference between an object and the larger set of which it is a member (class)
- A red, Ford Taurus sedan is an instance, or object, of general class of automobiles

Program Translation (1/2)

- **C++ source program:** Set of instructions written in C++ language
- **Machine language:** Internal computer language
  - Consists of a series of 1s and 0s
- Source program cannot be executed until it is translated into machine language
  - Interpreted language translates one statement at a time
  - Compiled language translates all statements together

Program Translation (2/2)
Function vs. Class Names (1/3)

- **Modular programs**: Segments arranged in logical order to form an integrated unit
- **Module**: Segments of modular program
- **Function**: Name of a C++ procedure
  - Composed of sequence of C++ instructions
  - Function interface is its inputs and outputs
  - Method of converting input to results is encapsulated and hidden within function

Function vs. Class Names (2/3)

![FIGURE 1.7 A Well-Designed Program is Built Using Modules](image)

Function vs. Class Names (3/3)

![FIGURE 1.8 A Multiplying Function](image)

Function/Class Naming Conventions

- **Identifiers**: Names that convey an idea of the purpose of function or class
- **Identifier composition rules**:
  - First character must be a letter or underscore
  - Only letter, digit or underscore may follow
  - Blank spaces allowed
  - Identify component words with initial capitalization
  - Cannot be C++ keyword
  - Should be a mnemonic
C++ Keywords

<table>
<thead>
<tr>
<th>auto</th>
<th>default</th>
<th>goto</th>
<th>public</th>
<th>this</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>do</td>
<td>if</td>
<td>register</td>
<td>template</td>
</tr>
<tr>
<td>case</td>
<td>double</td>
<td>inline</td>
<td>return</td>
<td>typedef</td>
</tr>
<tr>
<td>catch</td>
<td>else</td>
<td>int</td>
<td>short</td>
<td>union</td>
</tr>
<tr>
<td>char</td>
<td>enum</td>
<td>long</td>
<td>signed</td>
<td>unsigned</td>
</tr>
<tr>
<td>class</td>
<td>extern</td>
<td>new</td>
<td>sizeof</td>
<td>virtual</td>
</tr>
<tr>
<td>const</td>
<td>float</td>
<td>overload</td>
<td>static</td>
<td>void</td>
</tr>
<tr>
<td>continue</td>
<td>for</td>
<td>private</td>
<td>struct</td>
<td>volatile</td>
</tr>
<tr>
<td>delete</td>
<td>friend</td>
<td>protected</td>
<td>switch</td>
<td>while</td>
</tr>
</tbody>
</table>

C++ Identifiers (1/2)

- Examples of valid identifiers:
  - grosspay
taxCalc
  - addNums
degToRad
  - multByTwo
salesTax
  - netPay
bessel

C++ Identifiers (2/2)

- Examples of invalid identifiers:
  - 4ab3 (begins with a number)
  - e*6 (contains a special character)
  - while (is a keyword)

The main Function (1/3)

- Each C+ program must have one and only one function named **main**
- Called a driver function because it drives the other modules
The main Function (2/3)

main

You go first
I'm done
You go first
I'm done
You go first
I'm done
You go first
I'm done

1st module
2nd module
3rd module
last module

The main Function (3/3)

- First line of function is called header line
  - What type of data, if any, is returned from function
  - The name of function
  - What type of data, if any, is sent into function

- Data transmitted into function at run time are referred to as **arguments** of function

main Function Composition

Program 1.0

```
#include <iostream>
using namespace std;
int main( )
{
    // Function Body
}
```

The cout Object

- The **cout** object sends data to the standard output display device
  - The display device is usually a video screen
  - Name derived from Console OUTput and pronounced “see out”

- Data is passed to **cout** by the insertion symbol
  `cout << “Hello there, world!”;`
### C++ Sample Code using `cout`

**Task:** passing a message to `cout`  

**Program 1.1**

```c++
#include <iostream> using namespace std; int main() {
    cout << "Hello, world!";
    return 0;
}
```

What will be displayed on screen?  

**Hello, world!**

### Newline Escape Sequence

- Instructs the display device to move to a new line  
- A newline caused when the characters `\` and `n` are used together  
- Backslash provides an “escape” from the normal interpretation of the character that follows  

Newline escape sequences can be placed anywhere within a message to `cout`.

### Preprocessor Command

- Performs an action before the compiler translates source code to machine code  
- An example is: `#include <iostream>`  
- Causes the `iostream` file to be inserted wherever the `#include` command appears  

- `iostream` is part of the `iostream` standard library  
  - `iostream` is part of the C++ standard library  
  - Include two important classes:  
    - `istream`: declarations and methods for data input  
    - `ostream`: declarations and methods for data output

### Namespaces

- Files accessed by compiler when looking for prewritten classes or functions  
- Sample namespace statement:  
  - `using namespace std;`  
  - `iostream` contained in a namespace called `std`  
  - Compiler uses `iostream`’s `cout` object from `std` wherever `cout` is referenced
More C++ Sample Code (1/2)

```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "Computer, computers everywhere";
    cout << " as far as I can see";
    return 0;
}
```

What will be displayed on screen?
Computer, computers everywhere
as far as I can see

More C++ Sample Code (2/2)

```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "Computers everywhere\n as far as\n I can see";
    return 0;
}
```

What will be displayed on screen?
Computer everywhere
as far as
I can see

Syntax

- The set of rules for formulating grammatically correct C++ language statements
  - Compiler accepts statements with correct syntax without generating error message

- A program statement can syntactically correct and logically incorrect
  - Compiler will accept statement
  - Program will produce incorrect results

Style of C++ Programs

- Every C++ program must contain one and only one main() function
  - Statements included within braces {}

- C++ allows flexibility in format for the word main, the parentheses (), and braces {}
  - More than one statement can be put on line
  - One statement can be written across lines

- Use formatting for clarity and ease of program reading
Standard C++ Program Form (1/2)

- Function name starts in column 1
  - Name and parentheses on their own line
- Opening brace of function body on next line
  - Aligned with first letter of function name
- Closing brace is last line of function
  - Aligned with opening brace
- Standard form highlights the function as a unit

Standard C++ Program Form (2/2)

- Within function, indent statements 4 spaces (=1 tab)
  - Creates uniform look for similar statement groups
  - Good programming practice
- Final program form should be consistent
  - Proper format improves program readability and understandability

Bad Program Format

```cpp
int main()
{
  first statement; second statement; third statement;
  fourth statement;
  return 0;
}
```

Good Program Format

```cpp
int main()
{
  program statements in here;
  return 0;
}
```
Comments

- Explanatory remarks written within program
  - Clarify purpose of the program
  - Describe objective of a group of statements
  - Explain function of a single line of code
- Computer ignores all comments
  - Comments exist only for convenience of reader
- A well-constructed program should be readable and understandable
  - Comments help explain unclear components

Comment Structure

- **Line comment**: Begins with 2 slashes (//) and continues to the end of the line
  - Can be written on line by itself or at the end of line that contains program code
    // this is a line comment
- **Block comment**: Multiple line comment begins with the symbols /* and ends with the symbols */
  /* This is a block comment that span across two lines */

Common Programming Errors (1/2)

- Omitting parentheses after `main`
- Omitting or incorrectly typing the opening brace `{`
  - Opening brace signifies start of function body
- Omitting or incorrectly typing the closing brace `}`
  - Closing brace signifies end of function
- Misspelling the name of an object or function
  - Example: Typing `cot` instead of `cout`

Common Programming Errors (2/2)

- Forgetting to close a string sent to cout with a double-quote symbol
- Omitting the semicolon at the end of each statement
- Forgetting `\n` to indicate a new line
A C++ program consists of one or more modules
- One module must be the function `main()`
- `main()` is starting point of C++ program

The simplest C++ program has the form:
```cpp
#include <iostream>
using namespace std;
int main()
{
    program statements;
    return 0;
}
```

C++ statements are terminated by a semicolon
- C/C++ standard library contains many functions and classes
  - Standard Library provided with C++ compiler
  - Include `<iostream>` for input and output
- `cout` object displays text or numeric results
  - Stream of characters is sent to `cout` by:
  - Enclosing characters in double quotes
  - Using the insertion ("put to") operator, `<<`