C++ Programming:

Basic Elements of C++
Introduction

- **Computer program**: sequence of statements designed to accomplish some task
- **Programming**: planning/creating a program
- **Syntax**: rules that specify which statements (instructions) are legal
- **Programming language**: a set of rules, symbols, and special words
- **Semantic rule**: meaning of the instruction
C++ Programs

• A C++ program is a collection of one or more subprograms, called functions

• A subprogram or a function is a collection of statements that, when activated (executed), accomplishes something

• Every C++ program has a function called `main`

• The smallest individual unit of a program written in any language is called a token
Symbols

• Special symbols

+    ?
-
*    ,
/    <=
.    !=
;    ==
    >=
Symbols (continued)

• Word symbols
  – Reserved words, or keywords
  – Include:
    • int
    • float
    • double
    • char
    • void
    • return
Identifiers

• Consist of letters, digits, and the underscore character (_)
• Must begin with a letter or underscore
• C++ is case sensitive
• Some predefined identifiers are `cout` and `cin`
• Unlike reserved words, predefined identifiers may be redefined, but it is not a good idea
Legal and Illegal Identifiers

- The following are legal identifiers in C++:
  - first
  - conversion
  - payRate

<table>
<thead>
<tr>
<th>Illegal Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee Salary</td>
<td>There can be no space between employee and Salary.</td>
</tr>
<tr>
<td>Hello!</td>
<td>The exclamation mark cannot be used in an identifier.</td>
</tr>
<tr>
<td>one + two</td>
<td>The symbol + cannot be used in an identifier.</td>
</tr>
<tr>
<td>2nd</td>
<td>An identifier cannot begin with a digit.</td>
</tr>
</tbody>
</table>
Data Types

- **Data Type**: set of values together with a set of operations is called a data type

- C++ data can be classified into three categories:
  - Simple data type
  - Structured data type
  - Pointers
Simple Data Types

- Three categories of simple data
  - Integral: integers (numbers without a decimal)
  - Floating-point: decimal numbers
  - Enumeration type: user-defined data type
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Values</th>
<th>Storage (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>-2147483648 to 2147483647</td>
<td>4</td>
</tr>
<tr>
<td>bool</td>
<td>true and false</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>-128 to 127</td>
<td>1</td>
</tr>
</tbody>
</table>
int Data Type

- Examples:
  - 6728
  - 0
  - 78
- Positive integers do not have to have a + sign in front of them
- No commas are used within an integer
- Commas are used for separating items in a list
bool Data Type

- bool type
  - Has two values, true and false
  - Manipulate logical (Boolean) expressions
- true and false are called logical values
- bool, true, and false are reserved words
**char Data Type**

- The smallest integral data type
- Used for **characters**: letters, digits, and special symbols
- Each character is enclosed in single quotes
- Some of the values belonging to **char** data type are: 'A', 'a', '0', '*', '+', '$', '&'
- A blank space is a character and is written ' ', with a space left between the single quotes
Floating-Point Data Types

- C++ uses scientific notation to represent real numbers (floating-point notation)

<table>
<thead>
<tr>
<th>Real Number</th>
<th>C++ Floating-Point Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>75.924</td>
<td>7.592400E1</td>
</tr>
<tr>
<td>0.18</td>
<td>1.800000E-1</td>
</tr>
<tr>
<td>0.0000453</td>
<td>4.530000E-5</td>
</tr>
<tr>
<td>-1.482</td>
<td>-1.482000E0</td>
</tr>
<tr>
<td>7800.0</td>
<td>7.800000E3</td>
</tr>
</tbody>
</table>
Floating-Point Data Types (continued)

- **float**: represents any real number
  - Range: -3.4E+38 to 3.4E+38
- Memory allocated for the float type is 4 bytes
- **double**: represents any real number
  - Range: -1.7E+308 to 1.7E+308
- Memory allocated for double type is 8 bytes
- On most newer compilers, data types double and long double are same
Floating-Point Data Types (continued)

- Maximum number of significant digits (decimal places) for float values is 6 or 7
- Float values are called single precision
- Maximum number of significant digits for double is 15
- Double values are called double precision
- **Precision**: maximum number of significant digits
Arithmetic Operators

- **C++ Operators**
  - `+` addition
  - `−` subtraction
  - `∗` multiplication
  - `/` division
  - `%` remainder (mod operator)
- `+`, `−`, `∗`, and `/` can be used with integral and floating-point data types
- **Unary operator** – has only one operand
- **Binary Operator** – has two operands
Order of Precedence

• All operations inside of ( ) are evaluated first
• *, /, and % are at the same level of precedence and are evaluated next
• + and − have the same level of precedence and are evaluated last
• When operators are on the same level
  – Performed from left to right
Expressions

• If all operands are integers
  – Expression is called an integral expression
• If all operands are floating-point
  – Expression is called a floating-point expression
• An integral expression yields integral result
• A floating-point expression yields a floating-point result
Mixed Expressions

- Mixed expression:
  - Has operands of different data types
  - Contains integers and floating-point

- Examples of mixed expressions:
  
  2 + 3.5
  
  6 / 4 + 3.9
  
  5.4 * 2 – 13.6 + 18 / 2
Evaluating Mixed Expressions

• If operator has same types of operands
  − Evaluated according to the type of the operands

• If operator has both types of operands
  − Integer is changed to floating-point
  − Operator is evaluated
  − Result is floating-point
Evaluating Mixed Expressions (continued)

• Entire expression is evaluated according to precedence rules
  
  − Multiplication, division, and modulus are evaluated before addition and subtraction

  − Operators having same level of precedence are evaluated from left to right

  − Grouping is allowed for clarity
Type Conversion (Casting)

- **Implicit type coercion**: when value of one type is automatically changed to another type

- Cast operator provides explicit type conversion

- Use the following form:
  
  ```
  static_cast<dataTypeName>(expression)
  ```
Expression

static_cast<int>(7.9)  Evaluates to
static_cast<int>(3.3)  7
static_cast<double>(25)  3
static_cast<double>(5+3)  25.0
static_cast<double>(15)/2  =static_cast<double>(8) = 8.0
(because static_cast<double>(15) = 15.0)
=15.0/2  =static_cast<double>(7) (because 15/2 = 7)
= 7.5
= 7.0

static_cast<double>(15/2)

=static_cast<int>(7.8 + 7.5)

=static_cast<int>(15.3)
= 15

static_cast<int>(7.8 + static_cast<double>(15/2))

=static_cast<int>(7.8 + 7.0)
=static_cast<int>(14.8)
= 14
string Data Type

- Programmer-defined type supplied in standard library
- Sequence of zero or more characters
- Enclosed in double quotation marks
- Null: a string with no characters
- Each character has relative position in string
- Position of first character is 0, the position of the second is 1, and so on
- Length: number of characters in string
Input

• Data must be loaded into main memory before it can be manipulated

• Storing data in memory is a two-step process:
  1. Instruct the computer to allocate memory
  2. Include statements to put data into allocated memory
Allocating Memory

- **Named Constant**: memory location whose content can’t change during execution

- The syntax to declare a named constant is:

  ```
  const dataType identifier = value;
  ```

- In C++, `const` is a reserved word
const double CONVERSION = 2.54;
const int NO_OF_STUDENTS = 20;
const char BLANK = ' ';
const double PAY_RATE = 15.75;
**Variable**

Variable: memory location whose content may change during execution

The syntax for declaring one variable or multiple variables is:

```plaintext
dataType identifier, identifier, . . . ;
```

**EXAMPLE 2-12**

```plaintext
double amountDue;
int counter;
char ch;
int x, y;
string name;
```
Assignment Statement

- The assignment statement takes the form:
  ```
  variable = expression;
  ```
- Expression is evaluated and its value is assigned to the variable on the left side.
- In C++, `=` is called the assignment operator.
int i, j;
double sale;
char first;
string str;

i = 4;
j = 4 * 5 - 11;
sale = 0.02 * 1000;
first = 'D';
str = "It is a sunny day.";
A C++ statement such as:

\[ i = i + 2; \]

evaluates whatever is in \( i \), adds two to it, and assigns the new value to the memory location \( i \).
Declaring & Initializing Variables

• Variables can be initialized when declared:
  ```
  int first=13, second=10;
  char ch=' ';  
  double x=12.6, y=123.456;
  ```
• first and second are int variables with the values 13 and 10, respectively
• ch is a char variable whose value is empty
• x and y are double variables with 12.6 and 123.456, respectively
Input (Read) Statement

• **cin** is used with `>>` to gather input
  ```
  cin >> variable >> variable. . .; 
  ```
• The **extraction operator** is `>>`
• For example, if `miles` is a **double** variable
  ```
  cin >> miles; 
  ```
  ─ Causes computer to get a value of type **double**
  ─ Places it in the memory cell **miles**
Input Statement (continued)

• Using more than one variable in `cin` allows more than one value to be read at a time
• For example, if `feet` and `inches` are variables of type `int` a statement such as:

```
cin >> feet >> inches;
```

  − Inputs two integers from the keyboard
  − Places them in locations `feet` and `inches` respectively
Example 2-17

```cpp
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string firstName;    //Line 1
    string lastName;     //Line 2
    int age;             //Line 3
    double weight;       //Line 4
    cout << "Enter first name, last name, age, "
                        << "and weight, separated by spaces."
                   << endl;    //Line 5
    cin >> firstName >> lastName;    //Line 6
    cin >> age >> weight;            //Line 7
    cout << "Name: " << firstName << " "
          << lastName << " "    //Line 8
            << endl;
    cout << "Age: " << age << " "    //Line 9
        << endl;
    cout << "Weight: " << weight << " "    //Line 10
        << endl;
    return 0;    //Line 11
}```
Sample Run:

Enter first name, last name, age, and weight, separated by spaces.
Sheila Mann 23 120.5
Name: Sheila Mann
Age: 23
Weight: 120.5
Increment & Decrement Operators

- Increment operator: increment variable by 1
- Decrement operator: decrement variable by 1
- Pre-increment: \( ++\text{variable} \)
- Post-increment: \( \text{variable}++ \)
- Pre-decrement: \( --\text{variable} \)
- Post-decrement: \( \text{variable}-- \)
Increment & Decrement Operators (continued)

- `++count;` or `count++;` increments the value of `count` by 1
- `--count;` or `count--;` decrements the value of `count` by
- If `x = 5;` and `y = ++x;`
  - After the second statement both `x` and `y` are 6
- If `x = 5;` and `y = x++;`
  - After the second statement `y` is 5 and `x` is 6
Output

• The syntax of `cout` and `<<` is:
  
  `cout<< expression or manipulator
     << expression or manipulator
     << ...;`

• Called an output (`cout`) statement

• The `<<` operator is called the **insertion operator**
  or the stream insertion operator

• Expression evaluated and its value is printed at the current cursor position on the screen
Output (continued)

- **Manipulator**: alters output
- **endl**: the simplest manipulator
  - Causes cursor to move to beginning of the next line
Output Example

- Output of the C++ statement
  
  \[
  \texttt{cout} \ll a;
  \]

  is meaningful if \(a\) has a value

  For example, the sequence of C++ statements,

  \[
  a = 45;
  \texttt{cout} \ll a;
  \]

  produces an output of \(45\)
The New Line Character

- The new line character is '\n'
- Without this character the output is printed on one line
- Tells the output to go to the next line
- When \n is encountered in a string
  - Cursor is positioned at the beginning of next line
- A \n may appear anywhere in the string
Examples

• Without the new line character:
  
  ```
  cout << "Hello there.";
  cout << "My name is James.";
  ```
  
  Would output:
  
  Hello there. My name is James.

• With the new line character:

  ```
  cout << "Hello there.\n"
  cout << "My name is James.";
  ```
  
  Would output
  
  Hello there.
  My name is James.
<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>Newline</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\r</td>
<td>Return</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>'</td>
<td>Single quotation</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quotation</td>
</tr>
</tbody>
</table>
Preprocessor Directives

- C++ has a small number of operations
- Many functions and symbols needed to run a C++ program are provided as collection of libraries
- Every library has a name and is referred to by a header file
- Preprocessor directives are commands supplied to the preprocessor
- All preprocessor commands begin with #
- No semicolon at the end of these commands
Preprocessor Directive Syntax

- Syntax to include a header file

  `#include <headerFileName>`

- Causes the preprocessor to include the header file `iostream` in the program

- The syntax is:

  `#include <iostream>`
Header Files

• In older versions of C++
  - Header files had the file extension \texttt{.h}
• ANSI C++ removes this extension
• The descriptions of the functions needed to perform I/O are contained in iostream
• The syntax is:
  
  \texttt{#include <iostream>}

Using `cin` and `cout` in a Program and namespace

- `cin` and `cout` are declared in the header file `iostream`, but within a namespace named `std`
- To use `cin` and `cout` in a program, use the following two statements:

  ```cpp
  #include <iostream>
  using namespace std;
  ```
Using the `string` Data Type in a Program

- To use the `string` type, you need to access its definition from the header file `string`

- Include the following preprocessor directive:

```cpp
#include <string>
```
Creating a C++ Program

- C++ program has two parts:
  1. Preprocessor directives
  2. The program

- Preprocessor directives and program statements constitute C++ source code

- Source code must be saved in a file with the file extension `.cpp`
Creating a C++ Program (continued)

• Compiler generates the object code
  – Saved in a file with file extension .obj

• Executable code is produced and saved in a file with the file extension .exe.
• Declaration Statements
  ```cpp
  int   a, b, c;
  double x, y;
  ```
  - Variables can be declared anywhere in the program, but they must be declared before they can be used.

• Executable Statements have three forms:
  ```cpp
  a = 4;       // assignment statement
  cin >> b;    // input statement
  cout << a << " " << b << endl; // output statement
  ```
#include <iostream>  //Line 1

using namespace std;  //Line 2
const int NUMBER = 12;  //Line 3
int main()  //Line 4
{
    int firstNum;  //Line 6
    int secondNum;  //Line 7
    firstNum = 18;  //Line 8
    cout << "Line 9: firstNum = " << firstNum << endl;  //Line 9
    cout << "Line 10: Enter an integer: ";  //Line 10
    cin >> secondNum;  //Line 11
    cout << endl;  //Line 12
    cout << "Line 13: secondNum = " << secondNum << endl;  //Line 13
    firstNum = firstNum + NUMBER + 2 * secondNum;  //Line 14
    cout << "Line 15: The new value of " << "firstNum = " << firstNum << endl;  //Line 15
    return 0;  //Line 16
}
Sample Run:
Line 9: firstNum = 18
Line 10: Enter an integer: 15

Line 13: secondNum = 15
Line 15: The new value of firstNum = 60
Program Style and Form

- The Program Part
  - Every C++ program has a function main
  - Basic parts of function main are:
    - The heading
    - The body of the function
- The heading part has the following form

```
typeOfFunction main(argument list)
```
Syntax

• Errors in syntax are found in compilation

```java
int x;       //Line 1
int y       //Line 2: syntax error
double z;   //Line 3
y = w + x;  //Line 4: syntax error
```
Use of Blanks

• Use of Blanks
  − One or more blanks separate input numbers
  − Blanks are also used to separate reserved words and identifiers from each other and other symbols

• Blanks between identifiers in the second statement are meaningless:

  ```
  int a, b, c;
  int a, b, c;
  ```

• In the statement: `inta, b, c;`
  no blank between the `t` and `a` changes the reserved word `int` and the identifier `a` into a new identifier, `inta`. 
Semicolons, Brackets, & Commas

- Commas separate items in a list
- All C++ statements end with a semicolon
- Semicolon is also called a statement terminator
- { and } are not C++ statements
Semantics

• Possible to remove all syntax errors in a program and still not have it run
• Even if it runs, it may still not do what you meant it to do
• For example,
  \[ 2 + 3 \times 5 \text{ and } (2 + 3) \times 5 \]
  are both syntactically correct expressions, but have different meanings
Form and Style

• Consider two ways of declaring variables:
  
  − Method 1
    
    ```
    int feet, inch;
    double x, y;
    ```
  
  − Method 2
    
    ```
    int a, b; double x, y;
    ```
  
  • Both are correct, however, the second is hard to read
Documentation

• Comments can be used to document code
  − Single line comments begin with // anywhere in the line
  − Multiple line comments are enclosed between /* and */

• Name identifiers with meaningful names

• Run-together-words can be handled either by using CAPS for the beginning of each new word or an underscore before the new word
Assignment Statements

• C++ has special assignment statements called compound assignment:

  +=, -=, *=, /=, and %= 

• Example:

  \( x *= y; \)
Programming Example

Write a program that takes as input a given length expressed in feet and inches. Convert and output the length in centimeters.

- **Analysis:**
  - **Input:** Length in feet and inches
  - **Output:** Equivalent length in centimeters
  - Lengths are given in feet and inches
  - Program computes the equivalent length in centimeters
  - One inch is equal to 2.54 centimeters
Programming Example (continued)

- Convert the length in feet and inches to all inches:
  - Multiply the number of feet by 12
  - Add given inches

- Use the conversion formula (1 inch = 2.54 centimeters) to find the equivalent length in centimeters
Programming Example (continued)

- The algorithm is as follows:
  - Get the length in feet and inches
  - Convert the length into total inches
  - Convert total inches into centimeters
  - Output centimeters
Variables and Constants

• Variables
  int feet;       //variable to hold given feet
  int inches;    //variable to hold given inches
  int totalInches;  //variable to hold total inches
  double centimeters;  //variable to hold length in
                        //centimeters

• Named Constant
  const double conversion = 2.54;
  const int inchesPerFoot = 12;
Main Algorithm

- Prompt user for input
- Get data
- Echo the input (output the input)
- Find length in inches
- Output length in inches
- Convert length to centimeters
- Output length in centimeters
Putting It Together

• Program begins with comments
• System resources will be used for I/O
• Use input statements to get data and output statements to print results
• Data comes from keyboard and the output will display on the screen
• The first statement of the program, after comments, is preprocessor directive to include header file iostream
Putting It Together (continued)

• Two types of memory locations for data manipulation:
  – Named constants
  – Variables

• Named constants are usually put before main so they can be used throughout program

• This program has only one function (main), which will contain all the code

• The program needs variables to manipulate data, which are declared in main
Body of the Function

• The body of the function main has the following form:

```c
int main ()
{
    declare variables
    statements
    return 0;
}
```
Writing a Complete Program

- Begin the program with comments for documentation
- Include header files
- Declare named constants, if any
- Write the definition of the function main
Program Convert Measurements: This program converts measurements in feet and inches into centimeters using the formula that 1 inch is equal to 2.54 centimeters.

//header file
#include <iostream>
using namespace std;

//named constants
const double CENTIMETERS_PER_INCH = 2.54;
const int INCHES_PER_FOOT = 12;

int main ()
{
    //declare variables
    int feet, inches;
    int totalInches;
    double centimeter;

    //Statements: Step 1 - Step 7
    cout << "Enter two integers, one for feet and " << "one for inches: ";  //Step 1
    cin >> feet >> inches;  //Step 2
    cout << endl;
}
cout << endl;

cout << "The numbers you entered are " << feet << " for feet and " << inches << " for inches. " << endl;  //Step 3
totalInches = INCHES_PER_FOOT * feet + inches; //Step 4

cout << "The total number of inches = " << totalInches << endl;  //Step 5

centimeter = CENTIMETERS_PER_INCH * totalInches; //Step 6

cout << "The number of centimeters = " << centimeter << endl;  //Step 7
return 0;
}

Sample Run
Enter two integers, one for feet, one for inches: 15 7

The numbers you entered are 15 for feet and 7 for inches.
The total number of inches = 187
The number of centimeters = 474.98
Summary

- **C++ program**: collection of functions where each program has a function called `main`
- Identifier consists of letters, digits, and underscores, and begins with letter or underscore
- The arithmetic operators in C++ are addition (+), subtraction (-), multiplication (*), division (/), and modulus (%)
- Arithmetic expressions are evaluated using the precedence associativity rules
Summary (continued)

• All operands in an integral expression are integers and all operands in a floating-point expression are decimal numbers

• **Mixed expression**: contains both integers and decimal numbers

• Use the cast operator to explicitly convert values from one data type to another

• A named constant is initialized when declared

• All variables must be declared before used
Summary (continued)

- Use `cin` and stream extraction operator `>>` to input from the standard input device
- Use `cout` and stream insertion operator `<<` to output to the standard output device
- Preprocessor commands are processed before the program goes through the compiler
- A file containing a C++ program usually ends with the extension `.cpp`