Programming Fundamentals

Lecture 3

Aamina Batool

`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

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:

```
#include  <string>
```

- `Example:`
  - `String firstName;`
  - `firstName = “Aamina”`

Example

```
#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 << endl;                           //Line 8
    cout << "Age: " << age << endl;                     //Line 9
    cout << "Weight: " << weight << endl;               //Line 10
    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
```

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

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

  `cout << a;`

  is meaningful if `a` has a value

  For example, the sequence of C++ statements,

  ```
  a = 45;
  cout << 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 2-4 Commonly Used Escape Sequences

	Escape Sequence	Description
`\n`	Newline	Cursor moves to the beginning of the next line
`\t`	Tab	Cursor moves to the next tab stop
`\b`	Backspace	Cursor moves one space to the left
`\r`	Return	Cursor moves to the beginning of the current line (not the next line)
`\\`	Backslash	Backslash is printed
`\'`	Single quotation	Single quotation mark is printed
`\"`	Double quotation	Double quotation mark is printed

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>
```

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:

```
#include <iostream>

using namespace std;
```

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`.

Creating a C++ Program (continued)

- Declaration Statements

```
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:

```
a = 4;          //assignment statement
cin >> b;   //input statement
cout << a << " " << b << endl; //output statement
```

Example 2-28

```
#include <iostream>                                    //Line 1

using namespace std;                                   //Line 2
const int NUMBER = 12;                                 //Line 3
int main()                                             //Line 4
{                                                      //Line 5
    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

```
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 * 5` and `(2 + 3) * 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
- 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:

```
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
```

Different Operators

- **Arithmetic Operators (+ , -, /, %, *, =)**
- **Relational Operators (< , > , =, <=, >=, ==)**

Increment & Decrement Operators

- Increment operator: increment variable by 1
- Decrement operator: decrement variable by 1
- Pre-increment: `++variable`
- Post-increment: `variable++`
- Pre-decrement: `--variable`
- Post-decrement: `variable–`

- Unary operator – has only one operand
- Binary Operator – has two operands

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`

References

1. C++ Programming: From Problem Analysis to Program Design, Third Edition
2. https://www.just.edu.jo/~yahya-t/cs115/