Chapter 1 : Introduction to computer programming

Dr Kafi Rahman

email: kafi@truman.edu

1.4

What is a Program Made of?

What is a Program Made of?

- Common elements in programming languages:
  - Keywords
  - Programmer-Defined Identifiers
  - Operators
  - Punctuation
  - Syntax

Program 1-1

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     double hours, rate, pay;
8
9     // Get the number of hours worked.
10    cout << "How many hours did you work? ";
11    cin >> hours;
12
13    // Get the hourly pay rate.
14    cout << "How much do you get paid per hour? ";
15    cin >> rate;
16
17    // Calculate the pay.
18    pay = hours * rate;
19
20    // Display the pay.
21    cout << "You have earned $" << pay << endl;
22    return 0;
23 }
```

Key Words

- Also known as reserved words
- Have a special meaning in C++
- Keywords can not be used as variable names
- Keywords in the Program 1-1: using, namespace, int, double, and return

Key Words

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     double hours, rate, pay;
8
9     // Get the number of hours worked.
10    cout << "How many hours did you work? ";
11    cin >> hours;
12
13    // Get the hourly pay rate.
14    cout << "How much do you get paid per hour? ";
15    cin >> rate;
16
17    // Calculate the pay.
18    pay = hours * rate;
19
20    // Display the pay.
21    cout << "You have earned $" << pay << endl;
22    return 0;
23 }
```

Programmer-Defined Identifiers

- Names made up by the programmer
- Not part of the C++ language
- Used to represent various things: variable names(memory locations), function names, etc.
- In Program 1-1: hours, rate, and pay.

Operators

- Used to perform operations on data
- Many types of operators:
  - Arithmetic - ex: +,-,*,/
  - Assignment – ex: =

- Some operators in Program1-1:
  << >> = *

Operators

```
1   // This program calculates the user's pay.
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7      double hours, rate, pay;
8
9      // Get the number of hours worked.
10     cout << "How many hours did you work? ";
11     cin >> hours;
12
13     // Get the hourly pay rate.
14     cout << "How much do you get paid per hour? ";
15     cin >> rate;
16
17     // Calculate the pay.
18     pay = hours * rate;
19
20     // Display the pay.
21     cout << "You have earned $" << pay << endl;
22     return 0;
23  }
```

Punctuation

- Characters that mark the end of a statement, or that separate items in a list
- In Program 1-1: , and ;

Punctuation

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     double hours, rate, pay;
8
9     // Get the number of hours worked.
10    cout << "How many hours did you work? ";
11    cin >> hours;
12
13    // Get the hourly pay rate.
14    cout << "How much do you get paid per hour? ";
15    cin >> rate;
16
17    // Calculate the pay.
18    pay = hours * rate;
19
20    // Display the pay.
21    cout << "You have earned $" << pay << endl;
22    return 0;
23 }
```

Syntax

- The rules of grammar that must be followed when writing a program
- Controls the use of key words, operators, programmer-defined symbols, and punctuation

Variables

- A variable is a named storage location in the computer's memory for holding a piece of data.
- In Program 1-1 we used three variables:
  - The hours variable was used to hold the hours worked
  - The rate variable was used to hold the pay rate
  - The pay variable was used to hold the gross pay

Variable Declaration or Definitions

- To create a variable in a program we must first declare the variable.
- Here is the statement from Program 1-1 that defines the variables:

```
double hours, rate, pay;
```

Variable Declaration or Definitions

- There are many different types of data, which you will learn about in this course.
- In C++ programs, a variable holds a specific type of data.
- The variable declaration specifies the type of data a variable can hold, and the variable name.

Variable Definitions

- Once again, line 7 from Program 1-1:

  double hours, rate, pay;

- The word double specifies that the variables can hold floating point numbers
- By using the type information it can allocate specific number of bytes in the memory for that variable

1.5

Input, Processing, and Output

Input, Processing, and Output

- Three steps that a program typically performs:
  - Gather input data:
    - from keyboard
    - from files on disk drives
  - Process the input data
  - Display the results as output:
    - send it to the screen
    - write to a file

1.6

The Programming Process

The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

1.7

Procedural and Object-Oriented Programming

Procedural and Object-Oriented Programming

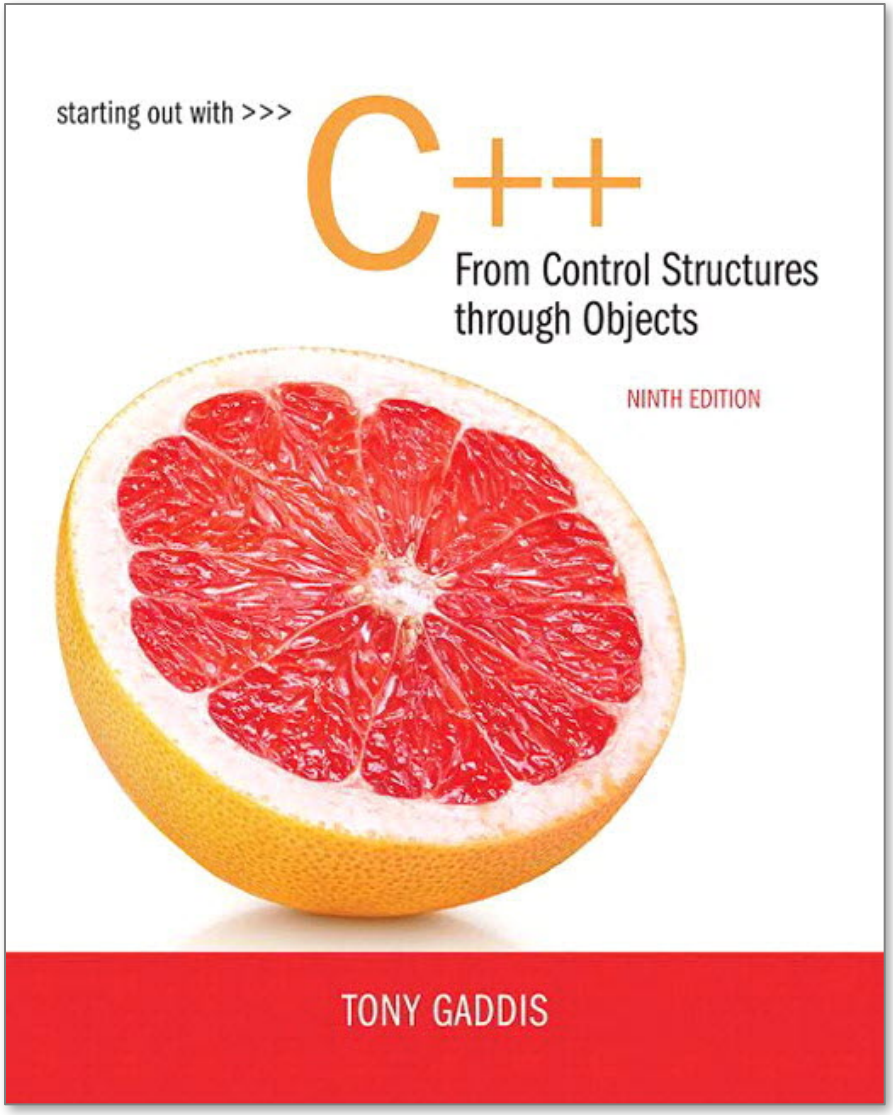
- Procedural programming: focus is on the process. Procedures/functions are written to process data.
- Object-Oriented programming: focus is on objects, which contain data and the means to manipulate the data. Messages are sent to objects to perform specific operations.

Chapter 2:

Introduction

to

C++



2.1

The Parts of a C++ Program

The Parts of a C++ Program

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, there!";
    return 0;
}
```

comment

preprocessor directive

which namespace to use

beginning of function named `main`

beginning of block for `main`

output statement

string literal

Send `0` to operating system

end of block for `main`

Special Characters

Character	Name	Meaning
//	Double slash	Beginning of a comment
#	Pound sign	Beginning of preprocessor directive
< >	Open/close brackets	Enclose filename in #include
( )	Open/close parentheses	Used when naming a function
{ }	Open/close brace	Encloses a group of statements
" "	Open/close quotation marks	Encloses string of characters
;	Semicolon	End of a programming statement

2.2

The `cout` Object

The `cout` Object

- Displays output on the computer screen

- You use the stream insertion operator `<<` to send output to `cout`:

```
cout << "Programming is fun!";
```

The `cout` Object

- Can be used to send more than one item to cout:

```
cout << "Hello " << "there!";
```

Or:

```
cout << "Hello ";
cout << "there!";
```

The `cout` Object

- This produces one line of output:

```
cout << "Programming is ";
cout << "fun!";
```

The `endl` Manipulator

- You can use the **`endl`** manipulator to start a new line of output. This will produce two lines of output:

```
cout << "Programming is" << endl;
cout << "fun!";
```

The `endl` Manipulator

```
cout << "Programming is" << endl;
cout << "fun!";
```

The `endl` Manipulator

- You do NOT put quotation marks around **`endl`**

- The last character in **`endl`** is a lowercase L, not the number 1.

`endl` ← This is a lowercase L

The `\n` Escape Sequence

- You can also use the **`\n`** escape sequence to start a new line of output. This will produce two lines of output:

```
cout << "Programming is\n";
cout << "fun!";
```

Notice that the **`\n`** is INSIDE the string.

The `\n` Escape Sequence

```
cout << "Programming is\n";
cout << "fun!";
```


Questions?