Getting Started

Intro Computer Science 1
Objectives

- Examine a Sample C++ Program
- Compiling a C++ Program
- Program Development Process
Sample C++ Program:
Examine Program Attributes

- **Basic Concept: Divide and Conquer: How?**

- **“Functions”**
  - **Definition** – Assigns a unique Name to a set of instructions (Statements)
  - **Call** – When you use the function name, the set of instructions associated with the Name are executed
Sample C++ Program:
Functions Definition Format

- Specifies returned value type
- Specifies unique Name
- Optional – for passing data
  
  return value type  function name (argument list)
  
  {  
  
  Function body // C++ statements
  
  }

Did you notice:
  • The left and right parenthesis
  • The braces are on separate lines
  • The indentations of the C++ Statements

We will look at functions in detail later
Sample C++ Program:
Empty Main Function

```
int main ()
{
}
```

Function returns a value of type `integer`

Function Name
/* Let's greet the world. */

#include <iostream>  // preprocessor instruction
using namespace std;

int main ()
{

    //Output “Hello World!”
    std: : cout << "Hello World!\n";
    //Send 0 back to OS indicating success
    return 0;
}

Explanation of Code
The Hello World Program

/* Lets greet the world. */

// preprocessor instruction

When Comment spans multiple lines

Single Line Comments

Comments are not executed
#include <iostream>

- **#include Directives** tell compiler where to find info about items used in the program [Note: NO space between “#” and “include”]

- #include <iostream> directs the processor to make cout (see-out), cin (see in) and other programs in iostream library available to your program

using namespace std;

Tells the compiler to use names in iostream in a standard way.
**Explanation of Code**

**The Hello World Program**

```cpp
main( )
```

A Special function name – It indicates where program execution starts
The OS calls the function `main( )` to start the C++ program

```cpp
int main( )
```

The function `main( )` returns a value of type `integer`
Explanation of Code
The Hello World Program Statements

```
std::cout << "Hello World!\n";
```

<table>
<thead>
<tr>
<th>std::</th>
<th>look in the namespace “std” for the name that follows</th>
</tr>
</thead>
<tbody>
<tr>
<td>cout &lt;&lt;</td>
<td>outputs to the monitor the text string “Hello World!” followed by a newline</td>
</tr>
<tr>
<td>Think of cout as a name (Identifier) for the monitor</td>
<td></td>
</tr>
<tr>
<td>; (semicolon) – Program statements must end with a semicolon</td>
<td></td>
</tr>
</tbody>
</table>
C++ source code is written with an editor
- vi
- Emacs
- Ultraedit

The Compiler converts your source code to Object code
The Linker combines all the object code into an executable program
## Compiling C++ Programs

### Command Line Compiler, `g++`

<table>
<thead>
<tr>
<th>Compiler Commands</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>g++ file.cpp</code></td>
<td>Compiles the file <em>file.cpp</em> and creates an executable named <em>a.out</em></td>
</tr>
<tr>
<td><code>g++ -- version</code></td>
<td>Displays the version of <code>g++</code></td>
</tr>
<tr>
<td><code>g++ file.cpp -o file2</code></td>
<td>Compiles the file <em>file.cpp</em>, creates an executable, and names the executable <em>file2</em> instead of <em>a.out</em></td>
</tr>
</tbody>
</table>
# Compiling C++ Programs

## Error Messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Error</td>
<td>The rules of the C++ language were not followed</td>
</tr>
<tr>
<td></td>
<td>if i=&lt; 100…</td>
</tr>
<tr>
<td></td>
<td>should be</td>
</tr>
<tr>
<td></td>
<td>If (i &lt;= 100) …</td>
</tr>
<tr>
<td>Logical Error</td>
<td>Wrong algorithm</td>
</tr>
<tr>
<td>Runtime Error</td>
<td>Program terminates abruptly: Div by 0; illegal memory access</td>
</tr>
</tbody>
</table>
Compiling and Running your C++ Program

- **C++ Program Source Code**
- **Compiler**
- **Computer**
- **editor**
- **Compile command**
- **Object Code for your Program**
- **linker**
- **Computer (Complete Object Code)**
- **Run**
- **Output of C++ Program**
- **Data for C++ Program**
- **Keyboard, file**
- **Other Object Codes**
Compiling and Running Your C++ Program

- Compile the code
- Fix errors the Compiler indicates and re-compile code
- Run the program again
  - Test and re-run
Program Development Phase
Step #1: Problem-Solving Phase

- Define Problem
  - Complex problem? → Adopt Problem Decomposition principle

- Design Algorithm  [Iterative process]
  *Just express your initial thoughts [on paper], you’ll refine them later:*
  - Draw/Sketch charts
  - Draw/Sketch Pictures (worth a thousand words!)
  - Narratives
  - Complex Algorithm? → Adopt Algorithm Decomposition Principle
  - Psuedocode
  - Verify that algorithm solves the problem in its entirety
Program Development Process

Step #2: Implementation Phase

- Translate Algorithm to C++
  - **START** writing your C++ Code
- Unit Test
  - Test individual functions for accuracy
  - Fix errors
- System Test
  - Test the entire program
  - Fix errors
Program Development Process

Problem-solving Phase

Start

Problem Definition

Algorithm Design

Verify Algorithm (Paper Testing)

Implementation Phase

Translate to C++

Testing

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

int main( )
{
    int number_pods, peas_per_pod, total_peas;
    cout << “Press return after entering a number. \n”;
    cout << “Enter the number of pods:\n”;
    cin >> number_of pods;
    cout << “Enter the number of peas in a pod:\n”;
    cin>> peas_per_pod;

    total_peas = number_of pods * peas_per_pod;
    cout << “If you have “;
    cout << number_of pods;
    cout << “ pea pods\n”;
    cout << “and”;
    cout << peas_per_pod;
    cout << “ peas in each pod, then\n”;
    cout << “you have “;
    cout << total_peas;
    cout << “ peas in all the pods.\n”;
    return 0;
}
```
#include <iostream>
using namespace std;

int main() {
    int number_pods, peas_per_pod, total_peas;

    cout << "Press return after entering a number. \n";
    cout << "Enter the number of pods:\n";
    cin >> number_of_pods;
    cout << "Enter the number of peas in a pod:\n";

    cin>> peas_per_pod;
    total_peas = number_of_pods * peas_per_pod;
    cout << "If you have \n"
    cout << number_of_pods;
    cout << " pea pods\n"
    cout >> peas_per_pod;
    cout << " peas in each pod, then\n"
    cout >> total_peas;
    cout << " peas in all the pods.\n"
    return 0;
}
#include <iostream>
using namespace std;

int main( )
{
    int number_pods, peas_per_pod, total_peas;
    cout << “Press return after entering a number. \n”;
    cout << “Enter the number of pods:\n”;
    cin >> number_of_pods;
    cout << “Enter the number of peas in a pod:\n”; cin>> peas_per_pod;
    total_peas = number_of_pods * peas_per_pod;
    cout << “If you have ”;
    cout << number_of_pods;
    cout << “ pea pods\n”; cout << “ and”;
    cout << peas_per_pod;
    cout << “ peas in each pod, then\n”; cout << “ you have ”;
    cout << total_peas;
    cout << “ peas in all the pods.\n”;
    return 0;
}

We will talk more about variables & Operators shortly
Sample C++ Code #2

Sample Dialogue

#include <iostream>
using namespace std;

int main() {
    int number_pods, peas_per_pod, total_peas;
    cout << "Press return after entering a number. \n";
    cout << "Enter the number of pods:\n";
    cin >> number_of_pods;
    cout << "Enter the number of peas in a pod:\n";
    cin >> peas_per_pod;

    total_peas = number_of_pods * peas_per_pod;
    cout << "If you have \n";
    cout << number_of_pods;
    cout << " pea pods";
    cout << " and ";
    cout << peas_per_pod;
    cout << " peas in each pod, then\n";
    cout << " you have \n";
    cout << total_peas;
    cout << " peas in all the pods.\n";
    return 0;
}

Sample Dialogue

Press return after entering a number
Enter the number of pods
10
Enter the number of peas in a pod
9
If you have 10 pea pods and 9 peas in each pod, then you have 90 peas in all the pods
#include <iostream>
using namespace std;

int main() {
    // Variable Declarations
    // Statement #1
    // Statement #2
    // Last Statement
    return 0;
}