Course Information

• Lecture:
  – James B D Joshi
  – Tuesdays/Thursdays: 11:00-12:15 PM
    • Office Hours: Wed 3:00-5:00PM/Appointment
  – TA: Samvith Srinivas
    • Office Hours: Mon 1:00-3:00PM/Appointment

• Pre-requisite
  – IS 0015 Data Structures and Programming Techniques
Course Information

• Textbook

• Course Description
  – An introduction to the development of programs using C++.
  – Emphasis is given to the development of program modules that can function independently.
    • Object-oriented design
  – The theory of data structures and programming language design is continued.

Grading

• Quiz 10%
• Homework/Programming Assignments 50%
• Exams 20%
• Project 20%
Course Policy

• Your work MUST be your own
  – Zero tolerance for cheating
  – Discussing problems is encouraged, but each must present his own answers
  – You get an F for the course if you cheat in anything however small
    – NO DISCUSSION

• Homework
  – There will be penalty for late assignments (15% each day)
  – Ensure clarity in your answers – no credit will be given for vague answers
  – Homework is primarily the GSA’s responsibility

• Check webpage for everything!
  – You are responsible for checking the webpage for updates

Computer Languages

• Machine language
  • Generally consist of strings of numbers - Ultimately 0s and 1s - Machine-dependent
  • Example: +130042774
             +1400593419

• Assembly language
  • English-like abbreviations for elementary operations
  • Incomprehensible to computers - Convert to machine language
  • Example: LOAD BASEPAY
             ADD OVERPAY
             STORE GROSSPAY

• High-level languages
  • Similar to everyday English, use common mathematical notations
  • Compiler/Interpreter
  • Example:
    grossPay = basePay + overtimePay
History of C and C++

- History of C
  - Evolved from two other programming languages
    - BCPL and B: “Typeless” languages
  - Dennis Ritchie (Bell Lab): Added typing, other features

- History of C++
  - Early 1980s: Bjarne Stroustrup (Bell Lab)
  - Provides capabilities for object-oriented programming
    - Objects: reusable software components
    - Object-oriented programs

- Building block approach” to creating programs
  - C++ programs are built from pieces called classes and functions
  - C++ standard library: Rich collections of existing classes and functions

Structured/OO Programming

- Structured programming (1960s)
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - E.g.Pascal:1971: Niklaus Wirth

- OOP
  - “Software reuse”
  - “Modularity”
  - “Extensible”
  - More understandable, better organized and easier to maintain than procedural programming
Basics of a Typical C++ Environment

- C++ systems
  - Program-development environment
  - Language
  - C++ Standard Library
- C++ program names extensions
  - .cpp
  - .cxx
  - .cc
  - .C

Phases of C++ Programs:
1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute

© 2003 Prentice Hall, Inc. All rights reserved.
Basics of a Typical C++ Environment

- Common Input/output functions
  - `cin`
    - Standard input stream
    - Normally keyboard
  - `cout`
    - Standard output stream
    - Normally computer screen
  - `cerr`
    - Standard error stream
    - Display error messages

- Comments: C’s comment /* .. */ OR Begin with //
- Preprocessor directives: Begin with #
  - Processed before compiling

A Simple Program: Printing a Line of Text

- Standard output stream object
  - `std::cout`
    - “Connected” to screen
    - `<<`
      - Stream insertion operator
      - Value to right (right operand) inserted into output stream

- Namespace
  - `std::` specifies that entity belongs to “namespace” `std`
  - `std::` removed through use of `using` statements

- Escape characters: \`
  - Indicates “special” character output
// Fig. 1.2: fig01_02.cpp
// A first program in C++.
#include <iostream>

// function main begins program execution
int main()
{
    std::cout << "Welcome to C++!\n";
    return 0; // indicate that program ended successfully
}
} // end function main

Welcome to C++!

---

A Simple Program: Printing a Line of Text

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>Newline. Position the screen cursor to the beginning of the next line.</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal tab. Move the screen cursor to the next tab stop.</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.</td>
</tr>
<tr>
<td>\a</td>
<td>Alert. Sound the system bell.</td>
</tr>
<tr>
<td>\</td>
<td>Backslash. Used to print a backslash character.</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quote. Used to print a double quote character.</td>
</tr>
</tbody>
</table>
Memory Concepts

- Variable names
  - Correspond to actual locations in computer's memory
  - Every variable has name, type, size and value
  - When new value placed into variable, overwrites previous value

  - `std::cin >> integer1;`
    - Assume user entered 45
    - `integer1` 45

  - `std::cin >> integer2;`
    - Assume user entered 72
    - `integer1` 45
    - `integer2` 72

  - `sum = integer1 + integer2;`
    - `sum` 117

Arithmetic

- Arithmetic calculations
  - `*` : Multiplication
  - `/` : Division
    - Integer division truncates remainder
      - `7 / 5` evaluates to 1
  - `%` : Modulus operator returns remainder
    - `7 % 5` evaluates to 2

<table>
<thead>
<tr>
<th>Operator(s)</th>
<th>Operation(s)</th>
<th>Order of evaluation (precedence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Parentheses</td>
<td>Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses “on the same level” (i.e., not nested), they are evaluated left to right.</td>
</tr>
<tr>
<td>*, /, or %</td>
<td>Multiplication, Division, Modulus</td>
<td>Evaluated second. If there are several, they’re evaluated left to right.</td>
</tr>
<tr>
<td>+ or -</td>
<td>Addition, Subtraction</td>
<td>Evaluated last. If there are several, they are evaluated left to right.</td>
</tr>
</tbody>
</table>
**Decision Making: Equality and Relational Operators**

- **if structure**
  - Make decision based on truth or falsity of condition
    - If condition met, body executed
    - Else, body not executed

- **Equality and relational operators**
  - Equality operators
    - Same level of precedence
  - Relational operators
    - Same level of precedence
    - Associate left to right

<table>
<thead>
<tr>
<th>Standard algebraic equality or relational operator</th>
<th>C++ equality or relational operator</th>
<th>Example of C++ condition</th>
<th>Meaning of C++ condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relational operators</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>x &gt; y</td>
<td>x is greater than y</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>x &lt; y</td>
<td>x is less than y</td>
</tr>
<tr>
<td>≥</td>
<td>&gt;=</td>
<td>x &gt;= y</td>
<td>x is greater than or equal to y</td>
</tr>
<tr>
<td>≤</td>
<td>&lt;=</td>
<td>x &lt;= y</td>
<td>x is less than or equal to y</td>
</tr>
<tr>
<td><strong>Equality operators</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>==</td>
<td>x == y</td>
<td>x is equal to y</td>
</tr>
<tr>
<td>≠</td>
<td>!=</td>
<td>x != y</td>
<td>x is not equal to y</td>
</tr>
</tbody>
</table>

© 2003 Prentice Hall, Inc. All rights reserved.
Algorithms / pseudocode

- Computing problems
  - Solved by executing a series of actions in a specific order
- Algorithm: a procedure determining
  - Actions to be executed
  - Order to be executed
  - Example: recipe
- Program control
  - Specifies the order in which statements are executed
- Pseudocode
  - Artificial, informal language used to develop algorithms
  - Similar to everyday English

Control Structures

- Sequential execution
  - Statements executed in order
- Transfer of control
  - Next statement executed not next one in sequence
  - Structured programming – “goto”-less programming
- 3 control structures to build any program
  - Sequence structure
    - Programs executed sequentially by default
  - Selection structures
    - if, if/else, switch
  - Repetition structures
    - while, do/while, for
Keywords

- **C++ keywords**
  - Cannot be used as identifiers or variable names

<table>
<thead>
<tr>
<th>Keywords</th>
<th>auto</th>
<th>break</th>
<th>case</th>
<th>char</th>
<th>const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords common to the C and C++ programming languages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>continue</td>
<td>default</td>
<td>do</td>
<td>double</td>
<td>else</td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>extern</td>
<td>float</td>
<td>for</td>
<td>goto</td>
<td></td>
</tr>
<tr>
<td>if</td>
<td>int</td>
<td>long</td>
<td>register</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td>short</td>
<td>signed</td>
<td>sizeof</td>
<td>static</td>
<td>struct</td>
<td></td>
</tr>
<tr>
<td>switch</td>
<td>typedef</td>
<td>union</td>
<td>unsigned</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>volatile</td>
<td>while</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C++ only keywords</strong></td>
<td>asm</td>
<td>bool</td>
<td>catch</td>
<td>class</td>
<td>const_cast</td>
</tr>
<tr>
<td>delete</td>
<td>dynamic_cast</td>
<td>explicit</td>
<td>false</td>
<td>friend</td>
<td></td>
</tr>
<tr>
<td>inline</td>
<td>mutable</td>
<td>namespace</td>
<td>new</td>
<td>operator</td>
<td></td>
</tr>
<tr>
<td>private</td>
<td>protected</td>
<td>public</td>
<td>reinterpret_cast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>static_cast</td>
<td>template</td>
<td>this</td>
<td>throw</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>try</td>
<td>typeid</td>
<td>typename</td>
<td>using</td>
<td>virtual</td>
<td></td>
</tr>
<tr>
<td>wchar_t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Control Structures

- **Flowchart**
  - Graphical representation of an algorithm
  - Special-purpose symbols connected by arrows (flowlines)
  - Rectangle symbol (action symbol)
    - Any type of action
  - Oval symbol
    - Beginning or end of a program, or a section of code (circles)

**Exercise:** Find greater of three numbers
**if/else Selection Structure**

- Ternary conditional operator (?:)
  - Three arguments (condition, value if true, value if false)
- Code could be written:
  ```cpp
  cout << ( grade >= 60 ? "Passed" : "Failed" );
  ```

**while Repetition Structure**

- Repetition structure
  - Counter-controlled
    - While/do while loop: repeated until condition becomes false
    - For: loop repeated until counter reaches certain value
  - Sentinel value
    - Indicates "end of data entry"
    - Sentinel chosen so it cannot be confused with regular input
- Example
  ```cpp
  int product = 2;
  while ( product <= 1000 ) {
    product = 2 * product;
    cout << product;
  }
  ```

*Flowchart representation?
What is the output?*
**switch** Multiple-Selection Structure

- Test variable for multiple values
- Series of `case` labels and optional `default` case

```c
switch ( variable ) {
    case value1:      // taken if variable == value1
        statements
        break;     // necessary to exit switch
    case value2:
    case value3:     // taken if variable == value2 or == value3
        statements
        break;
    default:          // taken if none matches
        statements
        break;
}
```

© 2003 Prentice Hall, Inc. All rights reserved.

**break and continue** Statements

- **break** statement
  - Immediate exit from `while`, `for`, `do/while`, `switch`
  - Program continues with first statement after structure
- Common uses
  - Escape early from a loop
  - Skip the remainder of `switch`
Logical Operators

- Used as conditions in loops, if statements
- **&&** (logical **AND**)
  - true if both conditions are true
  - if ( gender == 1 && age >= 65 )
    ++seniorFemales;
- **||** (logical **OR**)
  - true if either of condition is true
  - if ( semesterAverage >= 90 || finalExam >= 90 )
    cout << "Student grade is A" << endl;

Logical Operators

- **!** (logical **NOT**, logical negation)
  - Returns true when its condition is false, & vice versa
  - if ( !( grade == sentinelValue ) )
    cout << "The next grade is " << grade << endl;
  - Alternative:
    if ( grade != sentinelValue )
    cout << "The next grade is " << grade << endl;
Confusing Equality (==) and Assignment (=) Operators

- **Common error**
  - Does not typically cause syntax errors
- **Aspects of problem**
  - Expressions that have a value can be used for decision
    - Zero = false, nonzero = true
  - Assignment statements produce a value (the value to be assigned)

  ```cpp
  if == was replaced with =
  if ( payCode == 4 )
    cout << "You get a bonus!" << endl;

  What happens?
  ```

Confusing Equality (==) and Assignment (=) Operators

- **Lvalues**
  - Expressions that can appear on left side of equation
  - Can be changed
    ```cpp
    x = 4;
    ```
- **Rvalues**
  - Only appear on right side of equation
  - Constants, such as numbers (i.e. cannot write 4 = x;)
- **Lvalues** can be used as rvalues, but not vice versa

© 2003 Prentice Hall, Inc. All rights reserved.
Structured-Programming Summary

• Structured programming
  – Programs easier to understand, test, debug and modify

• Rules for structured programming
  – Only use single-entry/single-exit control structures
  – Rules
    1) Begin with the “simplest flowchart”
    2) Any rectangle (action) can be replaced by two rectangles (actions) in sequence
    3) Any rectangle (action) can be replaced by any control structure (sequence, if, if/else, switch, while, do/while or for)
    4) Rules 2 and 3 can be applied in any order and multiple times

Representation of Rule 3 (replacing any rectangle with a control structure)
Program Components in C++

- Modules: *functions* and *classes*
- Programs use new and “prepackaged” modules
  - New: programmer-defined functions, classes
  - Prepackaged: from the standard library
- Functions invoked by function call
  - Function name and information (arguments) it needs
- Function definitions
  - Only written once
  - Hidden from other functions