<table>
<thead>
<tr>
<th><strong>Course #:</strong></th>
<th>CSCI-C 101</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Title:</strong></td>
<td>Computer Programming I</td>
</tr>
<tr>
<td><strong>Course Type:</strong></td>
<td>Required core</td>
</tr>
<tr>
<td><strong>Prerequisites:</strong></td>
<td>M014 Basic Algebra or Level 3 on the math placement exam.</td>
</tr>
<tr>
<td><strong>Credits:</strong></td>
<td>4</td>
</tr>
<tr>
<td><strong>Text Book:</strong></td>
<td>Absolute C++, Second Edition, Walter Savitch, Addison Wesley</td>
</tr>
</tbody>
</table>

**References:**
- Handouts
- Program Development and Design Using C++, by Gary Bronson
- C++ How to Program, Fifth Edition, Deitel and Deitel

**Current Catalog Description:**
Fundamental concepts of computer literacy and computer programming, algorithm development, and basic data structuring. The programming language used will be C++.

**Course Goals**
The student who completes this course:
1. Will be able to understand the basic problem-solving process using algorithm and pseudo-code development.
2. Will be able to solve problems such as sorting, searching, and others via computer programs.
3. Will be able to implement computer programs in a computer programming language, specifically using C++.
4. Will be able to develop, test and debug programs in C++ using the Microsoft Visual Studio .NET IDE environment.
5. Will be able to read, use, test and debug C++ programs or components written by others.
6. Will build a good foundation for understanding and learning advanced programming concepts, such as, object-oriented programming, in the next level.

**Major Topics Covered in the Course**
1. Introduction to computers.
   - Types of computers and computer organization (CPU, memory, peripherals).
   - Basic operating systems.
   - Simple design concepts
   - Number representations and data storage (bits, bytes, words, addresses, $K = 1024$.)
   - Stored program concept. Machine instructions.
   - High level languages and compilers.
   - Ethical behavior in IUSB computer science courses.
   - Pseudo-code.
• Algorithm design.
• Top-down design. Modularity.
• Program style: indenting, choices for identifiers, using named constants, commenting.

3. Programming Environment Topics
• Using Windows 200X, XP and the Visual C++ integrated environment.
• Program testing.
• Debugging techniques, including using the debugger.

4. Fundamental Programming Techniques
• Swapping the values in two variables.
• Simple loop based algorithms, including counting, sentinel-controlled, input validation and general conditional.
• When to use "for", "while" and "do-while" loops.
• Searching: linear and binary search.
• Sorting: cover insertion sort and possibly one other sort.
• Processing character streams using cin.get(), EOF and cin.good().

5. Programming Assignments
• The students should be kept busy with programming assignments. Students seem to do better if assignments are due weekly. No assignment during a week when a test is scheduled allows students to study!

6. The C++ Programming Language
• Basic syntax (rules for identifiers, declarations, etc.).
• Libraries (cmath, cstdlib, string) and namespaces
• Basic, enumerated and aggregate data types. The inexactness of stored floats.
• I/O operators, functions in iostream and iomanip.
• Operators
• The "operator concept": All C++ operators return a value.
• Precedence of operators.
• Postpone discussion of following operators: scope resolution, new, delete, comma, and bitwise operators.
• Selection and repetition structures: if, if-else, switch, while, do-while, for, continue, break.
• Functions and parameter passing (value and reference parameters). Function overloading
• Scope: Static, external, and global variables and constants
• One-dimensional arrays and basics multi-dimensional array
• Pointer basics and their relationship with arrays.
• Basic strings and string operations
• Preprocessor directives: #define and #include.
• Basics of conditional compilation: ifndef, endif.

Laboratory projects (specify number of weeks on each)

Laboratory: 1 hour/week, totally about fourteen lab sessions. Laboratory projects vary by instructor, however, the lab exercise are designed to closely support the material covered in class.

Estimate Curriculum Category Content (Semester hours)

<table>
<thead>
<tr>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Comp. Arch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Structures</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Prog. Languages</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

Additional hours may be dedicated to curriculum categories not listed above. For example explanation of concepts and theories. Discussion of social and ethical issues, discussion of interpersonal relationships and working within groups.

Oral and Written Communications

Not a course objective.

Social and Ethical Issues

Ethical behavior is discussed during the first lecture.

Theoretical Content

Not a course objective.

Problem Analysis

In three of the class homework, students are required to follow the software development life cycle to develop programs, in which they need to document the problem analysis and algorithms.

Solution Design

Approximately ten programming assignments are assigned. Students must analyze the requirements, and design the solutions. Three of them require the students follow the software development life cycle to develop the programs.

The design steps would be the following:
1) Understand what the question or problem is asking.
2) Identify what is needed by the problem; all inputs, outputs and equations.
3) Determine the constraints or limits on the problem.
4) Write an algorithm to solve problem.
5) Translate the algorithm into a program.
6) Run and test the program.

Prepared By

Adaikkalavan, Zhang, Holloway, Hakimzadeh