COMPUTER SCIENCE CURRICULUM

AP COMPUTER SCIENCE A COURSE DESCRIPTION/STANDARDS

John Klimek, Math Coordinator, K - 12
Curt Witthoff, Math/Science Specialist, Secondary

Dr. Benjamin Marlin
Superintendent of Schools
Advanced Placement Computer Science A

Topic Outline
Topic Outline

Following is an outline of the major topics covered by the AP Examinations in Computer Science. The ordering here is intended to define the scope of the course, but not necessarily the sequence. The topics in the right-hand column will not be tested on the Computer Science A examination.

I. Program Design
The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.

<table>
<thead>
<tr>
<th>Computer Science A and AB</th>
<th>Computer Science AB only</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Problem definition</td>
<td></td>
</tr>
<tr>
<td>1. Specification of purpose and goals</td>
<td></td>
</tr>
<tr>
<td>2. Identification of objects and classes (abstract data types)</td>
<td></td>
</tr>
<tr>
<td>3. Identification of class responsibilities (operations on abstract data types)</td>
<td></td>
</tr>
<tr>
<td>B. Program design</td>
<td></td>
</tr>
<tr>
<td>1. Design of user/client interface</td>
<td></td>
</tr>
<tr>
<td>2. Choice of data structures and algorithms</td>
<td></td>
</tr>
<tr>
<td>3. Function decomposition</td>
<td></td>
</tr>
<tr>
<td>4. Identification of reusable components from existing code</td>
<td></td>
</tr>
</tbody>
</table>
II. Program Implementation

The overall goals of program implementation parallel those of program design. Modules of the program that fill common needs should be built so that they can be reused easily in other programs. Procedural and data abstraction are important parts of program implementation.

*Computer Science A and AB*  
*Computer Science AB only*

A. Implementation techniques
   1. Methodology
      a. Object-based development
      b. Top-down development
   2. Use of abstraction
      a. Abstract data types
         (including encapsulation and information hiding)
      b. Procedural abstraction

B. Programming constructs
   1. Declaration
      a. Constant declarations
      b. Variable declarations
      c. Class declarations
      d. Function declarations
      e. Parameter declaration
         i. Value
         ii. Reference
         iii. Constant reference
   2. Input and output
      a. Interactive
      b. Files
   3. Control
      a. Sequential
      b. Conditional
      c. Repetition
         i. Iteration
         ii. Recursion
      d. Functions (including member functions)

C. Generic data types and functions
   1. AP classes
   2. Templates
III. Program Analysis

The analysis of programs includes analyzing and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or the algorithms they implement so as to understand their time and space requirements when applied to different data sets.

Computer Science A and AB

A. Testing
   1. Testing classes and modules in isolation
   2. Identifying boundary cases and generating appropriate test data
   3. Integration testing

B. Debugging
   1. Categorizing errors: compile-time, run-time, logic
   2. Identifying and correcting errors
   3. Techniques: using a debugger, adding extra output statements, hand-tracing

C. Understanding and modifying existing code

D. Handling errors — robust behavior

E. Reasoning about programs
   1. Pre/post conditions
   2. Assertions

F. Analysis of algorithms
   1. Informal comparisons of running times
   2. Exact calculation of statement execution counts

G. Numerical limits
   Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error)

Computer Science AB only

3. Invariants

3. Big-Oh notation

4. Worst case/average case time and space analysis
IV. Standard Data Structures

Data structures are the means by which the information used by a program is represented within the program. Abstraction is an important theme in the development and application of data structures.

**Computer Science A and AB**

A. Simple data types (e.g., int, char, bool, double, strings)
B. Aggregate data types
   1. Heterogeneous (structs)
   2. Homogeneous (arrays)
C. Classes

**Computer Science AB only**

D. Linked lists
E. Stacks
F. Queues
G. Trees
H. Heaps
I. Priority queues

V. Standard Algorithms

Standard algorithms can serve as examples of good solutions to standard problems. Programs implementing them can serve as models of good program design. They provide examples for analysis of program efficiency. Many are intertwined with standard data structures.

**Computer Science A and AB**

A. Operations on data structures
   1. Traversals
   2. Insertion
   3. Deletion

**Computer Science AB only**

B. Operations on dynamic data structures
   1. Traversals
   2. Insertion
   3. Deletion
   4. Allocation/deallocation of memory
Computer Science A and AB  Computer Science AB only

C. Searching
1. Sequential (linear)
2. Binary

D. Sorting
1. Selection
2. Insertion
3. Mergesort merge algorithm
4. Quicksort partition algorithm

3. Hashing

5. Heapsort

VI. Computer Systems
A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the importance of considering the ethical and social implications of computing systems. These topics need not be covered in detail, but they should be considered throughout the course.

Computer Science A and AB  Computer Science AB only

A. Major hardware components
1. Primary and secondary memory
2. Processors
3. Peripherals

B. System software
1. Language translators
2. Separate compilation
3. Operating systems

C. Types of systems
1. Single-user systems
2. Networks

D. Responsible use of computer systems
1. System reliability
2. Privacy
3. Legal issues and intellectual property
4. Social and ethical ramifications of computer use
Case Studies

A case study is a document that includes the statement of a problem, one or more programs that solve the problem, and a written description of one expert's path from problem statement to solution program(s). The write-up describes the choices made for design and implementation and the justification for the choices that were made.

Case studies provide a vehicle for presenting many of the topics of the AP Computer Science courses. They provide examples of good style, programming language constructs, fundamental data structures, algorithms, and applications. Moreover, case studies provide an economical way to deal with large programs. Large programs give the student practice in the management of complexity and motivate the use of certain programming practices (including thorough procedural decomposition, intermodule communication through parameter passing, and selection of data structures tailored to the needs of the problem) in a much more obvious way than do small programs.

Case studies are most valuable, however, in teaching programming methodology. They allow the teacher to show concretely the design and implementation decisions leading to the solution of a problem and thus to focus more effectively on those aspects of the programming process. This approach gives the student a model of the programming process as well as a model program. The use of case studies also gives the student a context for seeing the importance of good design when a program is to be modified.

The 2003 AP Computer Science Examinations will include questions based on the case study described in the document "AP Marine Biology Case Study." Both the A and AB examinations will contain at least five multiple-choice questions and one free-response question covering material from the case study. Printed excerpts from the case study programs will accompany the examinations.

Questions will deal with activities such as the following:

a. modifying the procedural and data organization of the case study program to correspond to changes in the program specification;

b. extending the case study program by writing new code;

c. evaluating alternatives in procedural and data organization;

d. evaluating alternative incremental development strategies;

e. understanding how the components of the program fit together; and

f. developing test data.
For the AP Marine Biology Case Study appearing on the 2003 examinations, sample questions appear in the teacher's manual. The text and code for the "AP Marine Biology Case Study" are available for downloading from AP Central.

The Examinations

The AP Examinations for Computer Science A and Computer Science AB are each three hours long and seek to determine how well students have mastered the concepts and techniques contained in the respective course outlines. Before the examination date, students must decide which of the two examinations they will take. In most cases students will prepare during the year for one examination or the other. Some students enrolled in the AB course, however, may not feel comfortable with some of its more advanced topics. Such students might prefer to take the Computer Science A examination.

Each examination consists of two sections: a multiple-choice section (40 questions in 1 hour and 15 minutes), which tests proficiency in a wide variety of topics, and a free-response section (4 questions in 1 hour and 45 minutes), which requires the student to demonstrate the ability to solve problems involving more extended reasoning.

The multiple-choice and the free-response section of both AP Computer Science Examinations require students to demonstrate their ability to design, write, analyze, and document programs and subprograms. As noted in the AP C++ subset, the AB examination may include free-response questions that ask about design as well as implementation of classes. A design question would provide students with a description of the type of information and operations on that information that an object should encapsulate. Students would then be required to provide part or all of a class declaration to define such objects. An example of this type of question appears in the sample free-response questions for Computer Science AB (see question 2 on pages 58-59).
A design question may require a student to develop a solution that includes the following:

- declaration of constructor(s) and member functions with
  - meaningful names
  - appropriate parameters
  - appropriate return types
  - use of const qualifier for functions and parameters when appropriate
- appropriate data representation
- appropriate placement of data and member functions into public and private sections
  - all data should be private
  - all client accessible operations must be specified as public member functions

Minor points of syntax are not tested on the examinations. All code given in the exams is consistent with the AP C++ subset and AP C++ classes. All student responses involving code must be written in C++. Students are expected to be familiar with and able to use the standard AP C++ classes. For both the multiple-choice and the free-response sections of the examinations, a quick reference to both the case study and the AP C++ classes will be provided.

In the determination of the grade for each examination, the multiple-choice section and the free-response section are given equal weight. Because each examination is designed for full coverage of the subject matter, it is not expected that many students will be able to correctly answer all the questions in either the multiple-choice section or the free-response section.