The Reasoning for The Advanced Placement C++ Subset

Owen L. Astrachan
Duke University
Durham, NC
ola@cs.duke.edu

Gail Chapman
Educational Testing Service
Princeton, NJ
gchapman@ets.org

Susan H. Rodger
Duke University
Durham, NC
rodger@cs.duke.edu

Mark Allen Weiss
Florida International University
Miami, FL
weiss@cs.fiu.edu

Abstract

The programming language used for the Advanced Placement Computer Science (AP CS) courses and examinations will change from Pascal to C++ beginning with the 1998-1999 academic year. This decision, proposed by the AP Computer Science Development Committee and approved by the College Board, was made in 1994. The 5-year transition period was crucial in defining a C++ subset and providing time for the high school teachers teaching AP CS to attend professional development activities. In [1], the authors misunderstand the reasoning and the forces that helped define the AP C++ subset. This paper attempts to correct those.

1 Introduction

The College Board, not the Educational Testing Service (ETS) as mentioned in [1], is responsible for the Advanced Placement (AP) Program consisting of 32 college-level exams in 19 disciplines, including the Computer Science (CS) A and AB exams. ETS is responsible only for the administration and grading of the exam.

In 1994, the AP Computer Science Development Committee proposed to the College Board that the programming language used in the AP CS courses and examinations be changed from Pascal to C++, effective with the 1999 examination.[1] This decision was based on the increased use of C++ in CS 1 and CS 2 courses in colleges, and not based on industry's use of C++ as implied by [1]. An advisory committee, the Ad Hoc Committee, was formed and met in 1995-1996. This committee gave recommendations for the AP C++ subset. The draft subset was made available to the public [2] for comments in early 1996, and in early 1997 the final version was approved by the AP CS Development Committee.[2]

In the transition to C++, the Ad Hoc Committee had three main concerns in deciding on a C++ subset; these included providing standard AP C++ classes, providing professional development opportunities for high school teachers, and eliciting the involvement of the computer science community. The C++ subset and AP classes are described below. AP CS high school teachers must learn C++ and be familiar with the AP C++ subset and classes. Since existing C++ courses taught in various places are quite different, professional development workshops for high school teachers and College Board consultants (who will also lead workshops) in AP CS C++ began in June 1997. This allows for one year of acclimatization before teachers must begin using C++. Teachers teaching AP CS will begin using C++ in the fall semester 1998, with the first C++ exam administered in May 1999.

The final concern of the committee was to involve the computer science community. The composition of the Ad Hoc Committee consisted of high school teachers, college professors, representatives from SIGCSE and representatives from the AP CS Development Committee. Representatives from several areas clarified concerns from the different groups. A draft of the AP C++ subset was made available on the web and advertised to the different groups. In addition, a panel was held at SIGCSE 1997 and a listserv [4] was created for any comments on AP CS.

2 College Board Philosophy of AP CS

We give a shortened version of the College Board philosophy of AP CS. The complete version can be found in [3].

Since C++ is a large and complex language, the Advanced Placement Computer Science (AP CS) course will not cover the entire language; a restricted subset of C++ has been chosen for use in the AP CS course and examination. The criteria for defining the subset were as follows. It should support the activities, techniques, and constructs listed in the *AP Computer Science Topic Outline.* It should include classes and constructs that minimize or allow easy detection of common novice errors. It should be as small as possible, while still being extensive enough for the development of programs of sufficient size and complexity for the AP CS course.

An important feature of the subset is its inclusion of classes and related features that facilitate the design of programs based on modern concepts of data abstraction, encap-

[1] Originally the date for the change was the 1998 exam. The Ad Hoc Committee recommended waiting another year, and the Development Committee concurred.

[2] Note that in [1] the spelling of both authors' names of [2] was incorrect; the correct spelling is Astrachan and Horwitz.

sulation, and information hiding. Object-oriented programming in its full glory is not covered – in particular, features of C++ that allow classes to be related by inheritance are not part of the subset. The subset does, however, support the use of what is called "object-based" approach to programming, emphasizing data abstraction and encapsulation.

The subset omits a number of C++ features that are not necessary to design and implement large programs as part of an AP CS course. Some features, such as inheritance, are not included because they are not part of the AP CS course. Other language features either are redundant, easily replaced, or do not provide sufficient benefit to be included in the subset. What remains is sufficiently extensive to permit large and complex programs to be developed using generally accepted C++ idioms.

3 Reasons for Decisions on C++ Subset

The AP CS exam, like CS 1 and CS 2 is continually evolving. One of the principal forces that helped shape the C++ subset is the variety of current computing environments in high schools. Many schools use computers with little memory or drive space and are thus limited in which C++ compilers can be used. In [1] the authors are insensitive to the needs of high school teachers, stating "the standard will be finalized soon, and there is no reason to suppose that vendors will not support the full language in the near future." Furthermore, it will take time for high schools to update computers and add additional memory. For example, Visual C++ Version 5.0 uses 200 Mb of diskspace and requires 20 Mb of memory. Certain decisions in the C++ subset were made to allow more high schools to participate in AP CS. As compilers change and become standard, many of the following items listed below will change.

Topics not covered in the AP C++ subset can still be taught in an AP CS course. The subset is intended to clarify what will be tested on the AP exam so that students and teachers do not focus on details of the language that are not strongly emphasized.

3.1 bool

The file `bool.h` is supplied with the AP classes for use with those compilers that do not support `bool`. At least one commonly used compiler (Turbo 3.0/4.5) does not support the bool type.

3.2 Exceptions

Exceptions are appropriate for error-handling in many cases, but not all compilers support exceptions at this time. We recommend using the standard `assert` macro at this point. When exceptions are supported we will consider implementing an `apassert` function using exceptions.

3.3 apstring

The article [1] states that there is no need to rename the `string` class to `apstring`. The renaming is required in nearly all commonly used compilers on Macintosh and Wintel machines. A user-defined class string has a name conflict with the standard string class. The article is correct in stating that writing a null character in the middle of an `apstring` generates behavior inconsistent with the standard. This can be corrected.

3.4 Arithmetic precision

The article states we should mandate 32-bit integers. Since the C++ standard does not mandate this, and since most high schools are not currently using 32 bit systems, we could not make this a requirement. We do recommend use of long ints.

3.5 Casts

The article states that "Old-style casts are not necessary." While it is true that `long(i)` can be used in place of `(long int) i`, there are occasions where the old-style cast is appropriate, e.g., `(Node *) x`. The AP subset recommends the use of the `static_cast`$< .. >$ operator for casting, and many compilers support this form of cast.

3.6 apmatrix

There is no matrix class in the STL, so an `apmatrix` class using a vectors of vectors was created to support subscripting using `a[j][k]`. The vector of vector approach facilitates indexing using `[][]` and makes clear the advantages of implementing one class using another. Implementing the matrix class directly in terms of two-dimensional arrays, as suggested in the article, would require using `a(j,k)` notation to index rather than `a[j][k]`. Using the vector of vector approaches permits matrix access to look the same as two-dimensional array access, i.e., to use double subscripting with `a[j][k]`.

Note that using the vector of vector classes makes it possible to allow ragged matrices and to write `a[j]` to get an entire row of the matrix, which reflects how the built-in 2-d array type works. This is also possible when allocating 2-d built-in arrays in C++, and the Ad Hoc Committee wanted the use of the `apmatrix` class to be close to the built-in 2-d array type. Furthermore, we see nothing wrong with allowing ragged matrices if a programmer wants to use them.

3.7 Expressions

The subset restricts the use of increment (`++`) and decrement (`--`) to shorthand for `+=1` and `-=1` respectively, i.e., `x++` is used only as a shorthand for `x+=1` rather than in expressions like `a[x++]=y`. Understanding the semantic differences between `x++` and `++x` is something that students will do in more advanced programming, but is not necessary to write efficient, readable code. Limiting the decisions stu-

dents must make in writing code helps beginning programmers to be more effective.

3.8 Implementation of Stacks/Queues

The AP stack and queue classes were created to make testing of these common classes easier and to be consistent with the STL classes. The article [1] states that the authors prefer lists and that "vectors are not considered an appropriate implementation for queues." The authors are incorrect in stating that stack operations using the AP vector class are not amortized constant time. Vectors, as used in the `apstack` and `apqueue` classes, provide amortized constant time bounds for all stack and queue operations. Essentially, the constant time bounds derive from the manner in which a vector is resized: doubling the size, even with a copy of the original data into the new space, results in amortized constant time bounds.

3.9 Names of Member functions

The classes `apqueue` and `apstack` use member functions `isEmpty()` and `length()` whereas the STL uses `empty()` and `size()`. This decision was made because `length()` is also used with string classes and `isEmpty()` was prefered as a better identifier than `empty()`. Empty does not indicate whether it tests for emptiness or makes the object empty. Adding synonym functions such as `empty()` and `size()` to the AP classes so that code written with STL syntax can be used with the AP classes is a reasonable course of action. The final AP classes will support both styles.

3.10 List class

Of course a list class (and a deque class, and a map class) could be provided as part of the AP classes. However, the number of initial AP classes was kept small to help teachers get acclimated to C++ without worrying about needing to use and cover many classes in addition to a case study, computer science topics, and the new language.

3.11 Elementary types: enum and typedef

The type `enum`s are useful in providing symbolic names to a set of constants. Plans are in place to incorporate `enum`s as part of the final AP C++ subset. On the other hand, `typedef`s are not needed in C++ as they are in C because classes (and structs) put new names into the global name space in a different way. Students and teachers can use `typedef`s, but their use will not be tested on AP exams.

4 Conclusion

The article [1] states that we should be thinking "long term" about the subset. We disagree and believe that we should be thinking long term about the content of the course (meaning basic principles); the underlying language is by-and-large a more short-term decision. Nevertheless, we will certainly revisit the AP C++ subset each year; it will evolve as necessary.

In the last paragraph of [1] the authors point to a well-defined subset, Embedded C++. It is unclear if the authors mean for this subset to be used in place of the AP C++ subset, or as an example of how to define a subset. Clearly Embedded C++, designed for use in embedded systems and with support for neither templates nor for any operations dealing with files, cannot be used in any reasonable course in computer science. The Embedded C++ subset does include a rationale, a description of the subset, and other supporting documentation. The same documentation is supplied as part of the AP C++ subset.

Comments Welcome

The AP CS Development Committee will update the AP C++ subset as high school environments change and compilers are updated. Feedback is always welcome and can be sent to the members of the Development Committee or posted on the AP CS listserv for discussion. Information about AP CS and joining the AP CS listserv can be obtained from the College Board AP CS web site [4].

Authors

Owen L. Astrachan is an Associate Professor of the Practice at Duke University, was the Chair of the Ad Hoc Committee and the Chief Reader of the AP CS exam from 1990-1994.

Gail Chapman is a Program Director for the Advanced Placement Program at Educational Testing Service, is a consultant to the AP Computer Science Development Committee, and was a consultant to the Ad Hoc Committee.

Susan H. Rodger is an Associate Professor of the Practice at Duke University and is currently the Chair of the AP CS Development Committee.

Mark Allen Weiss is a Professor at Florida International University, was a member of the Ad Hoc Committee and is currently a member of the AP CS Development Committee.

References

[1] M. Ben-Ari and K. Henney, A Critique of the Advanced Placement C++ Subset, *SIGCSE Bulletin*, V. 29, N. 2, p. 7-10, 1997.

[2] Owen Astrachan and Susan Horwitz. Second Call for Comments: C++ in APCS, 1996, `http://www.cs.duke.edu/~ola/ap/cplus.html`.

[3] Advanced Placement Course Description, Computer Science, May 1998, May 1999, The College Board.

[4] College Board AP CS web site. `http://cbweb1.collegeboard.org/ap/computer-science/html/indx001.html`.