September 2, 1992

AIPS++ Familiarization

R.M. Hjellming and B.E. Glendenning

This note is a brief summary of recommendations for those who wish to become familiar with AIPS++ so they can participate in the analysis, design, implementation, and testing of AIPS++. While it was decided in June of 1991 that AIPS++ would be implemented in C++ with object-oriented techniques, the experience of the first six months of 1992 in Charlottesville made it clear that much more is involved that learning the syntax and semantics of C++ as a programming language, although that is indeed part of what is needed for work in AIPS++.

We divide the learning areas for AIPS++ workers into four categories:

1. Becoming familiar with object oriented concepts;

2. Learning to understand and work on O-O analysis and design;

3. Learning the tools of analysis, design, and programming; and

4. Understanding the current state of the analysis and design of AIPS++.

We also recognize that there are three levels of familiarity. The first is for those who simply want to be familiar with the work in AIPS++, the second is for those who are mainly going to code in AIPS++, and the third is for those who will actively contribute to the analysis and design of AIPS++ system and applications areas.

There is an exploding literature related to Object-Oriented software design and implementation. The AIPS++ project has adopted the conceptual approach of Rumbaugh, Blaha, Premerlani, Eddy, and Lorensen (1991): "Object-Oriented Modeling and Design". Many other books, particularly Booch's "Object-Oriented Design" can provide useful insights to O-O software, but the "methodologies" and notation that are offered can appear similar to a "Tower of Babel" for those trying to learn the O-O approach for the first time.

The AIPS++ Project Book contains the best description of current work on the project. It evolves on a day-to-day basis. It can be read in either its on-line current form (using ainfo or doing a Ghostview preview of the postscript version), or the postscript version can be printed out. Ghostview is a public domain program that works well on workstations and which allows: preview of postscript files on the screen; hunt and find of sections based on the indices at the end (negative pages in the Ghostview page display bar to the left of the page display); and selected printing of ranges of pages selected by a click and drag with selection from the action menu. This preview and selective printing is recommended in order to both keep up to date and to avoid excessive printing. Workers on the project should become familiar with all of the Project Book. (A postscript version of the project book will always be available via anonymous ftp, aips2.cv.nrao.edu:/pub/aips++/docs/PBook.ps).

In order to become familar with object-oriented concepts in AIPS++ the first step is a combination of reading Part 1 of Rumbaugh et al. (1991) and the Introduction and Software Design sections of the project book. You should thoroughly understand object-oriented concepts like "encapsulation," "inheritance," and "polymorphism" as well as the fundamental concepts of the AIPS++ design, in particular the relations between the "Telescope" and "Imaging" models. For those just wishing to be familiar with the work of the AIPS++ project that may be sufficient. For those wishing to do analysis, design, and implementation in AIPS++, the rest of the Rumbaugh book is recommended. Regular reading of the Journal of Object-Oriented Programming and/or Object Magazine is useful. The AIPS++ project has adopted the Mark V Systems ObjectMaker program as the CASE tool for analysis and design. Participation in analysis and design will be strongly aided by this tool, and communication between development sites will be largely in the form of the ACSII document files (.bde files) produced by this tool.

Proto-typing or coding in AIPS++ will require extensive familiarity with both C++ and the coding and documentation standards for the AIPS++ project. The latter will be in the Project Book when completed. Learning C++ is like learning any other computer language, even though it is most important to make class design, inheritance, and method polymorphism high on the list of things to be learned. A good textbook, like Lippman's "C++ Primer" (1991, 2nd Edition), and use of a reasonable C++ compiler for learning exercises, should suffice for any experienced programmer to learn the syntax and semantics of AIPS++. For learning purposes any compiler using the C-front 2.1 (or preferable 3.0) preprocessor will do, as will the Gnu C++ compiler, or Borland's C++ 3.0 compiler. The Centerline C++ (ObjectCenter, once known as Sabre) development environment is recommended as an excellent one for both learning and code development.

As the project proceeds, use of standard class libraries will be most important. At they are adopted or developed the Project book will document, or point the way to, these class libraries. Eventually most of AIPS++ will be a set of class libraries to be used, with a standard interface and architecture, to write AIPS++ applications. During middle and late 1992 even the most basic class libraries will be evolving. Some of the commercially available C++ class libraries, that are closest to the computational orientation that AIPS++ will have, are the Rogue Wave math.h++, matrix.h++, and linpack.h++ libraries.

Understanding object-oriented software involves special conceptual barriers and the need to put aside old habits dominated by the procedure-oriented approach of languages like FORTRAN, C, Pascal, etc. If there are not a series of "Aha, that is what they mean", you may be trying to fit new concepts into old ruts, like new wine in old wine skins. Good luck.