- **Section 12:** Polymorphic Management — Describes the mode of managing symbolic constants and run-time symbolic objects and packages. Also discusses the **Generic** class, which is inherited by most COOL classes and manipulates lists of symbols to manage type information.

- **Section 13:** Exception Handling — Describes COOL exception handling, which is a raise, handle, and proceed mechanism that uses the COOL symbolic computing capability.

- **Section 14:** COOL Methodology — Describes COOL methodology for managers and programmers who need a brief overview of COOL components, organization, style guide, and rules for extending the library.

Conventions

Certain conventions and syntax are used to simplify presentation of the material in this manual. The following typographical conventions are used:

- **Bold** type — Words in bold represent either a class name, macro name, or system-supplied function.

- *Italics* — Words in italics enclosed by angle brackets or parenthesis represent arguments that must be specified by the programmer .

- `Monowidth Font` — Program examples and output are distinguished by monowidth font of a smaller size than the normal manual text. (e.g., `#include <COOL/String.h>`)

Program examples illustrated and defined within this manual are supplied and can be found in the `~COOL/examples` subdirectory. Examples are given specific file names by appending a suffix of `.C` to its associated paragraph number. For example, the string class example in paragraph 2.4 is located in the `2.4.C` filename.

Each section includes a requirements paragraph that states the prerequisites needed to understand and use the components or functions being discussed.

Certain paragraphs document reference information about COOL classes. A special format is used that provides the following information:

- Class Name
- Synopsis
- Base Classes
- Friend Classes
- Constructors
- Member Functions
- Friend Functions
- Example Program

ABOUT THIS MANUAL

Introduction

This manual contains supporting software documentation for COOL (the C++ Object-Oriented Library), a collection of classes, objects, templates, and macros that extend the capabilities of the C++ language. This manual is written for high level C++ application programmers using COOL.

Organization

This manual is divided into the following sections and appendixes:

- **Section 1:** Overview of COOL — Contains an introduction to the *COOL User's Manual* and what to expect on the various classes, macros and other enhancements to C++ that are discussed in this manual.
- **Section 2:** String Classes — Describes a collection of classes that implement textual operations and functions.
- **Section 3:** Number Classes — Contains information on a collection of numerically oriented classes that augment the built-in numerical data types to provide extended precision, range-checked types, and complex numbers.
- **Section 4:** System Interface Classes — Includes classes for calculating the date and time in different time zones and countries.
- **Section 5:** Parameterized Templates — Describes classes that allow a programmer to design and implement a class template without specifying the data type.
- **Section 6:** Ordered Sequence Classes — Describes classes that are a collection of basic data structures that implement sequential-access data structures as parameterized classes.
- **Section 7:** Unordered Sequence Classes — Describes classes that are a collection of basic data structures that implement random-access data structures as parameterized classes.
- **Section 8:** Set Classes — Describes classes that implement two basic data structures for random-access set operations.
- **Section 9:** Node and Tree Classes — Describes classes that are a collection of basic data structures that implement several standard tree data structures as parameterized classes.
- **Section 10:** Macros — Describes COOL macro facilities that are an extension to the standard ANSI C macro preprocessor functions and that support constant symbols, keyword and body arguments, parameterized templates and complex expression evaluation.
- **Section 11:** Symbols and Packages — Describes functions that manage error message textual descriptions, provide polymorphic extensions to C++ for object type and contents queries, and support sophisticated symbolic computing.

Printed on: Wed Apr 18 07:18:42 1990

Last saved on: Tue Apr 17 12:13:29 1990

Document: preface

For: skc