Foreword

Ironically, it was in Waterloo that the STL was adopted as part of the ISO/ANSI Standard C++ Library, and from that day on it went onto a triumphal march. Alexander Stepanov and Meng Lee had proposed the result of years of research at Hewlett-Packard, a standard template library, to the standards committee. The committee gracefully adopted the STL as part of the C++ Standard at a committee meeting in Waterloo in the summer of 1994, after countless controversial discussions and much work spent by committee members on making the STL fit for a standard. Most importantly, the adoption was tied to the condition that the source code had to be made publicly available. Since then the STL has become more and more popular in the C++ community and conquered the hearts of quite a number of programmers. Personally, I know of software developers who cannot imagine getting their work done anymore without a general-purpose library like the STL. Obviously, not all Waterloos are the same. This Waterloo was in Ontario – seemingly a good omen.

Much of the merit, however, is not seriously due to picking the right location for presenting a library. The STL is an invaluable foundation library that makes programmers more productive in two ways. It contains a lot of different components that can be plugged together, so it provides a flexible and extensible framework. Plus, it has an elegant, consistent, and easy to comprehend architecture.

When Ulrich asked me in fall 1995 whether I would feel like writing this book with him, my first thought was: Does the world really need another STL book? Three books had already been out at that point in time; I had volunteered for writing a regular column about the STL for a magazine of high renown like *C++ Report*; numerous conference organizers invited me to speak about the STL; even my employer had me prepare and conduct courses on the STL. In sum, there were countless resources available to meet the growing interest in the C++ community. I simply questioned the need for yet another STL tutorial. About a year later, I held the German edition of his book in my hands, skimmed through the pages, and started reading – with increasing enjoyment. And I must admit, he convinced me. This book goes beyond the tutorials I had seen up to then and has an approach and appeal of its own: it explains techniques for building your own data structures and algorithms on top of the STL and this way appreciates the STL for what it is – a framework. I had been looking for this aspect in tutorials, often in vain.

As can be expected, the book starts off with an introduction to the STL. Already the initial explanations provide you with insights into the internals of the STL that you miss in other introductory material. For instance, Ulrich explains what the implementation of an iterator typically looks like. This kind of information is profound enough to lay the foundations for leaving the realm of simple STL usage, and enables you to understand and eventually extend the STL framework by adding your own containers and algorithms. The most distinguishing part of this book is Part III: Beyond the STL. You will see demonstrations of elegant and sophisticated usage of the STL – well-known data structures like matrices and graphs built on top of the STL, as well as examples of additions to the STL, like hash-based containers.

I would also want to acknowledge that this revised English edition of the book is one of the most accurate and up-to-date sources of information on the STL currently available. It reflects the ISO/IEC C++ Standard which was published in September 1998. Keep up with the language standard and learn how the STL will improve your programs. In sum, I enjoyed the book and appreciate it as a sound and serious reference to the STL. I hope you will also.

Angelika Langer
June 1999