Глава 1

Об этой книге

Несмотря на то что шаблоны входят в C++ уже добрый десяток лет (и практически столько же времени доступны в том или ином виде в различных компиляторах), их использование до сих пор сопряжено с непониманием, неправильным применением или противоречиями. В то же время значение шаблонов как мощного инструмента для создания эффективного, быстрого и гибкого программного обеспечения с каждым днем возрастает. Сегодня концепция шаблонов уже стала краеугольным камнем ряда новых парадигм программирования на C++.

Авторам пришлось столкнуться с тем, что в большинстве существующих книг и статей трактовка теоретических положений и применения шаблонов C++ является в лучшем случае поверхностной. И даже в тех немногих публикациях, где дается квалифицированный обзор различных технологий программирования, основанных на шаблонах, описание их поддержки средствами языка оставляет желать лучшего. В результате как начинающие, так и опытные программисты на C++ при работе с шаблонами сталкиваются с трудностями, пытаясь понять, почему их код работает не так, как ожидалось.

Именно эти соображения и послужили толчком к созданию данной книги. Однако оба автора пришли к этой идее независимо друг от друга, и их позиции несколько различаются.

- Целью Дэвида было создание самодостаточного справочника, включающего подробное описание механизма языка шаблонов и основных современных приемов программирования на базе шаблонов. Самое важное, с точки зрения Дэвида, — полнота и точность изложения материала.
- Нико ставил перед собой задачу написать книгу, которая помогла бы ему и другим программистам использовать шаблоны на практике. Его кредо можно было бы выразить так: интуитивно понятное изложение материала и акцент на практических аспектах применения шаблонов.

В какой-то степени у нас получился альянс ученого и инженера: мы оба имеем дело с одной и той же областью знаний, но сферы наших интересов несколько различны (хотя, разумеется, и имеют много общего).

Начало нашему сотрудничеству положило издательство Addison-Wesley, и сейчас перед вами результат этого сотрудничества. Нам хочется верить, что у нас получи-

лась основательная работа, сочетающая в себе тщательно проработанное учебное пособие по шаблонам и детальный справочник. В качестве учебного пособия она охватывает не только введение в элементы языка, но и развитие понимания методов конструирования, лежащих в основе практических решений. Точно так же эта книга является не только детальным справочником по синтаксису и семантике шаблонов C++, но и кратким руководством по идиомам и технологиям языка — как широко известным, так и малознакомым.

1.1. Что необходимо знать, приступая к чтению этой книги

Чтобы получить максимальную пользу от работы с книгой, читатель должен быть знаком с C++. В данной книге дается детальное описание конкретного средства языка программирования, но не основ самого языка. Предполагается знакомство читателя с концепцией классов и наследования, а также умение писать программы на C++ с использованием таких компонентов, как потоки ввода-вывода и контейнеры из стандартной библиотеки C++. Кроме того, при необходимости в данной книге рассматриваются различные тонкие вопросы, которые могут не иметь прямого отношения к шаблонам. Таким образом обеспечивается доступность изложенного здесь материала как для квалифицированных специалистов, так и для программистов среднего уровня.

Изложение материала базируется на языке C++, соответствующем стандарту 1998 года [31], с учетом уточнений, приведенных в *первом списке технических опечаток* Комитета по стандартизации C++ (C++ Standardization Committee) [32]. Если вы чувствуете, что ваши знания основ C++ несколько устарели или отстали от современного уровня и их необходимо освежить, рекомендуем обратиться к дополнительным источникам информации, в частности [17, 18, 33]. В этих книгах содержится отличное введение в современный язык программирования C++ и его стандартную библиотеку. Кроме того, можно использовать публикации, перечисленные в библиографии.

1.2. Структура книги в целом

Цель данной книги — предоставить читателю базовые знания, необходимые для работы с шаблонами и использования в полной мере их преимуществ; кроме того, книга призвана обеспечить читателей информацией, которая даст опытным программистам возможность преодолеть современные ограничения в этой области. Исходя из этого, мы разбили материал книги на четыре части.

- Часть I, “Основы”. Здесь описаны основные концепции, положенные в основу шаблонов. Эта часть написана в стиле учебника.

- Часть II, “Углубленное изучение шаблонов”. Здесь представлены детальные сведения о языке. Эта часть является неплохим справочником по конструкциям, связанным с шаблонами.
- Часть III, “Шаблоны и конструирование”. В этой части рассматриваются фундаментальные приемы конструирования, поддерживаемые шаблонами C++, простирающиеся от тривиальных идей до сложных идиом (возможно, нигде до этого не опубликованных).
- Часть IV, “Нетрадиционное использование шаблонов”. Основана на предшествующих ей двух частях и рассматривает различные распространенные применения шаблонов.

Каждая из перечисленных частей книги состоит из нескольких глав. Книга также включает несколько приложений, которые охватывают материал, относящийся не только к шаблонам (например, вопросы перегрузки в C++).

Главы, входящие в состав первой части книги, требуют последовательного изучения. Например, глава 3 основана на материале, рассмотренном в главе 2. Однако в других частях книги связь между главами выражена не столь явно. Например, можно сначала проработать главу, посвященную функциям (глава 22, “Объекты-функции и обратные вызовы”), а уже потом приступать к чтению главы, посвященной интеллектуальным указателям (глава 20, “Интеллектуальные указатели”).

1.3. Как читать эту книгу

Если вы являетесь программистом на C++ и хотите получить общее представление о концепции шаблонов и поближе познакомиться с ней, то вам следует тщательно изучить часть I, “Основы”. С этим материалом имеет смысл хотя бы бегло ознакомиться даже тем, кто с шаблонами уже “на ты”, чтобы прочувствовать стиль и освоиться с используемой в книге терминологией. Эта часть также охватывает некоторые “материально-технические” аспекты, касающиеся организации исходного кода, содержащего шаблоны.

В зависимости от того, какой метод изучения материала вы предпочитаете, можно либо основательно изучить детальную информацию о шаблонах из части I, либо познакомиться с приемами практического программирования в части III (обращаясь к части II, если возникнут какие-либо вопросы). Последнее представляется особенно целесообразным в случае, если вы приобрели эту книгу для конкретных практических целей. Часть IV во многом подобна части III, но акцент в ней делается не на приемах конструирования, а на понимании роли шаблонов в конкретных приложениях. Следовательно, прежде чем приступать к изучению части IV, имеет смысл сначала ознакомиться с материалом, изложенным в части III.

Приложения содержат большое количество полезной информации, на которую сделано много ссылок в основной части книги. Кроме того, мы старались сделать их интересными и в качестве самостоятельного материала.

Опыт подсказывает, что лучше всего новые знания усваиваются на примерах. Поэтому в книге вы найдете большое количество примеров. Иногда это всего лишь несколько

строк кода, иллюстрирующих теоретическое положение, иногда — полноценные программы, реализующие конкретное применение материала. В последнем случае примеры снабжены комментариями C++ с описанием пути к файлу, в котором содержится код программы. Все эти файлы можно найти на Web-узле данной книги по адресу: `http://www.josuttis.com/tmplbook/`.

1.4. Некоторые замечания о стиле программирования

У каждого программиста на C++ свой стиль программирования, и авторы данной книги также не составляют исключения. Понятие стиля включает обычные вопросы: где помещать пробелы, разделители (скобки, фигурные скобки) и т.п. В целом мы старались придерживаться единого стиля, хотя иногда по ходу изложения приходилось делать исключения. Например, чтобы придать коду больше наглядности, в разделах руководства были широко использованы пробелы и осмысленные имена, в то время как при рассмотрении более сложных вопросов предпочтение отдавалось компактности.

Хотелось бы обратить внимание читателя на то, что в данной книге применяется несколько необычный подход к записи объявлений типов, параметров и переменных. Очевидно, что при объявлении возможно использование нескольких стилей:

```
void foo(const int &x);
void foo(const int& x);
void foo(int const &x);
void foo(int const& x);
```

Для обозначения целочисленной константы мы решили применять несколько непривычный порядок записи — `int const` вместо `const int`. Сделано это было по двум причинам. Во-первых, такой порядок обеспечивает более очевидный ответ на вопрос: "*Что* именно является константой?". "Что" — это всегда то, что находится перед модификатором `const`. Однако для выражения

```
int* const bookmark;  // Указатель не может изменяться,
                      // однако может изменяться
                      // значение, на которое он указывает
```

не существует эквивалентной формы, в которой модификатор `const` стоял бы перед оператором указателя `*`, хотя

```
const int N = 100;
```

эквивалентно

```
int const N = 100;
```

В этом примере константой является сам указатель, а не целочисленное значение, на которое он указывает.

Вторая причина связана с синтаксической подстановкой, часто встречающейся при работе с шаблонами. Рассмотрим два следующих определения типов[1]:

```
typedef char* CHARS;
typedef CHARS const CPTR; // Константный указатель на char
```

Если вместо CHARS буквально подставить его значение, то смысл второго объявления не изменится:

```
typedef char* const CPTR; // Константный указатель на char
```

Однако при обратном порядке записи, т.е. если const стоит *перед* именем определяемого типа, этот принцип неприменим. Рассмотрим вариант определений, альтернативный представленным выше определениям типа:

```
typedef char* CHARS;
typedef const CHARS CPTR;  //Константный указатель на char
```

В результате буквальной подстановки значения CHARS получается другой тип:

```
typedef const char* CPTR; // Указатель на константу char
```

Очевидно, что сказанное выше справедливо и для спецификатора volatile.

Что касается расстановки пробелов, то мы решили помещать пробел между амперсандом и именем параметра:

```
void foo(int const& x);
```

Это сделано для того, чтобы подчеркнуть разделение между типом параметра и именем параметра. Однако при такой записи еще более запутанными становятся объявления наподобие

```
char* a, b;
```

Здесь согласно правилам, унаследованным из C, a является указателем, а b — обычной символьной переменной. Чтобы исключить такого рода путаницу, мы просто стараемся избегать объявления нескольких переменных приведенным образом.

Данная книга не посвящена стандартной библиотеке C++, однако в ряде приведенных в ней примеров эта библиотека задействована. В общем случае мы используем заголовочные файлы C++ (например, <iostream>, а не <stdio.h>). Исключение составляет <stddef.h>. Мы применяем его вместо <cstddef> и, следовательно, не определяем size_t и ptrdiff_t с помощью префикса std::, поскольку таким образом обеспечивается большая переносимость, а применение std::size_t вместо size_t не дает никаких преимуществ.

---

[1] Заметим, что в C++ синоним типа определяет псевдоним, а не новый тип, например:

```
typedef int Length; //Length определяется как псевдоним int
int    i = 42;
Length l = 88;
i = l; // Корректно
l = i; // Корректно
```

1.5. Стандарт и практика

Несмотря на то что стандарт C++ доступен с конца 1998 года, вплоть до 2002 года не существовало широкодоступного компилятора, о котором можно было бы сказать, что он “обеспечивает полное соответствие стандарту”. Таким образом, сегодня поддержка языка в компиляторах реализована по-разному. Есть среди них такие, которые способны компилировать большую часть кода, приведенного в этой книге, однако многие достаточно популярные компиляторы с большинством наших примеров могут и не справиться. Для таких нестандартных реализаций C++, как правило, приводятся альтернативные приемы, которые должны обеспечить полное или частичное решение проблем, но некоторые из описанных в книге методик для таких компиляторов сегодня недоступны. Тем не менее авторы надеются, что эта проблема будет решена в широком масштабе, поскольку программисты требуют от производителей компиляторов строгого соответствия стандарту.

Однако даже с учетом упомянутых трудностей язык программирования C++ находится в постоянном развитии. Эксперты из сообщества C++ (независимо от того, входят они или нет в состав Комитета по стандартизации C++) уже обсуждают различные пути улучшения языка; при этом некоторые из потенциальных усовершенствований затрагивают шаблоны. Некоторые тенденции в этой области рассмотрены в главе 13, “Направления дальнейшего развития”.

1.6. Примеры кода и дополнительная информация

Получить доступ к демонстрационным программам и найти дополнительную информацию об этой книге можно на ее Web-узле по адресу: `http://www.josuttis.com/tmplbook/`.

Кроме того, большое количество дополнительной информации по рассматриваемой теме вы сможете получить на Web-узле Дэвида Вандевурда (`http://www.vandevoorde.com/Templates`) и в Web в целом. Начать советуем с просмотра списка литературы к данной книге.

1.7. Обратная связь с авторами

Мы приветствуем любые конструктивные отклики читателей — как отрицательные, так и положительные. Нам обоим пришлось основательно потрудиться, чтобы создать для вас эту книгу, которую, надеемся, вы оцените как отличную. Однако в определенный момент мы просто вынуждены были прервать работу над ней, поскольку подошел срок “выпуска продукта”. Следовательно, ни один из наших читателей не застрахован от того, что при изучении материала ему придется столкнуться с ошибками или несогласованностью, а также с отдельными моментами, которые

нуждаются в доработке или с тем, что отдельные темы в книге не освещены вообще. Ваши отклики дают нам возможность проинформировать всех читателей через Web-узел данной книги о найденных вами “узких местах” и улучшить таким образом ее последующие издания.

Связываться с нами лучше всего по электронной почте (`tmplbook@josuttis.com`); однако, прежде чем посылать сообщение, удостоверьтесь, пожалуйста, что найденная вами неточность отсутствует в списке опечаток на нашем Web-узле.

Заранее благодарим вас за сотрудничество.