

**Пышкин Е.В.**

# **ОСНОВНЫЕ КОНЦЕПЦИИ И МЕХАНИЗМЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**

БХВ-Петербург, Санкт-Петербург, 2005

## **Содержание**

### **Предисловие**

Преподавание: исторический экскурс  
Основная задача книги  
Благодарности

### **Введение**

Предпосылки  
Организация данной книги  
Исходные тексты программ и листинги программ  
О соответствии стандарту C++  
Одно замечание о терминологии

## **ЧАСТЬ I. ОСНОВАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**

### **Глава 1. Развитие языков как развитие абстрактных моделей**

- 1.1. Концепции проектирования и языки программирования
- 1.2. Абстрактные модели, лежащие в основе языков программирования
  - Абстрагирование в языках структурного императивного программирования
  - Другие абстрактные модели
- 1.3. Сущность объектно-ориентированного подхода
- 1.4. Вопросы и упражнения

### **Глава 2. Элементы объектной модели**

- 2.1. Смысл абстрагирования как элемента объектной модели
- 2.2. Смысл инкапсуляции
- 2.3. Отношения классов
  - Отношение обобщения
  - Отношение ассоциации
  - Отношение зависимости
  - Отношение реализации
- 2.4. Типы структурных иерархий
  - Структурная иерархия “is-part-of” (агрегирование как разновидность ассоциации)
  - Структурные иерархии “is-a” и “is-like-a”
- 2.5. Иерархии классов и модули: модель C++
- 2.6. Вопросы и упражнения

## **ЧАСТЬ II. КОНСТРУИРОВАНИЕ ТИПОВ**

### **Глава 3. Класс как основной механизм абстракции**

- 3.1. Структура и организация определения класса
  - Элементы-данные и элементы-действия
  - Определение класса
  - Управление доступом к членам класса

- Класс *QueueInt*: реализация инкапсуляции
- Порядок объявления открытых и закрытых членов класса
- Важное замечание об инкапсуляции
- 3.2. Инициализация объектов. Конструкторы
  - Конструктор как инструмент гарантированной инициализации
- 3.3. Спецификация и реализация класса
  - Спецификация класса
  - Реализация класса
- 3.4. Альтернативные способы инициализации объектов
  - Перегрузка конструкторов
  - Понятие о конструкторе по умолчанию
  - Права доступа в связи с конструкторами
  - Конструкторы как неявные преобразования
- 3.5. Разрушение среды функционирования объектов. Деструкторы
- 3.6. Основные возможности в связи с определением членов класса
  - Константные члены класса
    - Константные данные
    - Константные функции
  - Статические константы, переменные и функции
  - Члены класса, требующие обязательную инициализацию
  - Инициализация статических членов класса
  - Понятие о константном указателе *this*
  - Встроенные функции-члены класса
- 3.7. Перегруженные функции и операции-члены класса
  - Перегружаемые операции-члены класса
  - Перегружаемые операции, не являющиеся членами класса
  - Перегружаемые операции-друзья класса
  - Перегрузка операций ввода-вывода
  - Перегрузка других операций
  - Несколько замечаний по поводу перегрузки операций *new* и *delete*
- 3.8. Дружественный доступ. Локальные и вложенные классы
  - Несколько дополнительных замечаний о дружественном доступе
  - Локальные и вложенные классы
- 3.9. Копирование объектов класса
  - Проблемы копирования объектов и понятие поверхностного копирования
  - Реализация корректной семантики копирования объектов
    - Копирующий конструктор и перегрузка операции присваивания
    - Запрет копирования объектов данного класса
- 3.10. Вопросы и упражнения

## **Глава 4. Наследование и иерархии классов**

- 4.1. Наследование как основная форма отношения обобщения
  - Можно ли обойтись без наследования
  - Какие преимущества получает программист, используя отношение наследования классов
- 4.2. Методы производного класса в отношении к методам базового класса
  - Переопределение функций-членов базового класса в производном классе
  - Инициализация объектов производных классов
  - Наследование конструкторов и операций
  - Порядок вызова конструкторов и деструкторов в связи с иерархией классов
- 4.3. Управление доступом к членам класса в связи с наследованием
  - Наследование как многократное использование интерфейса
- 4.4. Реализация наследования: модели, отличные от C++
  - Однокоренная иерархия классов

- Соккрытие имен
- Неизменные методы и классы
- 4.5. Вопросы и упражнения

## Глава 5. Объекты классов и полиморфизм

- 5.1. Понятие о статическом и динамическом связывании
  - Статическое связывание. Установление типов объектов во время компиляции
  - Динамическое связывание. Установление типов объектов во время выполнения
- 5.2. Виртуальные функции – механизм реализации полиморфизма в C++
  - Общие сведения
    - Пример со студентами и аспирантами
    - Пример с геометрическими фигурами
    - Требования к сигнатуре виртуальной функции
  - Таблица виртуальных функций – основной элемент механизма реализации позднего связывания
    - Таблица виртуальных функций
    - Выбор корректной версии виртуальной функции
    - Инициализация указателя на таблицу виртуальных функций
    - Простой пример для иллюстрации затрат памяти на реализацию полиморфизма
  - Чисто виртуальные функции и абстрактные классы
  - Виртуальные функции: некоторые подробности
    - Разрушение среды функционирования полиморфных объектов
    - Виртуальные деструкторы
    - Вызов виртуальной функции из функции-члена класса
    - Можно ли в производном классе определить не виртуальную функцию вместо виртуальной функции базового класса?
    - Функции-не члены класса, работающие подобно виртуальным функциям
- 5.3. Динамическое связывание и приведение типов
  - RTTI (определение типа во время выполнения) и понижающее приведение типов
  - RTTI: проблемы использования
- 5.4. Несколько замечаний о преобразованиях типа в C++
  - Преобразование типов во время компиляции (преобразование *static\_cast*)
  - Преобразование типов во время выполнения (преобразование *dynamic\_cast*)
  - Преобразование типов для избавления от константности (*const\_cast*)
  - Преобразование «на свой страх и риск» (*reinterpret\_cast*)
- 5.5. Вопросы и упражнения

## Глава 6. Обработка ошибок на основе использования механизма исключений

- 6.1. Варианты обработки ошибок, не связанные с использованием исключений
  - Обработка ошибки на месте
  - Использование возвращаемых значений
  - Использование глобальных переменных или переменных-членов класса, исполняющих роль индикатора состояния приложения или объекта
  - Использование специально разработанных функций
  - Использование функций обратного вызова
- 6.2. Обработка исключений
  - Основная идея подхода
  - Что такое «исключение» и как оно образуется
  - Исключения следует перехватывать по ссылке

- Преимущества встраивания в язык обработки исключительных ситуаций
- 6.3. Особенности обработки исключений в языке C++
  - Типы исключений
    - Использование встроенных типов языка
    - Использование типов, определяемых пользователем
    - Группирование исключений
  - Обработчики исключений
    - Порядок записи обработчиков
    - Повторная генерация исключения
    - Использование обработчика *catch(...)*
    - Когда исключение считается обработанным?
  - Исключения в связи с ошибкой выделения памяти
  - Исключения в конструкторах и деструкторах
    - Конструкторы и исключения
    - Деструкторы и исключения
  - Спецификация исключений
  - Неожидаемые и неперехваченные исключения
  - Стандартные исключения
- 6.4. Особенности обработки исключений в языках Java и C#
  - Java: обязательная спецификация и обработка исключений
  - C#: необязательная обработка исключений
- 6.5. Вопросы и упражнения

## ЧАСТЬ III. ОРГАНИЗАЦИЯ И ВЗАИМОДЕЙСТВИЕ ТИПОВ

### Глава 7. Множественное наследование и интерфейсы

- 7.1. Множественное наследование для реализации суммы функциональностей
  - Обычное множественное наследование
  - Множественное наследование с общим базовым классом
- 7.2. Виртуальные базовые классы
- 7.3. Дискуссия о множественном наследовании. Понятие интерфейса
  - Реализация семантики наследования с интерфейсами на языке C++
  - Реализация семантики наследования с интерфейсами на языке C#
  - Конфликты имен и сигнатур при реализации интерфейсов
- 7.4. Роль интерфейсов в компонентном программировании.
- 7.5. Вопросы и упражнения

### Глава 8. Пространства имён в связи с модульностью и иерархией

- 8.1. Области действия и пространство имен
  - Программный проект: модули и интерфейсы
- 8.2. Пространства имён в C++
  - Помещение программных объектов в пространство имен
  - Использование программных объектов, объявленных в пространстве имен
  - Пространство имен как общая среда реализации программного проекта и пространство имен как внешний интерфейс для пользователей
  - Пространства имен: технические подробности
    - Разрешение конфликтов имен
    - Псевдонимы пространств имен
    - Открытость пространств имен
    - Объединение пространств имен
    - Вложенные пространства имен
    - Пространства имен как механизм управления версиями
  - Стандартное пространство имен. Пространство имен и код, использующий библиотеки C

- Проблемы пространств имен в C++
- 8.3. Пространства имён в Java
  - Именованние пакетов
  - Создание и использование пакета
  - Права доступа в связи с модульной организацией
- 8.4. Пространства имён в C#. Введение в архитектуру приложения .NET
- 8.5. Вопросы и упражнения

## **Глава 9. Введение в обобщённое программирование**

- 9.1. Обобщённое программирование без использования шаблонных функций
  - Определение функции сортировки простыми обмeнами для сортировки массива целых чисел
  - Определение универсальной функции сортировки простыми обмeнами
- 9.2. Шаблонные функции
  - Определение шаблонной функции
  - Использование шаблонной функции. Инстанцирование
- 9.3. Шаблонные классы – основной инструмент параметрического полиморфизма
  - Определение и использование шаблонного класса
  - Параметры шаблонов
- 9.4. Контейнеры и итераторы
  - Проектирование специализированных контейнеров
    - Определение специализированного контейнерного класса
    - Итерируемые контейнеры
    - Использование итератора специализированного контейнера
  - Проектирование стандартных контейнеров
    - Обобщенное представление стандартного контейнера и итератора контейнера
    - Определение контейнерного класса в стиле STL
    - Обобщенные функции для обработки стандартных контейнеров
    - Стандартная библиотека шаблонов (STL)
- 9.5. Контейнеры, построенные на основе однокоренной иерархии класса (модели Java, C#)
- 9.6. Иерархии шаблонных классов
  - Организация размещения данных в памяти
  - Постановка задачи
  - Разработка классов
- 9.7. Вопросы и упражнения

## **ЧАСТЬ IV. РАЗВИТИЕ МОДЕЛЕЙ**

### **Глава 10. Организация вычислительного процесса в управляемых средах**

- 10.1. Управляемый и неуправляемый код
  - Управляемый код в Java
  - Взаимодействие Java с неуправляемым кодом
  - Ограничения, накладываемые управляемым кодом
  - Управляемый код платформы Microsoft .NET
  - Код, безопасный по отношению к типам
  - Вместо резюме
- 10.2. Встроенные, размерные и ссылочные типы
  - Объекты и встроенные типы
- 10.3. Автоматическое удаление объектов (сборка мусора)

## **Глава 11. Развитие объектно-ориентированной модели управления типами**

### 11.1. Реализация полиморфизма

Java: полиморфизм по умолчанию

C#: спецификатор *new*, версии классов и виртуальные функции

### 11.2. Свойства

Поддержка свойств на уровне языка (на примере C#)

Особый случай: индексирование

### 11.3. Функции обратного вызова и делегаты

Понятие делегата и реализация в C#

### 11.4. Программирование, ориентированное на события

Обработка сообщений приложением Windows

Объектно-ориентированная архитектура приложения

Поддержка модели программирования, ориентированной на события, средствами языка

### 11.5. Специализированные атрибуты

Применение атрибутов, используемых компилятором

Использование атрибутов в период выполнения

Атрибуты, определяемые пользователем

### 11.6. Отражение (рефлексия)

Извлечение информации о типе

Создание экземпляров типов

Отражение методов класса

### 11.7. Интеграция кода и документации

Документирующие комментарии: модель Java

Документирующие комментарии: модель C#

## **Глава 12. Введение в компонентное программирование**

### 12.1. Понятие о компонентной архитектуре

Соккрытие реализации

Повторное использование кода

Динамическая компоновка и совместимость интерфейсов

### 12.2. Реализация основных элементов компонентной архитектуры на базе COM

Доступ к компоненту посредством запроса интерфейса

Идентификация интерфейсов, поддерживаемых компонентом

Управление временем жизни компонента

Явное выделение элементов архитектуры компонентного приложения

### 12.3. COM-сервер и COM-клиент: действующий макет

Реализация и использование DLL-компонента

Фабрики классов и регистрация компонентов

Преодоление языковой зависимости

Основные проблемы COM

### 12.4. Межязыковая и межплатформенная интеграция: проблемы и решения

### 12.5. Элементы компонентной архитектуры .NET Framework

Управляемые модули и сборки

Общая система типов .NET

Общезыковая спецификация

### 12.6. Вопросы и упражнения

## **Заключение**

## ПРИЛОЖЕНИЯ

### Приложение 1. Примеры заданий для практикума

#### Практикум. Задание №1

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Оформление результатов выполнения упражнения

#### Практикум. Задание №2

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Оформление результатов выполнения упражнения

#### Практикум. Задание №3

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Оформление результатов выполнения упражнения

#### Практикум. Задание №4

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Оформление результатов выполнения упражнения

#### Практикум. Задание №5

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Оформление результатов выполнения упражнения

#### Практикум. Курсовой проект

Цель задания

Основные умения, которые должны быть продемонстрированы студентом

Постановка задачи

Отчетность по заданию и подведение итогов

Рекомендуемые источники

### Приложение 2. Введение в потоковый ввод-вывод C++

#### П2.1. Иерархия классов ввода-вывода (заголовочный файл `iostream.h`)

Класс *ios* и производные классы

Классы для вывода *ostream*

Классы для вывода *ofstream*, *ostrstream*, *ostream\_withassign*

Класс для ввода *istream*

Классы для ввода *ifstream*, *istrstream*, *istream\_withassign*

Класс для ввода и вывода *iostream*

Классы для ввода и вывода *fstream*, *strstream*, *stdiostream*

#### П2.2. Вывод данных

Вывод данных встроенных типов

Вывод символов и строк

Буферизация

Форматирование вывода. Манипуляторы

Вывод для типов, не встроенных в C++

#### П2.3. Ввод данных

Ввод данных встроенных типов

Ввод символов и строк

Форматированный ввод. Манипуляторы

Ввод для типов, не встроенных в C++

П2.4. Файловые потоки

Ввод данных из файлового потока *ifstream*

Вывод данных в файловый поток *ofstream*

Использование потока *fstream* для реализации ввода и вывода при работе с одним и тем же файлом

Манипуляторы в связи с файловыми потоками

П2.5. Поточковый ввод-вывод в пространстве имён стандартной библиотеки C++

Иерархия классов стандартной библиотеки ввода-вывода

Класс *ios\_base*

Класс *basic\_ios* и структура *char\_traits*

Классы *basic\_istream* и *istream*

Классы *basic\_ostream* и *ostream*

Классы *basic\_iostream* и *iostream*

Организация работы с файловыми потоками

### **Приложение 3. Описание компакт-диска**

**Список источников**

**Предметный указатель**