Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П. Г. Демидова
Кафедра компьютерной безопасности и математических методов обработки информации

О. П. Якимова

Языки программирования
Часть 1

Лабораторный практикум

Рекомендовано
Научно-методическим советом университета для студентов, обучающихся по специальности Компьютерная безопасность

Ярославль 2010

Ярославский государственный университет

УДК 004.43
ББК З 973.2–018.1я73
Я 45

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009/10 года

Рецензент
кафедра компьютерной безопасности и математических методов обработки информации Ярославского государственного университета им. П. Г. Демидова

Я 45 **Якимова, О. П. Языки программирования. Ч. 1:** лабораторный практикум / О. П. Якимова; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2010. – 68 с.

Предназначен для студентов, обучающихся по специальности 090102.65 Компьютерная безопасность (дисциплина «Языки программирования», блок ОПД), очной формы обучения.

УДК 004.43
ББК З 973.2–018.1я73

Учебное издание

Якимова Ольга Павловна

Языки программирования
Часть 1

Лабораторный практикум

Редактор, корректор И. В. Бунакова
Верстка Е. Л. Шелехова

Подписано в печать 06.05.10. Формат 60×84 1/16.
Бум. офсетная. Гарнитура "Times NewRoman".
Усл. печ. л. 3,95. Уч.-изд. л. 2,16.
Тираж 50 экз. Заказ

Оригинал-макет подготовлен
в редакционно-издательском отделе Ярославского
государственного университета им. П. Г. Демидова.
Отпечатано на ризографе.
Ярославский государственный университет им. П. Г. Демидова.
150000, Ярославль, ул. Советская, 14.

Введение

Целью лабораторного практикума по курсу «Языки программирования» является получение практических навыков прикладного программирования с применением объектно ориентированного подхода (ООП) к проектированию и реализации программного обеспечения на единой универсальной платформе разработки приложений Microsoft .NET.

В задачи настоящего лабораторного практикума входит освоение таких основополагающих концепций ООП, как абстракция, наследование, инкапсуляция и полиморфизм. Разработка приложений ведется на языке объектно ориентированного программирования C# в среде Microsoft Visual Studio .NET.

Важным разделом практикума является исследование систем типизации среды .NET. При этом наибольшее внимание уделяется таким важным аспектам, как пространства имен, абстрактные типы данных, классы и методы, интерфейсы и обобщенные коллекции. В практикуме раскрываются вопросы, связанные с событийно-управляемым программированием. Краткое введение в теорию обработки событий сопровождается рядом примеров из практики программирования на языке C#. Отдельно исследуется обработка исключительных ситуаций.

В результате освоения практикума студенты получают возможность самостоятельной разработки широкого спектра прикладных программных решений в условиях современной компонентной архитектуры.

Лабораторная работа 1. **Объекты и классы (конструкторы, инкапсуляция, свойства, перегрузка операций)**

Цель работы: познакомиться с основой объектного подхода в языке C#, созданием объектов, классов и механизмом инкапсуляции на основе свойств.

Необходимые теоретические сведения

Классы и объекты

Формально класс – это пользовательский тип, состоящий из полей данных и методов, которые работают с этими данными. Множество полей данных представляет «состояние» экземпляра класса. Экземпляры класса иначе называются объектами.

Сила объектно ориентированных языков состоит в том, что в одном пользовательском типе за счет группировки данных и функциональности вы можете смоделировать поведение некоторой сущности реального мира. Например, для представления сотрудника в системе расчета зарплаты, необходимо создать класс, содержащий имя, текущую зарплату и идентификатор (табельный номер). Кроме того, класс Employee должен содержать методы GiveBonus() для начисления премии и DisplayStats() для вывода информации о сотруднике.

Синтаксис класса:

```
тип_доступа class имя_класса
{
тип_доступа тип имя_переменной1;
тип_доступа тип имя_переменной2;
…
тип_доступа возвращаемый_тип имя_метода1(список_параметров)
{
тело_метода
}
...
}
```

Модификаторы доступа определяют поле видимости данного класса. Для классов предназначены два модификатора или типа доступа:

- public – класс доступен для других компонент (сборок);
- internal – класс видим только внутри данной сборки (приложения).

По умолчанию применяется модификатор internal. Модификаторы доступа также указываются и перед полями и методами класса: private (по умолчанию), public, protected, internal и protected internal. Члены класса с типом доступа public доступны везде за пределами данного класса, с типом доступа protected – внутри членов данного класса и производных, с типом доступа private – только для других членов данного класса. Тип доступа internal применяется для типов, доступных в пределах одной сборки.

Классы в C# могут определять любое количество конструкторов. Конструктор класса – метод для инициализации переменных экземпляра класса объекта при его создании. Он имеет то же имя, что и его класс. В конструкторах тип возвращаемого значения не указывается явно. Конструкторы используются для присваивания начальных значений переменным экземпляра и для выполнения любых других процедур инициализации, необходимых для создания объекта.

Все классы имеют конструкторы независимо от того, определен он или нет. По умолчанию в C# предусмотрено наличие конструктора, который присваивает нулевые значения всем переменным экземпляра (для переменных обычных типов) и значения null (для переменных ссылочного типа). Но если конструктор явно определен в классе, то конструктор по умолчанию использоваться не будет. Приведем пример создания класса Employee, описанного выше.

```
public class Employee
{
 private string fullName; // Имя сотрудника
 private int empID; // табельный номер
 private float currPay; // зарплата
// Конструктор по умолчанию
```