Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра электронных приборов

ПРИКЛАДНАЯ ИНФОРМАТИКА

Методические указания к лабораторным работам
для студентов направления 200700.62 –
"Фотоника и оптоинформатика"

2013

Шандаров, Евгений Станиславович

Прикладная информатика: методические указания к лабораторным работам для студентов направлений «Фотоника и оптоинформатика» / Е.С. Шандаров; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Томский государственный университет систем управления и радиоэлектроники, Кафедра электронных приборов. - Томск : ТУСУР, 2013. - 14 с.

Предназначено для студентов очной и заочной форм, обучающихся по направлениям «Фотоника и оптоинформатика». по курсу «Прикладная информатика».

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и
радиоэлектроники»

Кафедра электронных приборов

УТВЕРЖДАЮ
Зав.кафедрой ЭП
_____С.М. Шандаров
«___» _____ 2013 г.

ПРИКЛАДНАЯ ИНФОРМАТИКА

Методические указания к лабораторным работам
для студентов направления 200700.62 - «Фотоника и оптоинформатика»

Разработчик

ст. преподаватель каф.ЭП
_____Е.С. Шандаров
«____»_____2013 г

2013

Содержание

Введение

Pascal – язык программирования высокого уровня, разработанный в 1970 г. Николаусом Виртом в качестве языка обучения структурному программированию. В ноябре 1970г вышла первая официальная публикация описания языка с изложением синтаксиса и семантики. Большую роль в развитие Pascal сыграла компания Borland International, создавшая Turbo-среду разработки.

Pascal – один из первых языков, основанный на строгой типизации и принципах структурного программирования.

В качестве компилятора в рамках курса предлагается использовать Free Pascal Compiler. Это свободно распространяемый, кроссплатформенный компилятор языка Pascal с открытыми кодами, совместимый с Turbo Pascal 7.0 и Delphi.

Лабораторная работа №1. Введение в объектно-ориентированное программирование. Наследование

1.1 Цель работы

Знакомство с основными принципами объектно-ориентированного программирования и синтаксисом языка Pascal для создания объектно-ориентированных приложений. Практическое применение принципа наследования – построение иерархической схемы классов.

1.2 Теоретическая часть

В языке Pascal основным элементом объектно-ориентированных приложений является объект. Синтаксис описания объекта:

```
type
    <имя объекта> = object
    <список членов(поля и заголовки методов)>;
end;
```

Члены класса можно разделить на две группы.

1. Поля данных – данные, определяющие состояние объекта
2. Методы – процедуры и функции, которые могут использовать поля объекта и внешние данные.

Каждый член класс имеет атрибут доступа, при помощи которого определяется “зона видимости” для члена класса. Всего таких атрибутов три:

public – член объекта может использоваться любой функцией(процедурой);

private – член объекта может использоваться только функциями-членами объекта;

protected – член объекта может использоваться функциями-членами объекта, а также членами объектов, для которого данный объект является базовым(предком).

Основные принципы ООП

Инкапсуляция(encapsulation). Комбинирование записей с процедурами и функциями, манипулирующими полями этих записей, формирует новый тип данных – объект.

Инкапсуляция производится таким образом, чтобы пользователь объекта мог видеть и использовать только интерфейсную часть класса (т. е. список декларируемых свойств и методов объекта и не вникать в его внутреннюю реализацию. Поэтому данные принято инкапсулировать в классе таким образом, чтобы доступ к ним по чтению или записи осуществлялся не напрямую, а с помощью методов.

Принцип инкапсуляции (теоретически) позволяет минимизировать число связей между объектами и, соответственно, упростить независимую реализацию и модификацию классов.

Наследование(inheritance). Наследованием называется возможность порождать один объект от другого с сохранением всех свойств и методов объекта-предка (прародителя) и добавляя, при необходимости, новые свойства и методы. Набор объектов,связанных отношением наследования, называют иерархией.

Наследование призвано отобразить такое свойство реального мира, как иерархичность.

Важно помнить то, что если характеристика однажды определена на каком-то уровне иерархии, то все объекты, расположенные ниже данного уровня, содержат эту характеристику.

Полиморфизм(polymorphism). Присваивание действию одного имени, которое затем совместно используется вниз и вверх по иерархии объектов, причем каждый объект иерархии выполняет это действие способом, именно ему подходящим.

Экземпляры объектных типов

Экземпляры объектных типов описываются в точности так же, как в Паскале описывается любая переменная, либо статическая, либо указатель, ссылающийся на размещенную в динамической памяти переменную:

```
var
```

```
    Emp: TEmployee;
```

Поля объектов

К полю объекта можно обратиться так же, как к полю обычной записи, либо с помощью оператора with, либо путем уточнения имени с помощью точки.

Например:

```
Emp.Rate := 10;
with Emp do
begin
    Name := 'Ivanov';
    Title := 'programmer';
end;
```

Методы

Внутри объекта метод определяется заголовком процедуры или функции, действующей как метод:

```
type
    TEmployee = object
    Name, Title: string[25];
    Rate: Real;
    procedure Init (AName,
 ATitle:
 String;
 ARate:
 Real);
end;
```

Примечание: Поля данных должны быть описаны перед первым описанием метода.

При определении метода после описания объекта имени метода должно предшествовать имя типа объекта, которому принадлежит этот метод, с последующей точкой:

```
procedure
 TEmployee.Init(AName,
 ATitle:
 string; ARate:
Real);
```

```
begin
    Name := AName;
    Title := ATitle;
    Rate := ARate;
end;
```

Создание объекта-наследника

```
<имя объекта> = object(<имя обекта-родителя>)
    <список членов>;
end;
```

При создании объекта-наследника, он наследует все поля и методы базового объекта, описанные атрибутами public и protected. Для этого класса нет необходмсоти снова определять эти поля и методы.

1.3 Экспериментальная часть

1.3.1 Задание на лабораторную работу

Создать приложение, реализующее следующую иерархию объектов

TPerson

/
\

TStudent
TTeacher

Объекты TStudent и TTeacher должны наследовать поля и методы объекта TPerson.

1.3.2 Порядок выполнения работы

Создать объект типа TPerson со следующими полями: фамилия, имя, отчество, дата рождения – и двумя методами, выводящими на экран ФИО (фамилию, имя, отчество) и дату рождения.

Создать объект TStudent – потомок класса TPerson, для которого определить новые поля (год зачисления в вуз, номер группы) и методы, выводящие эти поля на экран.

Создать объект

TTeacher – потомок класса

TPerson, для которого определить новое поле (должность) и метод,

выводящие это поле.

Создать одну переменную типа TPerson, две переменные типа TStudent и одну переменную типа TTeacher.

В основном блоке программы заполнить все поля всех экземпляров объектов (информацию пользователь вводит с клавиатуры) и вызвать методы.

1.3.3 Содержание отчета

Результат работы оформить в виде отчета, в котором обязательно привести:

- описание всех объектов, их полей и методов;
- скриншоты программы;
- листинг программы.

Лабораторная работа №2. Введение в объектно-ориентированное программирование. Конструкторы и деструкторы

2.1 Теоретическая часть

Существуют специальные методы объектов, которые отвечают за создание экземпляров объекта и их удаление. Это так называемые конструкторы и деструкторы. Конструкторы – методы, основная цель которых заключается в инициализации объекта и распределении памяти для хранения объекта. Как и любой другой метод, конструктор может иметь или не иметь параметров. Конструктор без параметров называется конструктором по умолчанию.

Деструктор разрушает созданный экземпляр. По своей форме конструкторы и деструкторы являются процедурами, но объявляются с помощью зарезервированных слов constructor и destructor.

Объекты могут размещаться в динамической памяти и ими можно манипулировать с помощью указателей. Паскаль включает несколько мощных расширений для выполнения динамического размещения и удаления объектов более легкими и более эффективными способами.

Free Pascal поддерживает расширенный синтаксис процедур new и dispose. В случае, когда нужно выделить память под динамическую переменную объектного типа, при вызове процедуры new может быть указано имя конструктора объекта:

```
Type
    TObj = object;
        Constructor Init
        Destructor Destroy;
        ...
    end;
    Pobj = ^TObj;
Var PP : Pobj;
```

Следующие 3 вызова эквивалентны:

```
pp := new (Pobj,Init);
new(pp,Init);
new (pp);
```

Таким же образом, для выполнения освобождения памяти деструктор можно вызывать как часть расширенного синтаксиса процедуры dispose:

```
dispose(pp, Destroy);
```

2.2 Экспериментальная часть

2.2.1 Задание на лабораторную работу

Создать приложение для работы с объектами типа “список”: создание объекта в динамической памяти, добавление и удаление элементов из списка, уничтожение объекта из динамической памяти.

2.2.2. Порядок выполнения работы

Создать объект типа TList со следующими обязательными методами: Init(конструктор объекта), Print(вывести список на экран), Add_Item(включение нового элемента в список), Delete_Item(удаление элемента из списка), Destroy(деструктор объекта) – и обязательным полем First, в котором хранится указатель на первый элемент.

В основном блоке программы в динамической памяти создать экземпляр объекта. Осуществить редактирование списка в форме диалога с пользователем (интерфейс должен быть эргономичным!!!). При выходе из программы очистить память, занимаемую списком.

2.2.3 Содержание отчета

Результат работы оформить в виде отчета, в котором кроме обязательных пунктов (титульный лист, цель работы, задание на работу с вариантом, заключение и листинг программы) обязательно привести:

– подробное описание всех объектов (назначение), их полей (назначение и тип данных) и методов (назначение и расширенный список формальных параметров);

– описание использования конструктора и деструктора объекта;

– скриншоты программы(на каждую операцию со списком).

Список рекомендуемой литературы

1. 1 Информатика. Базовый курс : учебное пособие для вузов / ред. С. В. Симонович. - 2-е изд. - СПб. : Питер, 2009
2. Лабораторный практикум по информатике : Учебное пособие для вузов/ В. С. Микшина, Г. А. Еремеева, К. И. Бушмелева и др; Ред. В. А. Острейковский. -М.: Высшая школа, 2003.-375 с.
3. PASCAL 7.0. Практическое программирование. Решение типовых задач: учебное пособие/ Лала Михайловна Климова. - 3-е изд., доп.. - М.: КУДИЦ-ОБРАЗ, 2002. - 516 с.
4. Офицеров Д.В. и др. Программирование на персональных ЭВМ. Практикум. -Минск, Высшая школа. 1993. -256 с.

Приложение А
Образец титульного листа отчета

Министерство образования и науки Российской Федерации

Федеральное государственное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления
и радиоэлектроники»

Кафедра Электронных приборов (ЭП)

Дисциплина «Прикладная информатика»

ОТЧЕТ
по лабораторной работе
«_____»

Выполнил студент гр. 348
_____И.О.
Фамилия
«_____» _____
20__ г

Проверил
_____
должность
_____И.О.
Фамилия
«_____»
_____20__ г

20___

Учебное пособие

Шандаров Е.С.

Прикладная информатика

Методические указания к лабораторным работам

Усл. печ. л. _____ Препринт
Томский государственный университет
систем управления и радиоэлектроники
634050, г.Томск, пр.Ленина, 40