МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»
(ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

Структуры и алгоритмы обработки данных в ЭВМ

Методические указания к выполнению курсового проекта для студентов специальности 230105 - "Программное обеспечение вычислительной техники и автоматизированных систем"

2012

Горитов А.Н.
Структуры и алгоритмы обработки данных в ЭВМ: методические указания к выполнению курсового проекта для студентов специальности 230105 – "Программное обеспечение вычислительной техники и автоматизированных систем" / А.Н. Горитов. – Томск: ТУСУР, 2012. – 11 с.

Методические указания разработаны в соответствии с решением кафедры автоматизированных систем управления

Составитель: д.т.н., профессор каф. АСУ А.Н. Горитов

Методические указания утверждены на заседании кафедры автоматизированных систем управлениям 28 июня 2012 г., протокол № 15

Содержание

1 Цель выполнения курсового проекта

Научиться описывать предметную область реального мира – объект и его атрибуты. Закрепить навыки использования основных структур данных, способов их описания и основных операций над ними. Освоить разработку удобного пользовательского интерфейса.

2 Теория

2.1 Непрерывная и дискретная информация

Информация о различных природных явлениях и технологических процессах воспринимается человеком (с помощью органов чувств и измерительных приборов) в виде тех или иных полей.

Математически такие поля представляются с помощью функций

$$y = f(x, t), \qquad (1)$$

где t – время, x – точка, в которой измеряется поле, y – величина поля в этой точке.

В большинстве случаев все скалярные величины, входящие в формулу (1), могут принимать непрерывный ряд значений, измеряемых вещественными числами (т.е. могут изменяться сколь угодно мелкими шагами). Поэтому информацию, представляемую таким способом, принято называть непрерывной информацией (или аналоговой).

Если же установить минимальные шаги изменения всех скалярных величин, входящих в формулу (1), то получим так называемое дискретное представление информации или дискретную информацию. Это вполне оправданный подход, т.к. точность измерений, как и человеческого восприятия, всегда ограничена. Т.е. даже непрерывную информацию мы всегда воспринимаем в дискретном виде. Но любая непрерывная информация может быть аппроксимирована дискретной информацией с любой степенью точности, поэтому дискретная форма представления информации – универсальна.

Результаты измерения любых скалярных величин представляются, в конце концов, в виде числа с конечным набором цифр (при заданной точности измерений). Поэтому дискретную информацию часто отождествляют с цифровой информацией.

2.2 Данные и ЭВМ

Обработка цифровой информации должна обеспечивать решение задач, связанных с реальным миром. Однако бесконечное множество объектов и связей между ними, составляющих наш мир, не может быть представлено в ограниченном объеме памяти любой машины, какой бы большой она ни была. Следовательно, необходимо построить некую ограниченную масштабную модель реального мира, которая учитывала бы только те данные или информацию и связи между объектами, которые касаются

именно рассматриваемой проблемы. Такая модель обычно имеет два уровня. Первый уровень является результатом абстракции и задает упрощенную логическую структуру данных. Второй уровень определяет преобразование этой структуры в физическую структуру данных, которая может быть непосредственно отображена в памяти машины и обработана с помощью программного обеспечения.

Структура данных – это набор правил и ограничений, которые показывают связи между отдельными элементами данных или группами данных. Она ничего не говорит об отдельных элементах данных, и любая информация о них является излишней при условии, что связи между ними установлены. Если некоторые данные в структуре сами являются структурами данных, то образуется некая иерархия структур данных.

2.3 Объекты предметной области

При решении конкретных проблем обычно ограничиваются той частью реального мира, которая является областью данной деятельности. Ее называют предметной областью (ПО). Для решения проблем деятельности нужна информационная модель ПО – описание структур данных на логическом уровне. Проектируя модель ПО, обычно связывают структуры данных с объектами ПО.

Объектами могут быть:

- люди, например, перечисленные в платежной ведомости;
- предметы, например, детали;
- построения, например, счета в задаче получения счетов.

Объект – это некая абстракция, которой можно дать уникальное и осмысленное *имя*. Оно отделяет конкретный объект от других подобных абстракций. Например, в ПО ВУЗ существует объект СТУДЕНТ – лицо, проходящее обучение в ВУЗе. В той же ПО существует объект ПРЕПОДАВАТЕЛЬ – лицо, обучающее СТУДЕНТов. Таким образом, объект есть **тип,** множество экземпляров, обладающих (в моделируемой ПО) сходными свойствами. Так, конкретный Иванов Виктор Леонидович является экземпляром объекта СТУДЕНТ.

Каждый объект обладает вполне определенным уникальным набором свойств. Эти свойства называются атрибутами. Атрибуты, как и объекты, имеют осмысленные имена. Так, атрибутами объекта СТУДЕНТ могут быть Фамилия, Имя, Отчество, Дата рождения, Номер студбилета,…, а атрибутами объекта ПРЕПОДАВАТЕЛЬ – Фамилия, Имя, Отчество, Дата рождения, Ученая степень,…Отметьте, что наборы атрибутов различных объектов могут пересекаться, но не могут совпадать. Таким образом, можно понимать объект как некоторый набор атрибутов.

Набор атрибутов должен быть достаточным для описания объекта в данной предметной области и не должен содержать избыточных атрибутов.

Каждый атрибут принимает значения из некоторого множества допустимых значений – домена. Поэтому говорят, что атрибут есть имя, определенное на домене. Границы доменов определяются смыслом данных и требованиями ПО. Так, атрибут *Пол* (человека) может принимать только два значения, поэтому его домен является двухэлементным множеством. Тип и значения соответствуют принятым в ПО, поэтому возможны различные варианты: {"М", "Ж" } или {"Муж.", "Жен."}, или {"1", "2"} и т.п.

Экземпляр объекта описывается конкретным набором значений атрибутов. Например, совокупность значений "Иванов", "Виктор", "Леонидович", "17.05.79", "9443625",… является описанием экземпляра объекта СТУДЕНТ. Такие совокупности называются кортежами. Поскольку кортеж соответствует экземпляру объекта, можно трактовать объект как множество кортежей.

2.4 Представление информации об объектах

Для любого объекта существует некоторая совокупность информации, которую можно назвать записью. Отдельные атрибуты образуют поля записи. Совокупность записей об объектах называют набором данных или файлом.

С точки зрения программиста объекту ПО соответствует тип записи. Отдельные атрибуты образуют поля записи.

Между этими категориями существует следующая связь:

- файл соответствует объекту;
- число экземпляров объекта равно числу записей в файле;
- число атрибутов, описывающих объект, равно числу полей в каждой записи.

Каждому полю соответствует имя поля. Поле записи имеет свое значение поля. Значение (содержимое) поля описывает атрибут.

3 Порядок выполнения работы и общие требования

3.1 Выбор темы курсовой работы

Тему курсовой работы студент выбирает либо из прилагаемого списка тем, либо предлагает свою тему и согласовывает ее с преподавателем, являющегося руководителем курсового проектирования. Считается, что студент приступил к выполнению курсового проекта только после утверждения темы руководителем.

3.2 Разрабатываемые вопросы

Во время работы над данным курсовым проектом необходимо разработать описание объекта выбранной предметной области. Описать структуру и атрибуты объекта средствами алгоритмического языка. Соста-

вить программу создания набора данных из записей об объектах. Составить программу формирования выходного документа.

3.3 Порядок выполнения

Выбор темы	1-е занятие
Определение атрибутов объекта и представление их в программе	2-е занятие
Разработка программы	3-6 занятия
Оформление пояснительной записки	7-е занятие
Защита курсового проекта	8-9 занятие

3.4 Требования к оформлению

Пояснительная записка должна отражать связь выполненной работы с изучаемым предметом и содержать следующие пункты:

1. Введение.
2. Содержательная постановка и описание задачи.
3. Атрибуты объекта и представление данных в программе.
4. Описание программы создания набора данных.
5. Описание программы формирования выходного документа.
6. Описание программы формирования списковой структуры.
7. Технология обработки данных.
8. Заключение.
9. Список литературы.

Кроме этого, в приложение необходимо включить:

1. Графическое описание данных.
2. Представление данных в памяти ЭВМ.
3. Рисунок списковой структуры.
4. Формат выходного документа.
5. Схема последовательности обработки данных.
6. Листинг программы.

В приложении приведены приметы оформления:

а) титульного листа;

б) описание структуры и атрибутов объекта средствами алгоритмического языка.

3 Темы курсовых работ

1. Электронный справочник «Домашняя фонотека»
2. Электронный справочник «Домашняя видеотека»
3. Электронный справочник домохозяйки
4. Электронный справочник «Библиотека музыкальной школы»
5. Электронный справочник транспортного агентства

4 Приложения

Приложение А

(справочное)

Пример оформления титульного листа курсового проекта

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»
(ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

(НАЗВАНИЕ)

Пояснительная записка к курсовому проекту по дисциплине "Структуры и алгоритмы обработки данных в ЭВМ"

Студент гр. (номер)
(подпись) И.О. Фамилия
(дата)

Руководитель проекта

(подпись) И.О. Фамилия
(дата)

(ГОД)

Приложение Б

(справочное)

Пример описания объекта и его атрибутов

3. ОПИСАНИЕ ОБЪЕКТА И ЕГО АТРИБУТОВ

Описание объекта - это совокупность значений всех атрибутов данного объекта, а атрибут - это некоторая характеристика объекта. В данной работе объектом является квартира. Каждая квартира имеет свою совокупность значений атрибутов. Число этих атрибутов одинаково у каждой квартиры. Различие между объектами заключатся в различных значениях их атрибутов. Ниже представлены характеристики объекта которые используются в программе.

1. Общая запись Haus.

1.1. Region - район расположения квартиры. Для него в записи выделяется строка длиной 20 байт.

1.2.TypeD - тип дома в котором расположена квартира. Для него выделяется один символ. Размер поля 1 байт.

1.3. NumberD - номер дома в котором расположена квартира. В записи используется тип Integer. Длина 2 байта.

1.4. FloorD - Число этажей в доме. В программе выделяется 1 байт. Используется тип Byte.

1.5. Внутренняя запись Flat.

1.5.1. Внутренняя запись Master.

1.5.1.1. Name - Имя владельца квартиры Для имени выделена строка длиной 10 байт

1.5.1.2. Famili - Фамилия владельца квартиры. Для фамилии выделяется строка длиной 10 байт.

1.5.2. Room - Число комнат в квартире. Представлено типом Byte. Выделяется 1 байт.

1.5.3. FloorK - Этаж на котором расположена квартира Представляется типом Byte. Выделяется 1 байт.

1.5.4. Area - Площадь квартиры. Представлена типом Real. Выделяется 6 байт.

1.5.5. Внутренняя запись Comfort.

1.5.5.1. BathRoom - Санузел. Представляется типом Boolean. Для него выделяется 1 байт.

1.5.5.2. Gas - Газ. Представляется типом Boolean. Для него выделяется 1 байт

1.5.5.3. Telefon - Телефон. Представляется типом Boolean. Для него выделяется 1 байт.

1.6. UnkNext - Указатель на следующий элемент структуры.

1.7. UnkPrev - Указатель на предыдущий элемент структуры. Для их представления используется тип Pointer.

Принимая во внимание все выше сказанное запись в программе примет вид:

```
type
Haus = record
Region : string[20];
TypeD : string[1];
NumberD : Integer;
FloorD : Byte;
   Flat record
      Master record
         Name : string[10];
         Famili : string[10];
      end;
     Room : Byte;
     FloorK : Byte;
     Area : Real;
     Comfort record
          BathRoom : Boolean;
          Gas : Boolean;
          Telefon : Boolean;
      end;
   end;
   UnkNext : Pointer;
   UnkPrev : Pointer;
end;
```