Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

А.Н. Стась

Технологии разработки программного обеспечения

Методические указания по выполнению практических, лабораторных работ и самостоятельной работе для магистрантов направлений 09.04.01 «Информатика и вычислительная техника» и 09.04.02 «Информационные системы и технологии»

Томск
2022

УДК 004.9
ББК 32.973.22
С 77

Рецензент:
Боровской И.Г., профессор кафедры экономической математики, информатики и статистики ТУСУР, докт. физ.-мат. наук

Стась, Андрей Николаевич
С 77 Технологии разработки программного обеспечения: метод. указания / А.Н. Стась. – Томск: Томск.гос. ун-т систем упр. и радиоэлектроники, 2022. – 18 с.

Методические указания по выполнению лабораторных работ и самостоятельной работе для магистрантов направлений 09.04.01 «Информатика и вычислительная техника», 09.04.02 «Информационные системы и технологии».

Одобрено на заседании каф. ЭМИС протокол № 1 от 30.08.2022 г.

УДК 004.9
ББК 32.973.22

Оглавление

Введение

Цель изучения дисциплины: Обучение студентов работе с современным оборудованием и программным обеспечением, развитие навыков получения, хранения, переработки и трансляции информации, в том числе при решении исследовательских задач.

Задачи дисциплины:

1) знакомство студентов с различными направлениями современных научных исследований в области информатики и вычислительной техники, их результатами и перспективами;

2) развитие у студентов способностей к эксплуатации современной компьютерной техники и к разработке программного обеспечения.

3) развитие навыков получения, хранения и переработки информации.

В результате освоения дисциплины обучающийся должен:

– знать понятие «жизненный цикл программного обеспечения», основные этапы жизненного цикла программного проекта, характеристики и особенности различных моделей, методы применения этих моделей в процессе разработки программного обеспечения;

– демонстрировать умения разрабатывать современное программное обеспечение; применять и создавать эффективные алгоритмы решения задач; конструировать информационные системы;

– иметь навыки участия в проектах по разработке современного программного обеспечения и руководства такими проектами;

– знать современные методы управления группами разработчиков программного обеспечения;

– демонстрировать умения профессионально организовать командную работу в проекте по разработке программного обеспечения;

– владеть методами и навыками командной работы над проектом и ее организации;

– знать основы управления проектами по разработке и модификации информационных систем, а также основные способы осуществления руководства коллективом;

– обладать управленческими умениями в области проектов по созданию программного обеспечения;

– владеть навыками управления и сопровождения проектов в области разработки и и модификации программного обеспечения;

– знать методы управления и руководства проектами в области разработки программного обеспечения на всех стадиях и этапах жизненного цикла;

– уметь управлять деятельностью команды разработчиков программного обеспечения на различных стадиях и этапах жизненного цикла проектов;

– владеть управленческими навыками в области разработки и поддержки комплексных проектов в области разработки программного обеспечения на всех стадиях и этапах.

1 Методические указания к практическим занятиям

ПРАКТИЧЕСКАЯ РАБОТА 1. Надежность программного средства и технологии его тестирования

Цель работы: получение навыков работы по организации тестирования программных средств.

Темы для предварительного изучения: Изучение опыта организации разработки программного обеспечения, извлеченный из практики ведущих разработчиков.

Задание 1.1

1. Изучить основные принципы организации тестирования.
2. Дать характеристику видов тестирования: функциональному и структурному.
3. Дать математическую оценку числа ошибок на основе алгоритмов «теории программного обеспечения».
4. Подчеркнуть важность тестирования. Дать примеры тестирования отдельных программ. Изучить вопрос об определении момента его окончания.
5. Изучить работу Холстеда по определению момента окончания тестирования и модифицируется его уравнение.
6. Обсудить метод определения момента окончания тестирования с помощью статистической оценки, полученной путем проверки за столом программы, в которую искусственно внесены ошибки. Найти, что эти статистические оценки существенно зависят от правил проверки программы за столом.

Задание 1.2

Организовать тестирование выбранного программного средства. Представить письменный отчет.

ПРАКТИЧЕСКАЯ РАБОТА 2. Технологии проектирования баз данных

Цель работы: получение навыков работы по проектированию базы данных, практическое освоение основных приемов и правил методологии информационного моделирования; получение навыков работы по созданию базы данных (БД), созданию и редактированию таблиц, созданию запросов на языке реляционных баз данных SQL.

Темы для предварительного изучения: Основные этапы разработки базы данных. Современные средства проектирования и программирования баз данных, освоение способов разработки концептуальной модели и реали-ация модели средствами Access. Назначение SQL. Порядок создания БД и таблиц. Функциональные категории команд SQL: DDL, DML, DCD.

Задание 2.1

Произвести анализ предметной области книжного издательства. Составить концептуальную модель (сущности, атрибуты, связи) и итоговую полноаттрибутную ER-диаграмму.

Введение. Основная цель системы обработки данных заключается в повышении эффективности работы компании, учреждения или организации. Система обработки данных должна: обеспечивать получение общих или детализированных данных по итогам работы; позволять легко определять тенденции изменения важнейших показателей; обеспечивать получение информации, критической по времени, без существенной задержки; выполнять точный и полный анализ данных. Одной из популярных среди настольных СУБД является Microsoft Access. Основными преимуществами являются: популярность среди многих конечных пользователей и осуществление высокой устойчивости данных, простота в освоении, использовании непрофессиональными программистами, возможность подготавливать отчеты

из баз данных различных форматов произвольной формы на основании различных данных; возможность разработки некоммерческих приложений.

Описание предметной области. База данных создаѐтся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчѐты. В соответствии с предметной областью система строится с учѐтом следующих особенностей: каждая книга издаѐтся в рамках контракта; книга может быть написана несколькими авторами; контракт подписывается одним менеджером и всеми авторами книги; каждый автор может написать несколько книг (по разным контрактам); порядок, в котором авторы указаны на обложке, влияет на размер гонорара; если сотрудник является редактором, то он может работать одновременно над несколькими книгами; у каждой книги может быть несколько редакторов, один из них – ответственный редактор; каждый заказ оформляется на одного заказчика; в заказе на покупку может быть перечислено несколько книг.

В результате анализа должны быть получены базовые сущности этой предметной области:

Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.

Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.

Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.

Контракты рассматриваются как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

Полученную модель реализовать в виде схемы БД MS Access путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таблицы БД).

Задание 2.2

Реализовать запросы в рамках БД:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
- получение списка редакторов, работающих над книгами;
- получение полной информации о книге (проекте);
- получение сведений о конкретном авторе (с перечнем всех книг);
- получение информации о продажах (по одному или по всем проектам);
- определение общей прибыли от продаж по текущим проектам;
- определение размера гонорара автора по конкретному проекту.

Задание 2.3

Аналогично заданиям 2.1–2.3 произвести проектирование ИС (выделить 4–5 базовых сущности, связи между ними, составить ER-диграмму), разработать БД и реализовать около 10 типовых запросов в соответствии с вариантом выданным преподавателем.

Варианты для задания 2.4: информационная система библиотеки, информационная система ВУЗа, информационная система швейного производства, информационная система

ресторана, информационная система больницы, информационная система склада, информационная система зоопарка, информационная система аэропорта, информационная система аптеки, информационная система автомастерской, информационная система школы, информационная система фотоцентра.

ПРАКТИЧЕСКАЯ РАБОТА 3. Использование CASE-средств для решения вопросов автоматизации разработки программного обеспечения

Цель работы: получение навыков работы с современными методами и средствами проектирования информационных систем.

Темы для предварительного изучения. Фирмы-поставщики CASE-средств. Программные средства поддержки жизненного цикла ПО. Оценка и выбор CASE-средств. Технология внедрения CASE-средств.

Задание 3.1

Ознакомление с назначением CASE-технологии на примере ErWin, предназначенного для построения логических и физических моделей предметных областей, проведения анализа и генерации готовых БД. Для создания моделей данных в ERwin можно использовать две нотации: IDEF1X и IE (Information Engineering). В данной работе будет использоваться нотация IDEF1X. Для внесения сущности в модель необходимо кликнуть по кнопке сущности на панели инструментов (ERwin Toolbox), затем кликнуть по тому месту на диаграмме, где Вы хотите расположить новую сущность. Кликнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт Entity Editor... можно вызвать диалог Entity Editor, в котором определяются имя, описание и комментарии сущности. Каждый атрибут хранит информацию об определенном свойстве сущности. Каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. Для описания атрибутов следует, кликнув правой кнопкой по сущности, выбрать в появившемся меню пункт Attribute Editor. Для установки связи между сущностями нужно воспользоваться кнопками в палитре инструментов. На логическом уровне можно установить идентифицирующую связь один ко многим, связь многие ко многим и неидентифицирующую связь один ко многим (соответственно кнопки – слева направо в палитре инструментов). Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Зависимая сущность изображается прямоугольником со скругленными углами. Экземпляр зависимой сущности определяется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности переносятся в состав первичного ключа дочерней сущности (миграция атрибутов). В дочерней сущности они помечаются как внешний ключ – (FK). При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности.

Порядок выполнения работы:

1. Ознакомиться с назначением и возможностями ErWin.
2. Разработать концептуальную модель издательства из задания 2.2.
3. Отобразить эту модель в среде ErWin.
4. Оформить отчет о проделанной работе.

Задание 3.2

Сгенерировать полученную модель в реальную СУБД на примере СУБД MS Access и mysql. Изучить особенности генерации SQL-кода.

Задание 3.3

С помощью CASE-средства ErWin осуществить проектирование ПО из задания 2.4 в соответствии с вариантом задания, выданным преподавателем.

2 Методические указания к лабораторным работам

ЛАБОРАТОРНАЯ РАБОТА 1. Методы разработки эффективных алгоритмов

Цель работы: Сравнение эффективных и неэффективных алгоритмов сортировки массивов.

Теоретический материал

Введем некоторые понятия и обозначения. Если у нас есть элементы

$$a_1, a_2, \text{K}, a_n$$

то сортировка есть перестановка этих элементов в массив

$$a_{k1}, a_{k2}, \text{K}, a_{kn}$$

где

$$a_{k1} \le a_{k2} \le \text{K} \le a_{kn}.$$

Считаем, что тип элемента определен как `INTEGER`.

```
Const n=???; //здесь указывается нужная длина массива
Var A: array[1..n] of integer;
```

Выбор `INTEGER` до некоторой степени произволен. Можно было взять и другой тип, на котором определяется общее отношение порядка.

Мы будем сначала классифицировать методы по их экономичности, т. е. по времени их работы. Хорошей мерой эффективности может быть C – число необходимых сравнений ключей и M – число пересылок (перестановок) элементов. Эти числа суть функции от n – числа сортируемых элементов. Хотя хорошие алгоритмы сортировки требуют порядка $n \cdot \log n$ сравнений, мы сначала разберем несколько простых и очевидных методов, их называют прямыми, где требуется порядка n^2 сравнений ключей.

Сортировка с помощью прямого включения

Такой метод широко используется при игре в карты. Элементы мысленно делятся на уже «готовую» последовательность a_1, K, a_{i-1} и исходную последовательность. При каждом шаге, начиная с $I = 2$ и увеличивая i каждый раз на единицу, из исходной последовательности извлекается i-й элементы и перекладывается в готовую последовательность, при этом он вставляется на нужное место.

Таблица 2.1 – Пример сортировки с помощью прямого включения

Начальные ключи	44	55	12	42	94	18	06	67
$I = 2$	44	55	12	42	94	18	06	67
$I = 3$	12	44	55	42	94	18	06	67
$I = 4$	12	42	44	55	94	18	06	67
$I = 5$	12	42	44	55	94	18	06	67
$I = 6$	12	18	42	44	55	94	06	67
$I = 7$	06	12	18	42	44	55	94	67
$I = 8$	06	12	18	42	44	55	67	94

ПРОГРАММА. СОРТИРОВКА С ПОМОЩЬЮ ПРЯМОГО ВКЛЮЧЕНИЯ

```
PROGRAM SI;
VAR
```

```
 I,J,N,X:INTEGER;
 A:ARRAY[0..50] OF INTEGER;
BEGIN
 WRITELN('Введите  длину  массива');
 READ(N);
 WRITELN('Введите  массив');
 FOR I:=1 TO N DO READ(A[I]);
 FOR I:=2 TO N DO BEGIN
  X:=A[I];
  A[0]:=X;
  J:=I;
  WHILE X<A[J-1] DO BEGIN
   A[J]:=A[J-1];
   DEC(J)
  END;
  A[J]:=X
 END;
 WRITELN('Результат:');
 FOR I:=1 TO N DO WRITE(A[I],' ')
END.
```

Сортировка с помощью прямого выбора

Этот прием основан на следующих принципах:

1. Выбирается элемент с наименьшим ключом.
2. Он меняется местами с первым элементом a_1.
3. Затем этот процесс повторяется с оставшимися $n-1$ элементами, $n-2$ элементами и т.д. до тех пор, пока не останется один, самый большой элемент.

Таблица 2.2 – Пример сортировки с помощью прямого выбора

Начальные ключи	44	55	12	42	94	18	06	67
$I = 2$	06	55	12	42	94	18	44	67
$I = 3$	06	12	55	42	94	18	44	67
$I = 4$	06	12	18	42	94	55	44	67
$I = 5$	06	12	18	42	94	55	44	67
$I = 6$	06	12	18	42	44	55	94	67
$I = 7$	06	12	18	42	44	55	94	67
$I = 8$	06	12	18	42	44	55	94	67

Такой метод сортировки – его называют прямым выбором – в некотором смысле противоположен прямому включению. Полностью алгоритм прямого выбора приводится в программе 2.3.

ПРОГРАММА. СОРТИРОВКА С ПОМОЩЬЮ ПРЯМОГО ВЫБОРА

```
PROGRAM SS;
VAR I,J,R,X,N:INTEGER;
    A:ARRAY[0..50] OF INTEGER;
BEGIN
 WRITELN('Введи  длину  массива');
 READ(N);
 WRITELN('Введи  массив');
 FOR I:=1 TO N DO READ(A[I]);
 FOR I:=1 TO N-1 DO BEGIN
  R:=I;
  X:=A[I];
  FOR J:=I+1 TO N DO IF A[J]<X THEN BEGIN
```

```
   R:=J;
   X:=A[R]
  END;
  A[R]:=A[I];
  A[I]:=X
 END;
 WRITELN('Результат:');
 FOR I:=1 TO N DO WRITE(A[I],' ')
END.
```

Сортировка с помощью прямого обмена

Как и в методе прямого выбора, мы повторяем проходы по массиву, сдвигая каждый раз наименьший элемент оставшейся последовательности к левому концу массива. Если мы будем рассматривать как вертикальные, а не горизонтальные построения, то элементы можно интерпретировать как пузырьки в чане с водой, причем вес каждого соответствует его ключу (см. таблицу 2.3.).

Таблица 2.3 – Пример пузырьковой сортировки

$I = 1$	2	3	4	5	6	7	8
44	06	06	06	06	06	06	06
55	44	12	12	12	12	12	12
12	55	44	18	18	18	18	18
42	12	55	44	42	42	42	42
94	42	18	55	44	44	44	44
18	94	42	42	55	55	55	55
06	18	94	67	67	67	67	67
67	67	67	94	94	94	94	94

Такой метод сортировки известен под именем «пузырьковая сортировка». Он представлен в следующей программе.

ПРОГРАММА. ПУЗЫРЬКОВАЯ СОРТИРОВКА

```
PROGRAM BS;
VAR I,J,X,N:INTEGER;
    A:ARRAY[0..50] OF INTEGER;
BEGIN
 WRITELN('Введи  длину  массива');
 READ(N);
 WRITELN('Введи  массив');
 FOR I:=1 TO N DO READ(A[I]);
 FOR I:=2 TO N DO  FOR J:=N DOWNTO I DO IF A[J-1]>A[J] THEN BEGIN
  X:=A[J-1];
  A[J-1]:=A[J];
  A[J]:=X
 END;
 WRITELN('Результат:');
 FOR I:=1 TO N DO WRITE(A[I],' ')
END.
```

Сортировка с помощью дерева

Метод сортировки с помощью прямого выбора основан на повторяющихся поисках наименьшего ключа среди n элементов, среди $n-1$ оставшихся элементов и т.д. Как же усовершенствовать упомянутый метод сортировки? Этого можно добиться, действуя согласно следующим этапам сортировки.

1) Оставлять после каждого прохода больше информации, чем просто идентификация единственного минимального элемента. Проделав $n-1$ сравнений, мы можем построить дерево выбора вроде представленного на рисунке 1.

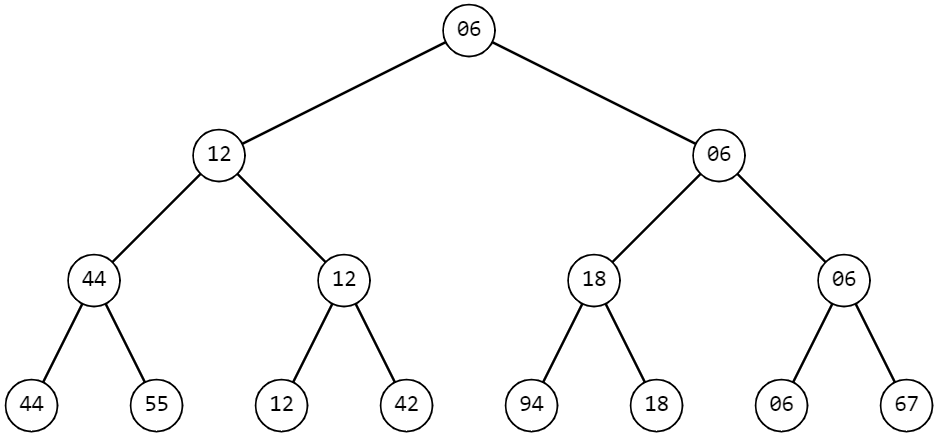

Рисунок 2.1 – Повторяющийся выбор среди двух ключей

2) Спуск вдоль пути, отмеченного наименьшим элементом, и исключение его из дерева путем замены либо на пустой элемент в самом низу, либо на элемент из соседней ветви в промежуточных вершинах (см. рисунки 2.2 и 2.3).

Д. Уилльямсом был изобретен метод *Heapsort*, в котором было получено существенное улучшение традиционных сортировок с помощью деревьев. Пирамида определяется как последовательность ключей $a[L],\ a[L+1],\ \ldots,\ a[R],$ такая, что

$$a[i] \le a[2i] \text{ и } a[i] \le a[2i+1] \text{ для } i = L\text{K}\ \frac{R}{2}.$$

Р. Флойдом был предложен некий «лаконичный» способ построения пирамиды на «том же месте». Здесь $a[1]\text{K}\ a[n]$ – некий массив, причем $a[m]\text{K}\ a[n]$ $\left(m = [n\ \text{DIV}\ 2]+1\right)$ уже образуют пирамиду, поскольку индексов i и j, удовлетворяющих соотношению $j = 2i$ (или $j = 2i+1$), просто не существует.

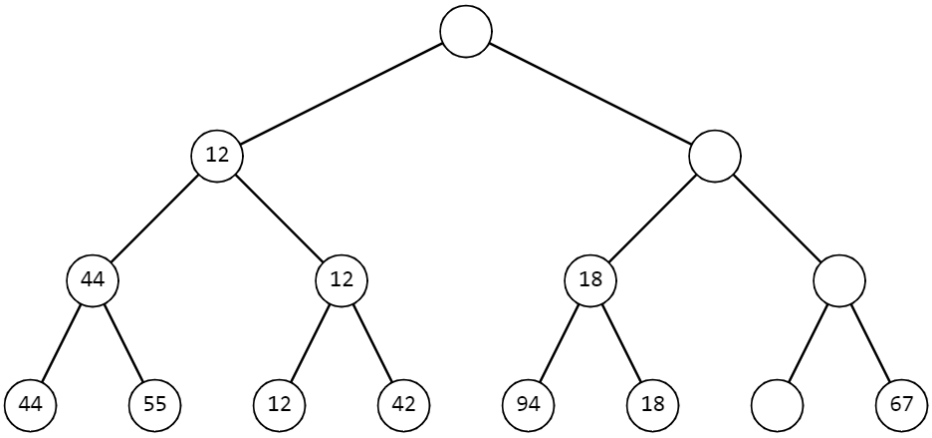

Рисунок 2.2 – Выбор наименьшего ключа

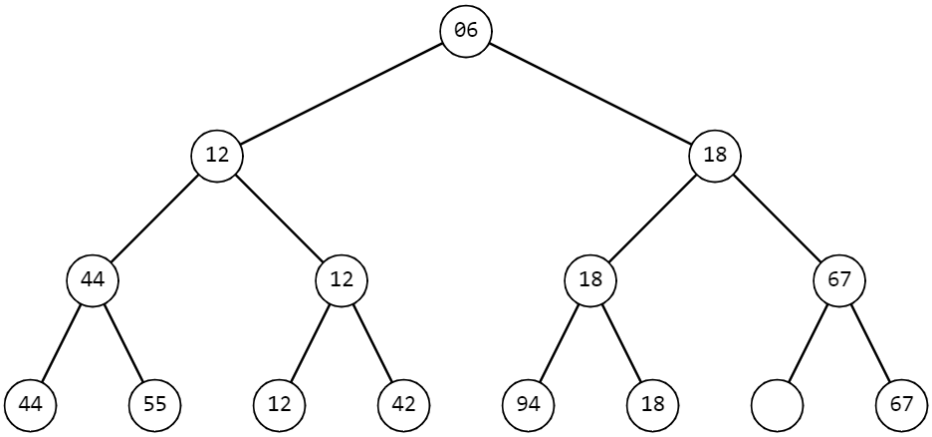

Рисунок 2.3 – Заполнение дырок

Эти элементы образуют какбы нижний слой соответствующего двоичного дерева (см. рисунок 2.4), для них никакой упорядоченности не требуется. Теперь пирамида расширяется влево; каждый раз добавляется и сдвигами ставится в надлежащую позицию новый элемент. Таблица 2.6 иллюстрирует этот процесс, а получающаяся пирамида показана на рисунке 2.4.

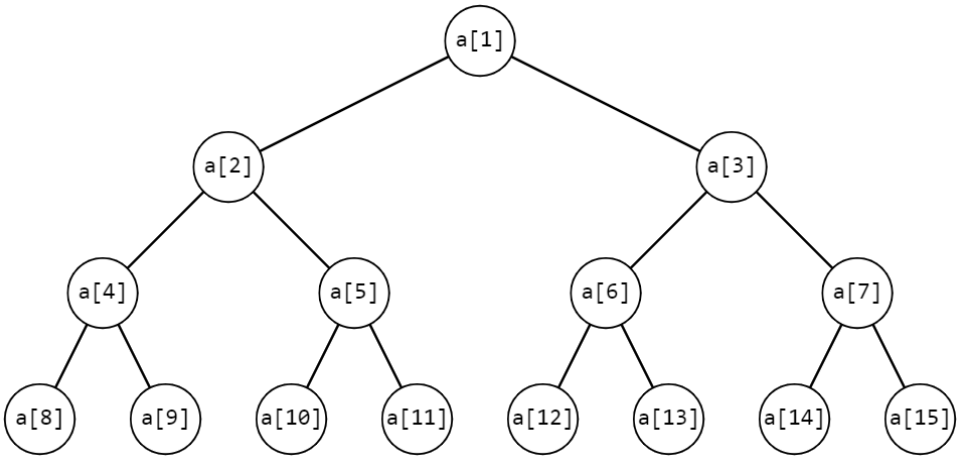

Рисунок 2.4 – Массив, представленный в виде двоичного дерева

Таблица 2.4 – Построение пирамиды

44	55	12	42)	94	18	06	67
44	55	12)	42	94	18	06	67
44	55)	06	42	94	18	12	67
44)	42	06	55	94	18	12	67
06	42	12	55	94	18	44	67

Каждый раз будем брать последнюю компоненту пирамиды (скажем x), прятать верхний элемент пирамиды в освободившемся теперь месте, а x сдвигать в нужное место. В таблице 2.5 приведены необходимые в этом случае $n-1$ шагов.

Таблица 2.5 – Пример процесса сортировки с помощью *Heapsort*

06	42	12	55	94	18	44	67
12	42	18	55	94	67	44	06
18	42	44	55	94	67	12	06
42	55	44	67	94	18	12	06
44	55	94	67	42	18	12	06
55	67	94	44	42	18	12	06
67	94	55	44	42	18	12	06

94	67	55	44	42	18	12	06

Пример из таблицы 2.5 показывает, что получающийся порядок фактически является обратным. Однако это можно легко исправить, и мы получаем программу, представленную ниже.

ПРОГРАММА. ПИРАМИДАЛЬНАЯ СОРТИРОВКА

```
PROGRAM HS;
VAR I,X,L,N,R:INTEGER;
    A:ARRAY[0..50] OF INTEGER;

PROCEDURE SIFT(L,R: INTEGER);
VAR
 I,J,X: INTEGER;
BEGIN
 I:=L;
 J:=2*L;
 X:=A[L];
 IF (J<R)AND(A[J]<A[J+1]) THEN INC(J);
 WHILE (J<=R)AND(X<A[J]) DO BEGIN
  A[I]:=A[J];
  A[J]:=X;
  I:=J;
  J:=2*J;
  IF (J<R)AND(A[J]<A[J+1]) THEN INC(J);
 END
END;

BEGIN
 WRITELN('Введи длину  массива');
 READ(N);
 WRITELN('Введи  массив');
 FOR I:=1 TO N DO READ(A[I]);
 L:=(N DIV 2)+1;
 R:=N;
 WHILE L>1 DO BEGIN
  DEC(L);
  SIFT(L,N)
 END;
 WHILE R>1 DO BEGIN
  X:=A[1];
  A[1]:=A[R];
  A[R]:=X;
  DEC(R);
  SIFT(1,R)
 END;
 WRITELN('Результат:');
 FOR I:=1 TO N DO WRITE(A[I],' ')
END.
```

Сортировка с помощью разделения

Этот улучшенный метод сортировки основан на обмене. Это самый лучший из всех известных на данный момент методов сортировки массивов. Его производительность столь впечатляюща, что изобретатель Ч. Хоар назвал этот метод *быстрой сортировкой* (*Quicksort*). В *Quicksort* исходят из того соображения, что для достижения наилучшей эффективности сначала лучше производить перестановки на большие расстояния. Предположим, что у нас есть *n* элементов, расположенных по ключам в обратном порядке. Их можно отсортировать

за *n*/2 обменов, сначала поменять местами самый левый с самым правым, а затем последовательно сдвигаться с двух сторон. Это возможно в том случае, когда мы знаем, что порядок действительно обратный. Однако полученный при этом алгоритм может оказаться и не удачным, что, например, происходит в случае *n* идентичных ключей: для разделения нужно *n*/2 обменов. Этих необязательных обменов можно избежать, если операторы просмотра заменить на такие:

```
WHILE  a[i] <= x  DO  i := i + 1  END;
WHILE  x <=  a[i] DO  j := j - 1  END
```

В этом случае *x* не работает как барьер для двух просмотров. В результате просмотры массива со всеми идентичными ключами приведут к переходу через границы массива.

Наша цель – не только провести разделение на части исходного массива элементов, но и отсортировать его. Будем применять процесс разделения к получившимся двум частям, до тех пор, пока каждая из частей не будет состоять из одного-единственного элемента. Эти действия описываются в следующей программе.

ПРОГРАММА. БЫСТРАЯ СОРТИРОВКА

```
PROGRAM QS;
VAR N,I:INTEGER;
    A:ARRAY[0..50] OF INTEGER;

PROCEDURE SORT(L,R: INTEGER);
VAR
 I,J,X,W: INTEGER;
BEGIN
 I:=L;
 J:=R;
 X:=A[(L+R) DIV 2];
 REPEAT
  WHILE A[I]<X DO INC(I);
  WHILE X<A[J] DO DEC(J);
  IF I<=J THEN BEGIN
   W:=A[I];
   A[I]:=A[J];
   A[J]:=W;
   INC(I);
   DEC(J)
  END
 UNTIL I>J;
 IF L<J THEN SORT(L,J);
 IF I<R THEN SORT(I,R);
END;

BEGIN
 WRITELN('Введи  длину  массива');
 READ(N);
 WRITELN('Введи  массив');
 FOR I:=1 TO N DO READ(A[I]);
 SORT(1, N);
 WRITELN('Результат:');
 FOR I:=1 TO N DO WRITE(A[I],' ')
END.
```

Требования к результатам выполнения работы:

– следует сравнить приведенные выше методы с точки зрения теории эффективности;

– реализовать приведенные выше методы на любом языки программирования;

– провести серию экспериментов по измерению времени, затрачиваемому на сортировку с использованием того или иного метода;

– сравнить полученные результаты с теоретическими выкладками.

ЛАБОРАТОРНАЯ РАБОТА 2. Разработка описания и анализ информационной системы

Цель работы: Описать и проанализировать информационную систему, распределить роли в группе разработчиков.

Составить и проанализировать требования к информационной системе, оформить техническое задание на разработку программного обеспечения.

3) Изучить методологии функционального моделирования IDEF0 и IDEF3.

Лабораторная работа направлена на ознакомление с процессом разработки требований к информационной системе и составления технического задания на разработку программного обеспечения, получение навыков по использованию основных методов формирования и анализа требований;ознакомление с процессом описания информационной системы и получение навыков по использованию основных методов анализа ИС;ознакомление с методологиями функционального моделирования IDEF0 и IDEF3, получение навыков по применению данных методологий для построения функциональных моделей на основании требований к информационной системе.

Требования к результатам выполнения работы:

– наличие описания информационной системы;

– наличие заключения о возможности реализации проекта, содержащего рекомендации относительно разработки системы, базовые предложения по объёму требуемого бюджета, числу разработчиков, времени и требуемому программному обеспечению;

– наличие диаграммы идентификации точек зрения и диаграммы иерархии точек зрения;

– наличие сценариев событий (последовательности действий);

– наличие пользовательских требований, четко описывающих будущий функционал системы;

– наличие системных требований, включающих требования к структуре, программному интерфейсу, технологиям разработки, общие требования к системе (надежность, масштабируемость, распределённость, модульность, безопасность, открытость, удобство пользования и т.д.);

– наличие составленного технического задания;

– модель должна отражать весь указанный в описании функционал, а также чётко отражать существующие потоки данных и описывать правила их движения;

– наличие в модели не менее трёх уровней;

– не менее двух уровней декомпозиции в стандарте IDEF0 (контекстная диаграмма + диаграммы A0);

– на диаграмме 1-го уровня (A0) не менее 4-х функциональных блоков;

– на диаграмме 2-го и далее уровнях должна быть декомпозиция в стандарте IDEF3, на каждой диаграмме не менее 2-х функциональных блоков.

3 Указания к самостоятельной работе студентов (СРС)

Виды самостоятельной работы:

1. Современные технологии программирования. Подготовка к тестированию.
2. Современные технологии программирования. Подготовка к устному опросу/собеседованию.
3. Современные технологии программирования. Выполнение практического задания.
4. Методы разработки эффективных алгоритмов. Подготовка к тестированию.
5. Методы разработки эффективных алгоритмов. Подготовка к лабораторной работе, написание отчета.
6. Проектирование информационных систем. Подготовка к тестированию.
7. Проектирование информационных систем. Подготовка к устному опросу/собеседованию.
8. Проектирование информационных систем. Подготовка к лабораторной работе, написание отчета.
9. Проектирование информационных систем. Выполнение практического задания.

Примеры заданий для самостоятельной работы

Задание 3.1Методология объектно-ориентированного моделирования

Цель работы: Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

Практическая работа направлена на ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML, получение навыков по применению данных элементов для построения объектно-ориентированных моделей ИС на основании требований.

Требования к результатам выполнения практической работы:

– модель системы должна содержать: диаграмму вариантов использования; диаграммы взаимодействия для каждого варианта использования; диаграмму классов, позволяющая реализовать весь описанный функционал ИС; объединенную диаграмму компонентов и размещения;

– для классов указать стереотипы;

– в зависимости от варианта задания диаграмма размещения должна показывать расположение компонентов в распределенном приложении или связи между встроенным процессором и устройствами.

Задание 3.2 Методология управление проектами

Цель работы: Изучение методологии управления проектами. Получение навыков по применению данных методологий для планирования проекта.

Работа направлена на ознакомление с основными понятиями методологии управления проектами, получение навыков по применению данных понятий при построении плана проекта, построения графика работ, распределения исполнителей, управления рисками.

Требования к результатам:

– построить модель управления проектом. Модель включает:
  - определение всех этапов проекта, зависимых этапов, определение длительности этапов;
  - построение на основе полученных данных сетевой и временной диаграмм;
  - построение диаграммы распределения работников по этапам;

– при определении этапа указывается его название, отражающее суть этапа (например, определение пользовательских требований, проектирование интерфейса и т.д.);

– этапов должно быть не менее 7, срок реализации проекта – 6 месяцев с 1.06.2007 по 31.12.2007;

– в проекте задействовано 6 человек персонала (фамилии необходимо придумать), некоторые из них участвуют на нескольких этапах проекта.