

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Севастопольский государственный университет»**

**Методические указания
к курсовому проектированию по дисциплине
«Информатика и программирование»
для студентов дневной и заочной форм обучения направлений
09.03.02 – «Информационные системы и технологии»
и 09.03.03 – «Прикладная информатика»**

Севастополь

2019

УДК 004.42 (075.8)

Методические указания к курсовому проектированию по дисциплине «Информатика и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика»; сост. В. Н. Бондарев, Т. И. Сметанина. — Севастополь: Изд-во СевГУ, 2018. — 72с.

Цель указаний: оказать методическую помощь студентам дневной и заочной формы обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика» в выполнении курсового проекта по дисциплине «Информатика и программирование»

Методические указания составлены в соответствии с требованиями программы дисциплины «Информатика и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика» и утверждены на заседании кафедры «Информационные системы» протоколом № от 2018 г.

Допущено научно-методической комиссией института информационных технологий и систем управления в технических системах в качестве методических указаний.

Рецензент: Кожяев Е.А., к.т.н., доцент кафедры информационных технологий и компьютерных систем

СОДЕРЖАНИЕ

1. Цель и задачи курсового проектирования.....	4
2. Тематика курсового проектирования и общие требования к программе.....	5
3. Выбор варианта задания и постановка задачи на проектирование.....	6
4. Порядок выполнения курсового проекта.....	7
5. Краткая характеристика основных этапов разработки программы.....	8
5.1. Постановка задачи.....	8
5.2. Выбор модели и метода решения задачи.....	9
5.3. Разработка структур данных.....	9
5.4. Нисходящее проектирование и модульное программирование.....	10
5.5. Структурное программирование (кодирование).....	11
5.6. Тестирование и отладка программы.....	12
5.7. Разработка программных документов	12
5.7.1. Единая система программной документации (ЕСПД).....	12
5.7.2. Требования к содержанию пояснительной записки.....	13
5.7.3. Требования к тексту программы и комментариям.....	15
6. Пример нисходящей разработки программы.....	16
6.1. Разработка внутренних структур данных.....	17
6.2. Разработка модульной структуры программы.....	18
6.3. Пошаговое кодирование и отладка программы.....	22
6.3.1. Кодирование и отладка функции чтения из файла.....	23
6.3.2. Кодирование и отладка функции инициализации меню.....	26
6.3.3. Кодирование и отладка выбора пунктов меню.....	28
6.3.4. Кодирование и отладка функции вывода базы на экран.....	30
6.3.5. Кодирование и отладка пунктов меню: добавить, удалить, исправить, найти.....	32
6.3.6. Кодирование и отладка функции выборки из базы.....	35
7. Правила оформления пояснительной записки.....	37
7.1. Общие требования.....	37
7.2. Структура пояснительной записки.....	37
7.3. Оформление отдельных частей текста.....	38
7.4. Правила оформления схем алгоритмов.....	41
Библиографический список.....	48
Приложение А. Варианты заданий к курсовому проекту.....	49
Приложение Б. Перечень стандартов, входящих в ЕСПД.....	53
Приложение В. Оформление титульного листа.....	54
Приложение Г. Бланк задания на курсовое проектирование.....	55
Приложение Д. Консольные функции	57
Приложение Е. Текст программы.....	58
Приложение Ж. Структура текста программного документа	68
Приложение З. Библиографическое описание источников информации.....	69

1. ЦЕЛЬ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ

Целью курсового проектирования является систематизация, закрепление и углубление знаний в области основ программирования и совершенствование практических навыков разработки программ на языках Паскаль или Си с использованием *методологии структурного программирования*. Для достижения цели на разных этапах курсового проектирования должны быть решены следующие **задачи**:

- выбор варианта задания и детализация поставки задачи;
- определение требований к функциям, выполняемым разрабатываемой программой;
- выбор типов и проектирование структур данных, определяющих способы представления, хранения и преобразования входных, выходных и промежуточных данных;
- разработка модульной структуры программы, определение функций модулей и способов их взаимодействия;
- написание текста (кодирование) программных модулей на алгоритмическом языке;
- разработка тестовых примеров;
- тестирование и отладка программы;
- разработка программных документов в соответствии с действующими стандартами.

Выполнение курсового проекта осуществляется по индивидуальному заданию, в соответствии с которым необходимо разработать программу для хранения и обработки информации, представленной в форме таблицы. Подобные программы являются важной составной частью различных автоматизированных информационных систем. Приобретение студентами практического опыта в разработке таких программ обеспечит хороший фундамент для освоения последующих дисциплин, связанных с разработкой информационных систем.

В рамках настоящего курсового проектирования при разработке программы используется методология структурного программирования, которая базируется на методах нисходящего проектирования и модульного программирования, структурного кодирования программ [4, 8]. Соблюдение этой методологии позволяет сократить время разработки программ и повысить их качество.

При выполнении курсового проекта необходимо учитывать, что разработка программных систем — это итеративный процесс. При этом основные фазы создания программы — проектирование, программирование и тестирование — нельзя строго разделить и зачастую они выполняются параллельно либо со значительным перекрытием во времени.

2. ТЕМАТИКА КУРСОВОГО ПРОЕКТИРОВАНИЯ И ОБЩИЕ ТРЕБОВАНИЯ К ПРОГРАММЕ

Темой курсового проекта является разработка программы для *обработки данных, представленных в форме таблицы*. Любая таблица отражает какую-либо форму человеческой деятельности. С этой точки зрения программа, написанная в результате курсового проектирования, может считаться упрощенным подобием *предметно-ориентированной базы данных*.

Широко распространенным видом услуг, которые эффективно реализуются с помощью компьютеров, является *информационно-справочное обслуживание*, которое подразумевает хранение сведений, прием новых сведений, обработку сведений, выдачу информации по *запросам*. Хранимые сведения в общем случае представляются записями. Для представления такого рода услуг создаются *автоматизированные информационные системы* (АИС) различного назначения. Основная задача, решаемая в ходе создания АИС, состоит в том, чтобы *организовать совместное хранение большого числа различных записей и выдавать по запросу любую из них независимо от того, какие записи и в каком порядке выдавались ранее*.

Наиболее типичной операцией в АИС является поиск и выдача затребованной записи. Для пользователя такой системы было бы обременительно знать и в каждом запросе указывать место хранения нужной записи — тем более, если в системе хранится очень большое количество записей.

Избежать этих неудобств позволяет структура данных, называемая *таблицей*. В таблице каждой записи соответствует определенное имя. При этом в запросе на выдачу нужной записи достаточно указать только ее имя, а реализация такой структуры данных должна сама обеспечивать достаточно быстрый поиск записи с указанным именем.

Таким образом, *таблица — это некоторый, вообще говоря, неупорядоченный набор именованных записей*. Имя записи в таблице называют *ключом*. Ключ должен быть таким, чтобы имела возможность сравнения любых двух записей. Поэтому в качестве ключей чаще всего используют целые числа и строки.

Над таблицей как структурой данных определены следующие операции:

- *поиск* записи с заданным ключом в таблице;
- *включение* в таблицу записи с заданным ключом (обычно считается, что если в таблице уже есть запись с таким ключом, то это означает замену старой записи на новую);
- *исключение* из таблицы записи с заданным ключом.

Помимо указанных операций должна быть реализована возможность *визуального поиска*. Для осуществления визуального поиска необходимо упорядочить записи (отсортировать). Сортировка может быть осуществлена по ключевому полю записи или по какому-либо другому полю в зависимости от вида визуального поиска. Так, например, список группы студентов сортируют по фамилиям, календарный план — по датам, физические эксперименты — по номеру и т. п.

В программе должно быть предусмотрено манипулирование данными — выполнение каких-либо действий над ними. Например, может потребоваться начис-

лить зарплату рабочему в зависимости от разряда, подсчитать количество рабочих первого разряда и т. д.

Для сохранения данных на внешнем носителе необходимо организовать работу с файлами. Разрабатываемая программа должна работать как с текстовыми, так и с типизированными файлами. Загрузка начальных данных из типизированного файла должна происходить автоматически при запуске программы. При завершении программы измененные данные записываются в типизированный файл. Кроме того, должна быть возможность записи и чтения текстовых файлов.

Следует помнить, что основная цель разработки — создание программного продукта, который позволит пользователю решать нужные ему задачи (в рамках заранее оговоренной предметной области). Поэтому немаловажным является обеспечение *дружественного интерфейса пользователя*.

3. ВЫБОР ВАРИАНТА ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧИ НА ПРОЕКТИРОВАНИЕ

Перечень вариантов заданий на курсовое проектирование приведен в приложении А. Вариант задания определяет структуру входной таблицы и производимые над ней действия.

В качестве примера приведем следующий вариант задания на курсовое проектирование. Даны записи о проданных товарах в виде:

- наименование товара;
- артикул;
- количество проданного товара;
- цена за единицу товара.

По каждой записи необходимо получить сумму выручки за данный вид товара. Необходимо обеспечить вывод на печать таблицы с исходными данными и таблицы с результатами расчетов. В конце таблицы с результатами расчетов следует вывести сумму, полученную от реализации всех товаров.

Студенты заочного отделения вариант задания определяют самостоятельно. Для этого вычисляется остаток от деления числа, образованного из двух последних цифр зачетной книжки, на 30. Например, две последние цифры в индивидуальном плане — 77. Тогда остаток от деления 77 на 30 равен 17, следовательно, номер варианта 17. **Студентам дневного отделения вариант задания назначается преподавателем** по порядковому номеру студента в журнале или по последним двум цифрам номера зачетной книжки.

Допускается выполнять курсовой проект по индивидуальным заданиям, связанным с разработкой научно-исследовательских тем, выполняемых кафедрами университета. **После выбора студентом варианта задания необходимо заполнить бланк технического задания** и подписать его у преподавателя, ведущего занятия. После этого задание на курсовое проектирование считается утвержденным.

В постановку задачи на проектирование для любого варианта задания входит выполнение следующих обязательных требований.

Разрабатываемая программа должна использовать **меню-ориентированный интерфейс**, обеспечивающий выполнение следующего минимального состава действий.

1. **Начальное создание таблицы.** При необходимости создания новой таблицы исходные данные считываются из текстового файла. Имя файла должен задавать пользователь.

2. **Просмотр таблицы.** При этом необходимо предусмотреть возможность скроллинга.

3. **Добавление новой записи в таблицу.**

4. **Удаление записи.** Удаляемый элемент выбирается по одному из полей таблицы (ключевому). Ключевое поле выбирает студент.

5. **Корректировка записи в таблице.** Корректируемую запись выбирают по одному из полей таблицы (ключевому).

6. **Сортировка таблицы.** Сортировка производится по одному из полей таблицы (ключевому). Метод сортировки выбирает студент.

7. **Поиск записи в таблице** по ключевому полю.

8. **Сохранение таблицы в текстовом файле.** Имя файла должен вводить пользователь. Сохранение таблицы в текстовом файле обеспечит при необходимости возможность её печати.

9. **Обработка таблицы и просмотр результатов обработки.** Результат обработки необходимо вывести на экран и в текстовый файл. Имя файла вводит пользователь.

10. **Выход** — завершение работы программы.

В процессе работы с одной и той же таблицей **она должна автоматически сохраняться в типизированном файле** и её загрузка из типизированного файла должна происходить автоматически при новом запуске программы.

По результатам разработки программы должна быть разработана **пояснительная записка**, объемом 25...35 страниц без учета приложений, а также разработаны схемы алгоритмов на листе формата А1. На лист выносится схема главной программы, а также схема алгоритма одной из основных функций или процедур.

4. ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОГО ПРОЕКТА

Курсовой проект по дисциплине АиП выполняется в течение семестра. Защита курсового проекта происходит на 15-16 неделе семестра по утвержденному графику. Чтобы получить допуск к защите, необходимо предъявить следующие документы и программные продукты:

- программу (на электронном носителе);
- пояснительную записку (ПЗ);
- презентацию.

Общий порядок работы над курсовым проектом определяется календарным планом, приведенным в бланке задания на курсовое проектирование. План учитывает особенности методологии структурного программирования и предусматривает выполнение следующих этапов разработки программы:

- постановка задачи (1...2 неделя семестра);
- выбор модели и метода решения задачи (1...2 неделя семестра);

- разработка структур данных (2...4 неделя семестра);
- нисходящее проектирование и модульное программирование (5...8 неделя семестра);
- структурное программирование (кодирование) (6...13 неделя семестра);
- нисходящее тестирование и отладка программы (6...13 неделя семестра);
- разработка программных документов (оформление ПЗ и графических материалов, 11...15 неделя семестра).

Сроки выполнения этапов указаны в скобках. Ввиду итеративности любого процесса проектирования, а также особенностей структурной разработки программ (технология нисходящего проектирования) отдельные этапы курсового проектирования совмещаются во времени и выполняются параллельно. Краткая характеристика содержания каждого из этапов изложена в следующем разделе.

Во время защиты проекта студент должен сделать доклад, продемонстрировать работу программы на компьютере, ответить на вопросы, относящиеся как к работе программы, так и к основным теоретическим положениям дисциплины АиП.

Курсовой проект оценивается по пятибалльной и рейтинговой шкалам. Критериями, влияющими на результирующую оценку, являются:

- полнота реализации требований к программе;
- удобство программного интерфейса;
- стиль написания программного кода;
- тщательность тестирования программы;
- качество оформления пояснительной записки и графических материалов;
- полнота и правильность ответов на вопросы во время защиты проекта;
- соблюдение календарного плана выполнения курсового проектирования.

5. КРАТКАЯ ХАРАКТЕРИСТИКА ОСНОВНЫХ ЭТАПОВ РАЗРАБОТКИ ПРОГРАММЫ

5.1. Постановка задачи

Создание любой программы начинается с постановки задачи. Описание общих требований к программе и постановка задачи на курсовое проектирование изложены в разделах 2 и 3. На этом этапе также определяется среда, в которой будет выполняться программа: требования к аппаратуре, используемая операционная система и другое программное обеспечение.

В общем случае постановка задачи завершается разработкой технического задания и внешней спецификации программы. В нашем случае достаточно ограничиться только спецификацией программы, включающей:

- описание исходных данных и результатов (типы, форматы, точность, способ передачи, ограничения);
- описание функций, реализуемых программой;
- способ обращения к программе;
- описание возможных аварийных ситуаций и ошибок пользователя.

Таким образом, на этом этапе программа рассматривается как «черный ящик», для которого определены основные функциональные требования, входные и выходные данные.

5.2. Выбор модели и метода решения задачи

На этом этапе анализируют условия задачи, строят математическую модель задачи и определяют общий метод ее решения. При наличии нескольких методов выбирают наилучший, исходя из критериев сложности, эффективности, точности и т. д.

В нашем случае на этом этапе следует составить и описать основные расчетные формулы, используемые при обработке записей в таблице.

5.3. Разработка структур данных

Так как содержание алгоритмов определяется тем, как организованы данные, то начинать проектирование программы надо не с алгоритмов, а с разработки структур, необходимых для представления входных, выходных и промежуточных данных в памяти ЭВМ. При этом принимаются во внимание многие факторы, например: ограничения на размер данных, необходимая точность, требования к быстродействию программы.

При решении вопроса о том, как будут организованы данные в программе, полезно задать себе следующие вопросы [9]:

1. Какая точность представления данных необходима?
2. В каком диапазоне лежат значения данных?
3. Ограничено ли максимальное количество данных?
4. Обязательно ли хранить их в программе одновременно?
5. Какие действия потребуется выполнять над данными?

Например, если максимальное количество однотипных данных, которые требуется обработать, известно и невелико, проще всего хранить их в статическом массиве. Если элементов в массиве много, то объема программного стека может не хватить и придется отвести под эти массивы место в динамической памяти.

Если максимальное количество элементов данных неизвестно и постоянно изменяется во время работы программы, то для их хранения используют динамические структуры. Выбор вида структуры зависит от требуемых операций над данными. Например, для быстрого поиска элементов лучше подходит бинарное дерево, а если данные требуется обрабатывать в порядке поступления, то применяется очередь [9 – 12].

В нашем случае будем полагать, что объем обрабатываемой таблицы позволяет поместить её целиком в основную (оперативную) память. Такое допущение существенно снижает произвол при выборе структур данных.

На этом этапе курсового проектирования следует выполнить обзор и обоснованный выбор структур данных, предназначенных для внутреннего представления табличной информации [2, 4 – 8]:

- статических и динамических массивов;
- различных видов списков;
- бинарных деревьев.

Кроме этого, следует привести полное описание всех остальных типов входных, выходных и промежуточных данных, используемых в будущей программе.

Таким образом, основная задача этого этапа — выбор типов и проектирование структур данных, определяющих способы представления, хранения и преобразования входных, выходных и промежуточных данных.

5.4. Нисходящее проектирование и модульное программирование

Под проектированием программы понимается определение её общей структуры и взаимодействия модулей. На этом этапе применяется **технология нисходящего проектирования**, основная идея которого проста: разбиение (декомпозиция) задачи на подзадачи меньшей сложности, пригодные для рассмотрения по отдельности. Таким образом, программа структурируется по уровням иерархии и разрабатывается путем последовательного уточнения деталей на каждом уровне иерархии [4, 8, 13].

Очень важным на этом этапе является **спецификация интерфейсов**, то есть определение способов взаимодействия подзадач. Спецификация интерфейсов заключается в формализованном описании входов и выходов функций, которые должны реализовываться отдельными частями программы (модулями). Поэтому кроме принципа иерархичности, на этом этапе также используют принцип **модульного программирования**, который заключается в разделении программы на функционально самостоятельные части (модули), обеспечивающие заменяемость, модификацию, удаление и дополнение составных частей (модулей) программы. Независимость модулей является основным принципом модульного программирования. Это означает, что программа должна быть поделена на модули таким образом, чтобы обеспечивалась минимизация их взаимодействия. Модули считаются независимыми, если они удовлетворяют таким требованиям:

- каждый модуль можно заменить другим функционально-эквивалентным модулем с таким же интерфейсом;
- взаимосвязи между модулями установлены в соответствии с иерархическим принципом;
- модуль имеет одну точку входа и одну точку выхода;
- доступ к внутренним структурам данных модуля выполняется через его интерфейс;
- модуль возвращает управление тому программному модулю, который его вызвал.

Таким образом, основная задача этого этапа — разработка модульной структуры программы, определение функций модулей и способов их взаимодействия.

Преимущества модульного программирования состоят в следующем.

Упрощается отладка программ, так как ограниченный доступ к модулю и однозначность его внешнего поведения исключают влияние ошибок в других, связанных с ним, модулях на его функционирование. Это дает возможность разрабатывать программу **методом сверху вниз (нисходящее программирование)**, постепенно присоединяя написанные модули к ранее отлаженным. После каждого

такого присоединения неверная работа программы сигнализирует о присутствии ошибки в новом модуле, а не в уже отлаженных.

Повышается надежность программы, так как относительно малый размер модулей и, как следствие, небольшая их сложность позволяют провести более полную проверку программы.

Обеспечивается возможность организации совместной работы больших коллективов разработчиков, так как каждый программист имеет дело с независимой от других частью программы.

5.5. Структурное программирование (кодирование)

Если методы нисходящего проектирования и модульного программирования регламентируют процесс построения программы на макроуровне, то методы структурного программирования работают на микроуровне, регламентируя процесс создания программного кода [9].

Основной задачей этого этапа является написание текста (кодирование) программных модулей на алгоритмическом языке с учетом правил структурного программирования.

Структурное программирование, позволяет путем соблюдения определенных правил уменьшить время разработки и облегчить возможность модификации программы. Программы, разработанные по правилам структурного программирования, обладают следующими свойствами:

- имеют модульную структуру, размер модуля ограничен (обычно не превышает 100 операторов языка высокого уровня);
- модули реализуются с использованием ***ограниченного числа допустимых управляющих конструкций*** языков программирования — последовательность, разветвление, выбор, цикл;
- модули содержат операторы безусловного перехода (go to) только в исключительных случаях.

Кодирование также организуется по принципу «сверху вниз»: вначале кодируются модули самого верхнего уровня и составляются тестовые примеры для их отладки. При этом на месте еще не написанных модулей следующего уровня ставятся так называемые ***заглушки*** — временные программы. Заглушки в простейшем случае просто выдают сообщение о том, что им передано управление, а затем возвращают его вызывающему модулю. В других случаях заглушка может выдавать значения, заданные заранее или вычисленные по упрощенному алгоритму.

Таким образом, сначала создается логический «скелет» программы, который затем обрывается необходимым кодом. Такую стратегию называют ***нисходящим программированием***. В этом случае исходные тексты модулей верхних уровней появляются при неполной проработке структуры программы на нижних уровнях иерархии. Нисходящее программирование может быть совмещено с выполнением отладки.

5.6. Тестирование и отладка программы

И проектирование, и программирование обязательно должны сопровождаться написанием тестов — проверочных исходных данных и соответствующих им наборов эталонных реакций.

Необходимо различать процессы тестирования и отладки программы. **Тестирование** — это процесс, посредством которого проверяется правильность программы. Тестирование носит позитивный характер, его цель — показать, что программа работает правильно и удовлетворяет всем проектным спецификациям. **Отладка** — процесс поиска и исправления *причин ошибок* в программе. Исправляют ошибки, обнаруженные при тестировании. При этом следует учитывать, что процесс обнаружения ошибок подчиняется закону насыщения, то есть большинство ошибок обнаруживается на ранних стадиях тестирования, и чем меньше в программе осталось ошибок, тем дольше искать каждую из них [9].

Идея нисходящего тестирования и отладки предполагает, что к тестированию программы приступают еще до того, как завершена ее разработка. Это позволяет раньше опробовать основные межмодульные интерфейсы, а также убедиться в том, что программа в основном удовлетворяет требованиям пользователя. Только после того, как логическое ядро испытано настолько, что появляется уверенность в правильности реализации основных интерфейсов, приступают к кодированию и тестированию следующего уровня программы.

Естественно, полная проверка программы, пока она представлена в упрощенной форме, невозможна, однако добавление каждого следующего уровня позволяет постепенно расширять область тестирования.

В разделе 6 приведен пример нисходящей разработки программы.

5.7. Разработка программных документов

5.7.1. Единая система программной документации (ЕСПД)

Разработка документации на программу — ответственный этап проектирования. От качества выполнения документации в значительной степени зависит не только эффективность использования программы, но и пригодность её к дальнейшему развитию и сопровождению.

Разработка программных документов выполняется в соответствии с *Единой системой программной документации (ЕСПД)*, которая представляет собой комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации. Перечень стандартов, входящих в ЕСПД, приведен в приложении Б. В соответствии с ЕСПД на разных стадиях проектирования программы составляются следующие *программные документы*: спецификация, техническое задание, пояснительная записка, текст программы, описание программы, программа и методика испытаний, различные эксплуатационные документы (формуляр, описание применения, руководство оператора, руководство программиста и др.). Общие требования к программным документам изложены в ГОСТ 2.105-95. *При выполнении курсового проекта разрабатывается только пояснительная записка.*

5.7.2. Требования к содержанию пояснительной записки

В рамках настоящего *учебного* проекта совмещены несколько стадий проектирования (эскизное, техническое и рабочее проектирование). Поэтому разрабатываемая *пояснительная записка* к курсовому проекту будет объединять в себе несколько документов, требования к которым определяются ГОСТ 2.105-95 (общие требования к текстовым документам). ПЗ к курсовому проекту должна содержать следующие структурные элементы и разделы:

- титульный лист;
- техническое задание (на специальном бланке);
- аннотация;
- содержание;
- перечень сокращений и условных обозначений (при необходимости);
- введение;
- назначение и область применения программы;
- технические характеристики программы;
- выполнение программы;
- выводы;
- перечень ссылок;
- приложения (**обязательное приложение — текст программы**).

Титульный лист пояснительной записки оформляют в соответствии с приложением В.

Техническое задание — документ на специальном бланке, в соответствии с которым выполняется курсовое проектирование. Образец заполнения бланка задания представлен в приложении Г. Бланк задания заполняется в начале проектирования и подписывается студентом и руководителем проекта.

В *аннотации* приводят сведения о назначении документа (в данном случае пояснительной записки) и краткое изложение его сути в виде 6...8 предложений.

В *содержании* указываются наименования разделов и подразделов пояснительной записки (кроме титульного листа и технического задания), а также номера соответствующих страниц.

Перечень сокращений и условных обозначений содержит все используемые в тексте пояснительной записки малораспространенные условные обозначения, символы, единицы, сокращения и термины. Независимо от этого *при первом появлении этих элементов в тексте необходимо привести их расшифровку*. Элементы перечня должны быть упорядочены по алфавиту.

Во *введении* следует указать наименование программы, а также документы, на основании которых ведется разработка с указанием организации и даты утверждения. В рамках настоящего курсового проектирования документ — это техническое задание, организация — Севастопольский государственный университет. Наименование программы и дата утверждения, указанные в разделе «Введение», должны соответствовать значениям, указанным на листе технического задания. В этом разделе обосновывается *актуальность* решаемой задачи, указываются *цель* и перечисляются *задачи*, которые необходимо решить для достижения постав-

ленной цели. Во введении также приводится краткое изложение последующих разделов пояснительной записки.

При формулировке цели следует указать наименование программы, важнейшие её характеристики и назначение.

Пример.

Целью данного курсового проекта является разработка программы расчета выручки за проданный товар, которая обеспечивает получение оперативных данных об эффективности работы торговых точек по продаже товаров бытового назначения.

Рекомендованный объем введения — 1 страница.

В разделе **«Назначение и область применения программы»** указывают назначение программы, краткую характеристику области применения программы (рекомендуемый объем — 0,5 страницы).

Раздел **«Технические характеристики программы»** должен содержать следующие подразделы:

- *постановка задачи на разработку программы* (предполагает формулировку варианта задания и детализацию требований к функциям, выполняемым программой);
- *применяемые математические методы* (описание расчетных формул и, при необходимости, описание допущений и ограничений, связанных с выбранным математическим аппаратом);
- *описание и обоснование выбора метода организации входных, выходных и промежуточных данных*;
- *разработка модульной структуры программы* с описанием процесса декомпозиции задачи и определения состава и функций модулей и способов их взаимодействия;
- *описание алгоритмов функционирования программы* с обоснованием выбора схемы алгоритма основной программы, а также алгоритмов всех модулей программы (процедур и функций) и их взаимодействия (описание выполняют с использованием схем алгоритмов и текста программы на исходном языке);
- *обоснование состава технических и программных средств*, необходимых для работы программы (обоснование выполняется на основании расчетов и анализа объемов памяти, требуемых для хранения входных и выходных данных на внешних носителях, а также объемов оперативной памяти, требуемых как для хранения внутренних структур данных, так и кода самой программы).

Рекомендуемый объем этого раздела — 15...18 страниц.

Раздел **«Выполнение программы»** должен содержать следующие подразделы:

- *условия выполнения программы* (указывается минимальный состав аппаратных и программных средств, необходимых для выполнения программы);
- *загрузка и запуск программы* (описывается содержимое носителя, на котором распространяется программа и указывается последовательность дей-

ствий оператора, обеспечивающих установку, загрузку, запуск и завершение программы);

- *проверка работоспособности программы* (описываются тестовые данные, порядок работы с программой, экранные формы и сообщения, подтверждающие выполнение программой всех предписанных ей функций).

Рекомендуемый объем раздела — 8...12 страниц.

В **выводах** указываются основные и смежные отрасли использования спроектированной программы, делаются выводы о соответствии её техническому заданию и о путях дальнейшего усовершенствования.

Выводы формулируют в прошедшем времени («получено», «достигнуто», «разработано» и т.п.) в обратном порядке по отношению к порядку следования задач во введении, т.е. от решенных задач к цели. При этом следует не только констатировать факт решения задачи (например, «разработана схема алгоритма»), но дать краткую характеристику результату (например, «в основу алгоритма положена структура данных в виде двунаправленного списка, позволяющая выполнять просмотр данных в двух направлениях»). В отдельном абзаце приводят фразу о факте достижения цели проекта, формулируют достигнутую цель и приводят рекомендации о дальнейшем использовании результатов проекта.

Пример.

Таким образом, цель курсового проекта достигнута. Разработанная программа расчета выручки за проданный товар может успешно применяться не только индивидуальными предпринимателями, но предприятиями малого и среднего бизнеса после соответствующей доработки.

Рекомендуемый объем выводов — 1 страница.

В разделе «**Перечень ссылок**» необходимо поместить перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов, на которые есть ссылки в основном тексте ПЗ.

В обязательном **приложении** приводится **текст программы с комментариями**. Текст программы содержит символическую запись программы на языке программирования с комментариями, отражающими структуру и назначение частей программы.

Помимо текста программы, в приложение могут быть включены и другие документы, использованные при разработке.

Поскольку оформление текста программы чрезвычайно важно для её эффективного сопровождения, рассмотрим требования к тексту программы и комментариям подробнее.

5.7.3. Требования к тексту программы и комментариям

Как отмечалось выше, документирование программы должно выполняться в процессе её разработки. Поэтому **составление комментариев** к программе в процессе её пошаговой разработки является важнейшим средством **оперативного документирования** программы. Обычно начинающие программисты пренебрегают этим. Вместе с тем хорошие комментарии существенно облегчают процесс сопровождения программы.

В идеале комментарии должны создаваться до написания программы — ими служит подробный алгоритм, изложенный на естественном языке [9]. Полезно до написания текста программы на алгоритмическом языке подробно указать, последовательность и содержание действий, которые должна выполнять программа. Это помогает в деталях продумать интерфейсы и алгоритмы работы модулей, избежать на самой ранней стадии серьезных ошибок. После такой работы написание текста программы сводится к вставке фрагментов кода между комментариями. Подходить к написанию программы нужно таким образом, чтобы ее можно было в любой момент передать другому программисту. Комментарии должны быть написаны простым языком и не подтверждать очевидное. Например, «объявление массива» или «вызов функции» и т.д.

Если комментарий занимает несколько строк, то его следует выровнять по вертикали и разместить до фрагмента соответствующего кода, а не справа от него. Отступ комментария должен соответствовать отступу комментируемого участка кода.

Для выделения подпрограмм или других логически законченных фрагментов кода рекомендуется пользоваться комментарием вида:

```
/*-----Подпрограмма 1-----*/
```

Текст программы после написания необходимо тщательно отредактировать: убрать ненужные участки кода, сгруппировать описания, отформатировать текст, оптимизировать проверки условий и циклы и т.д. Безусловно, текст программы должен быть написан с соблюдением правил структурного программирования.

6. ПРИМЕР НИСХОДЯЩЕЙ РАЗРАБОТКИ ПРОГРАММЫ

Рассмотрим следующую задачу. Пусть имеется база данных сотовых телефонов, которая хранится в текстовом файле. Каждая строка файла содержит запись о модели телефона, для которой указывается наименование телефона (20 позиций), вес телефона до сотых долей грамма, цена. Число записей в базе произвольно. Требуется написать программу выполняющую:

- а) чтение файла и отображение всей базы телефонов на экране;
- б) добавление записи в базу;
- в) корректировку записи;
- г) удаление записи из базы;
- д) поиск записи о модели телефона;
- е) выбор записей о моделях, попадающих в заданный пользователем диапазон цен.

Программа должна содержать меню и обеспечивать ввод-вывод в окна на экране. Необходимо предусмотреть контроль ошибок пользователя при вводе данных.

Так как требование обозримости учебной программы не позволяет учесть особенности рассматриваемой задачи во всех деталях, то напомним всего лишь **иллюстративную версию** программы на языке Си с элементами C++ [1, 3, 5]. Поэтому приведенный ниже пример не следует рассматривать как законченную разработку. Он лишь поясняет (конкретизирует) содержание указанных выше

этапов проектирования. Аналогичный пример программы, но на языке Паскаль можно найти в [8]. При разработке программы будем следовать основным этапам, указанным в разделе 5.

6.1. Разработка внутренних структур данных

Исходные данные хранятся в текстовом файле неопределенного размера, содержащем записи о телефонах. Каждая запись содержит название телефона (20 позиций), вес до сотых долей грамма, цену. Также к исходным данным относятся: название модели, минимальная и максимальная цены телефона, вводимые с клавиатуры при поиске записей в базе. В соответствии с описанием задачи для представления названия телефона будем использовать строку из 20 символов, а для представления веса телефона и цены будем использовать вещественные значения.

Поскольку в процессе работы программы придется вводить и просматривать базу на экране, возможно, в разных направлениях (с начала в конец или наоборот), и число записей неизвестно, то для её хранения в оперативной памяти компьютера будем использовать динамический двунаправленный список. Каждый элемент списка будет содержать информационную часть (модель телефона, вес, цена) и два указателя — на следующий и предыдущий элементы. Признаком конца списка служит константа NULL в поле указателя. Список доступен через указатели на его первый и последний элементы. Соответствующее описание элемента списка и указателей на языке Си представим в виде:

```
const int L_MODEL=20;           // длина названия телефона
struct tel {                    // структура для хранения данных о телефоне
    char model[L_MODEL];        //название телефона
    float weight, price;        //вес и цена телефона
};

struct elist{                  //элемент 2- направленного списка
    tel data;                   //данные о телефоне
    elist *left;               //указатель на элемент слева
    elist *right;              //указатель на элемент справа
};
elist *beg, *fin;              //указатели первого и последнего эл-та
```

К **результатам** работы программы относится та же база, выведенная в окно на экране, но содержащая только часть записей, удовлетворяющих критерию запроса. Выборка из базы также должна просматриваться на экране, поэтому для её хранения будем использовать отдельный двунаправленный список аналогичной структуры. Это позволит разделить процессы выборки записей и их отображения на экране и упростит программу.

Промежуточные величины, которые будут использоваться программой, будут связаны с реализацией интерфейса программы в виде меню. Для хранения пунктов меню программы будем использовать статический массив. При этом количество пунктов меню, длину пункта меню и позиции начала каждого пункта меню на экране буде задавать в виде констант. Это обеспечит при необходимости возможность быстрой модификации меню путем изменения значений констант. Названия пунктов меню будем представлять в виде строковых констант. В соот-

ветствии с условием задачи можно предложить следующие названия пунктов меню: "Просмотр", "Добавить", "Исправить", "Удалить", "Найти", "Выбрать", "Выход".

6.2. Разработка модульной структуры программы

На этом этапе требуется разработать модульную структуру программы. Положим в основу организации программы *принцип событийного управления*. В этом случае любое действие пользователя (нажатие клавиши, щелчок мышью), называемое событием, приводит к запрограммированной реакции программы. Структура программы, управляемой событиями, изображена на рис. 6.1. [8].

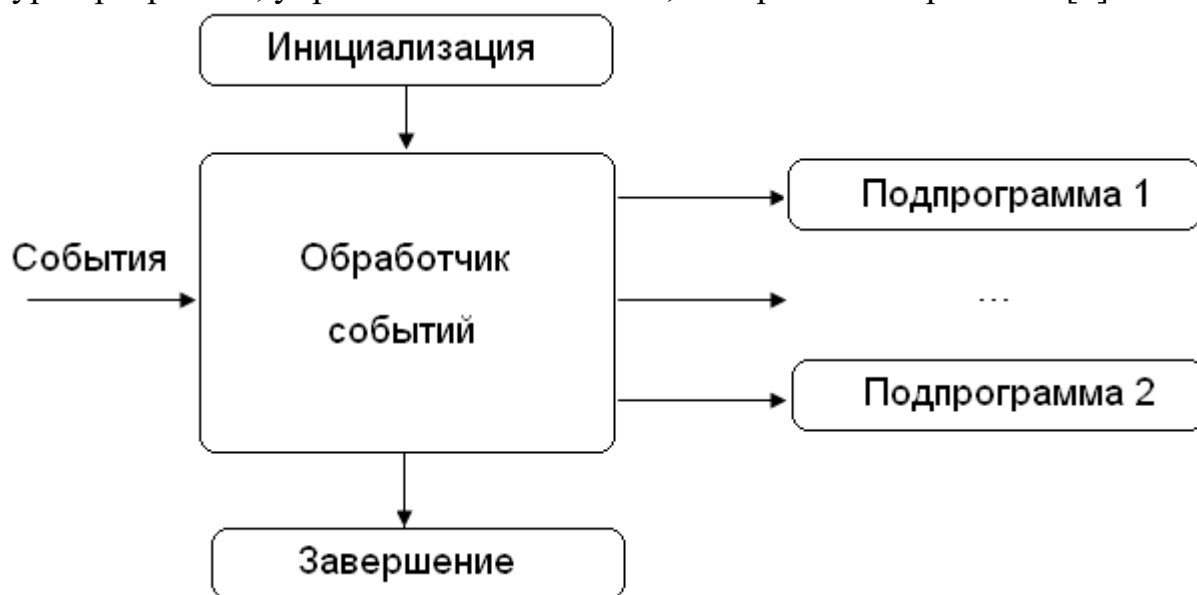


Рисунок 6.1 — Структура программы, управляемой событиями

Следуя технологии нисходящего проектирования, разработку модульной структуры программы начнем с самого верхнего уровня. Функциональная самостоятельность модуля достигается за счет рационального разбиения функций обработки данных на подфункции и оформления их в виде отдельных подпрограмм. В этом случае структура программы будет включать в себя программные модули, располагаемые на нескольких уровнях иерархии.

В соответствии с выбранным общим принципом построения программы (рис. 6.1) и постановкой задачи разрабатываемая программа должна обеспечивать выполнение следующих функций:

- инициализацию;
- вывод базы на экран;
- добавление записи о телефоне в базу;
- исправление записи о телефоне в базе;
- удаление записи о телефоне из базы;
- поиск записи о телефоне в базе;
- выбор записей о моделях телефонов, попадающих в заданный пользователем диапазон цен;
- выход из программы.

Поэтому определение функций программы на первом уровне иерархии никаких трудностей не вызывает.

Процесс дальнейшей декомпозиции функций первого уровня иерархии в функции второго и третьего уровней иерархии представлен в табл. 6.1. Например, инициализация состоит из двух независимых функций: считывания базы из файла и инициализации меню программы. Ввод базы связан с проверкой существования исходного текстового файла, считывания одной записи из файла и добавления её в двунаправленный список, который будет использоваться для хранения базы данных в оперативной памяти компьютера. Выбор из базы данных записей о моделях телефонов, попадающих в заданный пользователем диапазон цен, реализуется на втором уровне иерархии с помощью трех функций: ввод критерия отбора, формирование выборки из базы и вывод этой выборки. Поскольку дальнейшая детализация первых двух функций является нецелесообразной, то далее, на третьем уровне детализируется только вывод выборки, который реализуется посредством отображения на экране одной страницы базы данных и управления прокруткой для отображения других страниц. Аналогичным образом выполняется декомпозиция остальных функций первого уровня, приведенных в табл. 6.1.

После такой декомпозиции функций программы можно легко выделить минимальный состав модулей, которые будут оформляться в виде отдельных подпрограмм и которые совместно обеспечат реализацию всех функций программы. Схема модульной структуры программы изображена на рис. 6.2. Вызов модулей (подпрограмм) производится из главной программы в процессе её инициализации либо при наступлении определенного события — нажатия той или иной клавиши. Схема построена с учетом принципа иерархичности — модули нижних уровней иерархии не могут вызывать модули, относящиеся к верхним уровням иерархии.

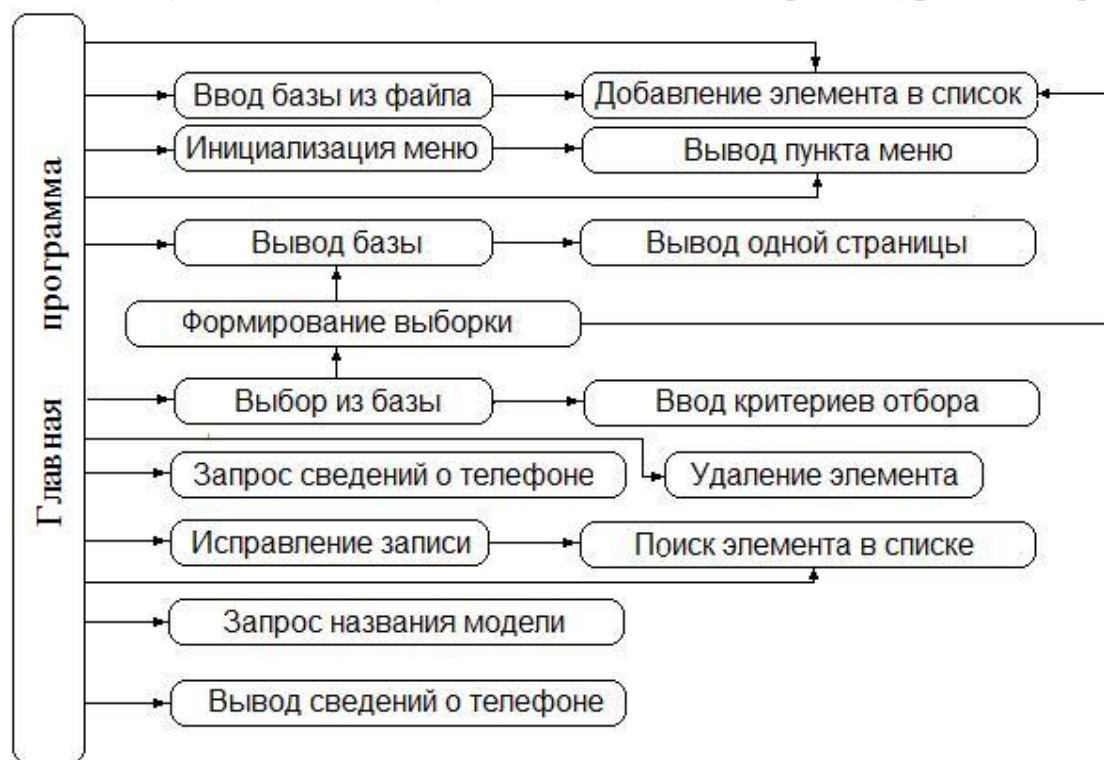


Рисунок 6.2 — Модульная структура программы

Таблица 6.1 — Декомпозиция функций программы на разных уровнях иерархии

Уровень 1	Уровень 2	Уровень 3
Инициализация	Ввод базы из файла ----- Инициализация меню	Проверка существования файла Считывание одной записи из файла Добавление элемента в список ----- Задание цветов и режимов Вывод пунктов меню
Вывод базы на экран	Вывод базы на экран	Вывод одной страницы Управление прокруткой
Добавление записи о телефоне	Запрос сведений о телефоне Добавление элемента список	Запрос сведений о телефоне Добавление элемента в список
Коррекция записи о телефоне	Запрос сведений о телефоне ----- Исправление записи	Запрос сведений о телефоне ----- Поиск элемента в списке Исправление элемента списка
Удаление записи о телефоне	Запрос названия модели телефона ----- Удаление из списка	Запрос названия модели телефона ----- Поиск элемента в списке Удаление элемента из списка
Поиск записи о телефоне в базе	Запрос названия модели телефона ----- Поиск элемента в списке ----- Вывод сведений о телефоне	Запрос названия модели телефона ----- Поиск элемента в списке ----- Вывод сведений о телефоне
Выбор из базы моделей телефонов, попадающих в заданный пользователем диапазон цен	Ввод критериев отбора ----- Формирование выборки ----- Вывод выборки (базы)	Ввод минимальной цены с проверкой ввода ----- Ввод максимальной цены с проверкой ввода ----- Формирование выборки ----- Вывод одной страницы Управление прокруткой
Выход	Выход	Выход

Эффективность модульной структуры зависит от того, насколько удастся обеспечить независимость программных модулей, которая достигается за счет уменьшения *связности* модулей друг с другом и увеличения *цельности* (внутреннего единства) модулей. Приведенная модульная структура предполагает связь модулей через передаваемые простые либо структурные параметры. Такая связь характеризуется наименьшей силой. При этом каждый модуль обладает *последовательно-функциональной цельностью*, т.е. в модуле может выполняться несколько последовательных функций, и выходные данные какой-либо функции являются входными данными для следующей функции. Во всех случаях надо стремиться к получению модулей с *функциональной цельностью*. Такие модули выполняют одну определенную функцию [9].

Схема модульной структуры программы должна быть дополнена описанием внешних характеристик программных модулей. Описание внешних характеристик модуля называется *внешней спецификацией*. Внешняя спецификация наряду с описанием функций, выполняемых модулем, а также внешних эффектов, происходящих при работе модуля, должна содержать описание **интерфейсов модулей**, т.е. всех сведений необходимых для обращения к нему других модулей. На основе внешней спецификации модулей в дальнейшем осуществляется разработка логической структуры этих модулей.

Приведем внешнюю спецификацию разрабатываемых модулей (в виде функций языка Си). При этом в основном ограничимся кратким описанием их интерфейсов.

Сначала определим необходимые константы:

```
const int N_ITEMS=7;           // количество пунктов меню
const int L_MODEL=20;          // длина названия телефона
const int STEP=18;             // количество строк на странице
```

Ввод базы из файла. Функция считывает записи из файла, создает двусторонний список и возвращает указатели на его начало и конец:

```
int readfile( elist* &beg, elist* &fin);
```

Файловая переменная и имя файла, содержащего базу данных о телефонах, определяются внутри функции.

Инициализация меню. Параметрами этой функции будут являться цвета активного и неактивного пунктов меню:

```
void initmenu(unsigned int activecolor, unsigned int inactivecolor);
```

Функция осуществляет вывод пунктов меню в заданное окно экрана.

Вывод пункта меню. В функцию передаётся номер пункта меню и цвет, которым следует его вывести:

```
void drawitem(unsigned int item, unsigned int color);
```

Добавление элемента в список. Для добавления элемента в список функции передаются ссылки на указатели начала и конца списка, а также собственно добавляемый элемент. Передача ссылок на указатели позволит функции после внесения элемента в список вычислять новые значения указанных указателей и передавать их вызвавшей подпрограмме:

```
void add(elist* &beg,elist* &fin,tel telefon);
```

Запрос сведений о телефоне. Эта функция используется при добавлении записи о телефоне и при коррекции записи о телефоне в базе данных. Функция возвращает структуру, содержащую сведения о телефоне:

```
tel query();
```

Запрос названия модели телефона. Эта функция требуется при удалении записи о телефоне и при поиске сведений о модели телефона. Поэтому её целесообразно оформить в виде небольшой вспомогательной функции. Функция считывает название модели телефона с клавиатуры и возвращает его через указатель, передаваемый ей в качестве параметра:

```
void queryname(char *s);
```

Поиск элемента в списке (по названию телефона) необходим при исправлении значений элемента списка, при удалении элемента из списка, а также при поиске сведений о модели телефона в базе данных. Функция получает указатель на начало списка и название модели телефона. В качестве результата она возвращает указатель на найденный элемент или пустую ссылку в случае неудачного поиска:

```
elist * find(elist *p, char *s);
```

Удаление элемента из списка. Функции передаются три указателя: указатель на начало списка, указатель на конец списка и указатель на элемент, который требуется удалить. Первые два указателя передаются по ссылке. Это позволяет функции после удаления элемента вычислять новые значения указателей начала и конца списка и передавать их в подпрограмму, вызвавшую функцию:

```
void del(elist* &beg, elist* &fin, elist *p);
```

Исправление записи о телефоне. В функции выполняется поиск элемента в списке и его корректировка. Для этого функции передается указатель на начало списка и сведения о модели телефона.

```
void edit(elist* beg, tel telefon);
```

Вывод базы на экран. У функции единственный параметр — указатель на начало списка, содержимое которого требуется отобразить на экране:

```
int showbase(elist *beg);
```

Выбор из базы моделей телефонов, попадающих в заданный пользователем диапазон цен. Ввод критериев выборки элементов списка выполняется внутри функции. Поэтому ей передается только указатель на начало списка:

```
void select(elist *beg);
```

В результате выборки функция формирует список, указатели начала и конца которого определяются внутри функции.

Ввод критериев отбора. Функция возвращает введенные пользователем значения минимальной и максимальной цены телефона:

```
void queryprice(float &startprice, float &endprice);
```

6.3. Пошаговое кодирование и отладка программы

До написания кода программы следует подготовить текстовый файл с базой телефонов. Он должен быть достаточно длинным, чтобы отладить вывод в окно с возможностью прокрутки содержимого вверх и вниз. Количество строк в файле рекомендуется сделать не кратным количеству строк в окне.

Кодирование будем выполнять пошагово, начиная с главной программы, а на месте ненаписанных подпрограмм поставим заглушки. Главная программа состоит из двух частей: части инициализации, включающей чтение из файла и вывод меню, и части отображения сообщений цикла, в котором анализируется, какой пункт меню выбрал пользователь.

6.3.1. Кодирование и отладка функции чтения из файла

Напишем часть главной программы, в которой будет выполняться ввод базы данных из файла и вывод этой базы на экран. Для этого будем вызывать из главной программы функции `readfile` и `showbase`. Функция `readfile` будет осуществлять последовательное чтение записей из файла и их добавление в двунаправленный список путем вызова функции `add`. Функция `showbase` будет обеспечивать элементарный просмотр формируемого списка (пока без возможности его прокрутки).

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//-----константы-----
const int L_MODEL=20;           // длина названия телефона
const int STEP=18;             // количество строк на странице

//-----типы и глобальные переменные-----
struct tel {                    //структура для хранения данных о телефоне
    char model[L_MODEL];        //название телефона
    float weight, price;        //вес и цена телефона
};

struct elist{                   //элемент 2- направленного списка
    tel data;                   //данные о телефоне
    elist *left;                //указатель на элемент слева
    elist *right;               //указатель на элемент справа
};

unsigned int defaultmode;       //текущий режим экрана
elist *beg,                     //указатель на первый эл-т списка
      *fin,                     //указатель на последний эл-т списка
      *p;                       //указатель на текущий эл-т списка
char name[L_MODEL];             //название модели телефона
tel telefon;                    //данные телефона

//-----прототипы функций-----
void add(elist* &beg,elist* &fin,tel telefon); //добавление элемента в список
void error(char message[]);      //вывод сообщения ошибки
int readfile( elist* &beg, elist* &fin);      //ввод базы из файла
int showbase(elist *beg);        //вывод базы на экран

//=====Главная функция=====
int main() {
    int n;
    defaultmode=LASTMODE;        //запомнить текущий текстовый режим экрана
    textmode(C80);               //установить режим 80x25 цветной

    beg=0; fin=0;                //обнулить указатели на начало и конец списка
    n=readfile(beg, fin);        //ввести базу из файла в список
    clrscr();                     //очистить экран
    n=showbase(beg);             //вывод базы на экран
    printf("n=%d",n);
    textmode(defaultmode);
    return 0;
}
```

```

//-----Вывод сообщения ошибки-----
void error(char message[]) {
    window(10,6,70,12);
    textbackground(LIGHTGRAY);
    textcolor(RED);
    clrscr();
    gotoxy(15,4);
    cputs(message);
    getch();
    textmode(defaultmode);
}

//-----Добавление элемента в конец списка-----
void add(elist* &beg,elist* &fin,tel telefon){
    elist *p;                                //указатель на создаваемый элемент
    p=new elist;                              //создание нового элемента
    p->data=telefon;                          //заполнение его информационной части
    p->right=0;                               //заполнение его указателей
    p->left=fin;
    if (!beg) beg=p;                         //если список был пустым
    else fin->right=p;                        //привязка нового эл-та к последнему
    fin=p;                                    //обновление указателя на конец списка
}

//-----Ввод базы из файла-----
int readfile( elist* &beg, elist* &fin){
    FILE *f;
    tel telefon;

    /*----проверка открытия файла base.txt ----*/
    f=fopen("c:\\bc5\\work\\base.txt","r");
    if(!f) {
        error("Файл base.txt не найден");
        exit(1);
    }

    /*----чтение записей из файла----*/
    while (!feof(f)) {
        fgets(telefon.model,L_MODEL,f);      //чтение названия модели
        fscanf(f,"%f%f\n",&telefon.weight,&telefon.price); //чтение веса и цены

        /*---контрольный вывод---*/
        printf("%s\t%5.1f\t%15.2f\n",telefon.model,telefon.weight,telefon.price);
        getch();

        add(beg,fin, telefon);                //добавление в список
    }
    fclose(f);
    return 0;
}

//-----Вывод базы на экран-(предварительная версия)-----
int showbase(elist *beg){
    int i;                                    //счетчик кол-ва строк

    if (!beg) {                              //если список пустой,
        cputs("Список пустой");              //то вывести сообщение
    }
}

```



```

    getch();
    return -1;
}

/*----задать параметры рабочего окна----*/
window(1,4,80,24);
textbackground(LIGHTGRAY);
textcolor(WHITE);
clrscr();

/*----циклический вывод записей из списка----*/
gotoxy(1,1);
cprintf(" Model      Weight   Price");
gotoxy(1,2);
i=2;
while (beg) {
    cprintf("%-22.20s %5.1f %15.2f",beg->data.model,beg->data.weight,beg->data.price);
    beg=beg->right;                               //переход к следующему элементу
    i++;
    gotoxy(1,i%step);
    if (i%STEP==0) {getch(); clrscr();}             //выведена одна страница
}
getch();
return i;
}

```

В функции `readfile` может возникнуть аварийная ситуация, обусловленная вводом неверного имени файла. Для её обработки введена вспомогательная функция `error`, которая выводит в текстовое окно на сером фоне красным цветом сообщение об ошибке и ожидает нажатия любой клавиши (рис. 6.3).



Рисунок 6.3 — Вывод сообщения об ошибке

Для организации оконного интерфейса программы рекомендуется использовать *функции работы с консолью* (содержащиеся в `conio.h`), описание которых приведено в приложении Д [1]. Вызов функции `window(10,6,70,12)` задает окно размером 6 строк по 60 символов, а вызовы `textbackground(LIGHTGRAY)` и `textcolor(RED)` устанавливают фон серого цвета и красный цвет отображаемого текста. Функция `gotoxy` устанавливает курсор в строку номер 4 и столбец 15 относительно верхнего левого угла текущего окна. Вызов функции `crputs(message)` обеспечивает вывод сообщения.

В тексте функции `readfile` содержится контрольный вывод. Убедившись в правильности работы функции, его следует удалить. Функцию следует запустить несколько раз, задав имя существующего файла и имя несуществующего файла. Функция `showbase` позволяет проконтролировать правильность работы функции добавления элементов в список. В дальнейшем она будет переписана.

6.3.2. Кодирование и отладка функции инициализации меню

Добавим в текст программы две функции, обеспечивающие инициализацию меню. Новые строки, добавленные в основную программу и в разделы объявлений, ниже выделены курсивом. Тексты уже отлаженных функций для экономии места не приводятся.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//-----константы-----
const int N_ITEMS=7;           //количество пунктов меню
const int L_MODEL=20;          //длина названия телефона
const int STEP=18;             //количество строк на странице

//-----типы и глобальные переменные-----
struct tel {                   //структура для хранения данных о телефоне
    char model[L_MODEL];        //название телефона
    float weight, price;        //вес и цена телефона
};

struct elist{                  //элемент 2- направленного списка
    tel data;                   //данные о телефоне
    elist *left;                //указатель на элемент слева
    elist *right;               //указатель на элемент справа
};

unsigned int defaultmode,      //текущий режим экрана
             activecolor,      //цвет активного пункта меню
             inactivecolor;    //цвет неактивного пункта меню

elist *beg,                   //указатель на первый эл-т списка
       *fin,                  //указатель на последний эл-т списка
       *p;                    //указатель на текущий эл-т списка
char name[L_MODEL];           //название модели телефона
tel telefon;                  //данные телефона

//-----прототипы функций-----
void add(elist* &beg,elist* &fin,tel telefon);    //добавление элемента в список
void error(char message[]);    //вывод сообщения ошибки
int readfile( elist* &beg, elist* &fin);          //ввод базы из файла
int showbase(elist *beg);      //вывод базы на экран
void drawitem(unsigned int item, unsigned int color); //вывод пункта меню
void initmenu(unsigned int activecolor, unsigned int inactivecolor); //инициализация меню

//=====Главная функция=====
int main() {
    int n;
    defaultmode=LASTMODE;      //запомнить текущий текстовый режим экрана
    textmode(C80);             //установить режим 80x25 цветной

    beg=0; fin=0;              //обнулить указатели на начало и конец списка
    n=readfile(beg, fin);      //ввести базу из файла в список
    clrscr();                  //очистить экран

    activecolor=LIGHTGREEN;      //цвет активного пункта меню
    inactivecolor=GREEN;        //цвет неактивного пункта меню
```

```

initmenu(activecolor,inactivecolor);    //инициализация меню
n=showbase(beg);                        //вывод базы на экран
cprintf("n=%d",n);
textmode(defaultmode);
return 0;
}

//-----Вывод пункта меню-----
void drawitem(unsigned int item, unsigned int color) {
const d=12;                            // длина поля пункта меню
const char * items[N_ITEMS]=           //названия пунктов
    {"Просмотр","Добавить","Исправить","Удалить","Найти","Выбрать","Выход"};
const int pos[N_ITEMS]={1,d,2*d,3*d,4*d,5*d,6*d}; //позиции начала полей пунктов
window(1,1,80,2);                      //окно для отображения пунктов
textbackground(LIGHTGRAY);              //цвет фона в окне светло-серый
textcolor(color);                       //цвет букв пункта
gotoxy(pos[item],1);                   //позиция начала пункта меню
cprintf("%s",items[item]);              //отобразить название пункта
}

//-----Инициализация меню-----
void initmenu(unsigned int activecolor, unsigned int inactivecolor){
unsigned int item;                      //номер пункта меню
window(1,1,80,2);                      //окно из двух строк для вывода пунктов
textbackground(LIGHTGRAY);              //цвет фона в окне светло-серый
clrscr();
/*----отобразить названия пунктов меню----*/
drawitem(0,activecolor);                 //активным цветом «Просмотр»
for (item=1;item<N_ITEMS;item++)
    drawitem(item,inactivecolor);        //все остальные неактивным

/* ----отобразить границу окна меню, подчеркнув его снизу----*/
gotoxy(1,2);
textcolor(inactivecolor);
cputs("_____");
gotoxy(1,1);                            //курсор на «Просмотр»
}

```

Названия пунктов меню программы хранятся в статическом массиве `items`, а позиции курсора, начиная с которых требуется отображать названия пунктов, — в массиве `pos`. Оба массива определены локально в функции `drawitem` и инициализированы необходимыми значениями. Функция обеспечивает отображение названия пункта меню по переданному ей номеру пункта. В функцию также передается цвет букв, который следует использовать при отображении пункта меню.

«Прорисовка» пунктов меню реализуется вызовом функции `initmenu` (рис. 6.4). Для этого с помощью вызова функции `window` в верхней части экрана определяется окно высотой 2 строки и шириной во весь экран. Функция `textbackground` задает для этого окна светло-серый фон с помощью константы `LIGHTGRAY`. Затем пункт меню «Просмотр» с помощью вызова `drawitem(0,activecolor)` отображается активным цветом, а остальные пункты меню с помощью циклического вызова этой же функции — неактивным цветом. В качестве активного цвета используется светло-зеленый цвет, а в качестве неактивного — зеленый. Повторная установка

текстового окна внутри функции `drawitem` обеспечивает её независимость от места вызова.

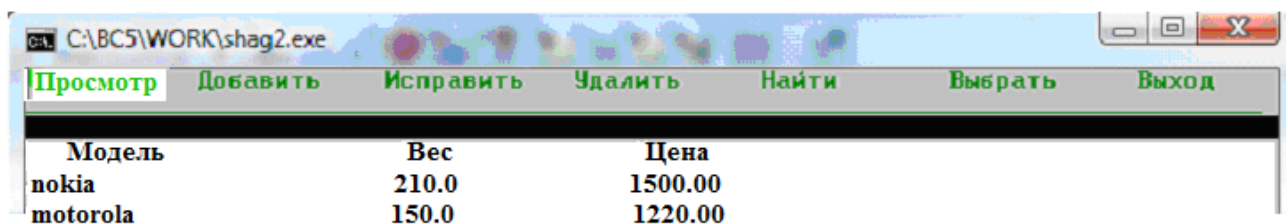


Рисунок 6.4— Внешний вид меню, формируемого программой

6.3.3. Кодирование и отладка выбора пунктов меню

Для обеспечения возможности выбора пунктов меню добавим в раздел описания переменных следующие строки:

```
int key;                //код клавиши
unsigned int item;      //номер пункта меню
unsigned int prev;      //номер предыдущего пункта меню
```

Кроме этого, опишем функцию-заглушку `stub`, которая просто будет выводить в рабочее окно сообщение о том, что вызван соответствующий пункт меню. Так как после завершения работы этой функции рабочее окно придется очищать от выведенного сообщения, то определим вспомогательную функцию `clear`. Заголовки функций

```
void clear(void);        //очистка текстового окна
void stub(unsigned int item); //заглушка
```

следует поместить в раздел описания прототипов функций, а определения функций

```
//-----Очистка рабочего окна -----
void clear() {
    window(1,3,80,25);
    textattr(WHITE);
    clrscr();
}
// -----Заглушка-----
void stub(unsigned int item) {
    window(1,4,80,24);
    textbackground(LIGHTGRAY);
    textcolor(WHITE);
    clrscr();
    printf("Вызван пункт меню %d ",item);
    getch();
}
```

разместить после кода главной функции.

Сам код главной функции тоже изменится и запишется в виде (добавленные строки выделены курсивом):

```
//=====Главная функция=====
int main() {
    defaultmode=LASTMODE;    //запомнить текущий текстовый режим экрана
    textmode(C80);           //установить режим 80x25 цветной
```

```

beg=0; fin=0; //обнулить указатели на начало и конец списка
readfile(beg, fin); //ввести базу из файла в список
clrscr(); //очистить экран
activecolor=LIGHTGREEN; //цвет активного пункта меню - светло-зеленый
inactivecolor=GREEN; //цвет неактивного пункта меню - зеленый
initmenu(activecolor,inactivecolor); //инициализация меню
tem=0; prev=0; //обнулить номера тек. и пред. пунктов меню
/* ---- цикл выбора пункта меню ---- */
while (1) {
    key=getch(); //ввести код нажатой клавиши
    if (key==0) key=getch(); //если введен расширенный код
    switch (key) { //проверить код нажатой клавиши
        /* ----выполнение пункта меню----*/
        case 13: // нажата Enter
            switch (item) { //анализ номера пункта
                case 0: showbase(beg); //вывод базы на экран
                    break;
                case 1: stub(item); break; // вызов заглушки
                case 2: stub(item); break;
                case 3: stub(item); break;
                case 4: stub(item); break;
                case 5: stub(item); break;
                default: // выход из программы
                    textmode(defaultmode);
                    exit(1);
            };
            break;
        /*----перемещение курсора влево----*/
        case 15: case 75: // нажаты Shift+Tab или Left
            prev=item;
            if (item==0)
                item=N_ITEMS-1;
            else item--;
            break;
        /*----перемещение курсора право----*/
        case 9: case 77: //нажаты Tab или Right
            prev=item;
            if (item==N_ITEMS-1)
                item=0;
            else item++; break;
    }
    clear();
    drawitem(prev, inactivecolor); //вывод предыдущего пункта
    drawitem(item, activecolor); //вывод выбранного пункта
}
}

```

Теперь главная функция будет реагировать на события, вызванные нажатием клавиш на клавиатуре. Для этого в цикле с помощью функции `getch()` считывается код нажатой клавиши, который запоминается в переменной `key`. Анализируя значение переменной `key`, можно выяснить, какая клавиша была нажата. Например, клавиша **Enter** имеет код 13, клавиша **Esc** — 27, **Tab** — 9. Нулевое значение

этой переменной сигнализирует о том, что была нажата специальная клавиша или комбинация специальных клавиш, генерирующих **расширенные коды**. В этом случае при повторном вызове `getch()` можно считать из буфера ввода дополнительный байт и таким образом выяснить, какая специальная клавиша была нажата. Значения дополнительного байта для некоторых специальных клавиш приведены в табл. 6.2.

Таблица 6.2 — Значения дополнительного байта для специальных клавиш

Клавиша	Код	Клавиша	Код
End	79	←	75
PgUp	73	↑	72
PgDn	81	→	77
Shift+Tab	15	↓	80

Значение переменной `key`, анализируется инструкцией `switch (key)`. Если была нажата клавиша `Enter`, то выбран один из пунктов меню. В этом случае выполняется инструкция `switch(item)`, в которой в зависимости от номера выбранного пункта меню осуществляется вызов предварительной версии функции `showbase`, либо заглушки. Если была нажата клавиша `Left` или комбинация клавиш `Shift+Tab`, то это означает, что пользователь перемещается по системе пунктов меню влево. В этом случае необходимо просто определить новые значения текущего (`item`) и предыдущего (`prev`) пунктов меню. Аналогичные действия выполняются при перемещении вправо (нажаты клавиши `Right` или `Tab`). После выхода из управляющей конструкции `switch (key)` осуществляется «перерисовка» текущего и предыдущего пунктов меню соответствующими цветами. Для этого вызывается функция `drawitem`.

6.3.4. Кодирование и отладка функции вывода базы на экран

Перепишем функцию `showbase` так, чтобы она обеспечивала возможность просмотра записей базы как в прямом (от начала к концу), так и в обратном направлениях. Введем вспомогательную функцию `showpage`, которая будет отображать на экране одну страницу базы. Для этого в функцию передается указатель на первую запись выводимой страницы. После вывода заданного числа строк (`STEP`) функция возвращает в качестве результата указатель на первую запись следующей страницы. Вызов функции `showpage` выполняется из функции `showbase`. Исходные коды обеих функции приведены ниже:

```
//-----Вывод базы на экран-----
int showbase(elist *beg){
int i;                                //счетчик кол-ва строк
elist *p,                             //указатель на текущий элемент страницы
      *pn;                             //указатель на 1-ый эл-т страницы
int key;                              //код нажатой клавиши
elist * showpage(elist *p);           //прототип функции вывода одной страницы

if (!beg) {                           //если список пустой,
    message("Список пустой");         //то вывести сообщение
    return -1;
}
```

```

/*----задать параметры рабочего окна----*/
window(1,4,80,24);
textbackground(LIGHTGRAY);
textcolor(WHITE);

p=beg;                                //указатель установить в начало списка
while (1) {
    pn=p;                             //запомнить указатель первого эл-та страницы
    /*---вывести страницу и получить новое значение
        указателя текущего элемента---*/
    p=showpage(p);
    /*----определить код нажатой клавиши----*/
    key=getch();
    if (key==0) key=getch();           //если расширенный код

    /*----управление прокруткой-----*/
    switch (key) {
        case 27:                      //нажата Esc
            return 0;                 //выход из просмотра
        case 13: case 80: case 81:     //если нажаты Enter||Down||PgDn
            if (!p) p=pn;              //и список исчерпан, то
            break;                    //отображать ту же страницу
        case 72: case 73:              //если нажаты Up или PgUp,
            p=pn;                     //то указатель в начало стр.
            for(i=1;(i<=STEP)&&(p);i++)
                p=p->left;             //на STEP записей вверх
            if (!p) p=beg;             //если 1-ая запись
            break;
    }
}
}

//-----Вывод одной страницы-----
elist* showpage(elist *p){
    int i;                            //номер строки вывода
    clrscr();
    /*---вывести в первую строку окна заголовков таблицы----*/
    gotoxy(1,1);
    sprintf("Модель      Вес  Цена");
    /*---начиная со второй строки окна, выводить элементы списка (базу)---*/
    gotoxy(1,2);
    i=2;
    while (p) {
        sprintf("%-22.20s %5.2f %10.2f",p->data.model,p->data.weight,p->data.price);
        p=p->right;
        i++;
        gotoxy(1,i);
        /*----выйти из функции, если выведена одна страница----*/
        if (i>STEP) return p;         //и вернуть указатель текущего эл-та списка
    }
    return p;                         //выход при достижении конца списка
}

```

Управление прокруткой, реализованное в showbase, основано на перемещении указателя первой записи (первого элемента) страницы вверх или вниз с шагом, равным размеру страницы. При вычислении значений указанного указателя выполняется анализ нажатия соответствующих клавиш. В случае нажатия клавиши Esc происходит выход из функции.

Если просматриваемая база окажется пустой, то выводится соответствующее сообщение с помощью вспомогательной функции message, которая размещает текст выводимого сообщения в диалоговом окне, формируемом другой вспомогательной функцией dlgwindow. Диалоговое окно, формируемое этой функцией, понадобится также при реализации оставшихся пунктов меню. Поэтому его формирование реализовано в виде самостоятельной функции.

```
//----- Отображение окна диалога-----
void dlgwindow(){
    window(10,6,70,12);
    textattr(GREEN+LIGHTGRAY*16);
    clrscr();
}
//-----Вывод сообщения-----
void message(char message[]) {
    dlgwindow();
    gotoxy(20,4);
    cputs(message);
    getch();
}
```

Включение в текст программы указанных вспомогательных функций, требует внесения в раздел прототипов функций следующих двух строк:

```
void dlgwindow(void);           //отображение окна диалога
void message(char *message);    //вывод сообщения
```

6.3.5. Кодирование и отладка пунктов меню: добавить, удалить, исправить, найти

Для реализации действий, указанных в заголовке, определим следующие дополнительные функции, которые будут вызываться из соответствующих пунктов меню:

```
tel query();                    //запрос сведений о телефоне
void queryname(char *s);        //запрос названия модели телефона
void edit(elist* beg, tel telefon); //исправление записи о телефоне
elist * find(elist *p, char *s); //поиск эл-та в списке по названию
void del(elist* &beg,elist* &fin,elist *p); //удаление эл-та из списка
void info(tel telefon);         //вывод сведений о телефоне
```

Функция query будет запрашивать у пользователя сведения о телефоне для того, чтобы в дальнейшем добавить их в виде новой записи в базу данных (с помощью функции add) или внести исправления в существующую запись с помощью вызова функции edit. Функция queryname будет запрашивать у пользователя только название модели телефона для поиска соответствующей записи в базе с целью её удаления из базы либо вывода имеющихся в базе сведений о данной мо-

дели. При этом поиск будет выполняться с помощью функции `find`, удаление и выдача сведений соответственно с помощью функций `del` и `info`.

Для реализации этих возможностей необходимо исправить указанные ниже ветви инструкции `switch (item)` главной функции:

```

case 1:                                     //добавление записи в базу
    telefon=query();
    add(beg,fin, telefon);
    break;
case 2:                                     //исправление записи
    telefon=query();
    edit(beg, telefon);
    break;
case 3:                                     //удаление записи
    queryname(name);
    p=find(beg,name);
    if (p) del(beg,fin,p);
    break;
case 4:                                     //поиск модели в базе
    queryname(name);
    p=find(beg,name);
    if (p) info(p->data);
    break;

```

А также дополнить программу следующими определениями функций:

```

//-----Запрос сведений о телефоне-----
tel query(){
char s[L_MODEL];                          //название модели телефона
tel telefon;                             //данные о телефоне
int i;                                    //индекс строки
int len;                                 //длина строки с названием модели
dlgwindow();                             //отобразить окно диалога

/*---ввод названия модели телефона---*/
do {
    gotoxy(2,1);
    cputs("Model?");
    gotoxy(2,2); clreol();
    cscanf("%s",s);
    len=strlen(s);
    for (i=len; i<L_MODEL-1; i++)
        s[i]=' ';                        //дополнение пробелами
    s[L_MODEL-1]='\0'; }
while (!len);
strcpy(telefon.model,s);

/*---ввод значения веса телефона---*/
do {
    gotoxy(2,3);
    cputs("Weight?");
    gotoxy(2,4); clreol();
    cscanf("%s",s);
    telefon.weight=atof(s);}
while (!telefon.weight);

```

```

/*---ввод значения цены телефона---*/
do {
    gotoxy(2,5);
    cputs("Price?");
    gotoxy(2,6); clreol();
    cscanf("%s",s);
    telefon.price=atof(s); }
while (!telefon.price);

getch(); //очистка буфера ввода
return telefon;
}
// ----- Исправление записи о телефоне -----
void edit(elist* beg, tel telefon) {
    elist *p; //указатель на найденный эл-т
    p=find(beg,telefon.model); //поиск телефона в списке
    if (p) { //если модель найдена,
        p->data.weight=telefon.weight; //то изменить вес
        p->data.price=telefon.price; // и изменить цену
    }
}
// -----Поиск эл-та в списке (по названию модели)-----
elist * find(elist *p, char *s){
/*---цикл по элементам списка ----*/
while (p) {
    if (strcmp(s,p->data.model)==0) //если модель найдена,
        return p; //то вернуть указатель
    p=p->right; //переход к след. эл-ту списка
}

/*---если модель не найдена----*/
gotoxy(2,6);
message("Такой модели в списке нет");
p=0;
return p;
}
// -----Запрос названия модели телефона-----
void queryname(char *name){
    int i; //индекс строки
    int len; //длина строки с названием модели
    char s[L_MODEL]; //название модели телефона
    dlgwindow(); //отобразить окно диалога

/*---ввод названия модели---*/
do {
    gotoxy(2,2);
    cputs("Model?");
    gotoxy(2,3); clreol();
    cscanf("%s",s);
    len=strlen(s);
    for (i=len; i<L_MODEL-1; i++)
        s[i]=' '; //дополнение пробелами
    s[L_MODEL-1]='\0'; }
while (!len);

```



```

elist *p;                                     //текущий эл-т исходного списка
queryprice(startprice,endprice);              //ввод начальной и конечной цены

/*---- инициализация указателей----*/
begs=0;
fins=0;
p=beg;
/*----цикл по элементам исходного списка и формирование нового списка----*/
while (p) {
    if ((p->data.price>=startprice)&&          //если цена в заданном диапазоне,
        (p->data.price<=endprice))
        add(begs,fins,p->data);              //то добавить эл-т в новый список
    p=p->right;                               //переход к следующему элементу
}
showbase(begs) ;                             //вывод выборки (списка) на экран
}

//-----Ввод критериев отбора-----
void queryprice(float &startprice, float &endprice) {
char *s=«    »;                             //вспомогат. строка для ввода числа
dlgwindow();                                //отобразить окно диалога
do {
    /*---ввод начальной цены---*/
    do {
        gotoxy(2,2);
        sprintf("Минимум: ");
        gotoxy(2,3);
        clreol();
        cscanf("%s",s);
        startprice=atof(s);                  //преобразовать строку в вещ. число
    }
    while (!startprice);

    /*---ввод конечной цены---*/
    do {
        gotoxy(2,4);
        sprintf("Максимум: ");
        gotoxy(2,5);
        clreol();
        cscanf("%s",s);
        endprice=atof(s); }                  //преобразовать строку в вещ. число
    while (!endprice);
}
while (startprice>endprice);                 //повторять ввод пока startprice>endprice
getch();                                    //очистка буфера ввода
}

```

С целью иллюстрации в разделе 7 приведена схема алгоритма функции select. На этом шаге кодирование и отладка завершаются. Полный текст программы приведен в приложении Е.

7. ПРАВИЛА ОФОРМЛЕНИЯ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

7.1. Общие требования

Пояснительная записка оформляется на листах белой бумаги стандартного формата А4 (210 × 297 мм) с одной стороны листа. Записка может быть написана от руки или напечатана на принтере.

Один лист должен содержать не более 40 строк. Следует соблюдать следующие размеры полей: верхнее, левое и нижнее поля — не менее 20мм, правое — не менее 10мм. Абзацный отступ должен быть одинаковым по всему тексту и равным 10...15 мм.

При **рукописном оформлении** текст пишется разборчивым почерком с одной стороны листа чёрными или фиолетовыми чернилами (пастой). При **компьютерном наборе** рекомендуется текст печатать шрифтом высотой от 12 до 14 пт. через 1,5 интервала (допускается также печатать текст через один или два интервала).

Нумерация страниц — сквозная, начиная с титульного листа. Все страницы, кроме титульного листа и страниц технического задания, **нумеруются** в правом верхнем углу цифрами без каких-либо дополнительных знаков.

Термины и определения, используемые в пояснительной записке должны быть едиными и соответствовать установленным стандартам, а при их отсутствии — общепринятым в научно-технической литературе. Текст должен быть кратким, четким, исключая возможность неверного толкования. Для выделения отдельных понятий допускается изменять интервалы между словами, а также печатать отдельные слова или части текста шрифтом, отличным от печати основного текста.

Сокращения слов в тексте не допускаются, за исключением общепринятых: и т.д. (так далее); и т.п. (тому подобное); и др. (другие); см. (смотри); разд. (раздел), п. (пункт); табл. (таблица); рис. (рисунок); стр. (страница).

7.2. Структура пояснительной записки

Материал пояснительной записки разбивается на **разделы и подразделы**, а при необходимости подразделы разбиваются на пункты и подпункты. Каждый раздел начинают с нового листа.

Заголовки разделов следует располагать в середине строки и печатать прописными буквами без точки в конце, не подчеркивая. В содержании заголовки разделов пишутся строчными буквами с первой заглавной буквы.

Заголовки подразделов, пунктов и подпунктов следует начинать с абзацного отступа и печатать строчными буквами, кроме первой прописной, не подчеркивая, без точки в конце.

Если заголовок состоит из двух или более предложений, их разделяют точкой. Переносы слов в заголовке раздела не допускаются. Заголовки разделов и подразделов печатаются полужирным шрифтом

Расстояние между заголовком и последующим текстом, а также между заголовками раздела и подраздела должно быть равно:

- при выполнении рукописным способом — 10 мм;

- при компьютерном наборе — одна пустая строка.

Расстояние между последней строкой текста и последующим заголовком должно быть равно:

- при выполнении рукописным способом — не менее 15 мм;
- при компьютерном наборе — две пустые строки.

Не допускается размещать наименование раздела, подраздела, а также пункта и подпункта в нижней части страницы, если после него расположена только одна строка текста.

Порядковые номера разделов, подразделов и пунктов обозначаются арабскими цифрами с точкой после каждой цифры, в том числе последней цифры (см. как пример нумерацию разделов и пунктов данных методических указаний). Не нумеруются введение, заключение, перечень ссылок.

Номер раздела состоит из одной цифры с точкой, соответствующей этому разделу. **Номер подраздела** составляется из номера раздела и подраздела, разделенных точкой. **Номер пункта** составляется из номера раздела, номера подраздела и номера пункта, разделенных точками.

Пример структуры текста программного документа и нумерации его разделов, подразделов, пунктов и подпунктов приведён в приложении Ж.

Текст может содержать **перечисления**. Перед перечислением ставится двоеточие, пункты перечисления оканчиваются точкой с запятой. Перед каждой позицией перечисления следует ставить дефис или строчную букву со скобкой (первый уровень детализации). Для дальнейшей детализации перечисления следует использовать арабские цифры со скобкой (второй уровень детализации).

Пример перечисления.

В тексте, таблицах и формулах необходимо соблюдать следующие требования к шрифтам:

- русские и греческие буквы набирать прямым шрифтом;
- латинские буквы набирать наклонным шрифтом (курсивом);
- в десятичных дробях целая часть отделяется запятой, а не точкой;
- нельзя путать разделительные символы:
 - 1) дефис, записываемый в отдельных словах (из-за, кто-то);
 - 2) минус, применяемый в математических выражениях (—);
 - 3) тире (—).

7.3. Оформление отдельных частей текста

В тексте документа можно применять **примечания**, которые приводятся сразу после абзаца. Одно примечание не нумеруется. После слова «Примечание» ставят точку. Несколько примечаний следует нумеровать по порядку арабскими цифрами с точкой. После слова «Примечания» ставят двоеточие.

Например:

П р и м е ч а н и е.

или

П р и м е ч а н и я:

- 1.
- 2.

Текст примечаний допускается печатать через один интервал.

Необходимые пояснения к тексту документа также могут оформляться в виде сносок. **Сноска** обозначается цифрой со скобкой, вынесенными на уровень линии верхнего обреза шрифта, например: «печатающее устройство²⁾...» или «бумага⁵⁾». Текст сноски располагают в конце страницы и отделяют от основного текста линией длиной 3 см, проведённой в левой части страницы.

Пример.

...структурное программирование¹⁾ предусматривает....

¹⁾ В узком его понимании.

В тексте документа и (или) в приложениях могут быть расположены **рисунки**. Оформлять рисунки допускается карандашом, подрисовочные подписи — черной пастой или чернилами (тушью). Нумерация рисунков должна быть двойной: первая цифра соответствует номеру раздела, а вторая — номеру рисунка в пределах раздела. В приложениях рисунки нумеруются в пределах каждого приложения в порядке, установленном для основного текста документа.

Ссылка на рисунок в тексте обязательна, например, «... как показано на рис. 1.1», «... выход из цикла происходит после проверки условия (рис. 1.2)». Рисунок располагают посередине страницы сразу после абзаца, в котором дается первая ссылка на него. Если рисунок занимает целую страницу, то его приводят сразу после страницы, на которой указана ссылка на рисунок. Рисунок, размещаемый на отдельной странице, может быть развернут на 90 ° против часовой стрелки относительно основного текста. Ссылки на рисунки, которые уже упоминались ранее, приводят с сокращенным словом «смотри», например, «см. рис. 1.2».

Рисунок имеет заголовок и подрисовочный поясняющий текст. Заголовок рисунка приводится под изображением рисунка и поясняющим текстом по центру страницы. Заголовок содержит слово «Рисунок», его номер, тире, название рисунка. Небольшой рисунок может не иметь названия, только номер.

Если рисунок не помещается на одной странице, можно переносить его на другие страницы, при этом название рисунка помещают на первой странице, поясняющие данные — на каждой странице, и под ними указывают: «Рисунок __, лист __».

Если на одном рисунке приводятся два и более изображения, то сверху-слева от каждого изображения подписывается его буквенное обозначение: а), б) и т.д. Названия этих изображений приводят, как правило, в названии всего рисунка, например: «Рисунок 1.1 — «Схема алгоритма с одним (а) и двумя (б) блоками решений». Примеры оформления рисунков смотри в тексте настоящих методических указаний.

Формулы в документе нумеруются арабскими цифрами в пределах раздела, номер ставят с правой стороны страницы, в скобках на уровне формулы. Ссылки в тексте на порядковый номер формулы дают в скобках, например: «в формуле (7.1)».

Значение символов и числовых коэффициентов, входящих в формулу, должны быть приведены непосредственно под формулой. Значение каждого сим-

вола печатают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова «где», без двоеточия после него. Если в программном документе приведен перечень этих символов и числовых коэффициентов, значения их под формулой допускается не приводить.

Пример.

Известно, что

$$Z = \frac{M_1 - M_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}, \quad (7.1)$$

где M_1, M_2 — математическое ожидание;

σ_1^2, σ_2^2 — среднеквадратическое отклонение прочности и нагрузки.

Таблица приводится сразу после абзаца со ссылкой на нее, например: «...данные представлены ниже (табл. 1.1)». Таблицы должны иметь заголовки и номер. Номер таблицы должен состоять из номера раздела и порядкового номера таблицы в пределах каждого раздела, записанного через точку. Заголовок таблицы приводится с абзацного отступа. Примеры нумерации и заголовков таблиц смотри в тексте настоящих методических указаний.

Если таблица занимает несколько листов, то на других листах вместо заголовка таблицы пишется только «Продолжение таблицы N». При этом заголовки граф повторяются или заменяются цифрами, в этом случае под первой строкой заголовков граф на первой странице выделяется строка с их номерами.

Сноски к таблицам располагают непосредственно под таблицей.

Пример.

Таблица 1.1 — Наборы данных, используемые для распечатки

Назначение	Стандартное имя	Используемое устройство
Для информационной распечатки	SSSSSSS ¹⁾	Печатающее устройство ²⁾
Для распечатки во время выполнения программы	PPPPPPPP	Печатающее устройство ²⁾

¹⁾ Имя SSSSSSS должно быть задано при настройке операционной системы.

²⁾ Для уменьшения простоев центрального процессора из-за операций ввода-вывода может быть использован накопитель на основе жесткого диска.

Перечень ссылок составляется в порядке очерёдности, в которой ссылки встречаются в тексте, либо в алфавитном порядке. Сведения об источниках информации оформляются с учётом требований действующих стандартов по библиотечному делу (см. приложение 3). Ссылки в тексте отчета на источники следует указывать порядковым номером по перечню ссылок, выделенным двумя квадратными скобками, например, «... в работах [1 – 7] ...».

Приложение должно иметь заголовок, напечатанный вверху строчными буквами с первой прописной симметрично относительно текста страницы. Посе-

редине строки над заголовком строчными буквами с первой прописной должно быть напечатано слово «Приложение __» и прописная буква, обозначающая приложение (А, Б, В и т.д.). Приложение обозначается буквой, даже если оно всего одно.

Содержание каждого приложения, при необходимости, разбивают на разделы, подразделы, пункты, нумеруемые отдельно по каждому приложению. Нумерация страниц документа и приложений, входящих в состав документа, должна быть сквозная, если приложения не выполняются отдельным документом. Иллюстрации и таблицы в приложениях нумеруют в пределах каждого приложения. На приложения должны быть даны ссылки в основном тексте документа. Все приложения должны быть перечислены в разделе «Содержание».

7.4. Правила оформления схем алгоритмов

Правила оформления схем алгоритмов, программ и данных устанавливает **ГОСТ 19.002-80**. Схемы алгоритмов, программ, данных и систем состоят из имеющих заданное значение символов, краткого пояснительного текста и соединяющих линий.

В стандарте используются следующие определения:

- **основной символ** — символ, используемый в тех случаях, когда точный тип (вид) процесса или носителя данных неизвестен или отсутствует необходимость в описании фактического носителя данных;
- **специфический символ** — символ, используемый в тех случаях, когда известен точный тип (вид) процесса или носителя данных или когда необходимо описать фактический носитель данных.

В общем случае при разработке программных систем разрабатывают: схемы данных, схемы программ (алгоритмов), схемы работы системы, схемы взаимодействия программ, схемы ресурсов системы.

В курсовом проекте разрабатываются только схемы программ. Эти схемы отображают последовательность операций в программе. Схема программы может называться также схемой алгоритма или процедуры (подпрограммы), судя по содержанию. **Не следует использовать давно устаревший термин «блок-схема».** Схема программы состоит из:

- символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);
- символов линий, указывающих поток управления;
- специальных символов, используемых для облегчения написания и чтения схемы.

На схемах программ применяют символы, приведенные в табл. 7.1.

Символ предназначен для графической идентификации функции, которую он отображает, независимо от текста внутри этого символа. Большинство символов задумано так, чтобы дать возможность включения текста внутри символа.

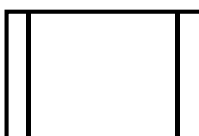
Формы символов, установленные рассматриваемым стандартом, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов. Символы должны быть, по возможности, одного размера. При необходимости определения ориентировочных размеров символов можно воспользоваться уже не действующим стандартом ГОСТ 19.003-80.

Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.


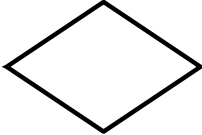

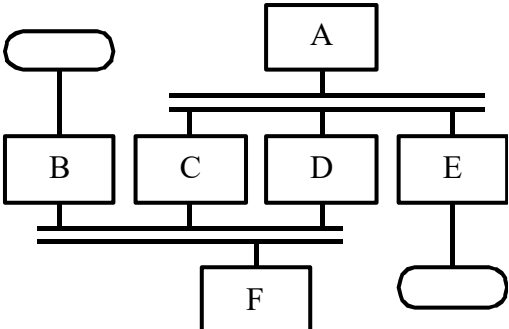
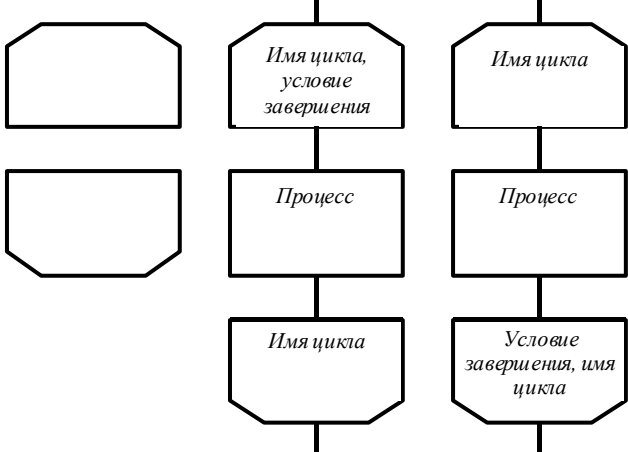


Минимальное количество текста, необходимого для понимания функции данного символа, следует помещать внутри данного символа. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока. Если объем текста, помещаемого внутри символа, превышает его размеры, следует использовать символ комментария.

Символ может иметь **идентификатор** (номер). Идентификатор символа должен располагаться слева над символом. Его используют для ссылки на символ, например, при описании схемы программы.

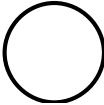

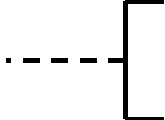

Таблица 7.1 — Символы, применяемые на схемах программ

Название и изображение символа	Назначение символа
1	2
Символы данных	
Данные (основной) 	Символ отображает данные, носитель данных не определен.
Символы процесса	
Процесс (основной) 	Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).
Предопределенный процесс (специфический) 	Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).

Продолжение таблицы 7.1

1	2
<p>Подготовка (специфический)</p> 	<p>Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы).</p>
<p>Решение (специфический)</p> 	<p>Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычислений могут быть записаны рядом с выходными линиями.</p>
<p>Параллельные действия (специфический)</p>  <p>Пример.</p> 	<p>Символ отображает синхронизацию двух или более параллельных операций.</p> <p>Примечание. Процессы С, D и E не могут начаться до тех пор, пока не завершится процесс А; аналогично процесс F должен ожидать завершения процессов В, С и D. Однако процесс С может начаться и (или) завершиться прежде, чем соответственно начнется и (или) завершится процесс D.</p>
<p>Граница цикла (специфический)</p> 	<p>Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.</p>
Символы линий	
<p>Линия (основной символ)</p> 	<p>Символ отображает поток данных или управления. При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.</p>
<p>Пунктирная линия (специфический)</p> 	<p>Символ отображает альтернативную связь между двумя или более символами. Кроме того, символ используют для обведения участка.</p>

Продолжение таблицы 7.1

1	2
Специальные символы	
Соединитель (специальный) 	Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.
Терминатор 	Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).
Комментарий 	Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры.
Пропуск 	Символ (три точки) используют в схемах для отображения пропуска символа или группы символов, в которых не определены ни тип, ни число символов. Символ используют только в символах линии или между ними. Он применяется в схемах, изображающих общие решения с неизвестным числом повторений.

В схемах может использоваться символ основного **процесса с полосой**. Символ с полосой указывает, что в этом же комплекте документации в другом месте имеется более подробное представление процесса. Символ с полосой представляет символ, внутри которого в верхней части проведена горизонтальная линия. Между этой линией и верхней линией символа помещен идентификатор, ссылающийся на подробное представление данного символа. В качестве первого и последнего символа подробного представления процесса должен быть использован символ-терминатор. Первый символ-терминатор должен содержать ссылку, которая имеется также в символе с полосой (рис. 7.1).

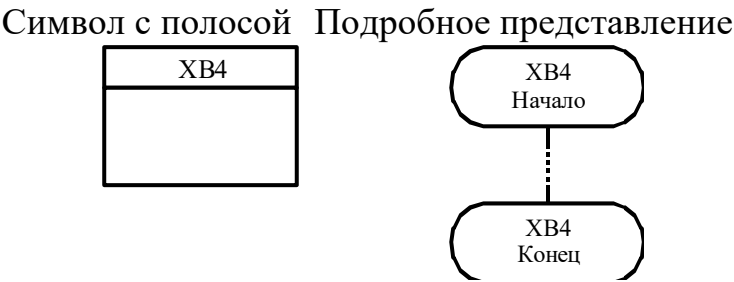


Рисунок 7.1 — Пример применения символа с полосой

Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным. В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используются стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

Две или более входящие линии могут объединяться в одну исходящую линию. Если две или более линии объединяются в одну линию, место объединения должно быть смещено (рис. 7.2).

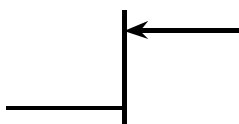


Рисунок 7.2 — Объединение входящих линий

Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

При необходимости линии в схемах следует разрывать во избежание излишних пересечений или слишком длинных линий, а также, если схема изображена на нескольких страницах. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва — внутренним соединителем. Ссылки на страницы могут быть приведены совместно с символом комментария (рис. 7.3).

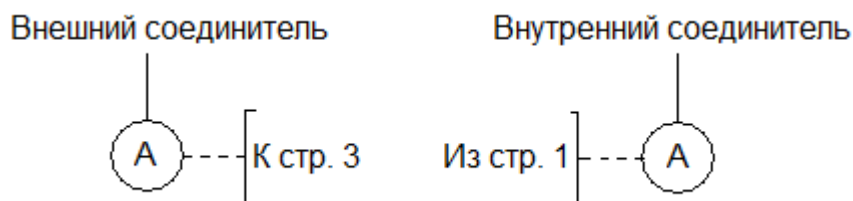


Рисунок 7.3 — Соединение линий потока между страницами

Несколько выходов из символа следует показывать:

- а) несколькими линиями от данного символа к другим символам;
- б) одной линией от данного символа, которая затем разветвляется в соответствующее число линий.

Каждый выход из символа должен сопровождаться соответствующими значениями условий (рис. 7.4), чтобы показать логический путь, который он представляет, с тем, чтобы эти условия и соответствующие ссылки были идентифицированы.

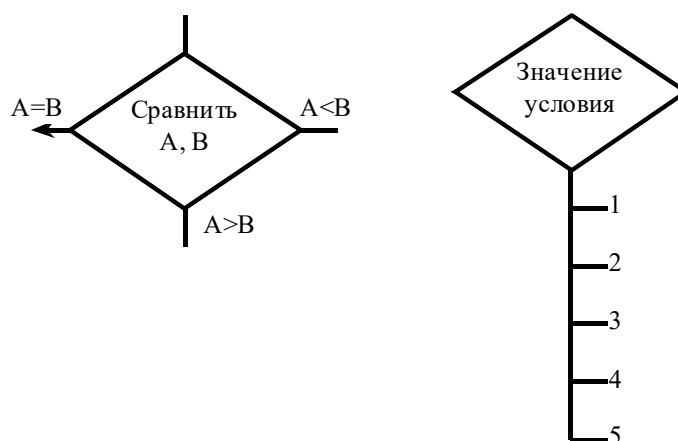


Рисунок 7.4 — Изображение символа с несколькими выходами

В качестве примера применения указанных выше символов и правил, рассмотрим схему алгоритма функции `select` (рис. 7.5), которая была определена в разделе 6.

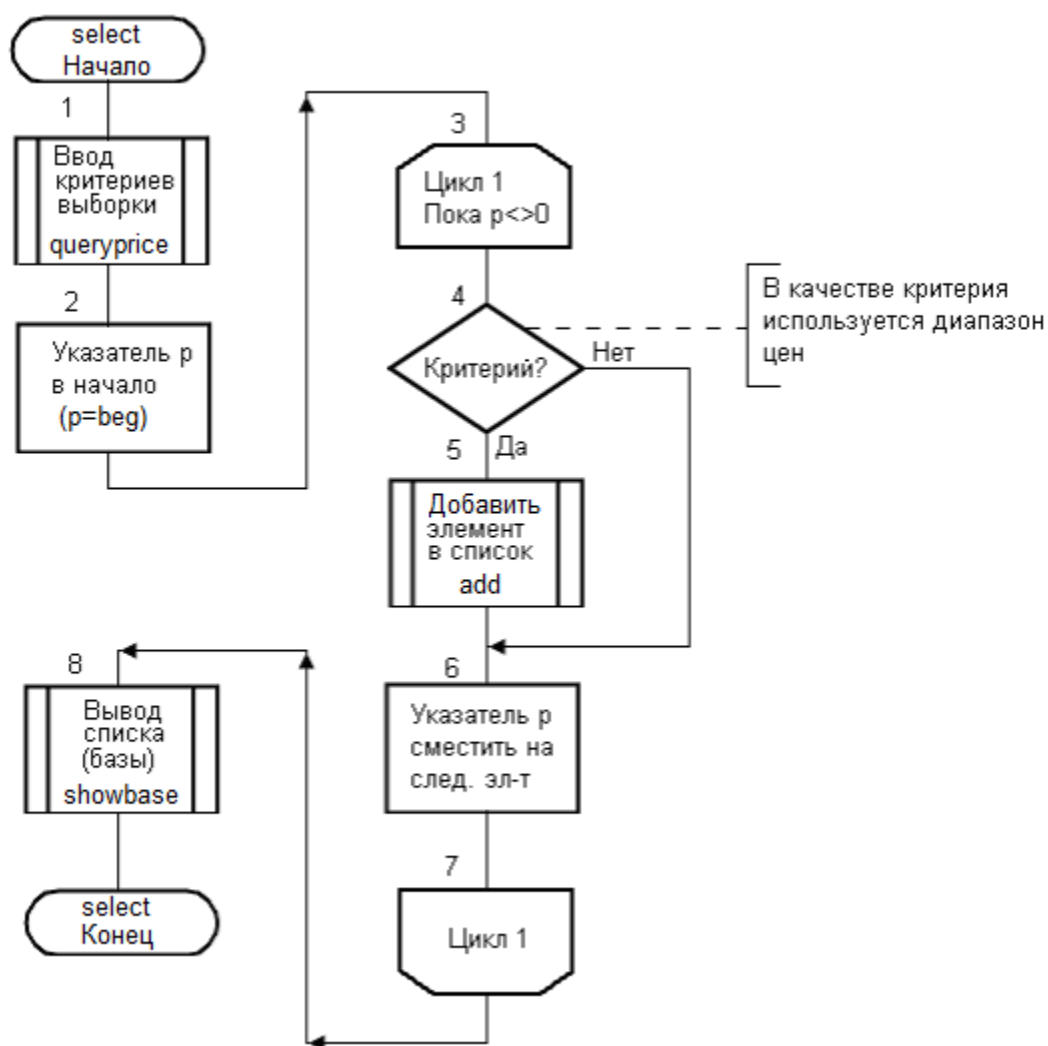


Рисунок 7.5 — Схема алгоритма функции `select`

Наличие на рисунке идентификаторов символов позволяет выполнять описание схемы.

Пример.

На рис. 7.5. изображена схема алгоритма. Здесь в блоке 1 осуществляется ввод критериев выборки записей из базы. Для этого вызывается функция `quegrise`. В блоке 2 выполняется инициализация указателя текущего элемента списка `p`, которому присваивается значение указателя на начало списка...

При изображении схем алгоритмов и программ, оформляемых отдельно от пояснительной записки *в виде демонстрационного материала*, они должны отвечать требованиям наибольшей наглядности и свободно просматриваться с расстояния 3...5 м. Для этого демонстрационный лист следует выполнять на чертежной бумаге стандартных форматов: минимальный формат листа — А3 (297x420 мм), максимальный — А1 (594x840 мм). При этом необходимо привести:

- рамку с основной надписью;
- название программы (подпрограммы);
- изображение схемы программы (подпрограммы);
- пояснительный текст (при необходимости).

Название программы помещают над схемой и пишут чертежным шрифтом. Пояснительный текст должен располагаться на свободном поле листа и выполняться также чертежным шрифтом.

В основной надписи (в рамке) на демонстрационных листах в графе "Обозначение документа" указывается обозначение программного документа, которое должно совпадать с обозначением на титульном листе ПЗ, а в графе "Наименование изделия" — название программы (подпрограммы).

Графическое представление алгоритмов и программ является одним из традиционных способов их описания. Однако этот способ обладает тем недостатком, что уже при количестве основных символов более 15...20 теряется наглядность, а при внесении изменений приходится перечерчивать схему. Поэтому в настоящее время этот способ описания алгоритмов и программ используется в основном при разработке простых программных систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: <https://e.lanbook.com/book/1261>. — Загл. с экрана.

2. Медведик, В.И. Практика программирования на языке Паскаль (задачи и решения) [Электронный ресурс] : учебное пособие / В.И. Медведик. — Электрон. дан. — Москва : ДМК Пресс, 2013. — 590 с. — Режим доступа: <https://e.lanbook.com/book/58700> — Загл. с экрана.

3. Подбельский, В.В. Курс программирования на языке Си [Электронный ресурс] : учебник / В.В. Подбельский, С.С. Фомин. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 384 с. — Режим доступа: <https://e.lanbook.com/book/4148>. — Загл. с экрана.

4. Кауфман, В.Ш. Языки программирования. Концепции и принципы [Электронный ресурс] / В.Ш. Кауфман. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 464 с. — Режим доступа: <https://e.lanbook.com/book/1270>. — Загл. с экрана.

5. Солдатенко, И.С. Практическое введение в язык программирования Си [Электронный ресурс] : учебное пособие / И.С. Солдатенко, И.В. Попов. — Электрон. дан. — Санкт-Петербург : Лань, 2018. — 132 с. — Режим доступа: <https://e.lanbook.com/book/109619>. — Загл. с экрана.

6. Андрианова, А.А. Алгоритмизация и программирование. Практикум [Электронный ресурс] : учебное пособие / А.А. Андрианова, Л.Н. Исмагилов, Т.М. Мухтарова. — Электрон. дан. — Санкт-Петербург : Лань, 2019. — 240 с. — Режим доступа: <https://e.lanbook.com/book/113933>. — Загл. с экрана.

ПРИЛОЖЕНИЕ А

(СПРАВОЧНОЕ)

ВАРИАНТЫ ЗАДАНИЙ К КУРСОВОМУ ПРОЕКТУ

Вариант 1

Даны записи о расходовании электроэнергии на заводах отрасли. Структура записи: номер завода, Ф.И.О. директора (15 символов), Ф.И.О. главного энергетика (15 символов), расход энергии по плану (в тыс. кВт·ч), израсходовано фактически (в тыс. кВт·ч). Получить по каждому заводу размер отклонения факта от плана (со знаком). Подсчитать суммарные значения по каждой позиции (для отрасли в целом) и отпечатать их в конце таблицы, содержащей исходные данные и результаты расчетов.

Вариант 2

Даны сведения о результатах разработки программ для курсовых проектов. Структура записи: шифр группы (6 символов), наименование дисциплины (10 символов), количество студентов в группе, число отлаженных программ. По каждой группе подсчитать процент отлаженных программ; подсчитать также суммарные показатели численности групп и числа отлаженных программ, среднее число и средний процент отлаженных программ в группе. Результаты оформить в виде таблицы.

Вариант 3

Даны сведения о расходовании на автобазах города топлива по следующему макету: номер автобазы, Ф.И.О. директора (15 символов), израсходовано топлива (в условных единицах), количество автомашин на базе. Подсчитать средний расход топлива на одну машину по каждой базе и в целом по городу.

Вариант 4

Даны сведения о перевозках авиапассажиров на рейсах одного аэропорта. Структура записи: номер рейса, название маршрута (например, «Севастополь – Лондон», 20 символов), марка самолета (6 символов), общие стоимостные затраты на рейс, количество пассажиров. Подсчитать среднюю стоимость перевозки одного пассажира на рейсе, а также итоговые данные по затратам и количеству пассажиров и среднюю стоимость провозки одного пассажира по аэропорту.

Вариант 5

Даны сведения за квартал о материальных ценностях в стоимостном выражении по филиалам завода. Структура записи: номер завода, номер филиала, фамилия ответственного за материальные ценности, наличие материальных ценностей на начало периода, получено материальных ценностей на сумму, выбыло на сумму. Получить ведомость движения материальных ценностей за отчетный период, содержащую: стоимость материальных ценностей по каждому филиалу на конец отчетного периода, по всему заводу на конец периода, а также итоговые цифры по каждому виду (на начало периода, получено, выбыло, на конец периода) и по всему заводу в целом. Проконтролировать двумя путями (по строке и столбцу) получение итоговой цифры: наличие материальных ценностей на конец отчетного периода в стоимостном выражении по заводу в целом.

Вариант 6

Даны сведения о времени выполнения заданий на ЭВМ (время измеряется в секундах). Структура записи: шифр задания (8 символов), код отдела (3 символа), Ф.И.О. программиста (15 символов), общее время прохождения задания, время центрального процессора – 5 знаков ($t_{\text{задания}} > t_{\text{цп}}$). Получить процент процессорного времени по каждому заданию, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям.

Вариант 7

Даны сведения об использовании машинного времени на ВЦ кафедрами института. Структура записи: код кафедры (5 символов), название кафедры (20 символов), количество сотрудников, расход машинного времени: по плану, фактически. Определить отклонения по расходу машинного времени по каждой кафедре, суммарные показатели планового и фактического расхода машинного времени в целом по институту и дать заключение: превышен или нет плановый фонд машинного времени. Результаты представить в абсолютных цифрах, а итоговую разность в абсолютном выражении (со знаком) и в процентах.

Вариант 8

Даны сведения за месяц о пропусках занятий студентами групп. Структура записи: шифр группы (6 символов), фамилия (15 символов), год рождения, пол (булевская переменная), пропущено часов, оправдано часов. Подсчитать количество неоправданных часов по каждому студенту, суммарные показатели по каждому виду, а также процент неоправданных пропусков в целом.

Вариант 9

Структура записей входного файла имеет следующий вид: табельный номер, Ф.И.О. (30 символов), год рождения, пол (булевская переменная), профессия (10 символов), стаж работы, разряд рабочего, номер цеха, номер участка, сумма заработной платы. Вид выходной таблицы представлен таблицей А.1.

Таблица А.1 – Вид выходной таблицы для варианта 9

Стаж работы	Количество рабочих по разрядам			Всего
	1	2	3	
До 6				
С 6 до 11				
С 11 до 16				
С 16 до 21				
С 21 до 25				
Свыше 25				

Вариант 10

Структура записей входного файла имеет следующий вид: название продукта, дата изготовления, срок годности, производитель, цена. Подсчитать количество просроченного товара и размер ущерба.

Вариант 11

Структура записей входного файла имеет следующий вид: номер зачетной книжки студента, ФИО студента, шифр группы (6 символов), дата рождения, год поступления, вступительный балл. Вывести фамилии 5 лучших студентов по годам поступления.

Вариант 12

Дан файл с информацией о результатах сданной сессии. Структура записей имеет следующий вид: номер зачетной книжки студента, ФИО студента, шифр группы (6 символов), пол, форма обучения, оценки за последнюю сессию (5 наименований) и даты сдачи экзаменов (5 наименований). Вывести фамилии студентов, получающих стипендию (если известна дата окончания сессии и балл, выше которого начисляется стипендия), подсчитать их количество на каждом курсе.

Вариант 13

Дан файл с информацией о потреблении продуктов животными зоопарка. Структура записей имеет следующий вид: номер вольера, кличка животного, порода, ареал обитания, тип продуктов, вес потребляемых продуктов, стоимость потребляемых продуктов, дата. Подсчитать сумму и вес употреблённых продуктов всеми видами пород животных (за указанный период).

Вариант 14

Дан файл с информацией о выданных книгах в библиотеке. Структура записей имеет следующий вид: ФИО абонента, автор книги, название книги, издательство, дата выдачи, стоимость книги. Выбрать ФИО и количество книг задолжников.

Вариант 15

Информация о продаже товаров подготовлена по следующему макету: номер магазина; номер секции; номер чека; наименование товара; артикул товара; цена товара; количество товара; дата продажи. Составить программу определения объема товарооборота по каждому магазину, т.е. общую сумму, вырученную от продажи всех товаров в данном магазине (данные содержат записи по нескольким магазинам). Разработать форму выходного документа.

Вариант 16

Дан файл с информацией о книгах в библиотеке. Структура записей имеет следующий вид: автор книги, название книги, издательство, жанр, дата поступления, стоимость книги. Выбрать 5 самых старых книг по всем жанрам.

Вариант 17

Структура записей входного файла имеет следующий вид: шифр группы (6 символов), шифр дисциплины (6 символов), количество отличных, хороших, удовлетворительных, неудовлетворительных оценок, пропущено лекций, пропущено практических занятий. Вид выходной таблицы представлен таблицей А.2.

Таблица А.2 – Вид выходной таблицы для варианта 17

Шифр группы	Средний балл	Количество оценок			
		Отл.	Хор.	Удовл.	Неуд.

Вариант 18

Дан файл с информацией о сотрудниках НИИ. Структура записи: номер отдела; табельный номер; фамилия; номер темы, над которой работает сотрудник; продолжительность работы над данной темой (в месяцах); код должности; размер заработной платы (в рублях). Определить перечень тем, разрабатываемых в НИИ, количество сотрудников, занятых в каждой теме, а также фонд заработной платы по каждой теме. Результаты напечатать в виде соответствующей таблицы.

Вариант 19

Дан файл с информацией о спортсменах различных команд. Структура записи: ФИО спортсмена, дата рождения, пол, вес спортсмена, разряд, наименование команды. Для каждой команды вывести 5 самых старших девушек и 5 самых тяжелых юношей.

Вариант 20

Дан файл с информацией о клиентах банка. Структура записей имеет следующий вид: ФИО клиента, дата рождения, пол, адрес проживания (улица, дом, квартира). Выберите 5 улиц, на которых проживают больше всего клиентов банка, и укажите средний возраст этих клиентов.

Вариант 21

Дан файл с информацией о студентах, проживающих в общежитии. Структура записей имеет следующий вид: номер зачетной книжки, ФИО студента, направление, группа, № общежития, № комнаты, долги за общежития. Выбрать 5 комнат с максимальным долгом проживающих.

Вариант 22

Дан файл с информацией о результатах сданной сессии. Структура записей имеет следующий вид: номер зачетной книжки студента, ФИО студента, шифр группы (6 символов), пол, форма обучения, оценки за последнюю сессию (5 наименований) и даты сдачи экзаменов (5 наименований). Вид выходной таблицы представлен таблицей А.2.

Таблица А.2 – Вид выходной таблицы для варианта 22

Средний балл	Количество студентов		Всего
	Мужской	Женский	
Меньше 60			
от 60 до 63			
от 64 до 73			
от 74 до 81			
от 81 до 89			
от 90 до 100			

Вариант 23

Структура записей входного файла имеет следующий вид: шифр группы (6 символов), номер зачетной книжки студента, ФИО студента, шифр группы (6 символов), пол, форма обучения, дата рождения, дата поступления, балл ЕГЭ. Выбрать в каждой группе по 5 девушек и юношей, имеющих высший балл.

Вариант 24

Структура записей входного файла имеет следующий вид: шифр группы (6 символов), номер зачетной книжки студента, ФИО студента, шифр группы (6 символов), пол, форма обучения, дата рождения, дата поступления, балл ЕГЭ. Выбрать в каждой группе самых старших 5 девушек и 5 юношей.

Вариант 25

Дан файл с информацией о студентах, проживающих в общежитии. Структура записей имеет следующий вид: номер зачетной книжки, ФИО студента, направление, группа, № общежития, № комнаты, долги за общежития. Для каждой группы подсчитать количество студентов, проживающих в общежитии и их суммарный долг.

ПРИЛОЖЕНИЕ Б
(СПРАВОЧНОЕ)
ПЕРЕЧЕНЬ СТАНДАРТОВ, ВХОДЯЩИХ В ЕСПД

- ГОСТ 19.002-80. ЕСПД. Схемы алгоритмов и программ. Правила выполнения
- ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам;
- ГОСТ Р 7.0.5-2008. СИБИД. Библиографическая ссылка. Общие требования и правила составления.
- ГОСТ Р 7.0.11-2011. СИБИД. Диссертация и автореферат диссертации. Структура и правила оформления;
- ГОСТ 7.32-2001. СИБИД. Отчет о научно-исследовательской работе. Структура и правила оформления;
- ГОСТ 2.051-2006 ЕСКД. Электронные документы. Общие требования;
- ГОСТ 2.111-68 ЕСКД. Нормоконтроль.

ПРИЛОЖЕНИЕ В
ОФОРМЛЕНИЕ ТИТУЛЬНОГО ЛИСТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и управления в технических системах
(полное название института)

кафедра «Информационные системы»
(полное название кафедры)

Пояснительная записка

к курсовой работе

на тему ПРОГРАММА РАСЧЕТА ВЫРУЧКИ ЗА ПРОДАННЫЙ ТОВАР

по дисциплине Информатика и программирование

Выполнил: студент II курса, группы: ИС/б-18-1-о

Направления подготовки (специальности) 09.03.02

Информационные системы и технологии

(код и наименование направления подготовки (специальности))

профиль (специализация) _____

Иванов Иван Иванович

(фамилия, имя, отчество студента)

Дата допуска к защите « _____ » _____ 20 18 г.

Руководитель _____

(подпись)

(инициалы, фамилия)

20 18 г.

ПРИЛОЖЕНИЕ Г

БЛАНК ЗАДАНИЯ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных технологий и управления в технических системах
Кафедра «Информационные системы»
Направление подготовки/специальность _____

Направленность/профиль/специализация _____

Курс _____ Группа _____ Семестр _____

З А Д А Н И Е

на курсовой проект (работу) студента

Иванова Ивана Ивановича

(фамилия, имя, отчество)

1. Тема работы (проекта) Программа расчета выручки за проданный товар
Вариант 14

руководитель работы (проекта) _____
(фамилия, имя, отчество, степень, звание, должность)

2. Срок сдачи студентом работы (проекта) _____

3. Входные данные к работе (проекту) _____

1. Язык программирования – C/ C++ (Borland Pascal)

2. Операционная система – Windows

3. Среда программирования – Borland C++ (Borland Pascal)

4. Класс вычислительной машины – персональная настольная ЭВМ

5. Меню-ориентированный интерфейс (использовать conio.h или модуль Crt)

6. Исходные данные – файл с записями о проданных товарах

7. Выходные данные – файл с записями о проданных товарах, сумма выручки

4. Содержание пояснительной записки (перечень вопросов, которые нужно разработать) _____
техническое задание, аннотация, содержание, введение, назначение и область применения
программы, технические характеристики программы, выполнение программы, выводы,
перечень ссылок, приложение (текст программы).

5. Перечень графического материала (с точным указанием обязательных чертежей) _____
презентация

6. Дата выдачи задания 01.09.2018

КАЛЕНДАРНЫЙ ПЛАН

№ п/п	Название этапов работы (проекта)	Срок выполнения этапов работы (проекта)	При- меча- ние
1	<i>Постановка задачи</i>	<i>01.09.18 – 07.09.18</i>	<i>2-ая неделя</i>
2	<i>Выбор модели и метода решения задачи</i>	<i>01.09.18 – 07.09.18</i>	<i>2-ая неделя</i>
3	<i>Разработка структур данных программы</i>	<i>01.09.18 – 21.09.18</i>	<i>2-4 неделя</i>
4	<i>Нисходящее проектирование и модульное программирование</i>	<i>21.09.18 – 12.10.18</i>	<i>5-8 неделя</i>
5	<i>Структурное программирование (кодирование)</i>	<i>12.10.18 – 16.11.18</i>	<i>8-13 неделя</i>
6	<i>Нисходящее тестирование и отладка программы</i>	<i>12.10.18 – 16.11.18</i>	<i>8-13 неделя</i>
7	<i>Разработка программных документов</i>	<i>16.11.18 – 30.11.18</i>	<i>13-15 неделя</i>
8	<i>Защита проекта</i>	<i>30.11.18 – 07.12.18</i>	<i>15-16 неделя</i>

Студент

(подпись)

(фамилия и инициалы)

Руководитель работы (проекта)

(подпись)

(фамилия и инициалы)

ПРИЛОЖЕНИЕ Д

(справочное)

КОНСОЛЬНЫЕ ФУНКЦИИ

В таблице Д.1. приведены функции для работы с дисплеем и клавиатурой (консолью), определенные в головном файле `conio.h` среды программирования Borland C++. Среда программирования Borland Pascal предоставляет практически такие же процедуры и функции при подключении модуля `Crt`.

Таблица Д.1 — Консольные функции

Шаблон функции	Назначение функции
<code>cgets(char *str);</code>	Чтение строки с консоли
<code>clreol(void);</code>	Удаление части строки от текущей позиции курсора до правой границы окна
<code>clrscr(void);</code>	Очистка экрана или окна
<code>cprintf (const char *format [arg, ...]); .</code>	Вывод строки в текстовое окно в соответствии с форматом
<code>cputs(const char *str);</code>	Вывод строки в текстовое окно
<code>cscanf (char *format [, address, ...]);</code>	Чтение данных с использованием форматного преобразования
<code>delline(void);</code>	Удаление текущей строки в текстовом окне
<code>getch(void);</code>	Считывание символа с консоли без эхо-печати
<code>getche(void);</code>	Считывание символа с консоли с эхо-печатью
<code>gettext(int left, int top, int right, int bottom, void*bufer);</code>	Копирование части текста с экрана в заданный буфер
<code>gettextinfo(struct text_info *r);</code>	Получение информации о текстовом режиме.
<code>gotoxy(int x, int y);</code>	Переместить курсор в заданную позицию на экране или окне
<code>highvideo(void);</code>	Установить высокую яркость пикселей
<code>kbhit(void);</code>	Проверить нажатие кнопки
<code>lowvideo(void);</code>	Установить низкую яркость пикселей
<code>movetext(int left, int top, int right, int bottom, int destleft, int desttop);</code>	Копирует текст из прямоугольной области экрана в другую прямоугольную область
<code>normvideo(void);</code>	Установить стандартную яркость пикселей
<code>putch(int ch);</code>	Вывести символ в текстовое окно
<code>puttext(int left, int top, int right, int bottom, void*bufer);</code>	Вывести текст из буфера на экран
<code>textattr(int newattr);</code>	Установить цвет текста и фона
<code>textbackground(int newcolor);</code>	Установить цвет фона для выводимого текста
<code>textcolor(int newcolor);</code>	Установить цвет текста, который выводится
<code>textmode(int newmode);</code>	Установить текстовый режим
<code>ungetch(int ch);</code>	Возвращает символ, введенный с клавиатуры
<code>wherex(void);</code>	Возвращает текущую координату X
<code>wherey(void);</code>	Возвращает текущую координату Y
<code>window(int left, int top, int right, int bottom);</code>	Установить координаты для текущего текстового окна

ПРИЛОЖЕНИЕ Е (ОБЯЗАТЕЛЬНОЕ) ТЕКСТ ПРОГРАММЫ

/*++++++

Севастопольский государственный университет
Кафедра «Информационные системы»

Программа для работы с базой моделей сотовых телефонов
Текст программы

РАЗРАБОТАЛ
Студент гр. ИС/б-11-о
Паскаль Б.

2018

++++++

Программа работает с базой данных сотовых телефонов, которая считывается из текстового файла. Каждая строка файла содержит запись об одной модели сотового телефона, для которой указывается название (20 символов), вес с точностью до сотых долей грамма, цена.

Основные функции программы:

- вывод базы на экран;
- добавление записи о телефоне в базу;
- исправление записи о телефоне в базе;
- удаление записи о телефоне из базы;
- поиск записи о телефоне в базе;
- выбор записей о моделях телефонов, попадающих в заданный пользователем диапазон цен.

Вариант задания 41. Утверждено 01.09.2018
Среда программирования Borland C++ version 5.02
Дата последней коррекции: 30.05.2018 .
Версия 1.0

++++++*/

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
```

```
//-----константы-----
const int N_ITEMS=7;           // количество пунктов меню
const int L_MODEL=20;          // длина названия телефона
const int step=18;             // количество строк на странице
```

```
//-----типы и глобальные переменные-----
struct tel {                   //структура для хранения данных о телефоне
    char model[L_MODEL];       //название телефона
    float weight, price;       //вес и цена телефона
};
```

```
struct elist{                  //элемент 2- направленного списка
    tel data;                  //данные о телефоне
    elist *left;               //указатель на элемент слева
    elist *right;              //указатель на элемент справа
};
```

```

};

unsigned int defaultmode,           //текущий режим экрана
               activecolor,         //цвет активного пункта меню
               inactivecolor;       //цвет неактивного пункта меню

int key;                            //код клавиши
unsigned int item;                  //номер пункта меню
unsigned int prev;                  //номер предыдущего пункта меню
elist *beg,                         //указатель на первый эл-т списка
        *fin,                       //указатель на последний эл-т списка
        *p;                         //указатель на текущий эл-т списка
char name[L_MODEL];                //название модели телефона
tel telefon;                       //данные телефона

//-----прототипы функций-----
void add(elist* &beg,elist* &fin,tel telefon); //добавление элемента в список
void error(char *message);           //вывод сообщения ошибки
int readfile( elist* &beg, elist* &fin);      //ввод базы из файла
int showbase(elist *beg);            //вывод базы на экран
void drawitem(unsigned int item, unsigned int color); //вывод пункта меню
void initmenu(unsigned int activecolor, unsigned int inactivecolor); //инициализация меню
void clear(void);                   //очистка текстового окна
void stub(unsigned int item);        //заглушка
void dlgwindow(void);               //отображение окна диалога
void message(char *message);         //вывод сообщения
tel query();                        //запрос сведений о телефоне
void edit(elist* beg, tel telefon);  //исправление записи о телефоне
elist * find(elist *p, char *s);     //поиск эл-та в списке по названию
void queryname(char *s);             //запрос названия модели телефона
void del(elist* &beg,elist* &fin,elist *p);  //удаление эл-та из списка
void info(tel telefon);              //вывод сведений о телефоне
void select(elist *beg);              //выбор телефонов в заданном диапа-
зоне цен
void queryprice(float &startprice, float &endprice); //ввод критериев отбора

//=====Главная функция=====
int main() {
defaultmode=LASTMODE;              //запомнить текущий текстовый режим экрана
textmode(C80);                     //установить режим 80x25 цветной

beg=0; fin=0;                      //обнулить указатели на начало и конец списка
readfile(beg, fin);                //ввести базу из файла в список

clrscr();                          //очистить экран
activecolor=LIGHTGREEN;            //цвет активного пункта меню - светло-зеленый
inactivecolor=GREEN;               //цвет неактивного пункта меню - зеленый
initmenu(activecolor,inactivecolor); //инициализация меню

item=0; prev=0;                    //обнулить индексы тек. и пред. пунктов меню

/* ---- цикл выбора пункта меню ---- */
while (1) { /**1**/
    key=getch();                    //ввести код нажатой клавиши
    if (key==0) key=getch();        //если введен расширенный код
    switch (key) { /**2**/
        /* ----выполнение пункта меню----*/
        case 13:                    // нажата Enter

```

```

switch (item) { /**3**/
    case 0: showbase(beg);
        break;
    case 1:
        в базу
        telefon=query();
        add(beg,fin, telefon);
        break;
    case 2:
        записи
        telefon=query();
        edit(beg, telefon);
        break;
    case 3:
        queryname(name);
        p=find(beg,name);
        if (p) del(beg,fin,p);
        break;
    case 4:
        queryname(name);
        p=find(beg,name);
        if (p) info(p->data);
        break;
    case 5:
        select(beg);
        item=5;
        break;
    default:
        textmode(defaultmode);
        exit(1);
}; /**3**/
break;
/*----перемещение курсора влево----*/
case 15: case 75:
    prev=item;
    if (item==0)
        item=N_ITEMS-1;
    else item--;
    break;
/*----перемещение курсора право----*/
case 9: case 77:
    prev=item;
    if (item==N_ITEMS-1)
        item=0;
    else item++;
    break;
} /**2**/
clear();
drawitem(prev, inactivecolor);
drawitem(item, activecolor);
} /**1**/
}

//-----Вывод пункта меню-----
void drawitem(unsigned int item, unsigned int color) {
    const d=12;
    const char * items[N_ITEMS]=

```

//проверка номера пункта меню
 //вывод базы на экран
 //добавление записи
 //редактирование
 //удаление записи
 //поиск модели в базе
 //выборка из базы по цене
 // выход из программы
 // нажаты Shift+Tab или Left
 //нажаты Tab или Right
 //вывод предыдущего пункта
 //вывод выбранного пункта
 // длина поля пункта меню
 //названия пунктов

```

    {"Просмотр","Добавить","Исправить","Удалить","Найти","Выбрать","Выход"};
const int pos[N_ITEMS]={1,d,2*d,3*d,4*d,5*d,6*d};           //позиции начала полей пунктов
window(1,1,80,2);                                           //окно для отображения пунктов
textbackground(LIGHTGRAY);                                  //цвет фона в окне светло-серый
textcolor(color);                                           //цвет букв пункта
gotoxy(pos[item],1);                                       //установить курсор в начало пункта
cprintf("%s",items[item]);                                  //отобразить название пункта
}

//-----Инициализация меню-----
void initmenu(unsigned int activecolor, unsigned int inactivecolor){
    unsigned int item;                                     //номер пункта меню
    window(1,1,80,2);                                     //окно из двух строк для вывода пунктов
    textbackground(LIGHTGRAY);                             //цвет фона в окне светло-серый
    clrscr();

    /*----отобразить названия пунктов меню----*/
    drawitem(0,activecolor);                               //активный цветом «Просмотр»
    for (item=1;item<N_ITEMS;item++)
        drawitem(item,inactivecolor);                     //все остальные неактивным

    /* ----отобразить границу окна меню, подчеркнув его снизу----*/
    gotoxy(1,2);
    textcolor(inactivecolor);
    cputs("_____");

    gotoxy(1,1);                                           //курсор на «Просмотр»
}

//-----Очистка рабочего окна -----
void clear() {
    window(1,3,80,25);
    textattr(WHITE);
    clrscr();
}

// -----Заглушка-----
void stub(unsigned int item) {
    window(1,4,80,24);
    textbackground(LIGHTGRAY);
    textcolor(WHITE);
    clrscr();
    cprintf("Вызван пункт меню %d ",item);
    getch();
}

//-----Вывод сообщения ошибки-----
void error(char message[]) {
    window(10,6,70,12);
    textbackground(LIGHTGRAY);
    textcolor(RED);
    clrscr();
    gotoxy(15,4);
    cputs(message);
    getch();
    textmode(defaultmode);
}

```

```

//----- Отображение окна диалога-----
void dlgwindow(){
    window(10,6,70,12);
    textattr(GREEN+LIGHTGRAY*16);
    clrscr();
}
//----- Вывод сообщения-----
void message(char message[]) {
    dlgwindow();
    gotoxy(20,4);
    cputs(message);
    getch();
}

//----- Добавление элемента в конец списка-----
void add(elist* &beg,elist* &fin,tel telefon){
    elist *p;                                //указатель на создаваемый элемент
    p=new elist;                             //создание нового элемента
    p->data=telefon;                          //заполнение его информационной части
    p->right=0;                              //заполнение его указателей
    p->left=fin;
    if (!beg) beg=p;                        //если список был пустым
    else fin->right=p;                       //иначе привязка нового эл-та к последнему
    fin=p;                                  //обновление указателя на конец списка
}

//----- Ввод базы из файла-----
int readfile( elist* &beg, elist* &fin){
    FILE *f;
    tel telefon;

    /*-----проверка открытия файла base.txt ----*/
    f=fopen("c:\\bc5\\work\\base.txt","r");
    if(!f) {
        error("Файл base.txt не найден");
        exit(1);
    }

    /*-----чтение записей из файла-----*/
    while (!feof(f)) {
        fgets(telefon.model,L_MODEL,f);      //чтение названия модели
        fscanf(f,"%f%f\n",&telefon.weight,&telefon.price); //чтение веса и цены
        add(beg,fin, telefon);               //добавление в список
    }
    fclose(f);
    return 0;
}

//----- Вывод базы на экран-----
int showbase(elist *beg) {
    int i;                                //счетчик кол-ва строк
    elist *p,                             //указатель на текущий элемент страницы
        *pn;                              //указатель на 1-ый эл-т страницы
    int key;                              //код нажатой клавиши
    elist * showpage(elist *p);           //прототип функции вывода одной страницы

```

```

if (!beg) {                                     //если список пустой,
    message("Список пустой");                     //то вывести сообщение
    return -1;
}
/*----задать параметры рабочего окна----*/
window(1,4,80,24);
textbackground(LIGHTGRAY);
textcolor(WHITE);

p=beg;                                           //указатель установить в начало списка
while (1) {                                     /**1**/
    pn=p;                                       //запомнить указатель первого эл-та страницы

    /*---вывести страницу и получить новое значение указателя текущего элемента---*/
    p=showpage(p);

    /*----определить код нажатой клавиши----*/
    key=getch();
    if (key==0) key=getch();                     //если расширенный код

    /*----управление прокруткой-----*/
    switch (key) { /**2**/
        case 27:                               //нажата Esc
            return 0;                           //выход из просмотра
        case 13: case 80: case 81:              //если нажаты Enter||Down||PgDn
            if (!p) p=pn;                       //и список исчерпан, то
            break;                              //отображать ту же страницу
        case 72: case 73:                      //если нажаты Up или PgUp,
            p=pn;                               //то указатель на начало стр.
            for(i=1;(i<=step)&&(p);i++)          //сместить на step записей вверх
                p=p->left;                      //ограничение для 1-ой записи
            if (!p) p=beg;
            break;
    } /**2**/
} /**1**/
}

//-----Вывод одной страницы-----
elist* showpage(elist *p){
    int i;                                     //номер строки вывода
    clrscr();
    /*---вывести в первую строку окна заголовки таблицы----*/
    gotoxy(1,1);
    cprintf("Модель      Вес  Цена");

    /*---начиная со второй строки окна выводить элементы списка---*/
    gotoxy(1,2);
    i=2;
    while (p) {
        cprintf("%-22.20s %5.2f %10.2f",p->data.model,p->data.weight,p->data.price);
        p=p->right;
        i++;
        gotoxy(1,i);
        /*----выйти из функции, если выведена одна страница----*/
        if (i>step) return p;                   //и вернуть указатель текущего эл-та списка
    }
    return p;                                   //выход при достижении конца списка
}

```

```

}

//-----Запрос сведений о телефоне-----
tel query(){
char s[L_MODEL];           //название модели телефона
tel telefon;               //данные о телефоне
int i;                     //индекс строки
int len;                   //длина строки с названием модели
dlgwindow();               //отобразить окно диалога

/*---ввод названия модели телефона---*/
do {
    gotoxy(2,1);
    cputs("Модель?");
    gotoxy(2,2); clreol();
    cscanf("%s",s);
    len=strlen(s);
    for (i=len; i<L_MODEL-1; i++)
        s[i]=' ';           //дополнение строки пробелами
    s[L_MODEL-1]='\0'; }
while (!len);
strcpy(telefon.model,s);

/*---ввод значения веса телефона---*/
do {
    gotoxy(2,3);
    cputs("Вес?");
    gotoxy(2,4); clreol();
    cscanf("%s",s);
    telefon.weight=atof(s);}
while (!telefon.weight);

/*---ввод значения цены телефона---*/
do {
    gotoxy(2,5);
    cputs("Цена?");
    gotoxy(2,6); clreol();
    cscanf("%s",s);
    telefon.price=atof(s); }
while (!telefon.price);

getch();                   //очистка буфера ввода
return telefon;
}

// ----- Исправление записи о телефоне -----
void edit(elist* beg, tel telefon) {
elist *p;
p=find(beg,telefon.model);
if (p) {
    p->data.weight=telefon.weight;
    p->data.price=telefon.price;
}
}

```



```

// -----Поиск эл-та в списке (по названию модели)-----
elist * find(elist *p, char *s){

/*---цикл по элементам списка ----*/
while (p) {
    if (strcmp(s,p->data.model)==0)                //если модель найдена,
                                                    //то вернуть указатель
                                                    //переход к след. эл-ту списка
        return p;
    p=p->right;
}

/*---если модель не найдена----*/
gotoxy(2,6);
message("Такой модели в списке нет");
p=0;
return p;
}

// -----Запрос названия модели телефона-----
void queryname(char *name){
int i;                //индекс строки
int len;              //длина строки с названием модели
char s[L_MODEL];      //название модели телефона
dlgwindow();          //отобразить окно диалога

/*---ввод названия модели---*/
do {
    gotoxy(2,2);
    cputs("Модель?");
    gotoxy(2,3); clrcl();
    cscanf("%s",s);
    len=strlen(s);
    for (i=len; i<L_MODEL-1; i++)
        s[i]=' ';                //дополнение строки пробелами
    s[L_MODEL-1]='\0'; }
while (!len);
strcpy(name,s);
getch();                //очистка буфера ввода
}

//-----Удаление эл-та из списка-----
void del(elist* &beg,elist* &fin,elist *p){
if ((p==beg)&&(p==fin))                //удаление единственного элемента
    { beg=0; fin=0;}
else
if (p==beg)                //удаление из начала списка
    {beg=beg->right; beg->left=0;}
else
if (p==fin)                //удаление с конца
    {fin=fin->left; fin->right=0;}
else {                //удаление из середины
    p->left->right=p->right;            //связать предыдущий со следующим
    p->right->left=p->left;            //связать следующий с предыдущим
}
free(p);                //освобождение памяти из под элемента
}

```

```

//-----Вывод сведений о телефоне-----
void info(tel telefon){
    dlgwindow();                                //отобразить окно диалога
    gotoxy(2,2); printf("Модель %s",telefon.model);
    gotoxy(2,4); printf("Вес %5.2f",telefon.weight);
    gotoxy(2,6); printf("Цена %6.2f",telefon.price);
    getch();                                    //ожидать нажатия любой клавиши
}

//----- Выбор из базы моделей телефонов в заданном диапазоне цен -----
void select(elist *beg) {
    float startprice,                            //начальная цена
          endprice;                              //конечная цена
    elist *begs,*fins;                          //указатели на первый и последний и
                                                // эл-ты формируемого списка
    elist *p;                                    //текущий эл-т исходного списка

    queryprice(startprice,endprice);            //ввод начальной и конечной цены

    /*---- инициализация указателей----*/
    begs=0;
    fins=0;
    p=beg;

    /*----цикл по элементам исходного списка и формирование нового списка----*/
    while (p) {
        if ((p->data.price>=startprice)&& (p->data.price<=endprice)) //если цена в треб. диапазоне цен
            add(begs,fins,p->data);                                //то добавить эл-т в новый список
        p=p->right;                                                //переход к следующему элементу
    }
    showbase(begs);                                                //вывод выборки (списка) на экран
}

//-----Ввод критериев отбора-----
void queryprice(float &startprice, float &endprice) {
    char *s=" ";
    dlgwindow();
    do {
        /*----ввод начальной цены----*/
        do {
            gotoxy(2,2);
            printf("Минимум: ");
            gotoxy(2,3);
            clreol();
            cscanf("%s",s);
            startprice=atof(s);
        }
        while (!startprice);

        /*----ввод конечной цены----*/
        do {
            gotoxy(2,4);
            printf("Максимум: ");
            gotoxy(2,5);
            clreol();

```

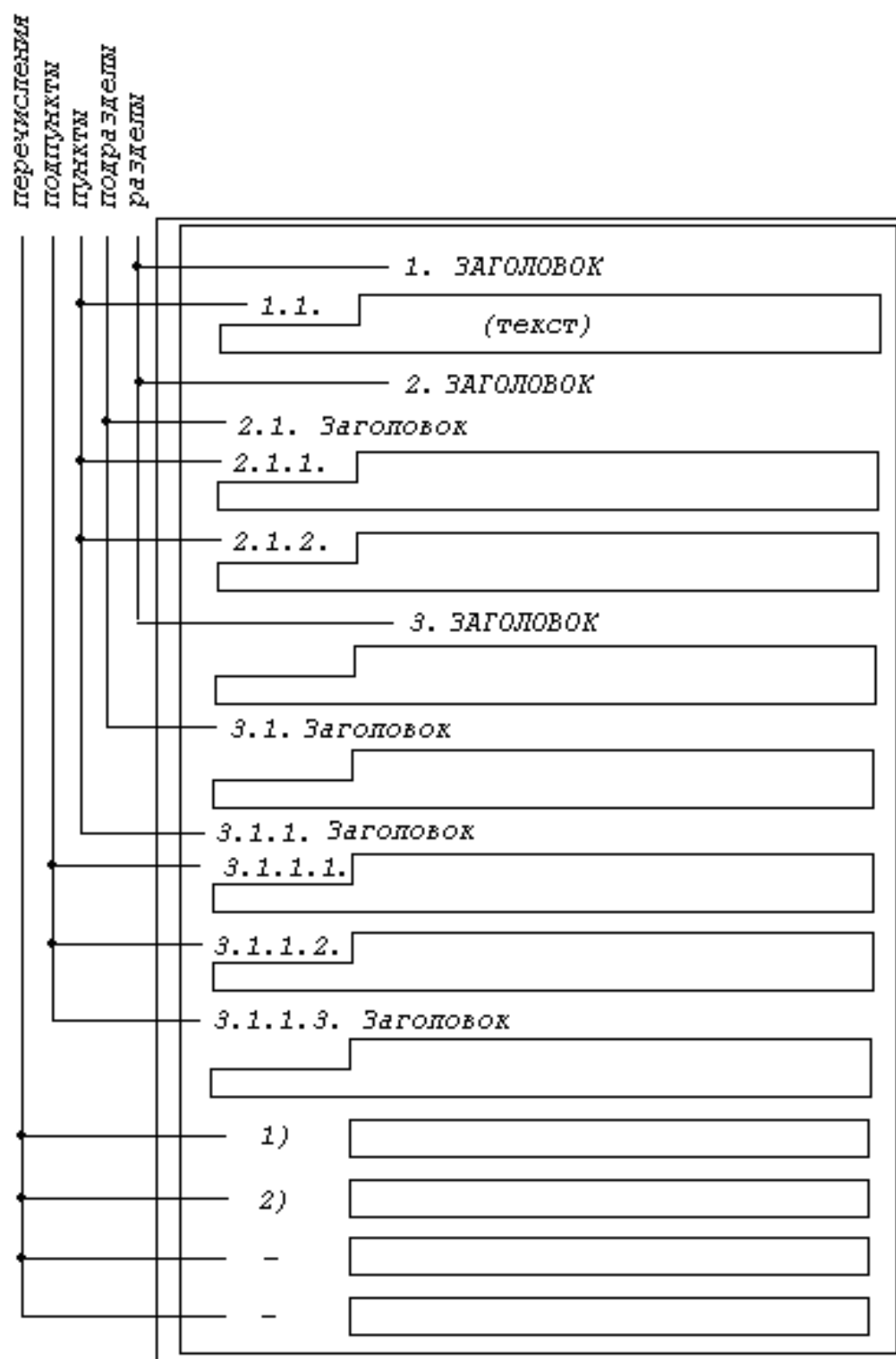
```
        cscanf("%s",s);
        endprice=atof(s); }
    while (!endprice);
}

while (startprice>endprice);
getch();
}
```

//преобразовать строку в вещ. число

//повторять ввод пока startprice>endprice
//очистка буфера ввода

ПРИЛОЖЕНИЕ Ж **(СПРАВОЧНОЕ)** **СТРУКТУРА ТЕКСТА ПРОГРАММНОГО ДОКУМЕНТА**



ПРИЛОЖЕНИЕ 3 **(СПРАВОЧНОЕ)** **БИБЛИОГРАФИЧЕСКОЕ ОПИСАНИЕ ИСТОЧНИКОВ** **ИНФОРМАЦИИ**

Библиографическое описание документов выполняется в соответствии с недавно введенным стандартом ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления» (ГОСТ 7.1-2003, IDT). Ранее действующие ГОСТы на библиографические описания отменены (ГОСТ 7.1-84, ГОСТ 7.16-79, ГОСТ 7.18-79, ГОСТ 7.34-81).

Сокращения слов при описании должно выполняться в соответствии со следующими стандартами:

- ГОСТ 7.12.93 «Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила»;
- ГОСТ 7.11–78 «Сокращения слов и словосочетаний на иностранных европейских языках в библиографическом описании».

Библиографическое описание литературных источников предусматривает наличие таких областей описания: область названия и сведений об ответственности, область издания, область специфических сведений, область физической характеристики, область серии, область примечаний, область стандартного номера и условий доступности. Область специфических сведений используется только при описании особых видов документов, например, электронных ресурсов, отдельных видов нормативных и технических документов, когда требуется в описании указать сведения об особенностях представления информации. Каждая из областей содержит обязательные и факультативные элементы. Обязательные элементы содержат библиографические сведения, обеспечивающие идентификацию документа, и их приводят в любом описании.

В соответствии со стандартом пунктуация в библиографическом описании выполняет две функции: обычных грамматических разделительных знаков и разделительных знаков, которые применяются для распознавания областей и элементов описания (знаки предписанной пунктуации).

Области описаний отделяют одну от другой точкой и тире. Для идентификации в описании знаков предписанной пунктуации используют пробелы, которые помещают слева и справа от такого знака. Исключением являются точка и запятая — пробел в этом случае оставляют только справа.

Первые слова сведений, которые относятся к сведениям об ответственности, записываются со строчной буквы, если они не являются именами собственными.

Сведения, заимствованные из других источников или выявленные в ходе аналитического анализа объекта описания, заключают в квадратные скобки.

В соответствии со старыми нормативами авторов, которые были указаны в заголовке библиографического описания, не указывали в области сведений об ответственности. Теперь это правило изменено. При этом в заголовке библиографического описания указывают фамилию автора, а потом имя (возможно отчество), а в области ответственности сначала указывают инициалы, а затем фамилию.

Простейшее описание библиографического источника будет содержать следующие области и элементы:

- сведения об авторе, которые приводятся в начале описания, если авторов не более трех;
- заглавие и дополнительные сведения (например, название серии), которые приводятся после двоеточия со строчной буквы;
- сведения об ответственности, отделяемые от заглавия знаком « / ». Это могут быть автор (авторы), ответственное лицо (составитель, редактор, переводчик и т.п.) или организация (фирма);
- если описываемый источник является составной частью другого источника (напр., статьей из журнала), то описание группового источника отделяется знаком « // »;
- выходные данные отделяются знаками « . — » и содержат сведения о месте издания, издательстве, годе издания и числе страниц.

Примеры описания источников различных видов приведены ниже в таблицах 3.1 — 3.3. Дополнительно в качестве примера описания источников можно использовать библиографический список к настоящим методическим указаниям.

Таблица 3.1 — Описание книг

Пример описания	Примечание
Вирт Н. Алгоритмы и структуры данных / Н. Вирт ; пер. с англ. — М. : Мир, 1989. — 360 с.	<i>Один автор</i>
Бондарев В. Н. Искусственный интеллект : учеб. пособие для студ. вузов / В. Н. Бондарев, Ф. Г. Аде. — Севастополь : Изд-во СевНТУ, 2002. — 615 с.	<i>Два, три автора</i>
Бондарев В. Н. Цифровая обработка сигналов: методы и средства : учеб. пособие для студ. вузов / В. Н. Бондарев, Г. Трёстер, В.С. Чернега. — 2-е изд. — Харьков : Конус, 2001. — 398 с.	
Основы создания гибких автоматизированных производств [Текст] / Л. А. Пономаренко, Л. В. Адамович, В. Т. Музычук, А. Е. Гридасов ; ред. Б. Б. Тимофеева. — К. : Техника, 1986. — 144 с.	<i>Четыре автора</i>
Психология менеджмента / П. К. Власов, А. В. Липницкий, И. М. Луцихина и др. ; под ред. Г. С. Никифорова. — [3-е изд.]. — Х. : Гуманитар. центр, 2007. — 510 с. : ил.,табл. — Библиогр. : с. 504—510. — ISBN 966-83243-34-X.	<i>Пять и более авторов</i>
Справочник конструктора РЭА : общие принципы конструирования / Под ред. Р. Г.Варламова. — М. : Сов. радио, 1980. — 480 с.	<i>Без автора</i>
Савельев И. В. Курс общей физики : в 4 ч. Ч.1. Механика / И. В. Савельев. — М. : Изд-во МГУ, 1968. — 568 с.	<i>Том многотомного издания</i>
Макс Ж. Методы и техника обработки сигналов при физических измерениях : в 2 т. / Жорж Макс ; пер. с франц. — М. : Мир, 1983. — Т.1. — 312 с.— Т.2. — 256 с.	<i>Многотомное издание</i>

Таблица 3.2 — Описание статей из сборников и журналов, материалы доклада

Пример описания	Примечание
Бондарев В. Н. Исследование частотно-импульсного метода определения спектральных коэффициентов / В. Н. Бондарев // 36. наук. пр. Севастопольского ВМІ им. П. С. Нахімова. — Севастополь, 2006. — Вип. 1(9). — С. 121–125.	<i>Статья из сборника</i>
Беляев Б. Г. Поле в области Френеля / Б. Г. Беляев // Изв. Вузов : радиоэлектроника. — 1985. — Т.28. — №9. — С. 8–13.	<i>Статья из журнала</i>
Akaida H. Block matrix inversion / H. Akaida // SIAM Journal. — 1973. — Vol.21. — No 3. — P. 234–241.	<i>Статья из журнала</i>
Синтез реактивной нагруженной антенны / А. Ф. Чаплин, Д. М. Сазонов, Б. Г. Беляев, Б. А. Мишустин // Радиотехника. — 1979. — Т.34. — №3. — С. 39–43.	<i>Статья из журнала, более трех авторов</i>
Бондарев В. Н. Анализ независимых компонент с помощью самоорганизующихся нейронных сетей / В. Н. Бондарев // Информационные технологии и информационная безопасность в науке, технике и образовании «ИНФОТЕХ-2004» : матер. междунар. науч.-практ. конф., Севастополь, 20 – 25 сент. 2004г. — Киев–Севастополь : НТО РЭС Украины, 2004. — С. 98–99.	<i>Материалы доклада</i>

Таблица 3.3 — Описание стандартов и электронных ресурсов

Пример описания	Примечание
ГОСТ 7.1-2003. Библиографическое описание источников информации. — М. : Изд-во стандартов, 2003. — 76 с.	<i>Стандарт</i>
Swanson E. Editing ISBD (SR): approach, scope, definitions [Electronic resource] // 68th IFLA Council and General Conference, August 18-24, 2002: Proceedings. — Mode of access: WWW.URL: www.ifla.org/IV/ifla68/papers/148-162e.pdf . — Last access: 30.05.2013. — Title from the screen.	<i>Электронный удаленный ресурс</i>
Технологии информационного общества и культура [Электронный ресурс] : Международные конференции и проекты / Центр ПИК. — Электрон. дан. — М., 2004. — 1 CD-ROM. — Загл. с этикетки диска.	<i>Электронный локальный ресурс</i>

Заказ № _____ от « _____ » _____ 2018г. Тираж _____ экз.
Изд-во СевГУ