Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

"КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ"

**ПРОГРАММИРОВАНИЕ**

Учебно – методическое пособие по курсовой работе для студентов,

обучающихся в бакалавриате по направлениям подготовки

"Информатика и вычислительная техника",

"Прикладная информатика"

Калининград

Издательство ФГБОУ ВПО "КГТУ"

2015

УДК 681.3.06

УТВЕРЖДЕНО

Ректором ФГБОУ ВПО

"Калининградский

государственный технический

университет"

АВТОРЫ: Гудков А.Л., к.т.н., доцент кафедры систем управления и вычислительной техники ФГБОУ ВПО "Калининградский государственный технический университет";

Калинина С.А., доцент той же кафедры.

Учебно – методическое пособие рассмотрено и одобрено кафедрой систем управления и вычислительной техники ФГБОУ ВПО "Калининградский государственный технический университет " 03 июля 2015 г., протокол № 7.

Учебно – методическое пособие рекомендовано к печати методической комиссией факультета автоматизации производства и управления ФГБОУ ВПО "Калининградский государственный технический университет" \_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г., протокол № \_\_\_.

**РЕЦЕНЗЕНТ –** кафедра систем управления и вычислительной техники ФГБОУ ВПО "Калининградский государственный технический университет"

**Введение**

Дисциплина "Программирование" является первой из цикла дисциплин образовательной программы подготовки бакалавров в области информатики и вычислительной техники (прикладной информатики), освоение которых необходимо для формирования способности разрабатывать прикладное программное обеспечение вычислительных систем. Она ориентирована на получение знаний, умений и навыков по структурной процедурно-ориентированной технологии программирования с использованием системы программирования Турбо-Паскаль (Turbo-Pascal).

Учебная программа дисциплины, помимо лекционных занятий и лабораторного практикума, предусматривает выполнение студентами курсовой работы - индивидуального задания по разработке программы учетной задачи. Основная цель этой работы – закрепление, расширение и углубление знаний, полученных в теоретическом курсе, приобретение навыков программирования в условиях большей, чем в лабораторном практикуме, самостоятельности. Курсовая работа предполагает комплексное использование студентом знаний по технологии, приемам и средствам программирования.

Задание на курсовую работу выдается после успешного выполнения студентом лабораторного практикума по дисциплине [1, 2].

1. **Задание по курсовой работе**

Курсовая работа предполагает разработку программы (на языке Турбо-Паскаль) учетной задачи, при решении которой по указываемому пользователем запросу (определенным исходным данным) формируются (в результате обработки предварительно созданных на магнитном носителе информации файлов) и представляются в виде таблицы определенные сведения. Конкретная сущность задачи (вариант задания) определяется преподавателем – руководителем работы ("заказчиком программы"), после чего студент в качестве предварительного этапа работы оформляет индивидуальное задание по ней (форма задания приведена в приложении 1, примеры индивидуальных заданий – в приложении 2).

Результаты курсовой работы представляются в виде:

1) пояснительной записки, в которой описаны поэтапные и итоговые результаты программирования;

2) программы задачи, исполнение которой может демонстрироваться для разных исходных данных (запросах пользователя).

Задача для программирования в курсовой работе является "функциональной", т.е. решается при существующих файлах. Процедуры же создания, корректировки и печати файлов в задачу не "входят". Тем не менее, программы создания и печати тестовых файлов, которые должны использоваться для проверки правильности (тестирования) и демонстрации исполнения программы, должны быть разработаны и приведены в приложениях к основному тексту пояснительной записки.

Требования к программе, представляемой в качестве результата курсовой работы:

1) программа должна быть многомодульной, с подпрограммами;

2) в программе должна предусматриваться проверка запроса пользователя на допустимость (характер проверки может уточняться преподавателем – руководителем работы).

**2. Методические указания по выполнению курсовой работы**

2.1. **Для успешного выполнения курсовой работы** необходимы знания, умения и навыки по алгоритмизации и программированию на языке Турбо-Паскаль в объеме, достаточном для:

- обработки на компьютере числовых и символьных данных (в т.ч. структурных значений, состоящих из них);

- организации диалога между пользователем и компьютером;

- создания и обработки типизированных файлов на магнитном носителе информации,

а также знаний основ технологии структурного процедурно-ориентированного программирования ("нисходящей" разработки многомодульных Паскаль-программ с подпрограммами). Уточнение, расширение и углубление же знаний об указанной технологии программирования, приобретение навыков ее реализации осуществляется в ходе и в результате собственно выполнения курсовой работы.

2.2. **Порядок выполнения курсовой работы.** После получения и уточнения задания по программированию следует:

- описать постановку задачи, для которой разрабатывается программа;

- провести проектирование программы (разработать схему иерархии процедур задачи, функциональную и модульную структуру программы, алгоритм решения задачи на компьютере);

- подготовить тестовые исходные данные – файлы и запросы пользователя для различных вариантов решения задачи, т.е. спланировать тестирование программы;

- написать исходный текст программы (головного модуля с основной программой и модулей с внешними подпрограммами и общими для них нестандартными типами данных);

- провести тестирование и отладку программы.

Результаты каждого этапа курсовой работы (разработки программы) необходимо описать в пояснительной записке по ней.

2.3. **Описание постановки задачи** для программирования должно содержать:

- **характеристику задачи** (она определяет, для получения каких сведений, при каких исходных данных решается задача на компьютере);

- **описание исходных данных,** необходимых для решения задачи на компьютере (файлов и запроса пользователя);

**- описание результатов решения** задачи на компьютере (представляемых пользователю в виде таблицы сведений и возможных сообщений).

По сути описание постановки задачи это более подробное изложение сведений о задаче, сформулированных в задании по курсовой работе (см. "Макет пояснительной записки по курсовой работе", приведенный в приложении 3).

2.4. При **проектировании программы** (а это важнейший этап ее разработки) следует сначала произвести "**декомпозицию задачи"**  - выделение в ней (точнее – в процессе ее решения на компьютере) отдельных определенных процедур. Результатом этого станет **схема иерархии процедур задачи.** При этом процедуры, из которых состоит процесс решения задачи "в целом" (процедуры "первого уровня" декомпозиции), в свою очередь, могут детализироваться путем выделения в них других, более простых процедур (процедур "второго уровня") и т.д. Делается это для того, чтобы упростить алгоритмизацию и программирование задачи, свести эту работу к определению и описанию на языке программирования алгоритмов достаточно простых процедур.

В учетной задаче курсовой работы могут быть выделены процедуры (рис. 1):

- ввода и проверки запроса пользователя на допустимость (если запрос состоит из нескольких значений, для каждого значения должна предусматриваться "своя" процедура);

- формирования (определения) искомых по запросу сведений;

- вывода результатов решения задачи в виде таблицы.

Проверки значений, вводимых пользователем в качестве запроса на получение сведений, могут выделяться в отдельные процедуры в составе процедур ввода данных.

Формирование сведений включает в себя процедуры обработки файлов на магнитном носителе и сортировки полученных сведений (возможны и другие процедуры в соответствии с индивидуальным заданием – см. примеры проектирования программ). При этом порядок процедур обработки файлов в зависимости от варианта задачи может быть разным. Для одних вариантов сначала следует определять обработку второго файла, содержащего "оперативную" информацию по задаче, а затем – первого файла со справочными данными (так показано на рис. 1), для других – сначала определяется обработка первого файла и получается начальная "таблица" в виде массива записей, следом – второго для получения нужных в таблице (массиве) показателей. Есть варианты, для которых процедура обработки файла со справочной информацией должна выполняться для проверки правильности запроса.

Рис. 1. Схема иерархии процедур задачи (пример).

Рис. 2. Функциональная структура программы (пример).

Процедура вывода результатов решения задачи тоже может детализироваться – "разбиваться" на процедуру вывода заголовка с "шапкой" таблицы и процедуру вывода страницы табличной части (в курсовой работе эту детализацию можно не проводить).

В схеме иерархии процедур задачи на рис. 1 используются обобщенные наименования процедур. В курсовой же работе следует давать конкретные наименования выделяемых при декомпозиции задачи процедур в соответствии с их сущностью (например, "Ввод и проверка даты", "Определение объемов продаж", "Получение фамилий авторов" и т.п.).

Декомпозиция задачи не имеет единственного решения, следует рассмотреть различные возможные варианты ее и выбрать лучший для дальнейшего программирования.

2.5. Вторым этапом проектирования программы должна стать разработка **функциональной структуры программы**, в которой программа представляется как взаимосвязанная совокупность основной программы и подпрограмм. Она получается на основе схемы иерархии процедур задачи – определяется, что процесс решения задачи "в целом" должен предусматриваться основной программой (модуль Program), а выделенные в нем процедуры – соответствующими подпрограммами.

Однако, не обязательно для каждой процедуры, выделенной при декомпозиции задачи, предусматривать подпрограмму. Возможны достаточно простые процедуры, для которых подпрограммы нецелесообразны. Например, подпрограмма для процедуры "Формирование сведений", определенной в схеме иерархии процедур учетной задачи на рис. 1, необязательна, т.к. вызовы подпрограмм процедур, из которых она состоит, могут предусматриваться непосредственно в основной программе (рис. 2). Поэтому функциональная структура программы этой задачи (рис. 2) отличается от схемы иерархии ее процедур и определяет необходимость пяти подпрограмм, вызываемых из основной программы (FormR). К ним относятся:

- подпрограмма InA для ввода и проверки запроса пользователя;

- подпрограмма DetR для определения количественных показателей результата;

- подпрограмма GetR для получения справочных данных;

- подпрограмма OrdR для сортировки сведений;

- подпрограмма OutR для вывода таблицы со сведениями.

Для каждой подпрограммы необходимо определить список формальных параметров (аргументы и результаты подпрограмм, их типы). Например, для подпрограммы ввода и проверки запроса InA (в случае, если он представляет собой одно одиночное значение) результатами (параметрами-переменными) должны быть сам запрос и признак его правильности. А для подпрограммы определения количественных показателей в получаемых сведениях путем отработки файла с оперативной информацией DetR запрос является аргументом, результатами же должны быть массив нужных показателей (его элементы при выводе станут строками соответствующей таблицы) и количество элементов в нем. Этот же массив (массив записей) должен быть входным и "одновременно" выходным параметром подпрограммы получения справочных данных из первого файла GetRD, т.к. она дополняет формируемые сведения этими справочными данными (наименованиями, фамилиями и т.п.), а также подпрограммы сортировки сведений OrdR, которая "сводится" к упорядочиванию этого массива. Входными параметрами подпрограммы вывода таблицы с полученными сведениями OutR должен быть запрос (он представляется в заголовке таблицы) и сформированный массив результатов с количеством элементов в нем (его значения представляются в табличной части выводимых сведений).

2.6. **На основе функциональной структуры программы**, которая определяет, какие процедуры должны выполняться в процессе решения задачи (при каких исходных данных и с какими результатами), полезно (а в курсовой работе необходимо) определить **алгоритмические решения** и представить их графически в виде схем:

- обобщенного алгоритма задачи:

- алгоритмов отдельных процедур.

В обобщенном алгоритме (он должен описываться в основной программе) предусматриваются порядок и условия выполнения процедур "первого уровня" (вызовов подпрограмм из основной программы). В качестве схем, детализирующих алгоритмы отдельных процедур (они должны описываться в подпрограммах), достаточно представить алгоритмы ввода и проверки запроса, обработки файлов для получения требуемых результатов. Представлять алгоритмы сортировки (упорядочивания) сформированных сведений и вывода их не нужно.

При алгоритмизации задачи и ее процедур не следует излишне подробно определять инструкции по выполнению необходимых операций и использовать в схемах символику языка программирования, а уж тем более – рисовать схемы после написания текста программы, указывая в них ее операторы (такие "алгоритмические решения" рассматриваются как неумение разрабатывать алгоритмы обработки данных на этапе проектирования программы).

На рис. 3 представлена схема обобщенного алгоритма учетной задачи, соответствующая предыдущим проектным решениям (рис. 1, 2).

да

нет

нет

да

да

Ошиб-ка?

Вывод сообще-ния об ошибке

Н

InA

OutR

OrdR

GetRD

DetR

Данные есть?

Сорти-ровка?

Вывод "нет данных"

К

нет

Рис. 3. Схема алгоритма учетной задачи (пример).

2.7. **Модульная структура программы**  определяется путем распределения подпрограмм по модулям программы – головному модулю Program и дополнительным к нему модулям Unit. Если каждая подпрограмма размещается в отдельном модуле Unit, то модульная структура программы "повторяет" функциональную (с одним отличием – в ней рекомендуется предусматривать модуль с описанием нестандартных типов данных – записей, массивов, используемых в разных фрагментах программы). Но можно в одном модуле записывать несколько подпрограмм, функционально "близких". Например, подпрограммы сортировки и вывода результатов решения учетной задачи могут располагаться в одном модуле (рис. 4). Важно при этом, чтобы тексты модулей состояли из обозримого количества предложений (модули из большого числа предложений труднее отлаживать и модифицировать).

Следует отметить, что если в функциональной структуре программы показано, из каких программных единиц какие подпрограммы вызываются к исполнению, то в модульной структуре - в каких модулях какие модули используются (использование модуля это использование имен объектов – констант, типов данных, переменных, подпрограмм, описанных в этом модуле).

Головной модульProgram

InA

U1

DetR

U2

GetRD

U3

OrdR

OutR

U4

Type..

U5

FormR

|  |  |  |  |
| --- | --- | --- | --- |
| U1, U2, U3, U4 – | | модули Unit с подпрограммами; | |
| U5 – | модуль Unit с описанием нестандартных типов данных, используемых в головном модуле и модулях U2, U3, U4. | |

Рис. 4. Модульная структура учетной задачи (пример).

2.8. **Тестовые файлы** на магнитном носителе информации (например, диске) должны состоять из небольшого числа элементов, но значения элементов должны обеспечивать проверку правильности программы для различных вариантов решения задачи (получения упорядоченных сведений, получения таблицы с одной строкой и т.п.). **Варианты запросов** при тестировании программы также должны быть ориентированы на различные варианты решения задачи, в т.ч. быть и ошибочными, и такими, для которых сведения в тестовых файлах отсутствуют.

Важно, чтобы при тестировании программы тестовые исходные данные позволяли:

- классифицировать получаемые по программе результаты как правильные или неправильные;

- обеспечивали выполнение каждого оператора программы хотя бы один раз.

При тестировании и отладке многомодульной программы можно проверять безошибочность подпрограмм по модулям (осуществлять автономное тестирование внешних подпрограмм). Для этого следует использовать специально разработанные для тестирования подпрограммы:

- вызывающую ее для теста программу (модуль Program);

- "подпрограммы-заглушки", имитирующие вызываемые подпрограммой, но непроверенные еще подпрограммы.

2.9. **Пояснительная записка по курсовой работе** должна состоять из титульного листа, информационной и основной частей, приложений.

Информационная часть записки включает аннотацию (сведения о назначении пояснительной записки и краткое изложение ее основной части) и "содержание" (перечень наименований разделов и подразделов основной части записки и приложений с указанием их номеров и номеров страниц, с которых эти структурные единицы записки начинаются).

Основная часть пояснительной записки должна содержать следующие нумеруемые по порядку (1, 2, 3, 4) разделы:

- описание постановки задачи;

- описание проектных решений;

- описание программы;

- описание результатов испытания программы.

Описание постановки задачи включает подразделы:

- характеристика задачи (1.1);

- исходные данные (1.2);

- результаты решения (1.3).

Описание проектных решений включает подразделы:

- функциональная структура задачи, программы (2.1);

- описание алгоритма задачи (2.2).

Описание программы включает подразделы:

- структура программы (3.1);

- текст программы (3.2);

- исполнение программы (3.3).

В приложениях (они нумеруются) приводятся:

- исходный текст программы;

- тексты тестовых файлов;

- тексты программ создания и печати тестовых файлов;

- контрольные примеры исполнения программы.

В приложении 3 (к данному учебно-методическому пособию) приводится "макет пояснительной записки по курсовой работе".

**3. Примеры проектирования программ**

В качестве примеров проектирования программ приводятся результаты проектирования для задач "Учет посещаемости учебных занятий" и "Учет успеваемости студентов" (см. приложение 2). Алгоритмические решения в примерах приведены не в полном объеме. В курсовой работе следует представлять схемы алгоритмов большего числа процедур.

3.1. **Проектирование программы задачи "Учет посещаемости учебных занятий"** осуществляется для получения результатов в виде таблицы, представленной на рис. 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сведения**  **о посещаемости учебных занятий в период с \_\_\_\_\_\_\_\_\_\_ по \_\_\_\_\_\_\_\_\_\_**  (список студентов, у которых пропуски занятий составляют более \_\_\_\_\_\_ час.)   |  |  | | --- | --- | | **Фамилия и инициалы студента** | **Количество часов пропущенных занятий** | |  |  | |

Рис. 5. Форма представления результатов задачи "Учет посещаемости учебных занятий"

На рис. 6 представлена **схема иерархии процедур этой задачи**, полученная в результате ее декомпозиции.

Рис. 6. Схема иерархии процедур задачи "Учет посещаемости учебных занятий"

В качестве процедур первого уровня декомпозиции выделены:

- ввод и проверка периода;

- ввод и проверка количества часов ("минимально допустимого");

- получение искомого списка студентов (у которых пропуски занятий составляют более указанного пользователем);

- сортировка (упорядочивание) искомого списка студентов;

- вывод искомого списка студентов.

В свою очередь процедура ввода и проверки периода включает в себя процедуру ввода и проверки даты (последняя должна выполняться дважды – для начальной и конечной дат периода).

К процедурам второго уровня декомпозиции задачи относятся также процедуры в составе процедуры "Получение искомого списка студентов":

- определение количества пропущенных студентами часов (в результате обработки второго исходного файла, в котором студенты представлены своими номерами);

- выбор в искомый список студентов (тех, у которых количество часов пропущенных занятий более "минимально допустимого", т.е. указанного в запросе пользователя);

- получение фамилий и инициалов студентов (для дополнения ими искомого списка – в результате обработки первого исходного файла).

Основная программа

Рис. 7. Функциональная структура программы для задачи "Учет посещаемости учебных занятий"

На рис.7 представлена функциональная структура программы решения задачи, полученная на основе схемы иерархии ее процедур. Наличие процедуры "Получение искомого списка студентов" в качестве отдельной, для которой пишется подпрограмма, признано нецелесообразным.

Помимо основной программы с именем RecAbsences, программа должна содержать следующие подпрограммы (указаны их заголовки):

1) Procedurе InPeriod (Var d1, d2: string; Var ℮: boolean);

обеспечивающая ввод и проверку периода (d1, d2 – начальная и конечная даты периода, ℮ - признак ошибки во введенных данных);

2) Procedure InDate (Var d: string; Var ℮: boolean);

обеспечивающая ввод и проверку даты d (℮ - признак ошибки в дате);

3) Procedure InMinH (Var h: integer; Var ℮: boolean);

обеспечивающая ввод и проверку минимально допустимого количества пропущенных часов занятий h (℮ - признак ошибки в этом показателе);

4) Procedure DetH (d1, d2: string; Var ArH: ta; Var n: integer);

обеспечивающая определение количества часов попущенных студентами занятий в указанный период – формирование массива записей ArH из n элементов (каждый элемент массива содержит номер студента и количество пропущенных им часов) в результате обработки второго исходного файла;

5) Procedure Select (h: integer; ArH: ta; n: integer; Var List: tl, Var m:

integer);

обеспечивающая выбор в искомый список тех студентов, у кого пропуски занятий более указанного в запросе минимально допустимого значения h, формирование m элементов массива List из массива ArH (m ≤ n);

6) Procedure GetFIO (Var List: tl; m: integer);

обеспечивающая дополнение массива List из m элементов фамилиями и инициалами студентов (они выбираются по номерам студентов из первого исходного файла);

7) Procedure OrdR (Var List: tl; m: integer);

обеспечивающая сортировку (упорядочивание) искомого списка студентов (массива записей List из m элементов) по убыванию количества пропущенных часов;

8) Procedure OutR (d1, d2: string; n: integer; List: tl; m: integer);

обеспечивающая вывод результатов задачи в требуемой форме (рис. 5).

В качестве параметров подпрограмм DetH, SelectL, GetFIO, OrdR и OutR используются массивы записей следующих типов:

Type teArH = record

nums: longint;

h: integer

end;

ta = array [1..50] of teArH;

teList = record

nums: longint;

fio: string [25];

h: integer

end;

tl = array [1..50] of teList;

В этих описаниях:

teArH – имя типа элемента массива записей ArH (nums - номер студента, совпадающий с номером элемента файла с фамилией и инициалами студента, h - количество часов пропущенных занятий);

ta - имя типа массива записей ArH;

teList - имя элемента массива записей List (fio - фамилия и инициалы студента);

tl - имя массива записей List.

Данное описание типов массивов записей (и их элементов) включается в модуль Unit с именем U4. U4 – модуль с описаниями общих нестандартных типов данных (для головного модуля с основной программой и определенных модулей с подпрограммами). Подпрограммы записываются в других модулях в соответствии с представленной на рис. 8 **модульной структурой программы:**

- в **модуле U1** подпрограммы InPeriod, InDate, InMinH;

- в **модуле U2** подпрограммы DetH, Selest, GetFIO;

- в **модуле U3**  - подпрограммы OrdR, OutR.

Головной модуль

Program

RecAbsences

InPeriod

InDate

InMinH

U1

OrdR

OutR

U3

Type…

U4

DetH

Select

GetFIO

U2

Рис. 8. Модульная структура программы задачи "Учет посещаемости учебных занятий"

При этом головной модуль Program (с основной программой RecAbsences) использует все модули Unit, a модули U2 и U3 используют модуль U4 (с общими для них типами данных).

На рис. 9 представлена **схема обобщенного алгоритма задачи** "Учет посещаемости учебных занятий". В ней предусмотрено выполнение выделенной на этапе декомпозиции задачи процедуры "Получение искомого списка студентов", но не в результате выполнения соответствующей подпрограммы (ее наличие было признано нецелесообразным). Сделано это для того, чтобы схема алгоритма для его наглядности поместилась на одной странице.

нет

нет

да

да

да

Получение искомого списка студентов

Вывод сообщения об отсутствии

Н

Ошибка?

InPeriod

InMinH

Ошибка?

Список есть?

Сорти-ровка?

OutR

К

Вывод сообщения об ошибке

Вывод сообщения об ошибке

OrdR

нет

нет

Рис. 10

Рис. 9. Схема обобщенного алгоритма задачи "Учет посещаемости учебных занятий"

На следующем рисунке (рис. 10) представлена детализация алгоритма этой процедуры (в этой схеме предусмотрены исполнения процедур по подпрограммам DetH, SelectL и GetFIO, которые будут вызваны в основной программе задачи – модуле Program).

нет

нет

Н

Данные есть?

да

DetH

Данные есть?

SelectL

GetFIO

K

да

Рис. 10. Схема процедуры получения искомого списка студентов.

На рис. 11 представлена схема алгоритма определения количества пропущенных студентами часов в результате обработки второго исходного файла (данный алгоритм должен быть описан в виде подпрограммы DetH).

нет

нет

да

С прочитанным номером

студента

Подготовка к обработке файла и формированию массива записей ArH

Элемент есть?

Подготовка процедуры

Чтение элемента файла

DetH

Дата в периоде?

Поиск элемента в массиве ArH

Есть элемент?

Ошибка?

Создание нового элемента в массиве

Увеличение количества часов в элементе массива

К

да

да

нет

Рис. 11. Схема алгоритма определения количества пропущенных часов у студентов.

Дополним результаты проектирования программы текстами головного модуля (рис.12) и интерфейсными частями модулей с подпрограммами и описанием общих типов данных (рис. 13).

|  |
| --- |
| Program ResAbsences; {Учет посещаемости учебных занятий}  Uses U1, U2, U3, U4;  Var d1, d2: string; h: integer; e: boolean;  Arh: ta; List: tl; n, m: integer;  Begin  InPeriod (d1,d2, e); {Ввод и проверка периода}  If e then Writeln ('Даты указаны с ошибкой') else  begin  InMinH (h, e); {Ввод и проверка количества часов}  If e then Writeln ('Недопустимый параметр') else  begin  {Получение искомого списка студентов}  {Определение количества пропущенных часов}  DetH (d1, d2, Arh, n);  If n > 0 then {Начальный список есть}  begin  {Выбор в искомый список студентов}  Select (h, Arh, n, List, m);  If m > 0 then {Список получен}  {Выбор фамилий и инициалов студентов}  GetFio (List, m);  end  else m := 0;  If m > 0 then {Список студентов получен}  begin  If m > 1 then {Необходима сортировка списка}  OrdR (List, m);  OutR (d1, d2, el h, List, m);  end  else {Сведений по запросу нет}  begin  If n = 0 then Writeln (' Нет пропусков занятий  в указанный период ')  else writeln ('Нет имеющих пропуски  занятий более указанного');  end;  end;  end;  End. |

Рис. 12. Текст основной программы (головного модуля).

|  |
| --- |
| **Unit U1;**  Interface  Procedure InDate (d: string; Var e: boolean);  Procedure InPeriod (Var d1, d2: string; Var e: boolean);  Procedure InMinH (Var h: integer; Var e: boolean);  Implementation  Procedure InDate ;  Begin  {Текст процедуры InDate}  end;  …  End.  **Unit U2;**  Interface  Uses U4;  Procedure DetH (d1, d2: string; Var ArH:ta; Var n: integer);  Procedure Select(h: integer; ArH:ta; n: integer; Var list: tl; Var m: integer);  Procedure GetFIO( Var list: tl; m: integer);  Implementation  …  End.  **Unit U3;**  Interface  Uses U4;  Procedure OrdR (Var List:tl; m: integer);  Procedure OutR (d1, d2: string; n: integer; List:tl; m: integer);  Implementation  …  End.  **Unit U4;**  Interface  Type teArh = record  nums: longint;  h: integer; end;  ta = array [1..50] of teArh;  telist = record  nums: longint;  fio: string [25];  h: integer; end;  tl = array [1..50] of telist;  Implementation  End. |

Рис. 13. Тексты программ модулей с подпрограммами и описанием общих типов.

3.2. **Проектирование программы задачи "Учет успеваемости студентов"** осуществляется для получения результатов в виде таблицы, представленной на рис.14.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сведения об успеваемости студентов  учебной группы <шифр группы>   |  |  |  |  | | --- | --- | --- | --- | | **Учебная дисциплина** | **Показатели успеваемости** | | | | **Число аттестованных студентов (чел.)** | **Число неаттестованных студентов (чел.)** | **Процент успеваемости** | |  |  |  |  | |

Рис. 14. Форма представления результатов задачи "Учет успеваемости студентов".

Рис. 15. Схема иерархии процедур задачи "Учет успеваемости студентов" (см. приложение 2).

На рис. 15 представлена схема иерархии процедур этой задач, полученная в результате ее декомпозиции. В качестве процедур первого уровня декомпозиции выделены:

- ввод и проверка шифра учебной группы;

- формирование сведений об успеваемости;

- вывод таблицы со сведениями.

Из них детализирована только вторая процедура, включающая в себя:

- подготовку к формированию сведений;

- определение показателей успеваемости по дисциплинам (см. рис. 14);

- сортировку (упорядочивание) сведений об успеваемости (по убыванию процента студентов, получивших положительные оценки по дисциплине).

Подготовка к формированию сведений предполагает:

- получение списка дисциплин из первого файла;

- подготовку определения показателей успеваемости (подсчета аттестованных и неаттестованных по каждой дисциплине студентов.

Определение показателей успеваемости по дисциплинам, в свою очередь "разбивается" на:

- определение количества студентов по категориям для каждой дисциплины (подсчет аттестованных и неаттестованных студентов в результате обработки второго файла);

- вычисление процентов успеваемости по дисциплинам.

На рис. 16 представлена **функциональная структура программы** решения задачи, полученная на основе схемы иерархии ее процедур. Наличие процедур "Формирование сведений об успеваемости", "Получение списка дисциплин", "Подготовка определения показателей успеваемости" "Определение количества студентов по категориям" в качестве отдельных, для которых пишутся подпрограммы, признано нецелесообразным.

Рис. 16. Функциональная структура программы "Учет успеваемости студентов".

Помимо основной программы с именем RecAtStud, программа должна содержать следующие подпрограммы (указаны их заголовки):

1) Procedure InGroup (Var Gr : string; Var ℮ : boolean);

обеспечивающие ввод и проверку шифра учебной группы Gr (℮ - признак ошибки в этом данном);

2) Procedure PrepFR (Var R : tr; Var n : integer);

обеспечивающая подготовку к формированию сведений (результатов) в виде массива записей R из n элементов (элемент массива должен содержать наименование дисциплины из первого исходного файла и нулевые значения показателей успеваемости по ней);

3) Procedure DetQR (Gr : string; Var R : tr; n : integer);

обеспечивающая определение (подсчет) показателей успеваемости студентов указанной группы (Gr - шифр группы) по дисциплинам в массиве R из n элементов (в результате обработки второго исходного файла);

4) Procedure CalcInt (Var R : tr; n : integer);

обеспечивающая вычисление процентов успеваемости по дисциплинам в массиве записей R из n элементов (вызывается к исполнению из программы DetQR, но возможен вариант вызова ее из основной программы);

5) Procedure OrdR (Var R : tr; n : integer);

обеспечивающая сортировку (упорядочивание) элементов массива записей R из n элементов по убыванию процента успеваемости по дисциплине;

6) Procedure OutR (Gr : string; R : tr; n : integer);

обеспечивающая вывод запроса и результатов решения задачи по нему.

В качестве параметра подпрограмм PrepFR, DetQR, CalcInt, OrdR, OutR используется массив записей следующего типа:

Type teR = record

sub : string [25];

sa, sna : integer;

rate : real

end;

tr = array [1..10] of teR;

В этом описании:

- teR - тип элемента массива записей R (sub - наименование дисциплины, sа – количество аттестованных по дисциплине студентов, snа - количество неаттестованных по дисциплине студентов, rate - процент успеваемости по дисциплине);

- tr - тип массива записей R.

Данное описание типа массива (и его элемента) включается в модуль Unit с именем U5 (модуль с общими для головного модуля с основной программой и определенных модулей с подпрограммами нестандартных типов данных). Подпрограммы записываются в других модулях в соответствии с представленной на рис. 17 **модульной структурой программы:**

- в **модуле U1** подпрограмма InGroup;

- в **модуле U2** подпрограмма PrepFR;

- в **модуле U3** подпрограмма DetQR и CalcInt;

- в **модуле U4** подпрограмма OrdR и OutR.

При этом головной модуль Program (с основной программой RecAtStud) использует все модули Unit, а модули U2, U3 и U4 используют модуль U5.

Отметим, что подпрограмма CalcInt может быть внутренней в подпрограмме DetQR.

Головной модуль

Program

IuGroup

U1

PrepFR

U2

DetHR

CalcInt

U3

OrdR

OutR

U4

Type..

U5

RecAtStud

Рис. 17. Модульная структура программы задачи "Учет успеваемости студентов".

На рис. 18 представлена схема обобщенного алгоритма задачи "Учет успеваемости студентов". На рис. 19 – схема алгоритма процедуры подготовки к формированию сведений об успеваемости (в т.ч. обработки первого исходного файла). На рис. 20 – схема алгоритма определения показателей успеваемости по дисциплинам в результате обработки второго исходного файла.

Дополним результаты проектирования программы задачи "Учет успеваемости студентов" текстами головного модуля и интерфейсных частей модулей с подпрограммами и описанием общих типов данных (рис. 21-23).

нет

да

да

да

Вывод сообщения об отсутствии

Н

Ошибка?

InPGroup

PrepFR

Дисципл. есть?

Сорти-ровка?

OutR

К

Вывод сообщения об ошибке

OrdR

нет

нет

DetQR

Подготовка к формированию

сведений

Определение

показателей

Вывод

результатов

Рис. 18. Схема обобщенного алгоритма задачи "Учет успеваемости студентов".

нет

Элемент файла есть?

Подготовка к обработке

файла

Чтение элемента файла

PrepFr

Формирование Rn

n=n+1

К

Subn=<элемент файла>

can=0, cann=0

да

Рис. 19. Схема алгоритма подготовки формирования сведений (об успеваемости студентов).

нет

нет

да

да

да

нет

Элемент есть?

Подготовка к обработке

файла

Чтение элемента файла

DetQR

i = 1, n

Оценкаi>2

Шифр группы?

sai = sai+1

snai = snai+1

K

CalcInt

Рис. 20. Схема алгоритма определения показателей успеваемости.

|  |
| --- |
| Procedure RecAtStud;  {Учет успеваемости студентов}  Uses U1, U2, U3, U4, U5;  Var Gr: string; e: boolean; R: tr; n: integer;  Begin  {Ввод и проверка шифра группы}  InGroup (Gr, e);  If e then Writeln ('Ошибка в шифре группы')  else begin  {Подготовка к формированию сведений}  PrepFR (R, n);  If n=0 then Writeln ('Списка дисциплин нет')  else  begin {Определение показателей успеваемости}  DetQR(Gr, R, n);  If n>1 then {Сортировка сведений} Ordr(R, n);  {Вывод сведений об успеваемости}  OutR (Gr, R, n);  end;  end;  End. |

Рис. 21. Текст головного модуля (основной программы).

|  |
| --- |
| **Unit U1;**  Interface  Procedure InGroup (Var GR: string; Var e: boolean);  Implementation  ...  End.  **Unit U2;**  Interface  Uses U5;  Procedure PrepFR (Var R: tr; Var n: integer);  Implementation  ...  End. |

Рис. 22. Тексты модулей U1и U2 с подпрограммами.

|  |
| --- |
| **Unit U3;**  Interface  Uses U5;  Procedure CalcInt ( Var R: tr; n: integer);  Procedure DetQR(Gr: string; Var R: tr; n: integer);  Implementation  ...  End.  **Unit U4;**  Interface  Uses U5;  Procedure OrdR ( Var R: tr; n: integer);  Procedure OutR ( Gr: string; R:tr; n: integer);  Implementation  ...  End.  **Unit U5;**  Interface  Type teR = record  sub: string [25];  sa, sna: integer;  rate: real;  end;  tr = array [1..10] of teR;  Implementation  End. |

Рис. 23. Тексты программ модулей U2, U3, U4 с подпрограммами и модуля U5 с описанием общих типов.

Приложения

Приложение 1

Бланк задания

Задание на курсовую работу по дисциплине

"Программирование"

студента группы <шифр группы> <фамилия, инициалы>

Разработать Паскаль-программу решения задачи

<наименование задачи>.

Исходные данные:

1) Файл <наименование файла>;

Структура элемента файла:

* <смысл данного> (<формат данного>);
* <смысл данного> (<формат данного>);
* …

2) Файл …

3) Запрос пользователя:

* <смысл данного> (<формат данного>);
* …

Результаты решения: сведения о …, упорядоченные по ….

Студент Руководитель работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ф.И.О./ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ф.И.О./

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Приложение 2

Примеры задания

**Задание 1.**

Разработать Паскаль-программу решения задачи **"Учет посещаемости учебных занятий".**

**Исходные данные:**

1) Файл "Список студентов"

Структура элемента файла:

* фамилия и инициалы студента (строка).

2) Файл "Сведения о пропусках занятий"

Структура элемента файла:

* дата (строка в формате ´ггггммчч´ - год, месяц, "число");
* номер студента (целое число – номер элемента из первого файла с фамилией и инициалами студента);
* количество часов пропущенных занятий (целое число).

3) Запрос:

* две даты, определяющие период времени (две строки в формате ´чч.мм.гггг´);
* количество часов (целое число).

**Результаты решения**: Список студентов, у которых количество часов пропущенных занятий в указанный период более указанного в запросе пользователя, упорядоченный по убыванию этого показателя.

**Задание 2.**

Разработать Паскаль-программу решения задачи **"Учет успеваемости студентов".**

**Исходные данные:**

1) Файл "Список дисциплин"

Структура элемента файла:

* наименование дисциплины (строка).

2) Файл "Экзаменационные оценки".

Структура элемента файла:

* шифр учебной группы (строка);
* фамилия и инициалы студента (строка);
* экзаменационные оценки студента по дисциплинам (целые числа, последовательность которых соответствует последовательности дисциплин в первом файле).

3) Запрос: шифр учебной группы (строка).

**Результаты решения:** Сведения об успеваемости студентов указанной учебной группы по дисциплинам, упорядоченные по убыванию показателя успеваемости – процента студентов, получивших положительные оценки по дисциплине.

Приложение 3

Макет пояснительной записки

по курсовой работе по дисциплине

"Программирование"

### Титульный лист

Калининградский государственный технический университет

Кафедра систем управления и вычислительной техники

Работа защищена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(дата)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и Ф.И.О. преподавателя)

<НАИМЕНОВАНИЕ ЗАДАЧИ>

Пояснительная записка по курсовой работе

по дисциплине "Программирование "

Работа допущена Работу выполнил студент

к защите учебной группы\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и Ф.И.О. (подпись и Ф.И.О студента)

руководителя работы)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(дата) (дата)

Калининград, 201\_

Аннотация[[1]](#footnote-1)

В настоящей пояснительной записке приведены результаты разработки программы решения задачи <наименование задачи>, выполненной в соответствии с заданием по курсовой работе по дисциплине "Программирование" – постановка указанной задачи, проектные решения и программа (на языке Турбо – Паскаль), результаты испытания (тестирования) программы.

## СОДЕРЖАНИЕ[[2]](#footnote-2)

Стр.

|  |  |
| --- | --- |
| 1. Описание постановки задачи | 3 |
| 1.1. Характеристика задачи | 3 |
| 1.2. Исходные данные | … |
| 1.3. Результаты решения | … |
| 2. Описание проектных решений | … |
| 2.1. Функциональная структура задачи, программы | … |
| 2.2. Описание алгоритма задачи | … |
| 3. Описание программы | … |
| 3.1. Структура программы | … |
| 3.2. Текст программы | … |
| 3.3. Исполнение программы | … |
| 4. Описание результатов испытаний программы | … |
| Приложения | … |
| Приложение 1. Исходный текст программы | … |
| Приложение 2. Тексты тестовых файлов | … |
| Приложение 3. Тексты программ создания и печати тестовых файлов | … |
| Приложение 4. Контрольные примеры исполнения программы | ... |

1. Описание постановки задачи

1.1. Характеристика задачи

Задача "... " решается с целью получения сведений о ……. Исходными для решения задачи данными являются файлы на внешнем носителе, содержащие информацию о …. В начале решения задачи задается (вводится с клавиатуры) запрос, представляющий собой … и определяющий для какого (каких) … должны быть получены указанные выше сведения (результаты решения задачи).

1.2. Исходные данные

Исходная для решения задачи "... " информация содержится в двух файлах и в указываемом (вводимом с клавиатуры) в начале решения задачи запросе пользователя.

Структура элемента файла "…" (дисковое имя – ‘...’):

<смысл данного> (<формат данного>),

...

Запрос пользователя представляет собой:

<смысл данного> (<формат данного>),

...

1.3. Результаты решения

Результатом решения задачи на компьютере является таблица "... " (рис. 1.1), которая содержит:

<смысл данного> (<формат данного>),

...

Данные (строки) в таблице располагаются в порядке убывания (возрастания) … (или "в алфавитном порядке .").

2. Описание проектных решений

2.1. Функциональная структура задачи, программы

На рис. 2.1 представлена схема иерархии процедур задачи "…", полученная в результате её декомпозиции, если в схеме используются условные обозначения процедур, даются необходимые пояснения.

На рис. 2.2 представлена функциональная структура программы решения задачи, полученная на основе схемы иерархии ее процедур.

Помимо основной программы с именем "…", программа должна содержать следующие подпрограммы (указаны заголовки подпрограмм):

1) Подпрограмму <имя подпрограммы> (<параметры подпрограммы>), которая обеспечивает ввод и проверку запроса (элемента запроса) - ….;

2) Подпрограмму <имя подпрограммы> (<параметры подпрограммы>), которая обеспечивает обработку файла "…" с целью получения … для заданного (ых)…;

3) Подпрограмму <имя подпрограммы> (<параметры подпрограммы>), которая обеспечивает сортировку (упорядочивание)… по …. ;

4) Подпрограмму <имя подпрограммы> (<параметры подпрограммы>), которая обеспечивает вывод … на экран монитора;

….

Вместо слова "Подпрограмму" указывается Procedure либо Function.

2.2. Описание алгоритма задачи

На рис. 2.3. – 2.\_\_ представлены схемы обобщенного алгоритма задачи "…" и алгоритмов отдельных её процедур. Смысл используемых в них условных обозначений (имен, идентификаторов) приведен в табл. 2.1. Обобщенный алгоритм (алгоритм задачи в целом) описывается в основной программе. Алгоритмы отдельных процедур – в соответствующих подпрограммах (см. 2.1.).

3. Описание программы

3.1. Структура программы

При написании программы решения задачи "…." использовалась система (язык) программирования Турбо-Паскаль (версия…).

Функциональная структура программы (состав и взаимосвязь программных единиц исходного текста её – основной программы и подпрограмм) определена в 2.1.

Программа является описанием алгоритма задачи, приведенного в 2.2.

Модульная структура программы (состав и взаимосвязь модулей исходного текста её) приведена на рис. 3.1. Ниже указано содержание модулей программы.

Головной модуль <имя модуля> содержит тексты основной программы и внутренних подпрограмм <перечень имен подпрограмм>.

Модуль… содержит подпрограмму (ы)…

Модуль… содержит описания общих типов данных.

3.2. Текст программы

Исходный текст программы решения задачи "…" (на языке Турбо-Паскаль, версия…) – её модулей – приведен в приложении 1.

3.3. Исполнение программы

Для исполнения программы решения задачи "..." необходим компьютер типа … с клавиатурой и монитором. Программа предварительно откомпилирована и собрана в исполняемый модуль с именем…, который вызывается для исполнения стандартными средствами.

До начала исполнения программы на внешнем носителе должны быть сформированы файлы с исходными данными, структура, содержание и имя которого описаны в 1.2. В начале исполнения программы необходимо ввести с клавиатуры запрос, конкретизирующий вариант (условия) решения задачи (см. 1.2).

4. Описание результатов испытаний программы

В ходе испытаний (тестирования и контрольных примеров исполнения) программы решения задачи "…" использовались тестовые файлы, тексты которых приведены в приложении 2 (программы создания и печати тестовых файлов приведены в приложении 3), а также запросы для разных вариантов решения задачи:

1. ….
2. …..
3. …..

…..

После устранения ошибок в программе для указанных исходных данных (контрольных примеров) получены правильные результаты решения задачи (приложение 4), что свидетельствует о правильности разработанной программы.

### Приложения

#### Приложение 1

Исходный текст программы решения задачи "…"

#### Приложение 2

Тексты тестовых файлов

#### Приложение 3

Текст программ создания и печати тестовых файлов

#### Приложение 4

Контрольные примеры исполнения программы

# Пример 1. Запрос: …

Результат: …

**Список учебной литературы**

1. Учебно – методическое пособие по лабораторным работам для студентов бакалавриата по направлениям подготовки "Информатика и вычислительная техника" "Прикладная информатика" (часть первая) – Калининград, ФГБОУ ВПО КГТУ, 2015. – с.
2. Учебно – методическое пособие по лабораторным работам для студентов бакалавриата по направлениям подготовки "Информатика и вычислительная техника" "Прикладная информатика" (часть вторая) – Калининград, ФГБОУ ВПО КГТУ, 2015. – с.
3. Немнюгин С.А. Turbo Pascal: учеб. – СПб.: Питер, 2000, - 491 с.
4. Павловская Т.А. Паскаль. Программирование на языке высокого уровня: Учебник для вузов – СПб.: Питер, 2007. – 393 с.: ил.
5. Аляев Ю.А., Гладков В.П., Козлов О.А. Практикум по алгоритмизации и программированию на языке Паскаль: учебное пособие – М.: Финансы и статистика, 2004. – 528 с.

# Интернет-ресурсы (ссылки на учебники и учебно-методические пособия):

http://progbook.net/pascal

http://sprinter.ru/10038

# Оглавление

Введение 3

Задание по курсовой работе 4

Методические указания по выполнению курсовой работы 4

Примеры проектирования программ 13

Приложения 32

Приложение 1. Бланк задания 32

Приложение 2. Примеры задания 33

Приложение 3. Макет пояснительной записки по курсовой работе

по дисциплине "Программирование" 35

Список учебной литературы 42

1. Печать на отдельной странице, страница с аннотацией не нумеруется [↑](#footnote-ref-1)
2. Печать на отдельной странице [↑](#footnote-ref-2)