

*Современный
Гуманитарный
Университет*

Дистанционное образование

0640.00.01;1

**ФАКУЛЬТЕТ ИНФОРМАТИКИ И
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

КАФЕДРА ИНФОРМАТИКИ

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ВЫПОЛНЕНИЮ КУРСОВОЙ
РАБОТЫ ПО ДИСЦИПЛИНЕ**

**“АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И
ПРОГРАММИРОВАНИЕ”**

МОСКВА 2002

Разработано Лабзиной Т.А.

Рекомендовано Министерством общего
и профессионального образования
Российской Федерации в качестве
учебного пособия для студентов высших
учебных заведений

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ

“АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И ПРОГРАММИРОВАНИЕ”

В предлагаемом пособии изложены основные требования и рекомендации по выполнению курсовой работы. Приводится перечень тем курсовых работ и список литературы.

Для студентов Современного Гуманитарного Университета

0640.007.00.99.01;1/06.14. Тир.1500

(С) СОВРЕМЕННЫЙ ГУМАНИТАРНЫЙ УНИВЕРСИТЕТ, 1999

ОГЛАВЛЕНИЕ

	стр.
1. Общие положения	4
2. Выполнение курсовой работы	4
3. Содержание разделов курсовой работы	4
3.1. Оглавление	5
3.2. Разработка эскизного и технического проектов программы	5
3.2.1. Введение	5
3.2.2. Назначение и область применения	6
3.2.3. Технические характеристики	6
3.2.3.1. Постановка задачи	6
3.2.3.2. Описание алгоритма	8
3.2.3.3. Организация входных и выходных данных	11
3.2.3.4. Выбор состава технических и программных средств	12
3.2.4. Источники, использованные при разработке	12
3.3. Разработка рабочего проекта	12
3.3.1. Разработка программы	12
3.3.2. Спецификация программы	19
3.3.3. Текст программы	20
3.3.4. Описание программы	25
3.3.5. Тестирование программы	25
3.4. Внедрение	26
3.5. Литература	26
4. Оформление пояснительной записки	26
5. Тематика курсовых работ	27
5.1. Базы данных	27
5.2. Динамические структуры	28
5.3. Игры	29
5.4. Строковые данные и текстовые файлы	31
6. Литература	32
7. Приложения	34

1. ОБЩИЕ ПОЛОЖЕНИЯ

Курсовая работа предусмотрена тематическим планом изучения дисциплины “Алгоритмические языки и программирование”. Курсовая работа является самостоятельной работой студента, позволяет оценить качество знаний и отражает приобретенные студентом практические навыки.

Курсовая работа позволяет расширить объем знаний студентов в области программирования и создать реальную основу использования своих знаний для решения на ЭВМ задач по другим дисциплинам и в своей дальнейшей практической деятельности.

Тема назначается руководителем курсовой работы и утверждается на заседании учебно-методической комиссии.

Перед студентом ставится задача разработать приложение для Windows с целью решения конкретной задачи. Результатом решения является:

- А) исполняемый файл программы;
- Б) пояснительная записка, составленная с учетом требования стандартов ЕСПД.

Для решения поставленной задачи студенту необходимо предварительно ознакомиться с литературой, посвященной теме задания. При этом следует обратить внимание на средства, используемые для решения аналогичных задач или для решения каких-либо ключевых моментов задачи. Этап работы с литературой должен закончиться обзором, в котором собраны полученные сведения из литературы, дан их анализ с точки зрения приложения к поставленной задаче.

Сформулированные в настоящих указаниях задания на курсовую работу представляют студенту простор для творчества. В текстах задач умышленно опущены некоторые детали и необходимые требования. После ознакомления с литературой студент должен оценить возможности языка программирования и вычислительной техники, на которой предлагается реализовать решение. Результатом этой работы должна быть точная формулировка задачи со всеми ограничениями и требованиями.

При решении задачи необходимо придерживаться техники пошаговой детализации, использовать стандартные структуры, не забывая при этом о развитии программного окружения программиста, расширяя возможности языка за счет включения новых процедур и функций.

При разработке алгоритма необходимо предусмотреть средства проверки и тестирования программы, удобство работы пользователя, возможные модификации.

При написании программы не следует забывать о хорошем стиле программирования, о таких понятиях, как читабельность, эффективность, надежность. Необходимо искать наиболее простые и естественные приемы и методы решения.

2. ВЫПОЛНЕНИЕ КУРСОВОЙ РАБОТЫ

Выполнение курсовой работы состоит из трех этапов.

1. Подготовительный этап (разработка эскизного и технического проектов).
2. Практическая работа за компьютером (разработка рабочего проекта).
3. Оформление пояснительной записки.

3. СОДЕРЖАНИЕ РАЗДЕЛОВ КУРСОВОЙ РАБОТЫ

Все этапы разработки программы отражаются в пояснительной записке.

Пояснительная записка состоит из следующих разделов:

1. Оглавление.
2. Разработка эскизного и технического проектов программы (ГОСТ 19.404–79).
 - 2.1. Введение.
 - 2.2. Назначение и область применения.
 - 2.3. Технические характеристики.
 - 2.4. Источники, использованные при разработке.
3. Разработка рабочего проекта.
 - 3.1. Разработка программы.
 - 3.2. Спецификация программы.
 - 3.3. Текст программы.
 - 3.4. Описание программы.
 - 3.5. Тестирование программы.
4. Внедрение.
5. Литература.

При написании пояснительной записки необходимо придерживаться требований единой системы программной документации (ЕСПД).

3.1. Оглавление

Оглавление составляется в соответствии с содержанием пояснительной записки и должно отражать все разделы курсовой работы. После написания пояснительной записки в оглавлении проставляются страницы.

3.2. Разработка эскизного и технического проектов программы

Стандарт ГОСТ 19.404–79 устанавливает требования к содержанию и оформлению программного документа “Пояснительная записка”, входящего в состав документов на стадиях разработки эскизного и технического проектов программы.

3.2.1. Введение

В разделе “Введение” указывается тема курсовой работы, прилагается документ, на основании которого ведется разработка, с указанием организации и даты утверждения.

Пример.

Задание на курсовую работу по дисциплине
“Алгоритмические языки и программирование”

Студент группы И–333

Ивочкин К. П.

Направление: “Информатика”, № контракта 003009812034

Тема: Разработка приложения для Windows, представляющего собой компьютерную игру “Лабиринт”.

Условие задачи:

Игра “Лабиринт” состоит в том, что играющий перемещается в двухмерном пространстве по помещениям здания, план которого неизвестен. Начиная с произвольного помещения, путешественник должен найти выход из здания. Каждое помещение может иметь четыре двери: север, восток, юг, запад. План здания необходимо считать из текстового файла в связанный список. Порядок следования помещений в списке должен быть произвольным. Находясь в N–ом помещении, игрок может получить справку о правильном направлении движения, если верно ответит на вопрос по теме “Алгоритмические языки и программирование”.

Задание выдано: (число и подпись).

3.2.2. Назначение и область применения

В разделе “Назначение и область применения” указывают назначение программы и краткую характеристику области применения программы.

В приведенном примере задания необходимо разработать развлекательную программу, представляющую собой игру. Область применения: досуг программиста. Поскольку ставится задача разработать приложение для Windows, то использоваться программа может только под управлением Windows 9x.

3.2.3. Технические характеристики

Раздел “Технические характеристики” должен содержать следующие подразделы:

1. Постановка задачи.
2. Описание алгоритма.
3. Организация входных и выходных данных.
4. Выбор состава технических и программных средств.

3.2.3.1. Постановка задачи

Решение задачи начинается с ее постановки. Дается точное описание исходных данных, условий задачи и целей ее решения. На этом этапе условия задачи, записанные в форме различных словесных описаний, необходимо выразить на формальном языке математики. Обычно математическая модель – это набор уравнений, неравенств и ограничений, приближенно описывающих задачу. При построении математической модели отбрасываются некоторые свойства реальной задачи, мало влияющие на решение.

В этом разделе могут быть описаны основные приемы программирования и типы данных, используемые при решении аналогичных задач. Например, если в задаче используются динамические структуры, то перечисляются виды динамических структур данных и основные процедуры по работе с динамическими структурами. Если задача заключается в формировании базы данных и дальнейшей работе с базой, то приводится описание используемых типов данных (характеристика данных записного типа) и приемы работы с файлами.

Далее описываются возможные пути решения задачи с указанием их достоинств и недостатков. Выбирается и обосновывается метод решения задачи. Описываются ограничения, накладываемые на исходные данные, необходимая разрядность и точность представления исходных данных и результатов решения. Указываются возможные пределы изменения входных параметров задачи.

Пример.

В условии задачи игры “Лабиринт” указывается на необходимость использования динамической структуры “связанный список” и текстового файла для хранения информации о плане помещений лабиринта, поэтому нужно в постановку задачи включить определения динамических структур и организации файлов, а также обзор методов работы с такими структурами. В задаче также требуется организовать подсказку для выбора направления движения в лабиринте. Подсказкой может служить правильно выбранный ответ по теме курса. Представляемые игроющему вопросы и варианты ответа также должны быть считаны из файла на диске. На организацию этого файла не накладываются дополнительные требования, поэтому студент должен сам принять решение о структуре файла.

После описания общих положений тематики работы, необходимо указать конкретные методы решения поставленной задачи. Математическая формулировка в данном случае заменяется словесным описанием.

Предположим, что помещения здания соединяются между собой так, как показано на рис. 1.

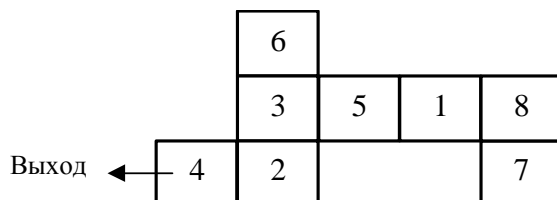


Рис. 1. Пример плана помещений

Информация о плане здания должна храниться в файле в последовательности: номер комнаты; номера комнат, смежных с данной. Смежные комнаты перечислены в порядке: север, восток, юг, запад (рис. 2). Порядок следования помещений в этом списке может быть любым, как и последовательность номеров помещений на плане.

Номер комнаты	Дверь в комнату			
	норд	ост	зюйд	вест
7	8	0	0	0
1	0	8	0	5
8	0	0	7	1
5	0	1	0	3
6	0	0	3	0
3	6	5	2	0
2	3	0	0	4
4	0	2	0	999
999	0	0	0	0

(0 = нет двери. 999 = выход)

Рис. 2. Информация о плане здания, представленном на рис. 3

Файл с данными считывается и помещается в список, исходя из номера помещения (рис. 3). Sp_i — указатель на начало списка.

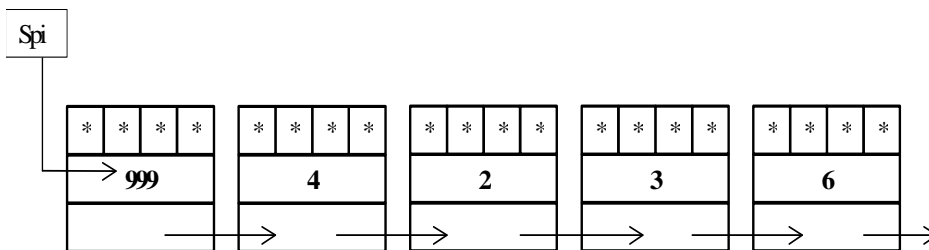


Рис. 3. Начало списка помещений после первого прочтения представленного на рис. 2 файла

Затем файл считывается еще раз и с учетом наличия дверей генерируется связь помещений (рис. 4).

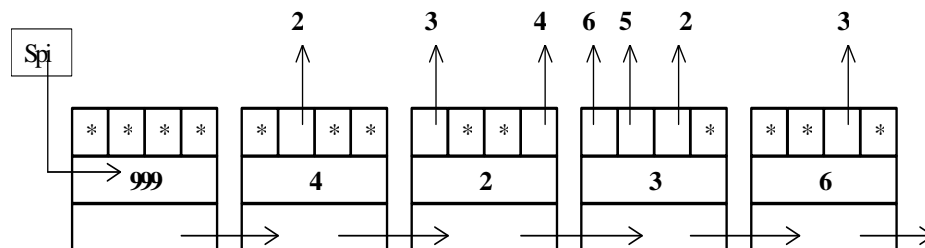


Рис. 4. Начало списка помещений после второго прочтения представленного на рис. 4 файла

Звездочкой помечены ссылки, равные Nil.

Перемещение игрока в лабиринте осуществляется путем выбора им одного из вариантов направления движения. Игрок вводит номер комнаты, с которой начинается путешествие. Если такого помещения в лабиринте нет, то об этом выдается сообщение. Если такая комната существует, то игроющему предлагается выбрать направление движения, которое можно определить, воспользовавшись подсказкой. Подсказка заключается в том, что на экран высвечивается вопрос по теме курса “Алгоритмические языки и программирование” и 4 варианта ответа. Правильный ответ указывает направление дальнейшего движения. Неправильный ответ дает неверное направление или вызывает сообщение о невозможности передвижения. Вывод подсказки на экран и выбор правильного ответа осуществляются с помощью специальной процедуры, которая обращается к файлу на диске. Файл-подсказку сделаем типизированным. Количество компонентов файла равно количеству помещений в здании-лабиринте. Номер компонента файла соответствует номеру помещения. Правильный ответ расположен в положении, определяющем верное направление. Компонент типизированного файла имеет тип “запись”, где отводится поле для хранения вопроса по теме курса и поле – массив предлагаемых ответов.

Запись:

Поле вопроса.

Поле 4-х ответов.

Признаком окончания игры является ссылка на помещение с номером 999.

В программу включим возможность создания и корректирования файлов плана помещения и подсказки.

3.2.3.2. Описание алгоритма

В разделе дается обобщенное словесное описание алгоритма решения поставленной задачи, излагаются основные требования к алгоритму и пути их реализации. Приводится схема алгоритма, состоящая из укрупненных модулей. Дается пояснение назначения и состава каждого модуля. Обобщенный алгоритм обычно использует обозначения и термины исходной задачи.

На следующем этапе каждый модуль детализируется. Выделяются укрупненные команды, реализуемые по вспомогательным алгоритмам. Тот же подход применяется при разработке вспомогательных алгоритмов.

Пример.

В программе решаются три независимые друг от друга подзадачи:

1. Перемещение по лабиринту.
2. Запись на диск плана помещений здания.
3. Запись на диск файла подсказки.

Задача “Перемещение по лабиринту” решается в два этапа: 1) формирование списка помещений; 2) выбор направления движения и переход из одного помещения в другое.

Обобщенная схема алгоритма приведена на рис. 5.

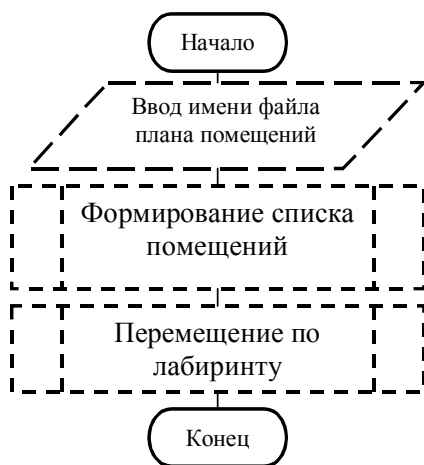


Рис. 5. Обобщенная схема алгоритма

При формировании списка плана помещений сначала производится считывание номеров комнат из текстового файла и построение связанного списка, затем построение многосвязанного списка, содержащего в себе план помещений здания (рис. 6).

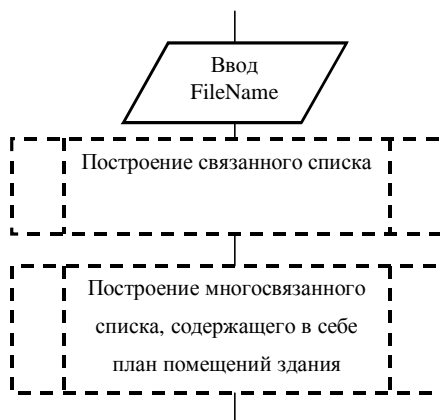


Рис. 6. Формирование списка плана помещений

Перемещение по лабиринту начинается с выбора номера помещения для начала путешествия. Если такого помещения нет, то игра заканчивается. Если такое помещение в лабиринте есть, то игроку предоставляется возможность выбрать направление движения или воспользоваться файлом подсказки. Укрупненная схема алгоритма приведена на рис. 7.

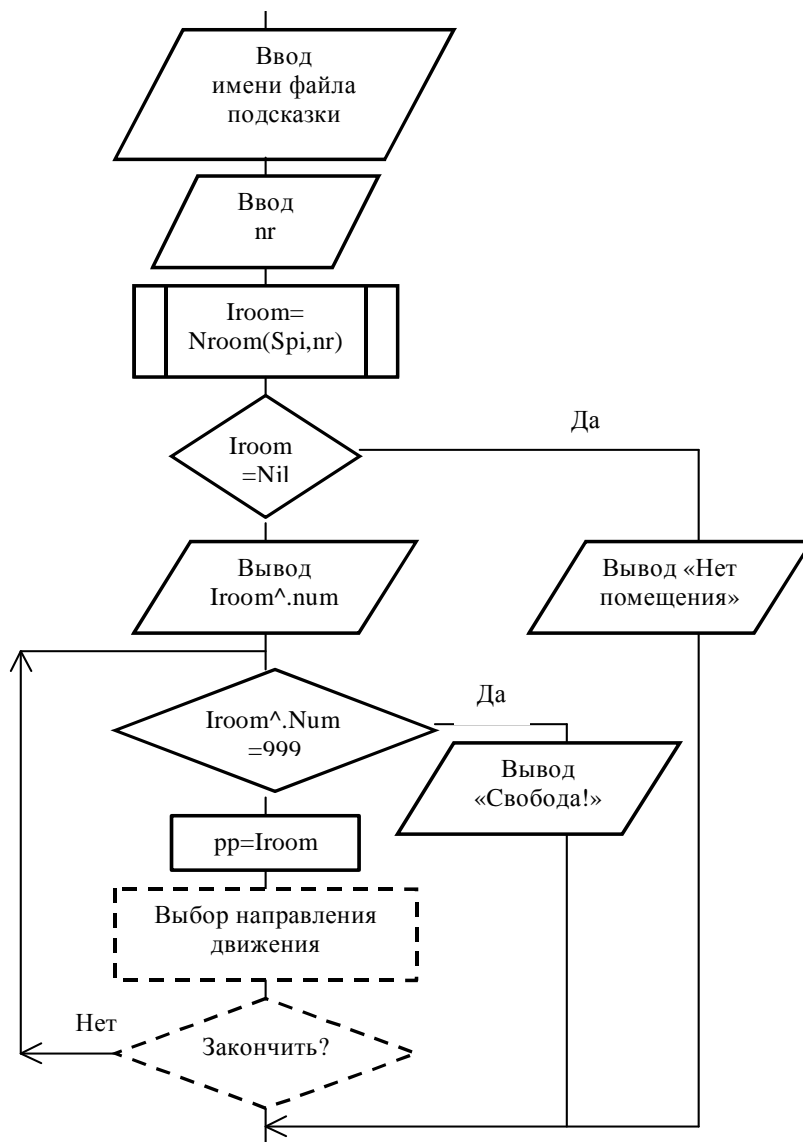


Рис. 7. Схема алгоритма этапа “Перемещение по лабиринту”

Блок “Выбор направления движения” может быть детализирован на этапе разработки программы.

Аналогично детализируются все составляющие алгоритма.

На этапе разработки технического проекта детализация на уровне операторов программы необязательна.

3.2.3.3. Организация входных и выходных данных

Данный раздел содержит описание и обоснование выбора метода организации входных и выходных данных.

В приведенном примере по условию задачи требуется использование файлов прямого доступа. Файл, содержащий план лабиринта, может быть организован непосредственно в приложении. Для этого разрабатывается отдельный фрагмент программы. Структура файла приведена на рис. 2. В текстовом файле в одной строке записаны номер комнаты и номера комнат смежных помещений. Смежные помещения располагаются в порядке: nord, ост, зюйд, вест. Если смежное помещение отсутствует, то вводится значение 0. Номера комнат в файле могут располагаться в произвольном порядке. Важным является то, что информация о комнате должна быть расположена в отдельной строке.

Разрабатываемое приложение предусматривает использование файлов прямого доступа. В файлах прямого доступа хранится информация подсказки. Файл–помощь может быть сформирован непосредственно в приложении. Для этого разрабатывается отдельный фрагмент программы. Количество компонентов файла равно количеству помещений в здании-лабиринте. Номер компоненты файла соответствует номеру помещения. Правильный ответ расположен в положении, определяющем верное направление. Компонент типизированного файла имеет тип “запись”, где отводится поле для хранения вопроса по теме курса и поле – массив предлагаемых ответов.

Запись:

Поле вопроса.

Поле 4–х ответов.

Для структуры помещений, приведенной на рис. 1, содержание файла-подсказки может быть следующим:

Стандартной файловой переменной является

Con

Prn

F

Input

Правильно описана файловая переменная текстового типа

F4: text80;

F1: File of String;

F: File of String[80];

F2: Text;

Для связи файловой переменной с физическим файлом на диске предназначена процедура

Reset

Rewrite

Assign

Read

Для чтения данных из типизированного файла предназначена процедура

Readln

ReadBlock

Input

Read

Если Var a, b: ^Real; то для a и b справедливо

New(a^)

a := a/b

a^ := a^ mod b^

a^ := sin(b^)

В результате выполнения New(p) p приобретает значение, соответствующее значению ноль,

типу переменной p,

адресу, начиная с которого можно разместить данные,

значению Nil

и т.д.

3.2.3.4. Выбор состава технических и программных средств

На основании разработанного алгоритма делается вывод о необходимости использования того или иного языка программирования. Перечисляются достоинства выбранной среды программирования. Определяются технические средства, необходимые для оптимальной работы будущей программы. В этой связи следует помнить, что задание на курсовую работу сформулировано, как разработка приложения для Windows.

В приведенном примере делается вывод о необходимости использования интегральной среды разработки программ Delphi. Среда Delphi позволяет достаточно быстро разрабатывать приложения для Windows. Технические характеристики компьютера: Pentium 100 и выше; объем оперативной памяти не менее 8 Мб; жесткий диск объемом не менее 500 Мб. Именно эти параметры создают условия для полноценной работы IDE Delphi 3 и программ, созданных в этой среде. Дополнительных средств (принтер, сканер, дополнительные дисководы и т.д.) не требуется.

3.2.4. Источники, использованные при разработке

Данный раздел должен присутствовать в пояснительной записке, если в основном тексте встречались ссылки на готовые разработки, используемые в программе. Здесь указывают перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов.

3.3. Разработка рабочего проекта

Этап разработки рабочего проекта включает в себя разработку программы и программной документации, а также испытание программы.

3.3.1. Разработка программы

Современные программы разрабатываются для функционирования в среде Windows. Приложение для Windows студенты разрабатывают в среде визуального программирования. Визуальное программирование строится на тесном взаимодействии двух процессов:

- процесс конструирования Windows-окна;
- процесс написания кода, придающего элементам этого окна и программе в целом необходимую функциональность.

Проект Windows-окна должен быть представлен в виде графической схемы, на которой расположены все визуальные и невидимые компоненты, разрабатываемого интерфейса. Компоненты на схеме должны быть пронумерованы. После схемы приводится расшифровка изображенных на схеме

компонентов: название и имя компонента; назначение в программе; события, на которые данный компонент откликается. Для каждого компонента должны быть указаны свойства, измененные при проектировании окна.

Пример.

Для разработки приложения игры “Лабиринт” используется среда визуального программирования “Delphi. Проект программы содержит три окна:

1. Form1 — перемещение по лабиринту.
2. Form2 — запись на диск плана помещений здания.
3. Form3 — запись на диск файла подсказки.

Окно Form1. Перемещение по лабиринту. (Рис. 8).

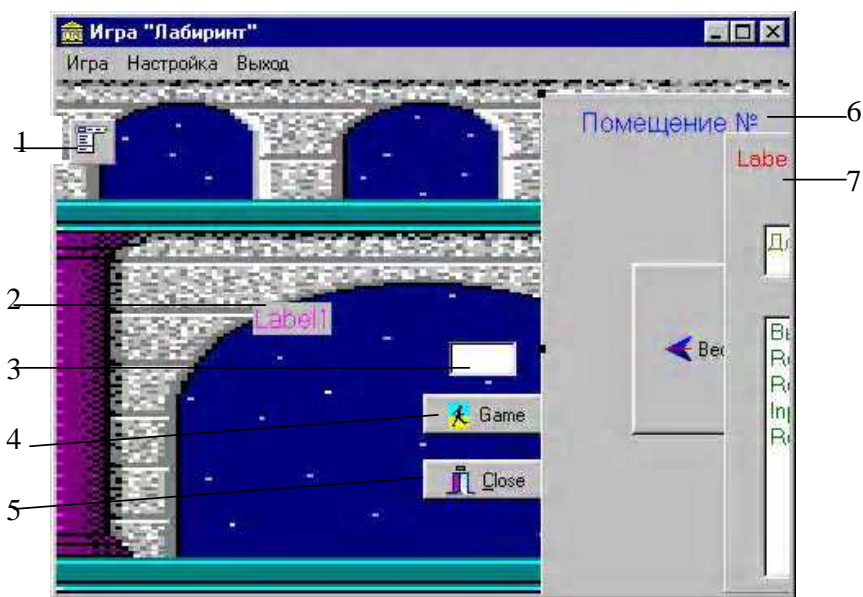


Рис. 8. Главное окно программы — Form1

Компонент Form1.

Свойства:

Caption — Игра “Лабиринт”;

Border — bsSingle;

BiSystemMenu — false;

BiMinimize — false;

BiHelp — False;

События: нет.

1 — компонент TMainMenu.

Свойства:

Items

- | | | |
|-----------|---------------------------|------------|
| Игра (N1) | Настройка (N2) | Выход (N3) |
| | Ввод плана помещения (N4) | |
| | Ввод файла помощи (N5) | |

События:
 Для N1 — событие BitBtn1Click (приводится ниже).
 N3Click — закончить работу с приложением.
 N4Click — перейти к работе со второй формой.
 N5Click — перейти к работе с третьей формой.
2 — компонент TLabel1.
 Свойства:
 Font — сиреневый, размер 12.
3 — компонент TEdit1.
 Свойства:
 Text — очистить.
 События:
 Edit1KeyPress — защита от ввода недопустимых символов.
4 — компонент TBitBtn1
 Свойства:
 Caption — Game;
 Glyph — Picture.Vmp.
 События:
 BitBtn1Click — ввод плана помещений из файла и организация структуры
 “Связанный список”. Выводит сообщение в метку Label1 и делает видимыми:
 Button1, Label1, Edit1.
5 — компонент TBitBtn2.
 Свойства:
 Kind — bkClose
6 — компонент TPanel1.
 Свойства:
 Align — alClient;
 Caption — очистить;
 Visible — False.
7 — компонент TPanel2.
 Свойства:
 Align — alClient;
 Caption — очистить;
 Visible — False.
Компонент Image1 (на схеме не пронумерован).
 Свойства:
 Picture — Arches.Vmp;
 Stretch — True;
Компонент Button1 (закрыт компонентом BitBtn1);
 Свойства:
 Caption — Продолжить.
 События:
 Button1Click — ввод номера помещения, ввод имени файла помощи,
 показывает Panel2, прячет Edit1, Label1, Button1.

Контейнер Panel1. (Рис. 9).



Рис. 9. Компонент Panel1

Перечисляются компоненты, расположенные в контейнере Panel1, их свойства и события.

Далее приводятся графические изображения других панелей и окон, перечисляются компоненты и их свойства.

Написание программы

Программную реализацию разработанных алгоритмов содержат обработчики событий. На этапе разработки рабочего проекта необходимая степень детализации алгоритмов обычно выбирается такой, чтобы предписания разработанных алгоритмов могли записываться на языке программирования, выбранном для составления текста программы. При детализации алгоритма необходимо перейти к обозначениям, принятым для разработки программ на алгоритмическом языке. При этом имена следует выбирать таким образом, чтобы они отражали сущность используемых параметров.

Кодирование должно быть простым. Изощренное программирование может обойтись слишком дорого при отладке и модификации программы. Необычное кодирование (например, использование скрытых возможностей машины) часто препятствует отладке программы и затрудняет ее использование другими программистами. Программа должна быть по возможности универсальной. Универсальные программы обеспечивают независимость программы от конкретного набора данных. Например, универсальная программа использует в качестве параметров переменные, а не константы, обрабатывает вырожденные случаи и т. д. Универсальность программы экономит время по дальнейшей работе над ней и обеспечивает широкие возможности по использованию. Разрабатывая такие программы, можно предвидеть будущие изменения в спецификациях этой программы.

Входные форматы должны быть разработаны с учетом максимального удобства для пользователя и минимальной возможности ошибок. Порядок переменных и форматы данных, привычные для пользователя, помогут избежать ошибок и облегчат использование программ.

При написании программы следует применять операторы, позволяющие использовать основные алгоритмические структуры. Оператор Goto желательно не использовать.

При написании программ не следует забывать о хорошем стиле программирования. После заголовка процедуры или функции записывается комментарий, содержащий поясняющий текст, а именно: назначение подпрограммы; перечень и назначение формальных параметров, их тип. Комментариями должны быть снабжены и основные смысловые блоки программы или подпрограммы. Для облегчения чтения текста программы отдельные операторы программы записываются с отступом.

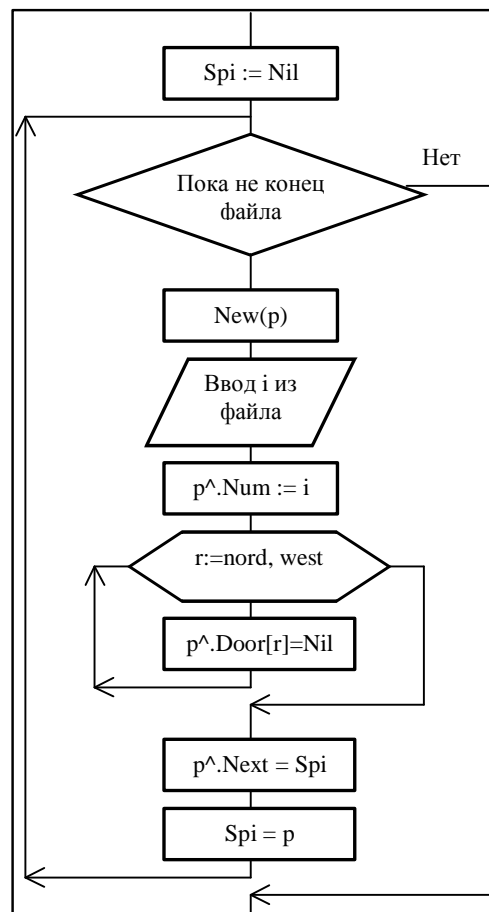


Рис. 10. Построение связанного списка

Пример.

Обработчик BitBtn1Click.

Для построения детальной схемы алгоритма необходимо определить структуру элемента списка и ввести обозначения.

Предлагается следующая структура элемента списка помещений:

Room = Record Num : Integer;

Door : Array [Trend] Of Uk;

Next : Uk;

End;

Поле Num хранит номер помещения; массив Door предназначен для хранения ссылок на смежные комнаты; поле Next содержит ссылку на следующий элемент списка. Trend — это перечисляемый тип данных, имеющий следующие значения: nord, ost, zued, west. Эти значения соответствуют направлениям дверей, которые могут быть расположены в каждом помещении. Указатель на начало списка помещений обозначим идентификатором Spi. Spi — глобальная переменная, которая описана в головной части программы.

Схема алгоритма построения связанного списка приведена на рис. 10.

Схема алгоритма построения многосвязанного списка приведена на рис. 11.

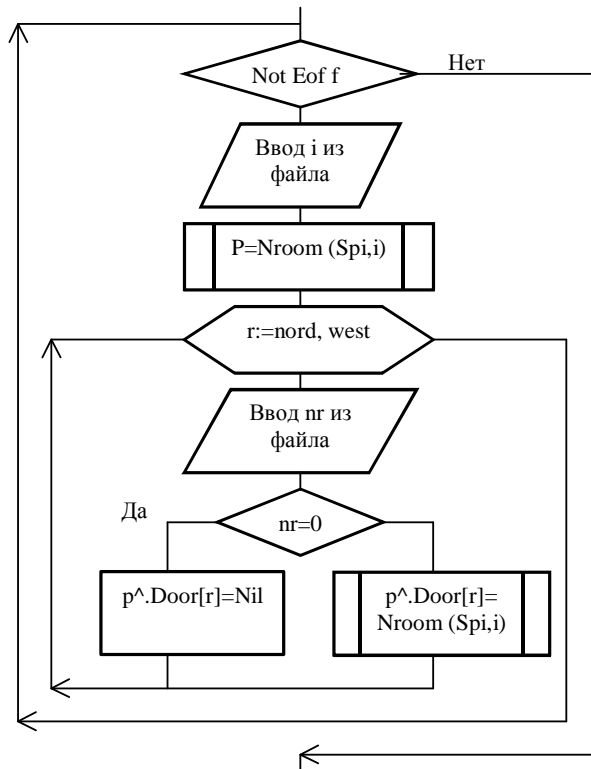


Рис. 11. Схема алгоритма построение многосвязанного списка, содержащего в себе план помещений здания

```

Программа.
procedure TForm1. BitBtn1Click (Sender: TObject);
// Построение связанного списка помещений
Var FileName : TNameFile; f : TextFile;
    p : Uk; i, nr: Integer;
begin
    FileName := InputBox('План помещений', 'Введите имя файла ','');
    AssignFile(f, FileName);
    Reset(f);
//Считываем только номера помещений и составляем связанный список
    Spi := nil;
    While not Eof(f) Do
        Begin
            New(p);
            Readln(f, i);
            p^.Num := i;
            For r := nord To west Do p^.Door [r] := nil;
            p^.next := Spi;
            Spi := p;
        End;
//Еще раз считываем из файла, но уже с дверьми
    Reset (f);
    While not Eof(f) Do
        Begin
            Read(f, i);
            p := Nroom(Spi, i);
            For r := nord To west Do
                Begin Read(f, nr);
                    If nr = 0 Then p^.Door[r] := nil
                        Else p^.Door[r] := Nroom(Spi, nr);
                End;
        End;
// Подготовка формы для ввода номера начала помещения
    label1.Caption := 'Введите № комнаты, с которой хотите начать движение';
    Edit1.Visible := True;
    label1.Visible := True;
    Button1.Visible := True;
end;

```

Функция Nroom — поиск ссылки на помещение с заданным номером. В начинающемся в Spi списке помещений по номеру помещения N отыскивается нужное. Значение функции указывает на это помещение. Схема алгоритма приведена на рис. 12.

```

Программа
Function Nroom(Spi : Uk; n : Integer): Uk;
// В списке помещений находим указатель на n - ое
Var q : Uk;
Begin
    q := Spi;
    While (q<>nil) and (q^.num<>n) Do q := q^.next;
    Nroom := q;
End;

```

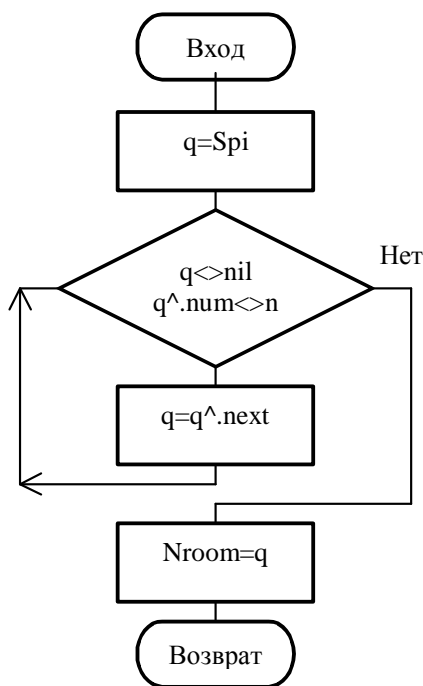


Рис. 12. Схема алгоритма функции Nroom

Аналогично уточняются все алгоритмы и разрабатываются обработчики событий.

Примечание. Не следует представлять в виде схем алгоритмов линейные вычислительные процессы. В детализированном виде представляются алгоритмы, имеющие сложную структуру. Сложной структурой будем считать алгоритмы, содержащие более одного разветвления или более одного цикла.

3.3.2. Спецификация программы

В разделе спецификация приводится точное название программы и ее состав. Форма спецификации приведена в приложении 2. Графы спецификации заполняют следующим образом:

в графе “Обозначение” указывают обозначение основных программных компонентов;

в графе “Наименование” указывают полное наименование соответствующего компонента;

в графе “Примечание” — дополнительные сведения, относящиеся к записанным в спецификации программам.

Пример.

Исполняемый файл программы “Игра “Лабиринт”” имеет название Labirint.exe и расположен на диске F в каталоге Student\Kurosov\Labirint. Состав проекта:

Наименование	Обозначение	Примечание
Ad	Файл плана помещений	Создается в приложении
Help1	Файл помощи	Создается в приложении
Labirint.Dof	Файл параметров проекта	Содержит текущие установки проекта: настройки компилятора и компоновщика, имена служебных каталогов, условные директивы
Labirint.Dpr	Файл проекта	Связывает все файлы, из которых состоит приложение
Labirint.Dsk	Файл, содержащий Desktop—настройки проекта	Содержит информацию о том, какие окна открыты и в каких позициях они расположены
Labirint.Res	Файл ресурсов	Содержит пиктограммы, графические изображения
Unit1.Pas	Файл программного модуля для формы №1	Определяет функциональность формы №1
Unit2.Pas	Файл программного модуля для формы №2	Определяет функциональность формы №2
Unit3.Pas	Файл программного модуля для формы №3	Определяет функциональность формы №3
Unit1.Dfm	Файл формы №1	Содержит список свойств всех компонентов, включенных в форму №1
Unit2.Dfm	Файл формы №2	Содержит список свойств всех компонентов, включенных в форму №2
Unit3.Dfm	Файл формы №3	Содержит список свойств всех компонентов, включенных в форму №3
Unit1.Dcu	Объектный файл для Unit1.Pas	Откомпилированная версия Unit1.Pas
Unit2.Dcu	Объектный файл для Unit2.Pas	Откомпилированная версия Unit2.Pas
Unit3.Dcu	Объектный файл для Unit3.Pas	Откомпилированная версия Unit3.Pas

3.3.3. Текст программы

Приводится полный листинг программы.

Пример листинга:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
```

```

StdCtrls, Buttons, Menus, ExtCtrls;
type
TForm1 = class(TForm)
  Image1: TImage;
  MainMenu1: TMainMenu;
  N1: TMenuItem;
  N2: TMenuItem;
  N3: TMenuItem;
  BitBtn1: TBitBtn;
  BitBtn2: TBitBtn;
  Panel1: TPanel;
  Label1: TLabel;
  Edit1: TEdit;
  Label2: TLabel;
  Label3: TLabel;
  BitBtn3: TBitBtn;
  Label4: TLabel;
  BitBtn4: TBitBtn;
  BitBtn5: TBitBtn;
  BitBtn6: TBitBtn;
  BitBtn7: TBitBtn;
  BitBtn8: TBitBtn;
  Panel2: TPanel;
  Memo1: TMemo;
  ListBox1: TListBox;
  Button1: TButton;
  N4: TMenuItem;
  N5: TMenuItem;
  Label5: TLabel;
  procedure BitBtn1Click(Sender: TObject);
  procedure Edit1KeyPress(Sender: TObject; var Key: Char);
  procedure BitBtn3Click(Sender: TObject);
  procedure BitBtn7Click(Sender: TObject);
  procedure ListBox1Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure N4Click(Sender: TObject);
  procedure N3Click(Sender: TObject);
  procedure N5Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var Form1: TForm1;
//Перечисляемый тип для определения направления движения
Type Trend = (nord, ost, zued, west);
// Структура элемента списка
Uk = ^Room;
  Room = Record Num : Integer;
           Door : Array [Trend] Of Uk;
           Next : Uk;
  End;
// Структура компонента файла помощи
THelp = Record v : String[100];

```

```

        otv : Array [1..4] Of String[100];
    End;
    TNameFile = String[20];
    Var Spi, pp, Iroom : Uk;
        r: Trend; ff: File Of THelp;
    // Spi — указатель на начало списка; pp, Iroom — рабочие указатели;
    // r — для определения направления движения;
    // ff — файловая переменная для обращения к файлу помощи.
    implementation
        Uses Unit2, Unit3;
    {$R *.DFM}
    Function Nroom(Spi : Uk; n : Integer): Uk;
    // В списке помещений находим указатель на n - ое
    Var q : Uk;
    Begin
        q := Spi;
        While (q<>nil) and (q^.num<>n) Do q := q^.next;
        Nroom := q;
    End;

    procedure TForm1.BitBtn1Click(Sender: TObject);
    // Построение связанного списка помещений
    Var FileName : TNameFile; f : TextFile;
        p : Uk; i, nr: Integer;
    begin
        FileName := InputBox('План помещений', 'Введите имя файла ', '');
        AssignFile(f, FileName);
        Reset(f);
        //Считываем только номера помещений и составляем связанный список
        Spi := nil;
        While not Eof(f) Do
            Begin
                New(p);
                Readln(f, i);
                p^.Num := i;
                For r := nord To west Do p^.Door [r] := nil;
                p^.next := Spi;
                Spi := p;
            End;
        //Еще раз считываем из файла, но уже с дверьми
        Reset (f);
        While not Eof(f) Do
            Begin
                Read(f, i);
                p := Nroom(Spi, i);
                For r := nord To west Do
                    Begin Read(f, nr);
                        If nr = 0 Then p^.Door[r] := nil
                            Else p^.Door[r] := Nroom(Spi, nr);
                    End;
            End;
        // Подготовка формы для ввода номера начала помещения
        labell.Caption := 'Введите № комнаты, с которой хотите начать движение';
        Edit1.Visible := True;
    end;

```

```

label1.Visible := True;
Button1.Visible := True;
end;
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
// Защита от недопустимых символов (вводятся только цифры)
begin
If not (key in ['1'..'9']) Then Key := #27;
end;

procedure TForm1.BitBtn3Click(Sender: TObject);
// Перемещение. BitBtn3 — север; BitBtn4 — запад;
//BitBtn5 — восток; BitBtn6 — юг.
Var aa: String[8]; c: Integer; pp: Uk;
//aa — имя кнопки; c — номер кнопки;
//pp — указатель текущего помещения.
begin
pp := Iroom;
aa := (sender as TBitBtn).Name;
c := StrToInt(Copy(aa,7,1));
Case c of
3: r := nord;
5: r := ost;
6: r := zued;
4: r := west;
End;
Iroom := Iroom^.Door[r];
If Iroom = nil Then
Begin ShowMessage('Это невозможно!');Iroom := pp; End
Else Label3.Caption := IntToStr(Iroom^.Num);
Label4.Caption := IntToStr(Iroom^.Num);
If Iroom^.Num = 999 Then
Begin ShowMessage('Свобода!!!');
Panel1.hide; label1.Hide; Button1.Hide;
Exit End;
end;
procedure TForm1.BitBtn7Click(Sender: TObject);
// Вывод содержимого справки в компоненты панели Panel2
Var i: Integer; a: THelp;
Begin
Panel1.Visible := False;
Panel2.Visible := True;
Label5.Caption := IntToStr(Iroom^.Num);
// Устанавливаем указатель текущего компонента файла
Reset(ff); Seek(ff, Iroom^.Num - 1);
Read(ff, a); Memo1.Lines.Add(a.v);
For i := 1 To 4 Do ListBox1.items[i]:= a.otv[i];
end;
procedure TForm1.ListBox1Click(Sender: TObject);
//Определение направления путем выбора верного ответа в списке
Var r1: Trend;
begin
pp := Iroom;
Panel2.Visible := False;
Panel1.Visible := True;

```

```

for r1 := nord To west Do
If ListBox1.ItemIndex = ord(r1) + 1 Then r := r1;
  Iroom := Iroom^.Door[r];
  If Iroom = nil Then
    Begin ShowMessage('Это невозможно!');Iroom := pp; End
    Else Label3.Caption := IntToStr(Iroom^.Num);
Label4.Caption := IntToStr(Iroom^.Num);
  If Iroom^.Num = 999 Then
    Begin Writeln('Свобода!!!');
    Panel1.hide; label1.Hide; Button1.Hide;
    Exit End;
end;
procedure TForm1.Button1Click(Sender: TObject);
// Ввод номера начала помещения
Var nr: Integer; FileName : TNameFile;
// nr — номер помещения для начала движения
// File Name — имя файла помощи
begin
If Edit1.Text = '' Then Exit;
nr := StrToInt(Edit1.Text);
Iroom := Nroom(Spi, nr);
If Iroom = nil Then
  Begin
    ShowMessage('Такого помещения нет!');
    Edit1.Text := ''; Label1.Caption := '';
    Exit;
  End;
panel1.Visible := True;
Label3.Caption := IntToStr(Iroom^.Num);
Label4.Caption := IntToStr(Iroom^.Num);
FileName := InputBox('Имя файла помощи', 'Введите имя файла помощи','');
AssignFile(ff, FileName);
Edit1.Hide; Label1.Hide; Button1.Hide;
end;
procedure TForm1.N4Click(Sender: TObject);
// Переход к форме №2, осуществляющей запись плана помещений на
диск.
Begin
Form1.Hide; Form2.Show;
End;
procedure TForm1.N3Click(Sender: TObject);
//Закреть приложение
begin
Form1.Close; Form2.Close; Form2.Close;
Application.Terminate;
end;
procedure TForm1.N5Click(Sender: TObject);
//Переход к форме №3, осуществляющей запись на диск файла помощи.
begin
Form1.Hide; Form3.Show;
end;
end.
Далее приводятся листинги остальных модулей.

```

3.3.4. Описание программы

Раздел “Описание программы” согласно ГОСТ 19.402–78* должен содержать следующие подразделы:

- общие сведения;
- функциональное назначение;
- описание логической структуры;
- используемые технические средства;
- вызов и загрузка;
- входные данные;
- выходные данные.

Отдельные разделы можно объединять. Некоторые пункты этого раздела повторяют разделы технического проекта. Такие повторения предусмотрены ГОСТом, так как на этапе рабочего проекта возникают некоторые дополнения или изменения в составе технических средств или программе. Здесь приводятся более конкретные и точные данные.

В подразделе “Общие сведения” должны быть указаны: обозначение и наименование программы; программное обеспечение, необходимое для функционирования программы; языки программирования, на которых написана программа.

В подразделе “Функциональное назначение” должны быть указаны классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение.

В подразделе “Описание логической структуры” должны быть указаны используемые методы; структура программы с описанием функций составных частей и связи между ними; связи программы с другими программами. Описание логической структуры программы выполняется с учетом текста программы на исходном языке.

В подразделе “Используемые технические средства” должны быть указаны типы ЭВМ и устройств, которые используются при работе программы.

В подразделе “Вызов и загрузка” должны быть указаны способ вызова программы с соответствующего носителя данных, входные точки в программу.

В подразделе “Входные данные” должны быть указаны: характер, организация и предварительная подготовка входных данных, формат, описание и способ кодировки входных данных.

В подразделе “Выходные данные” должны быть указаны: характер, организация и предварительная подготовка выходных данных, формат, описание и способ кодировки выходных данных.

3.3.5. Тестирование программы

Кратко описывается среда программирования. Приводятся основные команды, выполняемые при вводе и редактировании программы, команды записи программы на диск, чтения с диска. Перечисляются и описываются средства отладки.

Перечисляются требования, подлежащие проверке при испытании программы, а также порядок и методика их контроля. Приводятся исходные данные для решения контрольного примера и ожидаемые результаты.

Прилагается распечатка решения контрольного примера. Распечатка должна содержать фамилию, имя и отчество исполнителя, группу и дату.

Пример.

При разработке программы “Лабиринт” в качестве контрольного примера может быть использован пример, приведенный в разделе “Постановка задачи”.

3.4. Внедрение

В разделе описываются (руководство оператора ГОСТ 19.505–79):

- условия выполнения программы;
- выполнение программы;
- сообщения оператору.

В разделе “Условия выполнения программы” должны быть указаны условия, необходимые для выполнения программы (минимальный и/или максимальный состав аппаратурных и программных средств и т.п.).

В разделе “Выполнение программы” должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы. В разделе приводятся сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения. Перечисляется порядок и последовательность ввода исходных данных и получения результатов расчета.

В разделе “Сообщения оператору” должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора.

Содержание разделов допускается иллюстрировать поясняющими примерами, таблицами, схемами.

3.5. Литература

В разделе перечисляется литература, использованная при выполнении курсовой работы. При оформлении литературы необходимы следующие данные:

- фамилии и инициалы авторов через запятую;
- название книги или статьи;
- место издания (сокращенно);
- издательство или название журнала;
- год издания (для журнала добавляется номер журнала).

Источники нумеруются для того, чтобы на них можно было сослаться из текста пояснительной записки.

Например:

1. Фаронов В. В. Delphi 4. Учебный курс. -М.: Издательство “Нолидж”, 1998.
2. Дарахвелидзе П., Марков Е. Программирование в Delphi 4. – СПб.: БХВ — СПб., 1999.

4. ОФОРМЛЕНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Курсовая работа выполняется на стандартных листах формата 204 x 285. Текст пишут на одной стороне листа с полями: левое поле – 35 мм; правое – 10 мм; размеры нижнего и верхнего полей 20 – 25 мм. На титульном листе (см. приложение 1) указываются институт, кафедра, наименование изучаемой дисциплины, тема курсовой работы, фамилия и инициалы студента, группа, а также фамилия и инициалы руководителя.

Изложение пояснительной записки курсовой работы должно быть кратким, четким. В тексте допускаются общепринятые сокращения. Исходные данные, цифровые материалы, результаты решения необходимо свести в таблицы, форма которых либо задана, либо разработана студентом. Схемы алгоритмов и другой иллюстративный материал выполняются в карандаше с четким изображением, с обязательным использованием чертежных принадлежностей и размещаются сразу после ссылки на них в тексте. Нумерация страниц, рисунков и таблиц сквозная. Рисунки должны иметь подрисуночную подпись. Например, “Рис. 3. Схема алгоритма подпрограммы”. Каждая таблица должна иметь наименование,

под которым указывается “Таблица. . .” и ее порядковый номер. Ссылка на таблицу дается сокращенно – (табл. 3).

5. ТЕМАТИКА КУРСОВЫХ РАБОТ

5.1. Базы данных

1. Создайте программу введения базы данных “Сотрудники” научного учреждения “Прогресс”. База данных состоит из двух файлов. Файл № 1 – список сотрудников: фамилия, код должности, подразделение. Файл № 2 – справочник: наименование должности, зарплата за один час работы. Количество отработанных часов для каждого сотрудника вводятся с клавиатуры компьютера.

Ведение базы данных включает в себя следующие пункты:

- а) ввод информации о сотрудниках и запись ее на диск;
- б) удаление ненужной информации с файлов на диске;
- в) корректирование записей базы данных;
- г) вывод расчетной ведомости для каждого подразделения;
- д) расчет суммы выплаты заработной платы по институту в целом.

Расчетная ведомость имеет вид:

№	Фамилия	Начислено	Подходный налог	Отчисления в пенсионный фонд	К выплате
---	---------	-----------	-----------------	------------------------------	-----------

2. Создайте программу ведения базы данных торговой фирмы. Программа включает в себя: формирование и корректирование файлов данных; расчет комиссионного вознаграждения сотрудников фирмы. Файл данных о продавце включает его имя и фамилию, табельный номер, дату поступления на работу. Торговая фирма выплачивает продавцам комиссионное вознаграждение в размере 5%, если товара продано на сумму менее 1000 долл. в день, и 6%, если выручка составляет 1000 долл./день и выше. Продавцы, проработавшие в фирме более 10 лет, получают комиссионные на 1% больше. Сумма выручки за день для каждого продавца вводится с клавиатуры ЭВМ. Организуйте вывод общих итогов по сумме выручки и сумме комиссионного вознаграждения за месяц.

3. Разработать алгоритм и построить приложение справочной службы аэропорта. Программа должна выдавать справки об авиарейсах из Москвы в Санкт-Петербург. По требованию сообщать следующую информацию: о количестве свободных мест на заданное число; о проданных местах на заданное число; о проданных местах на весь месяц.

4. Разработать приложение “Помощник экзаменатора”. Экзаменационные вопросы и ответы к ним хранятся в файлах на диске. Каждый вопрос имеет балл сложности. Необходимо подобрать пять вопросов из разных разделов курса, имеющих в сумме балл сложности N, и вывести их на экран. Предусмотреть тренировочный режим работы, когда возможен вывод ответов на представленные вопросы. Доступ к тренировочному режиму работы предоставляется по паролю.

5. Разработать справочную систему по стандартным функциям Турбо Паскаля.

6. Разработать справочную систему по операторам языка Турбо Паскаль.

7. Разработать справочную систему по визуальным компонентам интегрированной среды разработки Delphi.

8. Разработать справочную систему по стандартным процедурам Object Pascal.

9. В альпинистском клубе ведется хроника восхождений. Записываются даты начала и завершения каждого восхождения, имена и адреса участвовавших в нем альпинистов, название и высота горы, страна и район, где она расположена. Создайте программу ведения базы данных альпинистского клуба, включающую в себя: а) ввод и корректирование исходных данных; б) вывод информации по запросам. Запросы:

1. Фамилии и адреса альпинистов, покоривших самую высокую вершину.

2. Наименование вершины, потребовавшей самого длительного восхождения, а также страны, где данная вершина расположена.

3. Фамилия и адрес альпиниста, совершившего наибольшее количество восхождений.

Для хранения информации использовать два файла.

10. Разработать программу ведения базы данных риэлторской фирмы. Данные о продаже квартир хранятся в двух файлах – основном и справочном. Основной файл содержит сведения о сделках и имеет следующие поля: дата сделки, покупатель, идентификационный номер продаваемой квартиры, цена. Файл-справочник содержит следующую информацию об уже проданных квартирах: общую площадь квартиры, число комнат. Номер записи в файле-справочнике соответствует идентификационному номеру квартиры. Предусмотреть ввод/корректирование исходной информации и вывод ведомости о сумме продаж, совершенных в каком-либо году или в каком-либо месяце. Ведомость о сумме продаж имеет вид:

Месяц	Сумма продаж	По числу комнат в квартире		
		1	2	3
Январь				
...				
Итого:				

5.2. Динамические структуры

1. Построить имитационную модель бензоколонки. На бензоколонке K стоек (1 стойка может обслуживать 1 автомобиль), каждый автомобиль обслуживается S сек. Интервал между моментами прибытия на бензоколонку автомобилей является случайной величиной, распределенной по закону $P(x)$. Если все стойки заняты, автомобиль становится в очередь. Для заданных $P(x)$ и S определить возможно меньшее значение K для того, чтобы очередь не удлинялась.

2. Написать подпрограмму-функцию $Form(S, X)$, где S – строка, X – вещественная переменная. В строке записано арифметическое выражение, содержащее переменную X , константы (целые или вещественные), операции $+$, $-$, $*$, $/$. Порядок операций определен скобками. Подпрограмма-функция возвращает значение арифметического выражения при заданном значении X .

3. Написать подпрограмму-функцию $Form(S, X, Y)$, где S – строка, X и Y – вещественные переменные. В строке записано арифметическое выражение,

содержащее переменные X и Y , константы (целые или вещественные), операции $+$, $-$, $*$, $/$. Порядок операций определен скобками. Подпрограмма–функция возвращает значение арифметического выражения при заданных значениях X и Y .

4. Задано выражение в постфиксной форме (обратная польская запись). Вычислить значение этого выражения для заданных значений входящих в него переменных.

5. Составить программу решения “задачи коммивояжера”. Необходимо определить минимальную стоимость проезда коммивояжера по N городам с возвращением в исходную точку. Каждый город входит в маршрут только один раз. Предположить, что стоимость проезда из города i в город j такая же, как и из j в i .

6. Разработать программу для составления списка заданий для параллельных процессоров. Три одинаковых центральных процессора могут выполнять M заданий. Каждое задание может быть выполнено на любом процессоре, и, если задание загружено в процессор, оно находится в нем до полного завершения (т.е. задания не могут прерываться или разделяться между двумя или более процессорами). При $i = 1, \dots, M$ задание i требует времени t_i для его выполнения. Для любого порядка заданий следующее задание из списка выполняется на первом освободившемся процессоре. Определить оптимальный порядок заданий, т.е. такой, который дает возможность завершить все задания в кратчайшее время.

7. Разработайте программу решения двух задач по работе с мультисписками. Даны две разреженные матрицы, хранящиеся в виде мультисписков. Напишите: 1) процедуру получения третьего мультисписка, являющегося матрицей–суммой первых двух; 2) процедуру удаления N -ой строки матрицы.

8. Разработайте процедуру исключения вершины из двоичного дерева.

9. Напишите программу, удаляющую из матрицы $[A]$ строку и столбец, содержащие наибольший элемент матрицы. Матрица $[A]$ является разреженной и хранится в виде мультисписков.

10. Написать программу, которая представит заданное арифметическое выражение в виде обратной польской записи и вычислит его значение. Для решения задачи использовать динамическую структуру стек

5.3. Игры

1. Напишите программу, которая генерирует или считывает шахматную позицию и определяет, не находится ли один из королей под шахом и не является ли шах матом. В программе предусмотреть два варианта ввода исходных данных: 1) шахматная позиция генерируется с помощью датчиков случайных чисел; 2) шахматная позиция вводится с клавиатуры ЭВМ.

2. Поле шахматной доски задается парой натуральных чисел: первое указывает номер вертикали при счете слева направо, второе – номер горизонтали при счете снизу вверх. Расстановка фигур задается таким образом, что в начале указываются поля, на которых стоят перечисленные белые фигуры, затем – поля, на которых стоят перечисленные черные фигуры.

А. На доске стоят два ферзя. Указать поля, на которые может пойти белый ферзь так, чтобы не попасть под удар черного ферзя.

Б. У белых на доске остался только король, у черных – король, конь, слон. Охарактеризовать положение белых с помощью слов: мат, шах, пат, обыкновенная позиция.

В. Получить m расстановок 8 ферзей на шахматной доске, при которых ни один из ферзей не угрожает другому.

3. Напишите программу составления кроссвордов. Исходными данными является конфигурация 6 на 6 (некоторое расположение пустых и заполненных квадратов) и список слов, состоящих из шести или менее букв. Результатом должно быть расположение этих слов, образующее общепринятый кроссворд, или сообщение о том, что такая конфигурация невозможна.

4. Разработать программу, моделирующую игру. Игра имеет следующие правила. Перед Вами большое число ящиков с деньгами. Сумма денег в каждом ящике – случайная величина, равномерно распределенная на отрезке $[0, 1]$. Вы выбираете ящик, открываете его и или берете деньги из ящика, или отказываетесь от них. Если Вы берете деньги, игра кончается. В противном случае Вы можете выбрать другой ящик. Эта процедура повторяется максимум до пяти ящиков (деньги из пятого ящика должны быть взяты, если он открыт).

5. Разработайте программу моделирующей игры. Два игрока, “нечетный” и “четный”, по очереди ставят единицы и нули в незанятые позиции поля N на N . Каждый из игроков может ставить 1 или 0 в произвольную свободную позицию, тем самым занимая ее. Игра продолжается до заполнения всех позиций. После этого суммируются числа вдоль каждой строки, каждого столбца и главных диагоналей. Число ODD нечетных сумм сравнивается с числом EVEN четных сумм. Если $ODD > EVEN$, выигрывает “нечетный”; если $EVEN > ODD$, выигрывает “четный”; если $ODD = EVEN$, результат считается ничейным. Если одним из игроков является ЭВМ, то постройте для нее выигрышную стратегию.

6. Разработать программу, моделирующую игру “Кости”. Играющий называет любое число в диапазоне от 2 до 12 и ставку, которую он делает в этот ход. Программа с помощью датчика случайных чисел дважды выбирает числа от 1 до 6 (“бросает кубик”, на гранях которого цифры от 1 до 6). Если сумма выпавших цифр меньше 7 и играющий задумал число меньше 7, он выигрывает сделанную ставку. Если сумма выпавших цифр больше 7 и играющий задумал число больше 7, он также выигрывает сделанную ставку. Если играющий угадал сумму цифр, он получает в четыре раза больше очков, чем сделанная ставка. Ставка проиграна, если не имеет место ни одна из описанных ситуаций. В начальный момент у играющего 100 очков. В программе должно присутствовать графическое изображение поверхности кубика при каждом ходе игрока.

7. Разработать программу, моделирующую игру “Морской бой”. На поле 10 на 10 позиций стоят невидимые вражеские корабли: 4 корабля по 1 клетке, 3 корабля по 2 клетки, 2 корабля по 3 клетки, 1 корабль в 4 клетки. Необходимо поразить каждую из клеток кораблей. Два игрока вводят позиции кораблей в виде цифр (1, 2, 3, 4) в соответствующие элементы матрицы, тем самым определяя конфигурацию и положение кораблей. Игроки по очереди “наносит удары” по кораблям противника. Если позиция корабля указана верно, то она помечается крестиком на поле. Предусмотреть вариант игры, когда одним из играющих является ЭВМ.

8. Разработать программу, моделирующую игру “Сбей самолет”. По экрану летят вражеские самолеты. Цель играющего — сбить их. Пусковая установка

находится в нижней строке экрана. Пусковую установку можно перемещать по строке вперед и назад.

9. Составить программу обучения работе с клавиатурой. Программа должна выдавать на экран буквы, цифры, слова и фразы, которые следует набрать на клавиатуре, и оценивать правильность и скорость набора. В программе предусмотреть три уровня подготовленности обучающегося.

10. Разработать программу, моделирующую игру “Скачки”. В игре участвуют 10 наездников; за каждый тур игры каждый из них продвигается вперед на расстояние от 1 до 5 км случайным образом. Длина дистанции — 50 км. Всего проводится 5 заездов, победителю каждого заезда начисляется 5 очков. Победителем считается наездник, набравший наибольшее количество очков во всех заездах. Перед началом заездов участник игры выбирает номер наездника, с которым он будет идентифицироваться во время игры. Количество участников игры не превышает 10. В каждом туре с вероятностью 0.1 каждый наездник может упасть, т.е. продвинуться за этот тур на ноль км. Передвижение наездников отобразить графически на экране. Предусмотреть возможность случайного распределения номеров наездников.

5.4. Строковые данные и текстовые файлы

1. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: 1) подсчитать, какие ключевые слова Паскаля и в каком количестве использованы в обрабатываемом тексте; 2) составить перечень имен простых переменных, используемых в левой части оператора присваивания; 3) представить перечень меток программы в алфавитном порядке.

2. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: 1) определить максимальную степень вложенности циклов в программе; 2) определить общее количество строк и количество символов, отличных от пробела; 3) удалить из текста программы все комментарии.

3. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: 1) первые буквы служебных слов сделать заглавными; 2) текст комментария заменить на номер комментария по порядку. Переписать текст программы в новый файл с минимальным количеством пробелов, сохранив пробелы только там, где они необходимы.

4. Создать программу, анализирующую правильность записи арифметического выражения с точки зрения синтаксиса Паскаля. Арифметическое выражение задается строковой переменной и вводится с клавиатуры компьютера.

5. Текст программы на Паскале хранится в файле на диске. Распечатать на экране текст программы таким образом, чтобы в каждой строке размещался только один оператор. Организовать смещение операторов относительно операторных скобок, как это принято в Паскале.

6. Составить программу, позволяющую сжимать текстовую информацию, а затем преобразовывать сжатую информацию в исходное состояние. В программе необходимо предусмотреть два варианта.

Для хранения текста в сжатом виде найти часто повторяющиеся

последовательности из двух букв и заменить их кодом. В качестве кода использовать символы, не встречающиеся в тексте. Составить таблицу кодов.

В заданном тексте найти слова, которые встречаются более трех раз, закодировать их и сжать текст, заменив слова кодами. Составить таблицу кодов.

7. Текстовый файл содержит данные по отпечаткам пальцев известных преступников. Разработать программу, которая печатает информацию обо всех преступниках, чьи отпечатки совпадают с отпечатками, найденными на месте преступления. Отпечаток пальцев, найденный на месте преступления, вводится с клавиатуры. Отпечаток пальцев может быть представлен кодом из 36 символов и 12 вещественных чисел, полученных в результате измерений отпечатка. Отпечатки пальцев считаются идентичными, если 35 символов из 36 совпадают и множества действительных чисел совпадают. Два действительных числа считаются “равными”, если разница между ними составляет не более 5% от большего числа. Два множества измерений считаются совпадающими, если не менее 9 из 12 пар “равны”.

8. Разработать программу “Выравнивание”. Исходная информация: текст, записанный в текстовом файле. Программа выводит этот текст с выравниванием по краям. Текст выводится без переносов слов. Параметры абзаца задаются в диалоговом режиме.

9. Текст программы на Паскале хранится в файле на диске. Считать программу с диска и получить распечатку программы. В распечатке программы отметить операторов, изменяющих значения заданных переменных (их имена вводятся). Степень вложенности операторов цикла пометить добавлением слева соответствующего количества пробелов.

10. Составить программу, определяющую наличие неописанных идентификаторов в тексте программы на Паскале. Текст программы хранится в файле на диске.

6. ЛИТЕРАТУРА

1. Гудман С., Хидетниemi С. Введение в разработку и анализ алгоритмов. М., 1981.
2. Ульман Дж. Базы данных на Паскале. М., 1990.
3. Доналд Алкок. Язык Паскаль в иллюстрациях. М., 1991.
4. Джонстон. Учись программировать. М., 1989.
5. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. М., 1982.
6. Мейер Б., Бодуэн К. Методы программирования. М., 1982.
7. Йенсен К., Вирт Н. Паскаль: руководство для пользователя. М., 1989.
8. Фаронов В.В. Турбо Паскаль. Книга 1. Основы Турбо Паскаля. М., 1992.
9. Турбо Паскаль 7.0. - Киев, 1996.
10. Абрамов С. А., Зима Е. В. Начала программирования на языке Паскаль. М., 1987.
11. Поляков Д. Б., Круглов И. Ю. Программирование в среде Турбо Паскаль. М., 1992.
12. Хершель Р. Турбо Паскаль. Вологда, 1991.
13. Перминов О. Н., Программирование на языке Паскаль. М., 1988.
14. Сергиевский М. В., Шалашов А. В., Турбо Паскаль 7.0. М., 1994.
15. Белецкий Я. Турбо Паскаль с графикой для персональных компьютеров. М., 1991.

16. Фаронов В. В. Турбо Паскаль 7.0. М., 1998.
17. Емелина Е. И. Основы программирования на языке Паскаль. М., 1997.
18. Марченко А. И. Программирование в среде Borland Pascal 7.0. Киев, 1996.
19. Фаронов В. В. Паскаль и Windows. М., 1995.
20. Руководство пользователя. Авторский коллектив. М., 1998г.
21. Мануйлов В. Г. Разработка программного обеспечения на Паскале. М., 1998г.
22. Фаронов В. В. Delphi 4. Учебный курс. М., 1998.
23. Дарахвелидзе П., Марков Е. Программирование в Delphi 4. СПб.: БХВ. СПб., 1999.
24. Федоров А. Г. Delphi 3.0 для всех. М., 1998.
25. Культин Н. Delphi 3. Программирование на Object Pascal. Киев, 1998.
26. Бронштейн И. Н., Семендяев К. А. Справочник по математике для инженеров и учащихся ВТУЗов. М., 1986.

СОВРЕМЕННЫЙ ГУМАНИТАРНЫЙ УНИВЕРСИТЕТ

Курсовая работа

По дисциплине “Алгоритмические языки и программирование”

Тема: _____

Выполнил студент: Ивочкин Константин Петрович

№ контракта 003009812034

Направление информатика

№ группы И-333

Подпись студента _____

Дата сдачи работы _____

Работа принята: Груздев Ф. А., методист _____

Подпись

Форма спецификации

Обозначение	Наименование	Примечание

.....

Примечание. Размеры таблицы допускается выполнять кратными шагу печатающих устройств.

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ
КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ**

**“АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И
ПРОГРАММИРОВАНИЕ”**

Редактор Н.С. Потемкина
Оператор компьютерной верстки Е.М. Кузнецова

Изд. лиц. ЛР № 071765 от 07.12.1998	Сдано в печать
НОУ “Современный Гуманитарный Институт”	
Тираж	Заказ
