УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра
«Программное обеспечение
информационных технологий»

Технология программирования

методические указания
для выполнения лабораторных работ
студентам направлений образования

5521900 – Информатика и информационные технологии,
5523500 – Защита информации,
5523600 – Электронная коммерция,
5811200 – Сервис (информационный сервис),
5811300 – Сервис (электронные и компьютерные технологии),
5320200 – Информатика и библиотековедение,
5140900 – Профессиональное образование (по направлению информатика и информационные технологии).

Ташкент 2009

Технология программирования (методические указания для выполнения лабораторных работ студентам направлений образования:

5521900 – Информатика и информационные технологии,
5523500 – Защита информации,
5523600 – Электронная коммерция,
5811200 – Сервис (информационный сервис),
5811300 – Сервис (электронные и компьютерные технологии),
5320200 – Информатика и библиотековедение,
5140900 – Профессиональное образование (по направлению информатика и информационные технологии)).

Настоящие методические указания являются основным руководством для выполнения лабораторных работ по дисциплине "Технология программирования" и содержат описание лабораторных работ, а также требования к их содержанию и оформлению.

ВВЕДЕНИЕ.

Методические указания для выполнения лабораторных работ по дисциплине «Технология программирования» предназначены для студентов направлений образования: 5521900 – Информатика и информационные технологии, 5523500 – Защита информации, 5523600 – Электронная коммерция, 5811200 – Сервис (информационный сервис), 5811300 – Сервис (электронные и компьютерные технологии), 5320200 – Информатика и библиотековедение, 5140900 – Профессиональное образование (по направлению информатика и информационные технологии)), являются основным руководством к выполнению лабораторных работ, содержат их описание, а также требования к их содержанию и оформлению.

Основной целью выполнения лабораторных работ является изучение технологии и методов разработки, эффективности, тестирования, отладки и эксплуатации конкретных программ. В результате выполнения лабораторных работ студенты должны научиться самостоятельно, разрабатывать программы на алгоритмическом языке программирования высокого уровня, оформлять документацию к ней и правильно ее оценивать. Следует заметить, что разработка программ может быть выполнена на любом алгоритмическом языке, в любой среде разработки по усмотрению студента - разработчика программы.

Самостоятельно разрабатывая программу, студент также должен научиться:

- проводить беседы с пользователем с целью определения требований к разрабатываемой программе и ее осуществимости;
- разрабатывать внешний и внутренний проект программы;
- выполнять тестирование и отладку разрабатываемой программы;
- составлять сопровождающую программу документацию.

С этой целью в дисциплину включены лабораторные работы:

№	Наименование темы и ее краткое содержание	Объем
1.	Предварительное проектирование программного обеспечения. Разработка постановки задачи.	4
2	Разработка программного обеспечения. Составление календарного плана разработки конкретной программы	2
3	Построение функциональной схемы системы ПО.	3
4	Внешнее проектирование программного обеспечения.	3
5	Разработка архитектуры программного обеспечения.	3
6	Описание алгоритма	3
7	Пошаговая разработка программы.	3
8	Программирование. Запись текстов программ на алгоритмическом языке высокого уровня.	4
9	Тестирование и отладка разработанной программы.	3
10	Составление документа «Руководство пользователю».	4
	Итого	32

ЛАБОРАТОРНАЯ РАБОТА № 1

Предварительное проектирование программного обеспечения

Цель работы:

- Проведение предварительного проектирования конкретной программы.
- Составить перечень требований и функциональных характеристик разрабатываемой программы.
- Разработка документа «Постановки задачи».

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо определить потребность в программном изделии, его назначение и основные функциональные характеристики; составить перечень требований к нему.

Работа должна быть оформлена в виде документа «Постановка задачи».

Теоретические сведения.

Определение полного комплекса требований к программному изделию является первоначальной задачей его разработки. Некачественное определение требований приводит к созданию программного изделия, которое будет правильно решать неверно сформулированную задачу, а программный продукт не будет соответствовать истинным потребностям заказчика.

Поэтому при определении требований к программному изделию требуется соблюдать максимально возможную аккуратность и точность, чтобы затем эти требования можно было транслировать в разрабатываемый проект с минимальным числом ошибок. Требования задаются на естественном языке и должны быть очень точно сформулированы.

Требования оформляются в виде документа, в котором письменно излагается то, что будет, и что не будет сделано при выпуске программного изделия. В учебном заведении такой документ называется " Постановка задачи ".

Постановка задачи пишется на естественном языке в терминах понятных и пользователю и разработчику программного обеспечения и может содержать следующие разделы:

1. **Заголовок к программе.**
2. **Условие задачи.**
   Формулируется условие задачи, краткое описание разрабатываемой программы, ее назначение и необходимые уточнения.
3. **Начало/окончание работы.**
   Указывается месяц и год начала/окончания разработки программы.
4. **Основание для разработки программы.**
   Основанием для разработки программы может быть заказ пользователя, задание администрации учебного заведения, контракт учебного заведения с другой организацией и пр.

5. Краткая характеристика объекта разработки.

Описывается объект разработки: как решается поставленная задача в настоящее время без разрабатываемой программы и какая часть ручной работы будет заменена программой.

6. Пользователь.

Указываются пользователи программы.

7. Цель и назначение разработки.

8. Основные требования.

Описываются требования пользователя к разрабатываемой программе.

Здесь же с точки зрения пользователя следует подробно перечислить функции программы.

9. Входная информация.

Перечисляются все входные данные программы с точки зрения их содержания и назначения - отчеты, файлы, записи, поля данных, таблицы... Их возможные носители и средства отображения информации и т.д.

10. Выходная информация.

Описываются выходные данные так же, как в пункте 9.

11. Требования к аппаратному и программному обеспечению.

Описывается конфигурация аппаратуры и программного обеспечения, в которых разрабатываемая программа может работать, другие программные продукты, от которых она зависит.

12. Внешние ограничения.

13. Эффективность.

Цели производительности, такие, как временные и объемные характеристики, пропускная способность, использование ресурсов и пр.

14. Безопасность данных от несанкционированного доступа.

15. Эргономические характеристики.

Эргономическими характеристиками изделия являются такие свойства, которые обеспечивают надежность, комфорт и продуктивность работы пользователей и операторов. Эргономика (греч.) - труд + закон - отрасль знания, изучающая трудовые процессы с целью создания наилучших условий труда.

16. Мобильность.

Описываются требования и цели обеспечения переноса программного продукта из одних рабочих условий в другие.

17. Окупаемость капиталовложений.

Определяется прибыль, которую даст создание программного продукта в понятиях, соответствующих целевому назначению организации.

18. Другие соглашения сторон.

19. Терминология.

Четко определяется вся терминология, которая может оказаться специфической для данной разработки.

ЛАБОРАТОРНАЯ РАБОТА № 2

Разработка программного обеспечения.

Цель работы:
- Определение этапов разработки конкретной программы.
- Разработка календарного плана создания конкретной программы.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо подробно проанализировать этапы разработки конкретной программы (ее жизненный цикл), начиная от возникновения потребности в ней до полного прекращения ее использования вследствие ее морального старения или потери необходимости решения соответствующих задач.

Работа должна быть оформлена в виде календарного плана разработки программы по форме:

№	Наименование этапа разработки программы	Срок исполнения		Примечания
		Начало	Окончание	

Теоретические сведения.

Обобщенная модель жизненного цикла программы может выглядеть так:

1. Системный анализ (предварительное проектирование ПИ)
- а) исследование
- б) осуществимость
  - эксплуатационная
  - экономическая
  - коммерческая

2. Проектирование программы
- а) конструирование программы
  - функциональная декомпозиция задачи
  - разработка архитектуры системы
  - внешнее проектирование программы
  - разработка архитектуры программы
  - проектирование базы данных
- б) программирование
  - внутреннее проектирование форм и модулей
  - определение свойств объектов и кодирование
  - отладка форм и модулей
  - компоновка форм и модулей в программу
- г) отладка программы в целом

3. Оценка (испытания) программы

4. Использование программного изделия

ЛАБОРАТОРНАЯ РАБОТА № 3

Построение функциональной схемы системы ПО.

Цель работы:

- проведение функциональной декомпозиции решаемой задачи;
- построение функциональной схемы;

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо провести функциональную декомпозицию решаемой задачи, построить соответствующую схему.

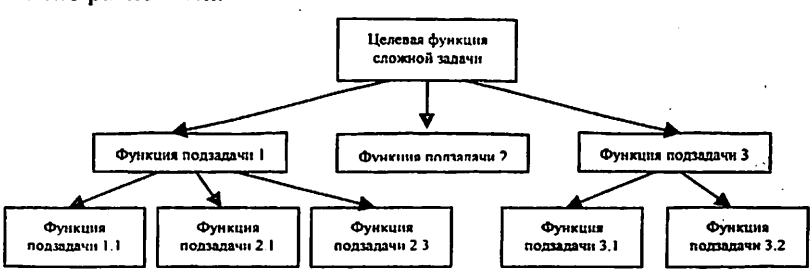
Работа должна быть оформлена в виде спецификации, содержащей функциональную схему решаемой задачи.

Теоретические сведения.

Проектирование программного обеспечения часто начинается с функциональной декомпозиции решаемой задачи.

Функциональная декомпозиция задачи представляет собой иерархическое разбиение сложной задачи на ряд проще решаемых небольших задач, которые, в свою очередь, разделяются на подзадачи до тех пор, пока каждая необходимая деталь в ней не будет определена достаточно ясно.

Концепция иерархической декомпозиции настолько естественна, что мы не всегда в состоянии осознать, как часто нам приходится использовать ее на практике. Она вытекает из человеческой потребности иметь дело с поддающимся управлению вполне определенным числом дискретных источников информации и производить «отсечение» информации до тех пор, пока число дискретных источников не станет приблизительно равно семи.



Строгая иерархическая декомпозиция подчиняется правилам:

1. На каждом уровне иерархии задача должна иметь законченный вид на данном уровне детализации;
2. На любом уровне иерархии каждое разбиение полностью охватывает отдельную задачу (функцию), соответствующую данному уровню детализации.

ЛАБОРАТОРНАЯ РАБОТА № 4

Внешнее проектирование программного обеспечения.

Цель работы:
- проведение внешнего проектирования конкретной программы;
- разработка взаимодействия разрабатываемой программы с пользователем: сценарий, экранные формы, набор подсказок, и пр.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо описать ожидаемое поведение разрабатываемой программы с точки зрения внешнего по отношению к нему наблюдателя (обычно - пользователя), то есть осуществить "конструирование" внешних взаимодействий будущей программы продукта с пользователем без конкретизации его внутреннего устройства.

Работа должна быть оформлена в виде внешней спецификации.

Теоретические сведения.

Внешнее проектирование мало, чем связано (если связано вообще) с программированием; более непосредственно оно касается понимания обстановки, проблем и нужд пользователя, психологии общения человека с машиной. Эта сторона внешнего проектирования становится все более значительной по мере того, как применение ЭВМ все больше начинает затрагивать пользователей, незнакомых с программированием.

Результаты внешнего проектирования программы отражаются во внешней спецификации, в которой может быть представлено описание следующих внешних аспектов программы:

- организация диалога программы с пользователем;
- состав меню, подменю ...;
- описание действий функциональных клавиш;
- все экранные формы или протокольные экранные сообщения;
- сообщения, выдаваемые пользователю во время проведения сеанса работы программы и выдаваемые пользователем на них ответы;
- сообщения об ошибках;
- подсказки пользователю, организация "помощи";
- структура и организация баз данных;
- описание и подготовка входных данных;
- выходные печатные формы;
- другие внешние сопряжения программы.

Внешняя спецификация должна быть написана на понятном пользователю и разработчику языке для уменьшения вероятности возможных недоразумений. Причем, проверку корректности и полноты спецификации необходимо проводить еще до начала программирования.

Основные правила организации диалога программы с пользователем.

1. Согласовывайте способ взаимодействия программы с пользователем, с его подготовкой и уровнем, с ограничениями, в условиях которых он работает.
2. Выходные данные должны выдаваться программой в требуемой форме и обязательно с комментариями. Нельзя, например, выдавать их в виде числа, а тем более - в виде набора чисел.
3. Обеспечьте концептуальную целостность для разных типов вводимых / выводимых сообщений. Например, все сообщения выдачи на экран, отчеты должны иметь одинаковые форматы, стиль и сокращения.
4. Старайтесь, чтобы пользователь вводил данные с клавиатуры как можно меньше. Будет лучше, если ему будет дана возможность выбора вводимых данных в виде меню, что исключит ошибки ввода пользователем. Сообщения, вводимые пользователем, должны быть как можно короче, но не настолько, чтобы исчезла их осмысленность.
5. Обеспечьте средства "помощи" - специальный набор функций (подсказки) по оказанию пользователю помощи, если тот запутался или забудет какое-либо правило взаимодействия.
6. Старайтесь, чтобы программа не рассердила пользователя. Избегайте оскорбительных сообщений. Общайтесь с пользователем на его языке, а не на тарабарском жаргоне программистов.
7. Помните о дизайне экрана. С эстетично оформленным экраном приятней работать. Экранная форма может быть разнообразной.
8. Старайтесь на каждое входное сообщение выдавать какое-либо уведомление. Программа должна принимать любые вводимые данные. Если данные не являются тем, что программа считает допустимым, то она должна информировать об этом пользователя.
9. Спроектируйте программу так, чтобы пользователь в любой момент работы с ней мог закончить эту работу или перейти в предыдущее состояние. Предполагается, что в первом случае программа успешно завершит свою работу (закроет открытые файлы, очистит переменные памяти и т.д.)
10. Ошибки пользователя должны обнаруживаться немедленно.
11. Не стремитесь исправлять входное сообщение пользователя. Например, в медицинской информационной системе пользователь случайно нажимает на лишнюю клавишу, вследствие чего входное сообщение принимает вид "Рэтиловый спирт" вместо сообщения "Этиловый спирт". Система исправляет это сообщение на "Метиловый спирт". Известно, что этиловый спирт опьяняет, а метиловый спирт убивает.
12. Любые действия пользователя, как правильные, так и неправильные, должны контролироваться программой. В качестве отрицательного примера можно привести программу, которая может вдруг аварийно, преждевременно закончить свою работу.

ЛАБОРАТОРНАЯ РАБОТА № 5

Разработка архитектуры программного обеспечения.

Цель работы:

- разработать архитектуру программного изделия.

Порядок выполнения работы и отчетность.

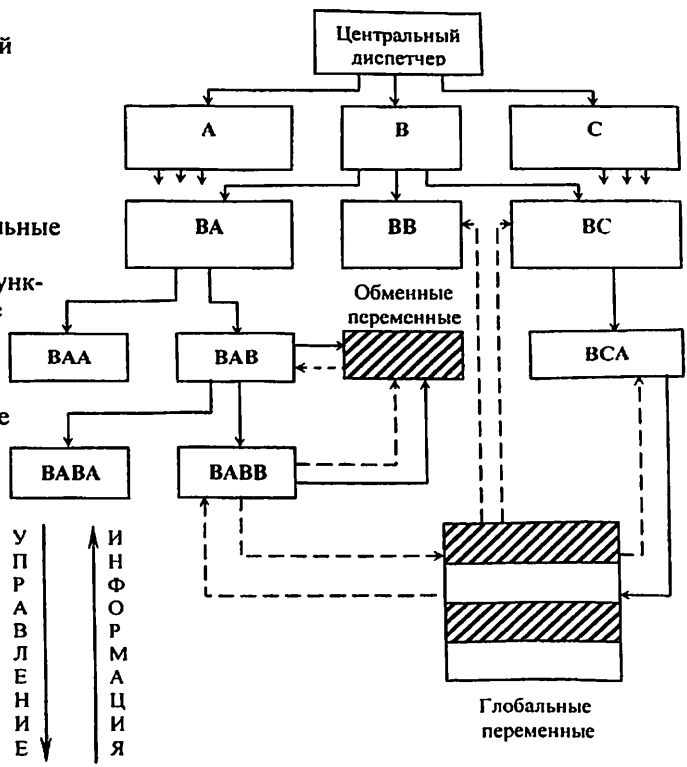
Во время выполнения лабораторной работы необходимо разработать архитектуру разрабатываемой программы: спроектировать структуру всех его компонент, его объектно - модульно - иерархическое построение.

Работа должна быть оформлена в виде спецификации, содержащей архитектуру разрабатываемой программы.

Теоретические сведения.

Типовая архитектура программы может иметь вид:

1. Уровень - центральный диспетчер
2. Уровень - местные диспетчеры
3. Уровень - функциональные программы
4. Уровень – функциональные программы
5. Уровень - стандартные программы, библиотеки программ



Общие правила структурного построения программных модулей.

1. Каждый модуль характеризуется функциональной законченностью, автономностью и независимостью в оформлении от модулей, которые его используют и которые он вызывает. Высокую степень независимости модулей можно достичь с помощью двух методов оптимизации:
   - усилением внутренних связей в каждом модуле, т.е. реализовать отдельные функции отдельными модулями (высокая прочность модуля).
   - ослаблением взаимосвязи между модулями, применяя формальный механизм передачи параметров (слабое сцепление модулей).
2. Применяются стандартные правила организации связей по управлению и информации с другими модулями (смотри далее).
3. Комплексы программ разрабатываются в виде совокупности небольших по количеству (до 100) программных модулей, связанных иерархическим образом, что дает возможность полностью и относительно просто уяснить функцию и правила работы отдельных частей и комплекса программ в целом.
4. Как правило, модуль содержит от 10 до 1000 выполняемых операторов языка высокого уровня. Размеры модуля влияют на степень независимости программы, легкость ее чтения и тестирования.
5. Модуль прочный. Прочность модуля измеряется его внутренними связями. Модуль - это замкнутая программа, которая выполняет одну или несколько функций, обладает некоторой логикой.
6. Модуль предсказуемый, т.е. модуль, работа которого не зависит от предыстории его использования. Модули не должны сохранять никаких "воспоминаний" о предыдущем вызове.
7. Определена структура принятия решений. Желательно, чтобы те модули, на которые прямо влияет принятое решение, были подчиненными (вызываемыми) по отношению к принимающему решение модулю.
8. Объем данных, на которые модуль может ссылаться, должен быть сведен к минимуму.
9. Внутренняя процедура (или подпрограмма) - это замкнутая программа, физически расположенная в вызывающем ее модуле. Их следует избегать, т.к. их трудно изолировать для автономного тестирования и они не могут быть вызваны из модулей, отличных от тех, которые их физически содержат. Когда возникает потребность во внутренней процедуре, проектировщик должен рассмотреть возможность оформления ее в виде отдельного модуля.
10. В параметры процедуры следует включать только те переменные, через которые идет обмен информацией с другими программными единицами. Другие переменные - это внутреннее дело процедуры. Процедуры, которые выдают в качестве результата только одно значение, оформляются как функции. Функция удобнее в использовании, так как ее результат непосредственно можно использовать в арифметическом и/или в логическом выражениях.

Правила связи программных модулей по управлению.

1. Передача управления вызываемому модулю всегда осуществляется через его начало, т.е. через первый оператор.
2. Выход из вызываемого модуля всегда происходит через его естественное окончание, т.е. после нормального его завершения.
3. По окончании исполнения вызываемого модуля управление передается в вызывающий модуль на оператор, следующий непосредственно за оператором вызова.
4. Модули низших уровней или одного уровня иерархии могут вызываться для исполнения только модулями высших уровней, т.е. модули низших уровней не могут вызывать модули высших уровней, а модули одного уровня - вызывать друг друга.
5. Если все же необходимо исполнить модуль с некоторой внутренней точки, то вызов все равно осуществляется стандартным образом (через его первый оператор), а точка начала задается в виде параметра. При этом в начале вызываемого модуля должен стоять переключатель, который обеспечивает передачу управления программой к его внутренним точкам по параметру, указанному при обращении к модулю.
6. В каждом модуле должна быть предусмотрена возможность подключения контрольных и отладочных средств; операторы, реализующие эти средства, обычно сосредотачиваются в конце модуля.

Правила связи программных модулей по информации.

1. Информация зон глобальных переменных доступна для использования любым модулям, входящим в комплекс программ или в группу программ в соответствии с областью действия зоны глобальных переменных, т.е. глобальные переменные, могут быть доступны не для всего комплекса программ, а лишь для указанной в описании группы модулей.
2. Локальные переменные доступны лишь в пределах того модуля, в котором они определены или объявлены.
3. Для взаимодействия вызываемых и вызывающих модулей создаются зоны обменных переменных, информация из которых доступна лишь модулям, непосредственно связанным по управлению.
4. После окончания работы вызываемого модуля считается, что соответствующие регистры не содержат информации, являющейся результатом его работы. Запрещается их использовать в вызывающем модуле.
5. Информация, находящаяся в регистрах вызывающего модуля, при вызове должна быть сохранена на период выполнения вызываемого модуля и восстановлена при возврате управления в вызывающий модуль. Сохранение регистров может осуществлять как вызывающий, так и вызываемый модуль.

ЛАБОРАТОРНАЯ РАБОТА № 6

Описание алгоритма.

Цель работы:

- разработка алгоритма решения задачи;
- запись алгоритма в блок схемной форме;

Порядок выполнения работы и отчетность.

Во время проведения лабораторной работы необходимо разработать алгоритм решения конкретной задачи и представить его в блок-схемной форме

Работа должна быть оформлена в виде спецификации, содержащей описание алгоритма конкретной программы.

Теоретические сведения.

Представленный в любой форме алгоритм пишется на естественном (разговорном) языке, используя термины отрасли, в которой решается задача и не должен содержать программистские термины; тем более алгоритм не может содержать операторы языка программирования. Текст алгоритма должен быть читаемым без дополнительных пояснений автора.

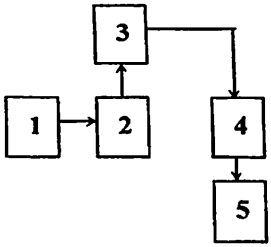
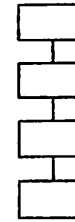
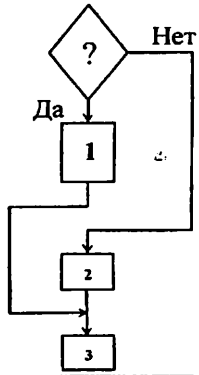
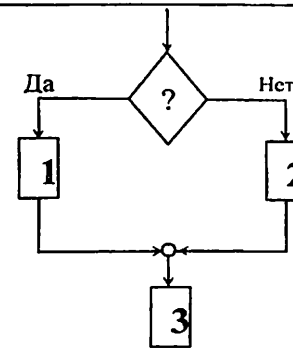
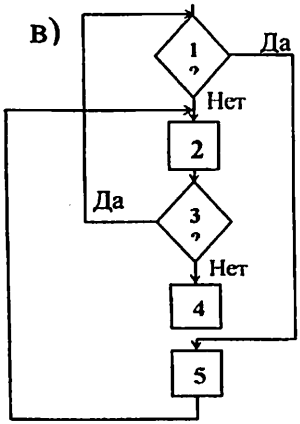
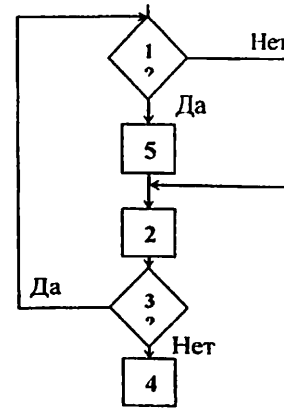
В алгоритме не показывают описания, т.к. описания - это особенности языка программирования, а не алгоритма. Алгоритм не может быть "привязан" к конкретному языку программирования. В алгоритме описываются конкретные действия - "что делается", а не "как это делается".

В блок-схеме обычно пишут "скупые" тексты. Поэтому ее дополняют пояснениями. В пояснении к схеме описывают функциональные действия всего алгоритма в целом, действия отдельных частей схемы и в случае крайней необходимости - действие отдельного блока. Описание каждого блока схемы бессмысленно. Пояснения должны содержать некоторую дополнительную информацию, а не перефразировку алгоритма.

Основные требования к блок-схеме.

- схема выполняется с соблюдением условий ГОСТа
- запись в блоках должна быть словесной или математической, а не в виде операторов языка программирования
- управление по схеме должно в основном идти вниз (вправо), возвращаясь назад только в циклах.
- альтернативно выполняемые ветви должны размещаться параллельно.
- переменные должны быть определены в каком-либо блоке.
- входные и выходные блоки процедур должны содержать, соответственно, входные и выходные (формальные) параметры.
- блоки можно объединять в более крупные пунктирными линиями, которые нужно комментировать - описать их назначения.
- на каждую подпрограмму (модуль) составляется отдельная схема.

Примеры неправильно и правильно составленных блок-схем.

Неправильно	Правильно
а) 	а) 
б) 	б) 
в) 	в) 

ЛАБОРАТОРНАЯ РАБОТА № 7

Пошаговая разработка программы.

Цель работы:

- Освоить метод пошаговой разработки программ.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо осуществить пошаговую разработку конкретной программы.

Работа должна быть оформлена в виде спецификации, содержащей описание пошаговой разработки конкретной программы.

Теоретические сведения.

Пошаговая разработка (пошаговая детализация) программы представляет собой простой процесс, предполагающий первоначальное выражение логики модуля в терминах гипотетического (условного) языка очень высокого уровня с последующей детализацией каждого предложения в терминах языка более низкого уровня, до тех пор, пока, наконец, не будет, достигнут уровень используемого языка программирования.

Достоинство пошаговой детализации состоит в том, что этот метод позволяет проектировщику упорядочить свои рассуждения - на каждом шаге детализации решается элементарная задача.

Пример метода пошаговой детализации.

ЗАДАЧА. Дана матрица размером 10*10 элементов. Для каждого столбца среди элементов, лежащих выше первого нулевого, и значения которых лежат в интервале [c,d], найти наименьший и наибольший элементы и их номера в строке. Если нулевого элемента в столбце нет, то обрабатывается весь столбец.

План решения задачи .

1. Входные / выходные переменные.
2. Основной алгоритм – перебор столбцов исходной матрицы (цикл по столбцам)
3. Перебор элементов столбца матрицы (внутренний цикл по строкам)
4. Обработка элементов матрицы
5. Поиск наибольшего и наименьшего элементов в столбце.
6. Обработка начальных и конечных операторов циклов
7. Оптимизация и шлифовка программы

Входные / выходные переменные.

A(10,10) - исходная матрица
C и D - границы интервала
max(10) - массивы, содержащие наибольшие и наименьшие значения
min(10) каждого столбца исходной матрицы.
Imax(10) - массивы, содержащие номера, строк в которых встречаются
Imin(10) найденные наибольшие и наименьшие значения в столбце.

а) Первый шаг.

Детализация ввода-вывода.

```
PROGRAMM PRIMER;
VAR
    A: ARRAY[1..10,1..10] OF REAL;        (*  Исходная матрица        *)
    C,D: REAL;                            (*  Границы интервала       *)
    I,J: INTEGER;                         (*  Номера строк и столбцов *)
    MAX,MIN: ARRAY[1..10] OF REAL;        (*  Значения наибольших     *)
                                          (*  и наименьших элементов  *)
    IMAX,IMIN: ARRAY[1..10] OF INTEGER; (*  и их номера строк *)

    Другие переменные
BEGIN
    WRITELN ('Введите элементы матрицы');
    FOR I:=1 TO 10 DO BEGIN
        FOR J:=1 TO 10 DO READ (A[I,J]);      WRITELN;
    END;
    WRITE('Введите границы интервала');       READ (C,D);

    Основной алгоритм

    FOR I:=1 TO 10 DO WRITELN ('MIN =', MIN[I], ', его номер', IMIN[I],
                               '; MAX=', MAX[I], ', его номер', IMAX[I]);
END.
```

б) Второй шаг.

Детализация основного алгоритма.

Необходимо выполнить одно и тоже для каждого столбца.

```
Начальные операторы
FOR J:=1 TO 10 DO
BEGIN
    Обработка столбца
END;
Конечные операторы
```

в) **Третий шаг.**

Обработка столбца исходной матрицы.

В этом столбце необходимо обработать элементы, лежащие выше первого нулевого элемента матрицы A.

```
Начальные операторы
FOR I:=1 TO 10  DO WHILE  A[I,J]#0  DO
BEGIN
    Обработка элементов матрицы
END;
Конечные операторы
```

г) **Четвертый шаг.**

Определение принадлежности элемента матрицы A заданному отрезку CD.

```
IF (C<A[I,J]) AND (A[I,J]<D)   THEN
BEGIN
    Поиск наибольшего элемента в столбце

    Поиск наименьшего элемента в столбце
END;
```

д) **Пятый шаг.**

Вычисление наибольшего и наименьшего элемента в столбце матрицы A и их номера в строке.

```
IF  A[I,J]>=MAXT  THEN  BEGIN  MAXT:=A[I,J];  IMAXT:=I;
END;
IF  A[I,J]<=MINT  THEN  BEGIN  MINT := A[I,J];  IMINT := I;
END;
```

Появились новые переменные: MAXT и MINT, IMAXT и IMINT, которые необходимо включить в описания переменных с присвоением типа REAL и INTEGER соответственно.

е) Шестой шаг.

Детализация начальных и конечных операторов циклов.

Движемся по программе изнутри циклов наружу и смотрим, что необходимо для работы циклов.

Чтобы сработал внутренний цикл, надо чтобы на первом шаге MAXT и MINT имели какие-то значения и что делать в случае отсутствия в столбце элементов из интервала (c,d)?

Примем, что индексы в этом случае равны нулю, а значения наибольших и наименьших элементов, попадающие в выходные массивы, несущественны.

Итак, вначале MAXT=C, MINT=D, IMAXT=0, IMINT=0.

Результаты этого цикла надо заслать в J-е элементы результирующих массивов, поэтому конечные операторы будут такими:

MAX[J]:=MAXT, MIN[J]:=MINT
IMAX[J]:=IMAXT , IMIN[J]:=IMINT

Для цикла по столбцам никаких начальных и конечных операторов не требуется.

ж) Последний шаг.

Шлифование и оптимизация программы.

Получив программу, можно заняться ее улучшением, чтобы она стала короче или выполнялась быстрее.

Например, если использовать MAX(J), MIN(J), IMAX(J), IMIN(J) вместо MAXT, MINT, IMAXT, IMINT, то строка конечных операторов не потребуется, но программа будет дольше выполняться (хотя ее текст сократится), т.к. участится выполнение операции "Обращение к элементу массива" (довольно длинная, с точки зрения ЭВМ, операция).

Можно выполнить другие оптимизационные действия.

Результирующая программа

```
PROGRAMM PRIMER;
VAR
      A: ARRAY[1..10, 1..10] OF REAL;          (* Исходная матрица  *)
      I, J: INTEGER                        (* Их номера строк и столбцов *)
      C,D: REAL;                                  (* Границы интервала *)
      MAX,MIN: ARRAY[1..10] OF REAL;   (* Значения наибольших и *)
                                                 (*  наименьших элементов *)
      IMAX,IMIN: ARRAY[1..10] OF INTEGER;  (*  и их номера строк *)
      MAXT,MINT: REAL;     (* Временные переменные, наибольший и *)
                                        (* наименьший значения элементов *)
      IMAXT,IMINT: INTEGER;                    (* и их номера в столбце *)
BEGIN
    WRITELN ('Введите элементы матрицы:');
    FOR I:=1 TO 10  DO
    BEGIN
        FOR J:=1 TO 10  DO  READ (A[I,J]);      WRITELN;
    END;
    WRITE('Введите границы интервала');          READLN (C,D);
    FOR J:=1 TO 10  DO            (* обработка столбцов матрицы A *)
    BEGIN
       IMAXT:=0;      IMINT:=0;       MINT:=C;       MINT:=D;
        (*  обработка элементов столбца матрицы A  *)
       FOR I:=1  TO  10  DO    WHILE  A[I,J]#0    DO
       BEGIN
          IF (C<=A[I,J]) AND (A[I,J]<=D) THEN   (* элемент матрицы *)
             BEGIN                                  (* принадлежит отрезку CD? *)
                IF  A[I,J]>=MAXT   (* элемент матрицы наибольший? *)
                   THEN  BEGIN  MAXT:=A[I,J];  IMAXT:=I; END;
                IF   [I,J]<=MINT      (* элемент матрицы наименьший? *)
                   THEN BEGIN MINT:=A[I,J];  IMINT:=I;  END;
             END;
       END;
       MAX[J]:= MAXT;              MIN[J]:= MINT;
       IMAX[J]:=IMAXT;             IMIN[J]:=IMINT;
    END;
    FOR I:=1 TO 10 DO WRITELN ('MIN=',MIN[I],',его номер', IMIN[I],
                          '; MAX=',MAX[I],',его номер',IMAX[I]);
END.
```

ЛАБОРАТОРНАЯ РАБОТА № 8

Запись текстов программ на алгоритмическом языке высокого уровня.

Цель работы:

- Запись текстов программ на алгоритмическом языке высокого уровня, используя правила хорошего стиля программирования.

Порядок выполнения работы и отчетность.

Во время проведения лабораторной работы необходимо написать программу на конкретном алгоритмическом языке программирования для решения конкретной задачи, используя приемы и методы программирования, которые используют, опытные программисты, чтобы получить правильные, надежные, эффективные, удобные для применения и легко читаемые программы.

В отчете должны быть представлены листинги текстов программ.

Теоретические сведения.

Программа должна быть составлена таким образом, чтобы ее могли прочитать в первую очередь люди, а не машины.

Программа - это документ для последующего использования, учебный материал по кодированию алгоритмов и средство для дальнейшей разработки более совершенных программ.

Как правило, к разработке программы приступают со скромными целями. В дальнейшем ее возможности расширяют. Трудно читаемые программы модифицировать сложно. Особенно, если это приходится делать не автору программы.

Программа должна передавать логику и структуру алгоритма настолько, насколько это возможно. Ее кодируют просто и рационально. Следует избегать всевозможных программистских трюков, т.к. чем их больше, тем труднее будет разобраться в логике программы самому автору, а кто-либо другой это сделать не сможет. На ранних этапах разработки сложной программы лучше без колебания переписать заново ее громоздкие блоки, если это ведет к ее упрощению.

Общая организация программы и ее запись

Также как разделение большого произведения на главы и параграфы облегчает чтение, так и разбиение большой программы на параграфы, разделы (подпрограммы и модули), путем выделения логических единиц улучшает восприятие программы; помогает избежать однообразия и хорошо организовать материал. Название раздела отражает его цель.

Структура программы хорошо реализуется с некоторым смещением. Для ЭВМ форма записи программы безразлична: смещение строк предназначено для удобства чтения программы человеком. Смещение строк не должно быть большим: двух-трех позиций вполне достаточно, чтобы выделить структуру программы.

Основные правила записи различных структурных конструкций на примере алгоритмического языка Pascal:

1. Короткий составной оператор:

```
BEGIN A1; A2; ...; An END;
```

2. Длинный составной оператор:

```
BEGIN
    A1; A2; ...; An
END;
```

3. Короткое ветвление и обход:

а)
```
IF e THEN A1
    ELSE A2;
```

б)
```
IF e THEN A3;
```

4. Более длинное ветвление:

```
IF e THEN A1
    ELSE A2;
```

5. Очень длинное ветвление:

```
IF e THEN BEGIN
        A1; A2; ...; An
    END
    ELSE BEGIN
        A1; A2; ...; An
    END;
```

6. Циклы короткие:

а) `WHILE e DO a;`

б) `REPEAT a UNTIL e;`

в) `FOR k:=b1 TO b2 DO a;`

г) `FOR k:=b2 DOWNTO b1 DO a;`

7. Циклы длинные:

а)
```
WHILE e DO BEGIN
        A1; A2; ...; An
    END;
```

б)
```
REPEAT
        A1; A2; ...; An
    UNTIL e;
```

в)
```
FOR k:=b1 TO b2 DO
BEGIN
        A1; A2; ...; An
END;
```

8. Циклы с общим началом и/или концом:

```
FOR k:=b1 TO b2 DO WHILE e DO
BEGIN
    A1; A2; ...; An
END;
```

9. Выбор:

```
CASE e OF
    N1,N2,...,Nk:: A1;
    .....
    Ne,.....,Nv:  An1;
END;
```

10. Длинные операторы. Если оператор не помещается в строку, то его продолжение в следующей строке следует записывать со смещением, причем небезразлично, в каком месте сделан перенос. Например:

```
WRITELN ('первый корень уравнения равен'
,X1,'второй корень уравнения равен',X2);
```

читается хуже, чем

```
WRITELN ('первый корень уравнения равен',X1,
                'второй корень уравнения равен',X2);
```

11. Объединение операторов в строку. Если язык программирования позволяет размещать несколько операторов в одной строке, то в строку следует группировать логически связанные операторы, и так, чтобы его можно было кратко откомментировать.

Вместо

```
I:=0;
X:=RO*Cos(FI);
Y:=RO*Sin(FI);
```

можно написать

```
I:=0;
X:=RO*Cos(FI);  Y:=RO*Sin(FI);  (* координаты вектора *)
```

Или вместо

```
WRITE('Введите коэффициенты:');
READLN(A,B,C);
```

можно написать

```
WRITE('Введите коэффициенты:');     READLN (A,B,C);
```

12. Увеличив интервалы между некоторыми операторами (добавив пустые строки), можно дополнительно улучшить восприятие программы.

13. Все метки оператора должны быть вынесены в начало строки и должны выступать из общего текста программы влево и в порядке возрастания, чтобы их можно было легко найти.

14. К началу строк выносятся заголовки блоков, начала крупных циклов и очень крупных составных операторов. Операторы END должны размещаться либо в той же строке, что и соответствующее ему DO или BEGIN либо под парным ему DO или BEGIN.

15. Использование достаточного количества пробелов, отступов, пустых строк в тексте программы является мощным средством, позволяющим сделать программу более выразительной.

ЛАБОРАТОРНАЯ РАБОТА № 9

Тестирование и отладка разработанной программы.

Цель работы:

- Осуществить тестирование и отладку разработанной ранее конкретной программы на алгоритмическом языке высокого уровня.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо составить набор тестов к разработанной ранее программе и провести ее отладку.

Составленный набор тестов необходимо представить в отчете.

Теоретические сведения.

Тестирование - это процесс выполнения программы с целью определения места некорректного ее функционирования. Оно включает преднамеренное конструирование трудных наборов входных данных, создающих наибольшие возможности для отказа программного изделия. Тестирование является основным методом обнаружения ошибок в программе. Результаты тестирования являются исходными данными для отладки.

Отладка программы - это этап ее разработки, на котором устраняются недостатки только что созданной программы.

Если программа правильно ведет себя для солидного набора тестов, нет оснований, утверждать, что в ней нет ошибок. Просто неизвестно, когда она не работает и можно говорить лишь о некотором уровне ее правильности.

Тесты, не способствующие обнаружению ошибок и только подтверждающие корректность функционирования программы, являются не эффективными, т.к. приводят к бесполезным затратам ресурсов и времени.

Тест - это просчитанный вручную или другим способом пример, промежуточные и конечные результаты которого используются для контроля правильности (живучести) программного изделия. Тест состоит из исходных данных и тех значений, которые должны выдать отладочные печати при работе по этому тесту. Эти значения должны быть записаны в точности в том виде, в котором их должна выдать ЭВМ. Эти значения желательно получить любым путем, но не тем, который реализован в программе, т.к. в последнем случае можно не заметить ошибки в алгоритмизации.

Комплект тестов должен быть таким:

- чтобы проверить все варианты внешнего эффекта программы и варианты ее внутренней работы алгоритма;
- чтобы все ветви алгоритма были пройдены, по крайней мере, по одному разу.
- чтобы проконтролировать предельные и вырожденные случаи.

Тестовые данные должны подбираться таким образом, чтобы программист был в состоянии вычислить правильный результат ещё до начала тестирования.

Процесс тестирования программы можно разделить на три этапа:

1. Проверка в нормальных условиях.
2. Проверка в экстремальных условиях.
3. Проверка в исключительных ситуациях.

Каждый из этих трех этапов проверки должен гарантировать получение верных результатов при правильных входных данных и выдачу сообщений об ошибках при неправильных входных данных.

Проверка в нормальных условиях

Случаи, когда программа должна работать со всеми возможными исходными данными, чрезвычайно редки. Обычно имеют место конкретные ограничения на область изменения данных, в которой программа должна сохранять свою работоспособность. Программа должна выдавать правильные результаты для характерных совокупностей исходных данных.

Проверка в экстремальных условиях

Тестовые данные этого этапа включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичные примеры очень большие числа, очень малые числа или отсутствие информации.

Проверка в исключительных ситуациях.

Используются исходные данные, значения которых лежат за пределами допустимой области их изменения. Наихудшая ситуация когда программа воспринимает неверные данные как правильные и выдаёт неверный, но правдоподобный результат. Программа должна отвергать любые данные, которые она не в состоянии обрабатывать верно.

Пример тестов.

Пусть требуется вычислить длину диагонали параллелепипеда по формуле $d=\sqrt{a^2+b^2+c^2}$ Необходимо сформировать тестовые данные для нормальных, экстремальных и исключительных условий.

Стороны Параллелепипеда			Примечание
1	1	1	Хороший нормальный тест $d \approx 1{,}7320508$
1	2	3	Тест в нормальных условиях $d \approx 3{,}7416577$
0	0	0	Результат должен быть равен нулю
0	1	2	Не параллелепипед. Что произойдет?
1	0	3	Неверные данные
2	1	0	Неверные данные
1	-6	3	Неверные данные
A	B	C	Неверные данные

ЛАБОРАТОРНАЯ РАБОТА № 10

Составление документа «Руководство пользователю».

Цель работы:

- Составить документ «Руководство пользователю» к разработанной ранее программы.

Порядок выполнения работы и отчетность.

Во время выполнения лабораторной работы необходимо составить документ «Руководство пользователю» к разработанной ранее программе.

Работа должна быть оформлена в виде документа «Руководство пользователю».

Теоретические сведения.

Формальные требования к документации программного обеспечения описаны в ЕСПД (Единая система программной документации), неформально: состав документации к программному обеспечению состоит из описания его внешнего эффекта и описания его внутреннего устройства.

Первая часть документации, так называемая «Инструкция пользователю» или «Руководство пользователю» предназначена для того, кто собирается использовать программное обеспечение (для пользователя), не вникая в подробности его внутреннего устройства.

Вторая часть - «Руководство программисту» необходима при модификации программного обеспечения или при необходимости исправить в нем ошибку.

В целом, документация к программному обеспечению может содержать ниже перечисленные сведения:

1. Наименование ПО и описание задачи, которую оно решает.
2. Область применимости ПО.
3. Режим работы ПО, сообщения, выдаваемые по ходу его работы, ответы пользователя на них (если это необходимо).
4. Исходные данные, необходимые для работы ПО; а также выдаваемые им результаты;.
5. Правила подготовки исходных данных на внешних носителях (если они применяются) и вид выдаваемой информации.
6. Описание структуры данных. Для любой переменной описывается ее назначение, атрибуты (тип, размер массива и т.д.), структура информации в ней, если она не очевидна. Описание переменных должно начинаться с тех, которые служат исходными данными и результатами.
7. Описания форм, объектов. Опись свойств форм и объектов.
8. Тексты программ, процедур (в виде распечатки ЭВМ) с комментариями.
9. Тесты.
10. Инструкция (руководство) пользователю.

Инструкция по использованию программы (или просто «Инструкция пользователю», или «Руководство для пользователя») - это выдержка из полной документации, предназначенная для эксплуатации программы. Она представляет собой независимый документ для пользователя программы, в котором описывается: что делает программа и как им пользоваться.

«Инструкция пользователю» должна содержать всю необходимую для пользователя информацию и должна быть ему понятна без дополнительных материалов (без обращения к другим спецификациям). Следовательно, необходимая для этой инструкции информация переписывается полностью из соответствующих спецификаций.

Первая часть инструкции является описательной и должна содержать:

- наименование программы;
- краткое описание программы;
- перечень выполняемых программой функций;
- краткую характеристику метода (или методов) решения поставленной задачи, его достоинство и недостатки;
- полную библиографическую ссылку на полное описание метода;
- описание входных и выходных данных.
- описание структуры базы данных (если она имеется), всех ее таблиц в словесной (вербальном) форме.

Вторая часть документа должна описывать порядок работы с программой. Она должна содержать описание всех режимов работы программы, а также содержание всех печатей и диагностических сообщений, которые выдаются по ходу выполнения программы.

Следует помнить, что пользователь по своей квалификации не является программистом и поэтому его работа с программой описывается на понятном ему языке и достаточно подробно, а именно:

- как запустить программу;
- как продолжить работу с программой (описывается подробный интерактивный режим работы пользователя с программой);
- подготовка и ввод исходных данных в программу;
- как реагировать на запросы программы;
- как вести работу в исключительных ситуациях;
- как реагировать на ошибки;
- как восстановить работу программы в случае аварийного его завершения;
- как получить требуемый результат;
- как правильно закончить работу с программой (запланированный программой выход);
- другие сведения, необходимые пользователю программы.

КОНТРОЛЬНЫЕ ВОПРОСЫ.

К лабораторной работе № 1 - Предварительное проектирование ПО

1. Предмет "Технология программирования", основные понятия.
2. Предварительное проектирование (Системный анализ).
3. Постановка целей проекта и продукта.
4. Определение требований.
5. Документ "Постановка задачи". ("Соглашение о требованиях")

К лабораторной работе № 2 - Разработка программного обеспечения.

1. Технология разработки программного обеспечения.
2. Требования, предъявляемые к «идеальной» технологии разработки программного обеспечения..
3. Объектно - ориентированные технологии разработки ПО.
4. Программное обеспечение как изделие.
5. Проектирование ПО. Обзор этапов проектирования ПО (жизненный цикл)

К лабораторной работе № 3 - Построение функциональной схемы системы ПО.

1. Методы проектирования программного обеспечения.
2. Восходящее и нисходящее проектирование программного обеспечения.
3. Объектно – ориентированное проектирование программного обеспечения.
4. Функциональная декомпозиция системы программного обеспечения.
5. Надежность программного обеспечения.

К лабораторной работе № 4 - Внешнее проектирование ПО.

1. Методическая, технологическая, инструментальная и организационная поддержка процесса проектирования ПО.
2. Конструирование программного обеспечения.
3. Внешнее проектирование ПО.
4. Правила организации диалога ПО с пользователем.
5. Создание интерфейса приложения.

К лабораторной работе № 5 – Разработка архитектуры ПО.

1. Конструирование архитектуры ПО.
2. Типовая архитектура программного обеспечения.
3. Общие правила структурного построения ПО.
4. Правила связи программных модулей по управлению.
5. Правила связи программных модулей по информации.

К лабораторной работе № 6 - Описание алгоритма.

1. Программирование. Проектирование программного модуля.
2. Планирование программирования.
3. Алгоритм. Формы описания алгоритмов.
4. Основные требования к блок-схеме.
5. Проектирование логики программного модуля.

К лабораторной работе № 7 - Пошаговая разработка программы.

1. Структурное программирование.
2. Метод пошаговой разработки программ.
3. Оптимизация программ. Основные приемы оптимизации программ.
4. Корректность (правильность) программы.
5. Эффективность программы. Эффективность или удобочитаемость?

К лабораторной работе № 8 -
Запись текстов программ на алгоритмическом языке высокого уровня.

1. Стиль программирования.
2. Типовая структура программного модуля.
3. Общая организация программы и ее запись.
4. Читабельность программы. Правила записи структурных конструкций. Комментарии. Блок комментариев. Идентификация имен переменных, функций, процедур, файлов.
5. Малый программистский стандарт. Правила сокращений имен.

К лабораторной работе № 9 - Тестирование и отладка программы.

1. Ошибки - их причины появления и последствия. Классы ошибок
2. Первичные и вторичные ошибки. Синтаксические и семантические ошибки. Защитное программирование. Программирование без ошибок.
3. Тестирование ПО. Понятие теста. Комплект тестов.Пример теста.
4. Методы тестирования. Этапы тестирования. Аксиомы тестирования.
5. Отладка ПО. Этапы, методы и инструменты отладки программ.

К лабораторной работе № 10 –
Составление документа «Руководство пользователю».

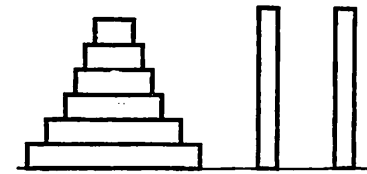
1. Использование программного обеспечения.
2. Сопровождение программного обеспечения.
3. Разработка программного обеспечения «Под ключ».
4. Документация к программному обеспечению.
5. Инструкция по использованию программного обеспечения.

Дополнительные вопросы.

1. Надежность программного обеспечения.
2. Программные изделия с малой и большой длительностью эксплуатации.
3. Системный анализ (предварительное проектирование).
4. Конструирование программного обеспечения.
5. Программирование программного обеспечения.
6. Оценка (испытание) программного обеспечения.
7. Использование (эксплуатация и сопровождение) ПО.
8. Разбиение программных проектов на группы с точки зрения разработки требований к ним.
9. Документ «Соглашение о требованиях».
10. Проблемы проектирования больших программных средств.
11. Система автоматизированного проектирования ПО (САПР ПО).
12. Объектно – ориентированное проектирование ПО.
13. Моделирование системы. Модель системы. Объектная модель. Объект, класс, свойства объектов, операции, методы.Обобщение и наследование.
14. Основные принципы проектирования программного обеспечения.
15. Конструирование объектной модели.
16. Документ «Внешняя спецификация».
17. Система автоматизации программирования (САП).
18. Документ «Внутренняя спецификация».
19. Оптимизация эффективности эксплуатации ПО.
20. Объектно-ориентированное программирование.
21. Понятия объектно-ориентированного программирования: объект, класс, событие, метод, инкапсуляция, наследование, полиморфизм, форма.
22. Распределение операций по классам.
23. Объектно-ориентированные языки программирования.
24. Визуальное программирование. Интегрированная среда разработки приложения. Мастер разработки приложения. Элементы управления приложением.
25. «Альфа» и «Бета» тестирования программного обеспечения.
26. Система автоматизации отладки программного обеспечения.
27. Оценка (испытания) программного обеспечения.
28. Предварительные и совместные испытания ПО.
29. Свойства качественного программного обеспечения.
30. Проблема оценки качества программных продуктов.
31. Определение качества программного обеспечения.
32. Качественные характеристики ПО: понятность, завершенность, осмысленность, мобильность, полезность, машинонезависимость, надежность, структурированность, эффективность, точность, доступность, модифицируемость, открытость, коммуникативность, информативность, расширяемость, учет человеческого фактора.

ВАРИАНТЫ ЗАДАЧ.

1. "Ханойская башня". Доска имеет три колышка. На первом нанизано m дисков убывающего вверх диаметра (см. рисунок). Расположить диски в том же порядке на другом колышке. Диски можно перекладывать с колышка на колышек по одному. Класть большой диск на меньший не разрешается .



2. "Пятнадцать". На квадратном поле размером 4x4 с помощью датчика случайных чисел расставлены 15 фишек с номерами от 1 до 15. Имеется одна свободная позиция. Расставить фишки по возрастанию их номеров, как показано на рисунке. Передвигать фишки можно только на соседнюю свободную позицию.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

3. "Вращающийся квадрат". Дан квадрат размером 4x4 клетки, в которых с помощью датчика случайных чисел расставлены буквы от A до P. Упорядочить буквы в квадрате по алфавиту. Квадрат имеет под квадраты, которые можно вращать по часовой стрелке на одну клетку. Под квадраты имеют размер 2x2, и указывается номером левой верхней клетки. Имеется операция, которая может быть выполнена один раз: обмен местами двух букв.

4. "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 меньшее 7, он также выигрывает сделанную ставку. Если сумма выпавших цифр 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.

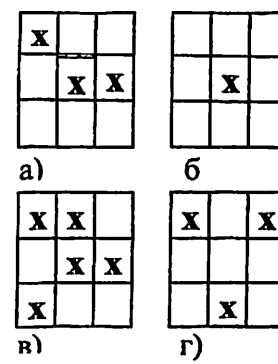
5. "Коровы и быки". Программа выбирает с помощью датчика случайных чисел четырехзначное число с разными цифрами. Угадать это число. На каждом шаге играющий называет четырехзначное число, а программа сообщает, сколько цифр числа угадано (быки) и сколько цифр угадано и стоит на нужном месте (коровы). Например, если программой задано число 1294, а играющий назвал 1423, он получит ответ "1 корова, 3 быка".

6. "Игра в слова". Программа выбирает слово и рисует на экране столько прочерков, сколько букв в этом слове. Отгадать, какое слово загадано программой. В каждый ход играющий указывает одну букву. Если названа буква, входящая в состав слова, она подставляется вместо соответствующего прочерка. В противном случае играющий теряет 1 очко. В начальный момент у играющего 15 очков.

7. "Ипподром". Играющий выбирает одну из трех лошадей, состязающихся на бегах, и выигрывает, если его лошадь приходит первой. Скорость передвижения лошадей на разных этапах выбирается программой с помощью датчика случайных чисел.

8. "Подбери ключи". Перед играющим четыре запертые двери. Открыть все двери, располагая десятью ключами, каждый из которых может открыть несколько дверей. Предоставляется 14 попыток.

9. "Жизнь". Игра моделирует жизнь поколений гипотетической колонии живых клеток, которые выживают, размножаются или погибают в соответствии со следующими правилами. Клетка выживает, если и только если она имеет двух или трех соседей из восьми возможных (рис. а). Если у клетки только один сосед или вовсе ни одного, она умирает в изоляции (рис. б). Если клетка имеет четырех или более соседей, она умирает от а б в г перенаселения (рис. в). В любой пустой позиции, у которой ровно три соседа, в следующем поколении появляется новая клетка (рис. г).



10. Требуется ввести курсор в область экрана (небольшой круг), расположение которого неизвестно играющему. Передвижение курсора сопровождается звуковым сигналом: если приближается к области, то звук становится выше; если удаляется - ниже.

11. Составить программу, помогающую в запоминании исторических дат. Программа должна предлагать вопросы, контролирующие знание дат исторических событий, например, "В каком году была Куликовская битва?". Если ответ правильный, должен быть предложен следующий вопрос. Если ответ не верен, программа подскажет правильный ответ, а позднее повторит этот вопрос еще раз.

12. Составить программу для тренировки памяти. Программа должна высветить на экране несколько точек, играющий - указать, в каком порядке эти точки были высвечены. Координаты точек выбираются в программе с помощью датчика случайных чисел.

13. "Сбей самолет". По экрану летят вражеские самолеты. Цель сбить их. Пусковая установка находится в нижней части экрана. Пусковую установку можно перемещать по строке налево и направо.

14. "Морской бой". На поле 10x10 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. Позиции указываются буквами от А до К (по строкам) и от 1 до 10 (по столбцам). Конфигурация и положение кораблей на поле выбираются с помощью датчика случайных чисел. Если клетка корабля угадана играющим, верно, она отмечается крестиком; в противном случае точкой.

15. Составить программу для заучивания слов иностранного языка. Программа должна предлагать слова из некоторого списка на одном языке, обучающийся - дать перевод этого слова на другой язык. Если ответ правильный, должен быть предложен следующий вопрос. Если ответ не верен, программа подскажет правильный ответ, а позднее повторит этот вопрос еще раз.

16. Составить программу для изучения созвездий. Программа должна построить на экране изображение созвездия, обучающийся - назвать его. Если ответ правильный, должен быть предложен следующий вопрос. Иначе - программа подскажет правильный ответ, а позднее повторит этот вопрос еще раз.

17. Составить программу, помогающую в изучении движения тела, брошенного под углом к горизонту с некоторой начальной скоростью. Играющий, зная расстояние от человека, бросающего камень, до лунки и ширину лунки, должен задать такие значения угла "альфа" и начальной скорости V, чтобы камень попал в лунку. На экране должны изображаться поверхность земли, лунка, камень и траектория полета камня. Расстояние от человека, бросающего камень, до лунки и ширины лунки выбирать с помощью датчика случайных чисел.

18. Составить программу обучения работе с клавиатурой. Программа должна выдавать на экран буквы, цифры, слова и фразы, которые следует набрать на клавиатуре.

19. "Угадай число". Один из играющих задумывает число от 1 до 1000, другой пытается угадать его за 10 вопросов вида: верно ли, что задуманное число больше такого-то числа. Написать программу, играющую за отгадчика.

20. Составить программу, помогающую в изучении колебаний математического маятника. Маятник должен двигаться на экране, совершая гармонические колебания, период которых выбран с помощью датчика случайных чисел. Играющий должен указать длину нити, на которой подвешен маятник. Ответ считается правильным, если ошибка не превышает 10%.

21. Написать программу вычисления определителя N -го порядка (N<=10), пользуясь формулой разложения определителя по i-й строке и зная формулу вычисления определителя 2-го порядка.

22. На местности имеется N населенных пунктов, пронумерованных от 1 до N (N<=10). Некоторые из пунктов соединены между собой дорогами. Информация о дорогах задается в виде последовательности пар чисел i, j (i<j), указывающих, что i-йи j-й пункты соединены дорогой, признак конца этой последовательности - пара нулей. Определить, можно ли попасть по этим дорогам из первого пункта в п -ый.

23. Кот Матроскин и Шарик загадывали четные и нечетные числа в произвольном порядке, пока не кончилось место на печке, где они записывали эти числа. Определить, каких чисел было загадано больше: четных или нечетных, если последним было записано число 0 (при подсчете это число не учитывать). Сколько четных чисел предшествовало первому нечетному?

24. В коробке перемешались кубики. На всех гранях каждого кубика нарисованы одинаковые буквы или цифры. Нужно разложить их в разные коробки: кубики с цифрами поместить в одну, а с буквами - в другую. Вывести на экран содержимое всех трех коробок (что вначале лежало в первой коробке, затем распечатать содержимое полученных двух коробок).

25. Написать программу наглядно демонстрирующую графики тригонометрических функций типа y=a sin(bx+c)+d, меняя значения параметров a,b,c и d. Провести исследование функции на четность, убывание и возрастания. Найти корни и экстремальные точки.

26. Написать программу наглядно демонстрирующую всевозможные варианты расстановки m (m>3) ферзей на шахматной доске m x m клеток, при которых ферзи не бьют друг друга.

28. Фунтик и Хрюша решили сравнить свои книжки «по толщине». Они составили список, указывая вместе с названиями и число страниц в каждой книге. Попутно выяснилось, что все книжки имели различное количество страниц, некоторые книги были как в библиотеке Фунтика, так и в библиотеке у Хрюши. Помогите Фунтику найти самую «толстую» его книжку среди тех, которых нет у Хрюши. (Книжки сравнивать по числу страниц.).

29. Написать программу обучающую детей понятиям «чет» и «нечет». Четные числа выводятся на красном квадратике, а нечетные числа – на синем. После ввода нескольких чисел составляется какой-нибудь узор (Например, выстроить их в линию в таком порядке: красный квадратик, синий квадратик, красный, синий ит.д.

30. Даны действительные числа a(1)...a(50). Они определяют интервалы на числовой оси [a(1), a(2)], . . . [a(49), a(50)]. Написать программу определяющую общие точки всех введенных интервалов. Отобразить на экране полученную картину.

31. У двух спортсменов были гири для тренировок. На каждой гтре был указан ее вес. Каждый раз после очередного занятия они расставляли их по возрастанию весов гирь. Потом решили объединить свои гири, сохраняя привычное расположение гирь. Требуется помощь в их расстановке (известен вес каждой гири в исходных наборах). Нарисовать на экране порядок гирь в исходных и в полученном наборах.

32. Написать программу наглядно демонстрирующую расстановку арифметических операцих + , - , * , / в арифметическом выражении 1*2*3*4*5 вместо звездочек так, чтобы получилось наперед заданное число.

33. Участники игры «О! Счастливчик» рассаживаются на пронумерованных стульях вокруг большого круглого стола. Посредством константы счета начинается их отсчет по часовой стрелке. Игрок, на которого попадает константа счета, обязан освободить место. Отсчет игроков продолжается до тех пор, пока не останутся два человека. При известном числе игроков N и константы счета C определить номера стульев, которые нужно занять игрокам, чтобы попасть в число двух «счастливчиков». Отобразить

34. Написать программу учета проведения футбольного чемпионата в виде таблицы размера N x N, в которой выигрыш команды отображается 2 очками, ничья - 1 очком, а проигрыш - 0 очками. Напечатать номер команды, занявшей первое место, номера команд, которые прошли чемпионат без поражений, подсчитать количество очков у каждой команды, определить номер команды -аутсайдера (при равном количестве очков у двух команда преимущество определяется из результата игры между этими командами).

35. Написать программу наглядно демонстрирующую графики квадратических функций типа $y=ax^2+bx+c$, меняя значения параметров a,b и c. Провести исследование функции на четность, убывание и возрастания. Найти корни и экстремальные точки.

36. Написать программу выводящую на экран таблицу умножения в заданной системе счисления (напрамер, в восьмеричной или в шестнадцатеричной).

37. Пусть заданы N окружностей на плоскости координатами их центра X,Y и радиусом R. Имеются ли среди данных окружностей две попарно пересекающиеся, отдельно стоящие? Если да, то напечатать их параметры (номера, координаты и радиусы) и отобразить их на экране.

38. Составить программу шифрования и дешифрования текста, пользуясь правилами по выбору: а) каждая буква заменяется на следующую по алфавиту (при этом буква «я» заменяется на букву «а»); б) буква заменяется на другую, которая следует за ней через N позиций (замена также циклическая).

39. Написать программу наглядно демонстрирующую вычисления интеграла методами Прямоугольника, Трапеций и Симпсона.

40. Написать программу учета посещаемости и успеваемости группы с учетом различных орг.выводов (например, начисление стипендии, объявление выговора, составление проекта приказа на отчисление и пр.).

41. Написать программу наглядно демонстрирующую некоторые задачи комбинаторики. Например, по заданному пятизначному числу N найти и напечатать все новые пятизначные числа, построенные из тех же цифр, а также выделить из них те числа, которые оказались больше исходного.

42. Написать программу демонстрирующую решение уравнений численными методами. Например, методами половинного деления, хорд Ньютона т пр.. Нарисовать графики соответствующих функций.

43. Написать программу, играющую в "крестики-нолики".

44. Написать шахматную программу, играющую за белых:
а) королем и ферзем против короля;
б) королем и двумя ладьями против короля.

45. "100 спичек" Из кучки, первоначально содержащей 100 спичек, двое играющих поочередно берут по несколько спичек: не менее одной и не более десяти. Проигрывает взявший последнюю спичку.

46. "Ним". Имеются три кучки спичек. Двое играющих по очереди делают ходы. Каждый ход заключается в том, что из одной какой-то кучки берется произвольное ненулевое число спичек. Выигрывает взявший последнюю спичку.

47. Компьютер и пользователь поочередно "называют" число не превышающее 10. Эти числа складываются один за другим. Выигрывает тот, кто первым достигнет 100. Обобщить задачу на ввод произвольных чисел (вместо 10 и 100).

48. 15 спичек расположить в ряд. Требуется собрать их в 5 кучек по 3 спички в кучке, перекладывая их по одной и каждый раз, перескакивая при этом по 3 спички. Обобщить задачу на ввод произвольных чисел вместо 15 и 3).

49. Десять спичек положены в один ряд. Распределить их попарно, всего 5 пар, перекладывая по одной спичке через две. Обобщить задачу на ввод произвольных чисел вместо 10 и 2).

50. Разложить спички в три кучки, например, по 12, 10 и 7 спичек в кучке. Компьютер и пользователь поочередно берут из кучек некоторое количество спичек, но только из одной кучки. Можно взять сразу всю кучу. Выигрывает тот, кто последним возьмет спички.

51. Написать программу наглядно демонстрирующую всевозможные варианты расстановки m (m>3) ладей на шахматной доске m x m клеток, при которых ладьи не бьют друг друга.

52. Написать программу "калькулятор".

53. Написать порограмму демонстрирующую вычисление наибольшего общего делителя, наименьшее общее кратное, количество простых чисел для заданных N чисел.

54. Написать программу демонстрирующую перестановки заданных N элементов без повторений. Например, ABC, ACB, BAC, BCA, CAB, CBA.

55. Написать программу реализующую различные алгоритмы сортировок больших массивов информации.

56. Написать программу демонстрирующую решение задачи: три мушкетера получили сведения, что 12 украденных алмазных подвесок хранятся в десяти различных замках Англии, причем, в каждом замке их по 12 штук (часть из них подлинные, а недостающие заменены фальшивыми). Известно, что у подлинных подвесок по 13 алмазов, у поддельных же различное количество камней, но не 13. Помоги мушкетерам найти все 12 подлинных алмазных подвесок королевы.

57. Написать программу наглядно демонстрирующую всевозможные варианты расстановки m коней (m>3) на шахматной доске m x m клеток, при которых кони не бьют друг друга.

58. Написать программу мониторинга среднесуточной температуры за месяц для метеорологической станции. которая бы, в частности, определяла: количество дней, когда температура была ниже 0°C, сумму положительных температур за месяц, среднюю температуру месяца, день, когда температура ближе всего подходило к среднемесячной и пр..

59. Маленький Мук пробовал печатать в текстовом редакторе. Определить, сколько различных символов он напечатал на экране, а также литеру, напечатанную максимальное число раз.

60. В стене имеется прямоугольное отверстие размерами X и Y. Пролезет ли в данную дыру кирпич, с размерами A, B и C, если проталкивать его таким образом, чтобы стороны кирпича были параллельны сторонам отверстия?

Приложение. Титульный лист отчета к лабораторным работам.

УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Программное обеспечение информационных технологий»

ЛАБОРАТОРНЫЕ РАБОТЫ

по дисциплине:

"Технология программирования"

Выполнил:
студент группы: 225-05 ИТ
Рашидов Ш.Ш.

Принял: доцент кафедры ПОИТ
Чернев Д. А.

Ташкент – 2009

ЛИТЕРАТУРА

1. Д.А.Чернев. Технология разработки программного обеспечения. Уч.пос. для вузов. Ташкент: Мехнат 2004.
2. Д.А. Чернев. Технология программирования. Учебник для колл. Ташкент: Укитувчи, 2003.
3. Г.С. Иванова "Технология программирования" М.: МГТУ им. Баумана, 2002.
4. Иванова Г.С. Основы программирования: Учеб. для вузов. - М.: МГТУ им. Баумана, 2001.
5. Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. Объектно-ориентированное программирование: Учеб. для вузов. - М.: МГТУ им. Баумана, 2001.
6. Канер С., Фояк Д., Нгуен Е.К. Тестирование программного обеспечения. - Киев:v "ДиаСофг", 2000.

Дополнительная литература

1. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование. - М.: ДМК Пресс, 2001.
2. Мандел Т. Разработка пользовательского интерфейса.М: ДМК Пресс, 2001
3. Липаев В.В. Надежность программных средств. - М.: "Синтег", 1998.
4. Липаев В.В., Филинов К.Н. Мобильность программ и данных в открытых информационных системах. - М.: Научная книга, 1997.
5. Международные стандарты, поддерживающие жизненный цикл программных средств. - М.: МП "Экономика", 1996.
6. Новоженов Ю.В. Объектно-ориентированные технологии разработки сложных программных систем. - М.: ДМК Пресс, 1996.
7. Липаев В.В. Управление разработкой программных комплексов. - М.: Финансы и статистика, 1993.
8. Липаев В.В., Позин Б.А., Штрик А.А. Технология сборочного программирования. - М.: Радио и связь, 1992.
9. В.В.Липаев "Проектирование программных средств",М:"ВШ",1991.
10. Л.В. Изосимов,А.Л.Рыленко "Метрическая оценка качества программ", М: "МАИ", 1989.
11. Липаев В.В. Тестирование программ. - М.: Радио и связь, 1986.
12. Фокс Д. Программное обеспечение и его разработка. М.: Мир, 1985.
13. Косимова Ш.Т., Чернев Д.А. Программалащ технологиясига кириш. Укув кулланма. Тошкент: ТошДТУ, 1997.
14. Н.Х.Латипова. Д.А.Чернев "Дастурлаш технологияси" фанидан тажриба ишларини утказиш буйича методик курсатмалар. Т: ТошДТУ, 2000.

СОДЕРЖАНИЕ

Технология программирования (методические указания для выполнения лабораторных работ студентам направлений образования:

5521900 – Информатика и информационные технологии,
5523500 – Защита информации,
5523600 – Электронная коммерция,
5811200 – Сервис (информационный сервис),
5811300 – Сервис (электронные и компьютерные технологии),
5320200 – Информатика и библиотековедение,
5140900 – Профессиональное образование (по направлению информатика и информационные технологии)).

Составители: Д.А.Чернев, Н.Х.Латипова, А.А.Абдурахманов.

Редактор Абдуазизов А.А.

Корректор Абдуллаева С.Х.

Бумага офсетная. Заказ № 330
Тираж 100
Отпечатано в типографии ТУИТ.
Ташкент 700084, ул. А. Темура - 108