Сценарий компьютерной игры «Абитуриент ИРНИТУ» для специальности ИСТб (актуально на 2023 год)

**Содержание**

[1. Модель игрового процесса 3](#_Toc124717683)

[2. Начало игры и пролог 4](#_Toc124717684)

[3. Семестр 1 5](#_Toc124717685)

[3.1 Программирование на ЯВУ 5](#_Toc124717686)

[3.2 Курсовая работа 1 – Введение в веб-технологии 8](#_Toc124717687)

[3.3 Сессия 1 – Программирование на ЯВУ; Введение в веб-технологии 10](#_Toc124717688)

[4. Семестр 2 12](#_Toc124717689)

[4.1 Архитектура ЭВМ и систем 12](#_Toc124717690)

[4.2 Курсовая работа 2 – Методы анализа данных 15](#_Toc124717691)

[4.3 Сессия 2 – Архитектура ЭВМ и систем; Методы анализа данных 17](#_Toc124717692)

[4.4 Учебная практика 19](#_Toc124717693)

[5. Семестр 3 23](#_Toc124717694)

[5.1 Инфокоммуникационные системы и сети 23](#_Toc124717695)

[5.2 Курсовая работа 3 – Базы данных 25](#_Toc124717696)

[5.3 Сессия 3 – Инфокоммуникационные системы и сети; Базы данных 28](#_Toc124717697)

[6. Семестр 4 31](#_Toc124717698)

[6.1 Анализ бизнес-процессов 31](#_Toc124717699)

[6.2 Курсовой проект 4 – ООП 32](#_Toc124717700)

[6.3 Сессия 4 – Анализ бизнес-процессов; ООП 35](#_Toc124717701)

[6.4 Производственная практика 37](#_Toc124717702)

[7. Семестр 5 39](#_Toc124717703)

[7.1 Нейросетевые технологии 39](#_Toc124717704)

[7.2 Курсовая работа 5 – Web-программирование 39](#_Toc124717705)

[7.3 Сессия 5 – Нейросетевые технологии; Web-программирование 39](#_Toc124717706)

[8. Семестр 6 40](#_Toc124717707)

[8.1 Основы мобильной разработки 40](#_Toc124717708)

[8.2 Курсовая работа 6 – Технология разработки программных комплексов 44](#_Toc124717709)

[8.3 Сессия 6 – Основы мобильной разработки; Технология разработки программных комплексов 47](#_Toc124717710)

[8.4 Научно-исследовательская практика 49](#_Toc124717711)

[9. Семестр 7 51](#_Toc124717712)

[9.1 Проектирование информационных систем 51](#_Toc124717713)

[9.2 Курсовая работа 7 – Интеллектуальные системы и технологии 52](#_Toc124717714)

[9.3 Сессия 7 – Проектирование ИС; Интеллектуальные системы и технологии 52](#_Toc124717715)

[10. Преддипломная практика 53](#_Toc124717716)

[11. ВКР – Создание ПЗ, создание проекта, защита ВКР 54](#_Toc124717717)

[12. Пересдача 56](#_Toc124717718)

1. Модель игрового процесса
2. Регистрация – уровень, на котором игрок подаёт заявку на поступление к ИРНИТУ на обучение по одной из специальностей.
3. Ознакомительный день – уровень, на котором игрок изучает управление в игре, предстоящий ему процесс прохождения игры и локацию университета.
4. Семестр – знакомство игрока с одним из профильных предметов по его направлению обучения. Делается акцент на теорию.
5. Курсовая работа – создание работы по одному из профильных предметов. Делается акцент на практику.
6. Сессия – проверка знаний игрока по пройденным предметам и курсовым работам.
7. Начало игры и пролог

Перед началом прохождения новый игрок выбирает модельку персонажа (парень или девушка).

После выбора запускается видеовставка с информацией о проекте и команде, его разрабатывающей и начинается день поступления.

**День поступления (регистрация)**:

Проигрывается вступительная заставка, игрок появляется у входа главного корпуса ИРНИТУ. В главном холле проходит запись абитуриентов на обучение в университете. Главный герой подходит к одному из участников приёмной комиссии. После диалога происходит регистрация игрока по форме, в основу которой взята анкета регистрации абитуриента в ИРНИТУ. После регистрации герой может уйти домой или пройтись по главному холлу.

**Ознакомительный день (обучение)**:

Герой приходит на вступительную лекцию. Игрок появляется в главном холле. Стоят несколько групп студентов с наставниками. Игрок подходит к своей группе, и следует за наставником в амфитеатр.

После того, как игрок приходит в кабинет, начинается лекция. Игроку выдаётся карта университета, зачётная книжка и пропуск. Представители профкома и преподаватели рассказывают про ВУЗ, местонахождение различных важных мест университета, мероприятия, процесс обучения. После окончания лекции игрок может уйти домой или пройтись по университету.

1. Семестр 1
   1. Программирование на ЯВУ

Игрок находится у на входе политеха, ему необходимо найти нужную аудиторию. У игрока есть карта. Он должен отыскать аудиторию как можно раньше, потому что у него может оказаться меньше времени на практическое задание.

На паре персонажу преподаватель рассказывает, что такое программирование, какой язык программирования он и его группа будут изучать в этом семестре. Преподаватель объяснит для чего нужен данный язык программирования, где будет применяться в будущем. Также, будут предоставлены примеры кода.

В конце пары игроку будет предложено сделать практическое задание. Оно будет представлять собой написание небольшого куска кода, с последующим пояснением.

1. **Язык программирования:**

C++ – компилируемый, статически типизированный язык программирования общего назначения.

1. **Для чего нужен? Где используется?**

С++ создавался на основе языка С, и при этом до определенного момента сохранял с ним совместимость. Следовательно, С++ вобрал в себя всю скорость языка С. C++ используется во всех сферах деятельности программирования: от высоконагруженных систем до программирования микроконтроллеров. На С++ можно написать как web-сервер, так и игры, любые компьютерные программы, компоненты и так далее.

1. **Теория**

***Типы данных.*** Сначала отвечу “зачем” существуют типы данных. Смотрите, допустим нам надо написать программу, которая выводит на экран данные о возрасте, весе и росте человека. Но, чтобы  программа могла обращаться к этим данным, их надо хранить где-то в оперативной памяти компьютера, а уже потом “попросить” показать их. Вот для этого нам и нужны типы данных – чтобы компилятор “понял” сколько памяти надо выделить для данных и что в этой памяти будет храниться (целое число (возраст), вещественное (вес и рост), символы и т.д.).

Типы данных можно разбить на три группы: числовые, символьные и логические. Для обозначения типов данных используются специальные зарезервированные (ключевые) слова.  Ниже мы их рассмотрим и укажем сколько памяти для них выделяется и какие данные они могут хранить.

Числовые типы данных: для хранения целых чисел (0, 33, -27 и т.д.)

**int** (4 байта) – хранит числа в диапазоне от -2 147 483 648 до 2 147 483 647

**short** (2 байта) – хранит числа в диапазоне от -32 768 до 32 767

**long** (4 байта) – хранит числа в диапазоне от -2 147 483 648 до 2 147 483 647

Числовые типы данных: для хранения вещественных чисел (с плавающей точкой: **–**435.332, 54.77, 3.0)

**float**(4 байта) – хранит дробные числа с точностью до 7 знаков после запятой

**double**(8 байт) – хранит дробные числа с точностью до 15 знаков после запятой

Символьный тип: для хранения одного символа

**char** (1 байт) – хранит один символ. Например: ‘f’, ‘+’ или ‘4’ (как символ). Одинарные кавычки обязательны.

Типа данных для хранения строк в С++ нет. Позже мы с вами познакомимся с тем, каким образом можно хранить строки в памяти.

Логический тип:

**bool**(1 байт) – может принимать только два значения **true** (истина) и **false** (ложь)

Сразу стоит обратить внимание, что все зарезервированные слова под типы данных пишутся не заглавными, а строчными (маленькими) буквами. Если вы объявите тип **bool**, как Bool – компилятор выдаст сообщение об ошибке. В редакторе, при написании кода, все зарезервированные слова С++ подсвечиваются синим цветом (это касается всех зарезервированных слов).

***Переменные и константы.***  Их названия говорят сами за себя. И переменная и константа – это определённые области в оперативной памяти, которые имеют имя (имя задаёт программист). Только переменная может менять своё значение в ходе выполнения программы, константа же определяется один раз и изменяться уже не может.

И переменным, и константам при объявлении обязательно надо присвоить имя. Существуют строгие правила, согласно которым даются имена:

- имя переменной может содержать маленькие и большие буквы английского алфавита, цифры, и **‘\_’** (нижнее подчеркивание). С нижним подчеркиванием имена читаются намного легче. Сравните **amountofapples**  и **amount\_of\_apples.**

- имя не может начинаться с цифры. **amount\_of\_apples1** – можно, **1amount\_of\_apples**– нет))

- именем не может быть зарезервированное слово (**int, bool** …). Если вы придумали имя переменной, ввели, а его подсветило синим цветом, значит такое слово в С++ является зарезервированным. Придумайте новое имя.

- имя переменной должно быть осмысленным (логичным тому, что будет хранить данная переменная). Т.е. вряд ли кто-то догадается, что в переменной **x** хранится значение количества яблок. Логичнее назвать такую переменную **amount\_of\_apples.**

***Объявление и инициализация переменных и констант.***

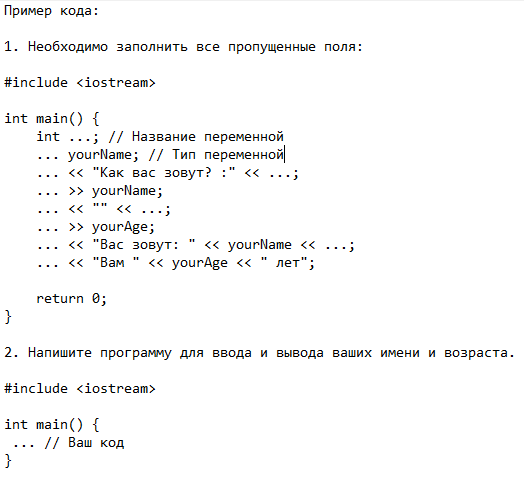
Как же можно создать переменную и указать какое значение она будет хранить? Для этого нам необходимо сначала указать тип данных, затем дать имя переменной (придерживаясь правил, рассмотренных выше), поставить знак = (означает присвоить) и указать значение. К примеру, создадим переменную, которая будет хранить значение количества яблок: int amount\_of\_apples = 7;

Присваивание значения переменной при её создании называется инициализацией. Желательно всегда инициализировать переменные при создании, даже если вы не знаете какое значение переменная примет в ходе работы программы. В этом случае инициализируйте её нулём: int amount\_of\_apples = 0;. Дело в том, что оперативная память свободной не бывает. И если вы просто создали переменную int amount\_of\_apples; и не присвоили ей никакого значения, она всё равно будет хранить какие-то остаточные данные от предыдущих программ. Поэтому, хотя и не является ошибкой сначала объявить переменную, а ниже присвоить ей значение, желательно присвоить значение этой переменно сразу, тем самым очистив её от “мусора”.

Что касается констант, то значение им должно быть присвоено сразу при создании. Например, определим константу, которая будет хранить количество дней в неделе: **const int** **daysInWeek = 7;** Чтобы дать понять компилятору, что это константа, а не обычная переменная, перед типом данных обязательно использовать ключевое слово **const**.

1. **Практическое задание**

Необходимо реализовать алгоритм, решающий конкретную задачу. Также даются комментарии по решению задач на программирование.



1. Пример написания задачи по программированию
   1. Курсовая работа 1 – Введение в веб-технологии

Игрок приходит в компьютерный класс, преподаватель рассказывает краткую теорию про HTML, после чего необходимо выполнить задание.

Цель курсовой – изучить основы языка HTML после чего проверить полученные знания на практике, создав простую веб-страницу.

HTML — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

Сложно, да? Попробуем проще.

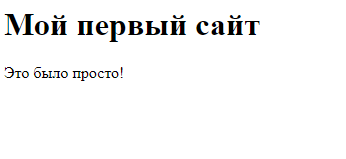
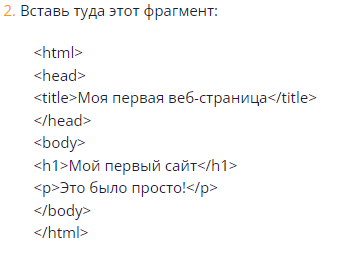
В интернете есть сайты, на сайтах есть страницы. HTML– это язык, на которых эти страницы пишутся. HTML отвечает за текстовое наполнение и структуру сайта.

В языке HTML теги – это «буквы», из которых складывается веб-страница. Их предназначение – рассказывать браузеру как он должен интерпретировать тот или иной фрагмент кода.

Все теги заключены вот в такие символы < >. Например, тег <i> – он превращает обычный текст в *курсив*. Теги бывают парными и одиночными. Парные теги состоят из двух частей: открывающего тега – < > и закрывающего – </ >. Если бы в примере выше мы не поставили бы после слова «курсив» закрывающий тег, то весь текст после открывающего тега *остался бы таким*.

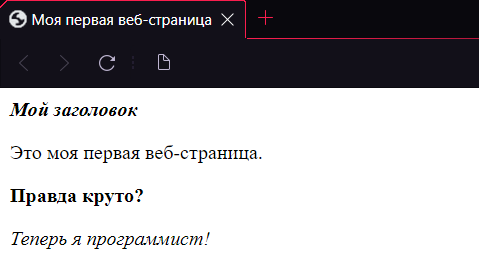
Существует несколько десятков тегов, но запоминать все необязательно. Если всерьез увлечешься программированием, то выучишь самые необходимые во время работы, а остальные всегда сможешь подсмотреть в интернете. Самые употребляемые вроде служебных <head> и <title> или текстовых <h1> и <p> запомнишь в любом случае.

У тегов есть определенная иерархия, порядок взаимодействия и атрибуты.

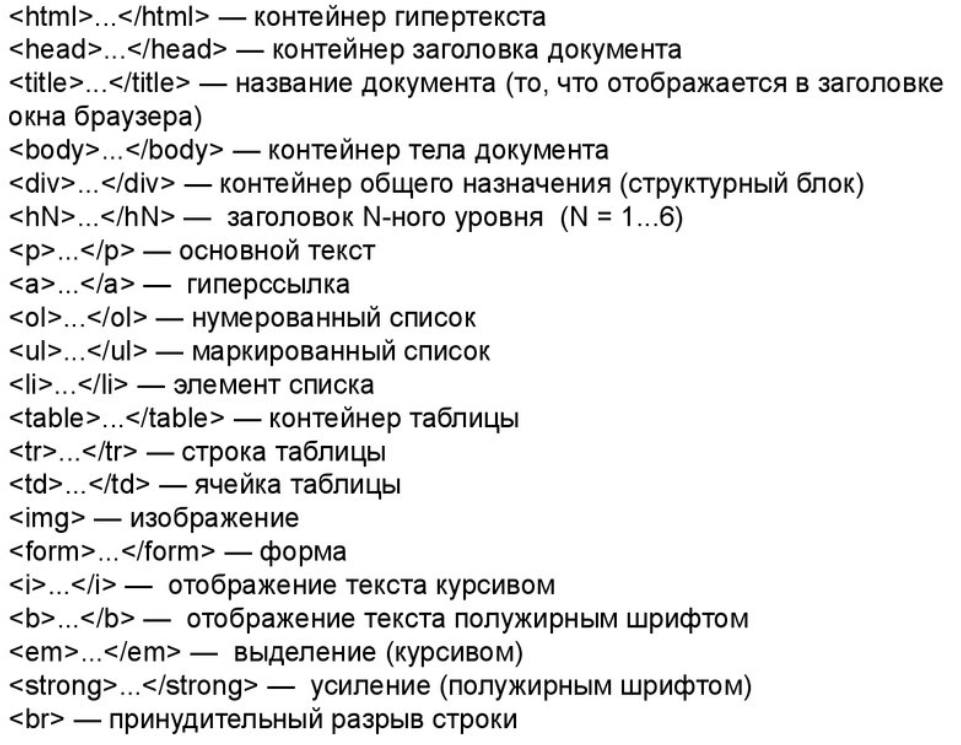


1. Пример создания страницы сайта на HTML

Задание – используя подсказку с тэгами (рисунок 4), напиши структуру веб-страницы, как на рисунке 3.



1. Страница, которую необходимо сделать



1. Подсказка с HTML тегами

В результате должен получиться такой код:

<html>

<Head><b><i>Мой заголовок</i></b></Head>

<title>Моя первая веб-страница</title>

<body>

<p>Это моя первая веб-страница.</p>

<p><b>Правда круто?</b></p>

<p><i>Теперь я программист!</i></p>

</body>

</html>

* 1. Сессия 1 – Программирование на ЯВУ; Введение в веб-технологии

Сессия – период, когда студент освобожден от занятий, но ему необходимо сдать экзамены и зачеты по пройденным предметам для проверки усвоенных им знаний. Тест представлен в виде таблицы ниже, где правильный ответ в столбце ответов выделен жирным шрифтом.

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Какой тип переменной использовать для дробного числа с 8 знаками после запятой? | long |
| float |
| **double** |
| Как лучше всего назвать переменную? | amountofcars |
| 1\_amount\_of\_cars |
| **amount\_of\_cars** |
| Нужно ли при создании константы присваивать ей значение? | **Да** |
| Нет |
| Не всегда |
| На какие группы делятся типы данных в C++? | Целочисленные, с плавающей точкой, символьные и логические |
| **Числовые, символьные и логические** |
| Символьные, числовые |
| Выберите верную инициализацию переменной | age = 10; |
| int age = five; |
| **int age = 0;** |
| Что такое HTML? | Язык описания внешнего вида документа, написанного с использованием языка разметки. |
| **Стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере** |
| Свободный набор инструментов для создания сайтов и веб-приложений |
| Зачем нужны теги в HTML? | **Теги говорят браузеру, как нужно отобразить определённый элемент документа** |
| Теги нужны для удобного чтения кода страницы |
| Для хранения информации |
| Укажите верную иерархию тегов | <HTML>  <HEAD >  <TITLE> Title </TITLE >  </HEAD>  <BODY> <P>…</P> <BODY>  </HTML> |
| **<HTML>**  **<HEAD >**  **<TITLE> Title </TITLE >**  **</HEAD>**  **<BODY> <P>…</P> </BODY>**  **</HTML>** |
| <HTML>  <HEAD >  <TITLE> Title </TITLE >  </HAED>  <BODY> <P>…</P> </BODY>  </HTML> |
| Как будет выглядеть данный текст в браузере: “<b><i>ТЕКСТ</i></b>”? | *ТЕКСТ* |
| ТЕКСТ |
| ***ТЕКСТ*** |
| Как получить такой текст на веб-странице: “**Я** *люблю* ***HTML***!”? | **<b>Я</b>**  **<i>люблю**  **<b>HTML</b></i>!** |
| <b>Я</b>  <i>люблю  <b>HTML!</b></i> |
| <b><i>Я</i></b> <b><i>люблю</i></b> <b><i>HTML!</i></b> |

1. Семестр 2
   1. Архитектура ЭВМ и систем

Игрок стоит в фойе. До начала пары игроку необходимо найти аудиторию В-214. На паре игроку будет рассказано про вычислительные машины, вычислительные системы, определение архитектуры ЭВМ и комплектующие ПК.

Существует множество самых различных определений терминов «вычислительная машина» (ВМ) и «вычислительная система» (ВС). Причина такой терминологической неопределенности кроется в невозможности дать удовлетворяющее всех четкое определение, достойное роли стандарта.

Термином «вычислительная машина» обозначается комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей.

Вычислительная система – совокупность взаимосвязанных и взаимодействующих процессоров или вычислительных машин, периферийного оборудования и программного обеспечения, предназначенная для подготовки и решения задач пользователей.

Таким образом, формально отличие ВС от ВМ выражается в количестве вычислителей. Множественность вычислителей позволяет реализовать в ВС параллельную обработку. С другой стороны, современные вычислительные машины с одним процессором также обладают определенными средствами распараллеливания вычислительного процесса. Иными словами, грань между ВМ и ВС часто бывает весьма расплывчатой, что дает основание там, где это целесообразно, рассматривать ВМ как одну из реализаций ВС. И напротив, вычислительные системы часто строятся из традиционных ВМ и процессоров, поэтому многие из положений, относящиеся к ВМ, могут быть распространены и на ВС.

Под архитектурой вычислительной машины обычно понимается логическое построение ВМ, т.е. то, какой машина представляется программисту. Впервые термин «архитектура вычислительной машины» (computer architecture) был употреблен фирмой IBM при разработке машин семейства IBM 360 для описания тех средств, которыми может пользоваться программист, составляя программу на уровне машинных команд. Подобную трактовку называют «узкой», и охватывает она перечень и формат команд, формы представления данных, механизмы ввода/вывода, способы адресации памяти и т.п. Из рассмотрения выпадают вопросы физического построения вычислительных средств: состав устройств, число регистров процессора, емкость памяти, наличие специального блока для обработки вещественных чисел, тактовая частота центрального процессора и т.д. Этот круг вопросов принято определять понятием «организация или структурная организация».

Архитектура и организация – это две стороны описания ВМ и ВС. В дальнейшем будем пользоваться термином «архитектура» в «широком» его толковании, объединяющем как архитектуру в узком смысле, так и организацию ВМ.

**Персональный компьютер, ПК** — это микро-ЭВМ (электронно-вычислительная машина), которая имеет эксплуатационные характеристики бытового прибора и широкий диапазон универсальных функциональных характеристик. Термин ПК на сегодняшний день означает персональный компьютер, который предназначается для индивидуальной работы и пользования.

Обычно он состоит из следующих составляющих:

**1. Блок питания**

Блок питания снабжает электроэнергией устройства компьютера.

**2. Материнская плата**

Материнская плата – это самая большая плата системного блока. На ней устанавливаются основные устройства компьютера: процессор, оперативная память, видеокарта, слоты (разъёмы), BIOS, с помощью шлейфов и кабелей к материнской плате подключаются жёсткий диск, клавиатура, мышь. Главная задача материнской платы – соединить все эти устройства и заставить их работать как одно целое.

**3. Процессор**

Центральный процессор (ЦП, CPU) является главным элементом компьютера, его **“мозгом”.** Он отвечает за все вычисления и обработку информации. Кроме этого, он выполняет управление всеми устройствами компьютера. От его мощности зависит быстродействие компьютера и его возможности.

**4. Кулер**

Кулера созданы для охлаждения процессора.

**5. Оперативная память**

Оперативная память (RAM или ОЗУ- оперативное запоминающие устройство) - в ней запоминается обрабатываемая процессором информация и запущенные пользователем программы. Оперативной она называется потому, что предоставляет процессору быстрый доступ к данным.

**6. Видеокарта**

Видеокарта – электронная плата, обеспечивающая формирования видеосигнала и тем самым, определяет изображение, показываемое монитором.

**7. Жёсткий диск**

Жёсткий диск, называемый так же HDD, или SSD предназначен для долговременного хранения информации. На жестком диске вашего компьютера хранится вся информация: операционная система, нужные программы, документы и прочие файлы. Он является основным устройством хранения информации в компьютере.

После теории игроку нужно будет собрать свой ПК (рисунок 5). Подойдя к столу перед игроком, будут комплектующие для ПК, и он же в собранном виде (рисунок 6).



1. Задание по предмету “Архитектура ЭВМ и систем”



1. Собранный пример компьютера
   1. Курсовая работа 2 – Методы анализа данных

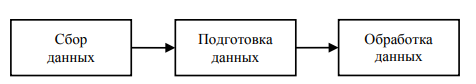
Современный этап развития человечества характеризуется экспоненциальным ростом количества накопленной информации. Согласно исследованию, к 2007 году человечество имело возможность хранения информации объемом 2.9·1020 байт. Большой объем данных порождают научные эксперименты. Так, в апреле 2016 года в открытый доступ поступили 300 Тбайт экспериментальных данных, полученных на большом адронном коллайдере. Функционирование многих технических систем также сопровождается сбором большого количества данных. Например, самолет Боинг-787 генерирует около 500 Гбайт данных за один полет. Однако для выделения из накопленных данных полезной информации требуется определенная обработка этих данных.

Также существует тенденция к переложению функции принятия решений – изначально функции человека – на так называемые экспертные системы (специализированные информационные системы). Экспертные системы позволяют повысить скорость и точность принятия решений. Как правило, функционирование экспертных систем связано с анализом большого объема данных.

В целом анализ данных можно определить как процесс поиска скрытых закономерностей и генерации новых знаний. К основным задачам анализа данных можно отнести прогнозирование, классификацию, поиск схожих черт, выдачу рекомендаций, выявление отклонений. Анализ данных – междисциплинарная область знаний, находящаяся на стыке математики, теории алгоритмов и информационных технологий.

Необходимость анализа больших объемов накопленных данных привела к созданию специализированных подразделений во многих компаниях. Некоторые компании, например Яндекс, реализуют собственные образовательные проекты в этой области.

Анализ данных включает три основных этапа.



Данные по виду можно подразделить на числовые и категориальные.

Числовые данные (Numerical Data) – это данные, характеризующие состояние какого-либо параметра изучаемого объекта. Наиболее часто такие данные бывают представлены вещественными числами. Примерами числовых данных являются заработная плата, население страны, артериальное давление, температура воздуха.

Категориальные данные (Categorical Data) – это данные, образующие признак принадлежности к какой-либо группе. Примерами категориальных данных являются экзаменационная оценка, цвет автомобиля, уровень образования человека.

Сбор данных – процесс формирования структурированного набора данных в цифровой форме. В некоторых случаях процесс сбора данных может включать также этап оцифровки.

Как правило, оцифрованные данные бывают представлены в виде:

- электронных таблиц в форматах XLS либо ODS;

- текстовых файлов в формате CSV

- веб-страниц в формате HTML;

- файлов в формате XML;

- базы данных с доступом по технологии JSON либо через специализированный интерфейс (API).

Для использования в системах анализа данные должны быть представлены в определенном, как правило, табличном виде. Однако зачастую наборы данных имеют следующие особенности:

- отличную от табличной форму представления;

- пропуски отдельных данных;

- некорректные значения;

- большие числовые значения;

- текстовые данные.

Перечисленные особенности могут либо привести к затруднениям в процессе дальнейшей обработки данных, либо сделать её невозможной.

Для устранения отмеченных несоответствий могут быть применены следующие операции:

- структурирование – приведение данных к табличному (матричному) виду;

- отбор – исключение записей с отсутствующими или некорректными значениями;

- нормализация – приведение числовых значений к определенному диапазону, например к диапазону 0...1;

- кодирование – это представление категориальных данных в числовой форме. Например, при бинарной классификации один из классов можно представить числом «0», а другой класс – числом «1». При множественной классификации система кодирования несколько усложняется: создается несколько числовых полей по количеству классов в выборке данных, каждый класс кодируется проставлением числа «1» в соответствующем поле.

Разведочный анализ данных (РАД) применяется для нахождения связей между переменными в ситуациях, когда отсутствуют (или недостаточны) априорные представления о природе этих связей.

Как правило, при разведочном анализе учитывается и сравнивается большое число переменных, а для поиска закономерностей используются самые разные методы.

Вычислительные методы разведочного анализа данных включают основные статистические методы, а также более сложные, специально разработанные методы многомерного анализа, предназначенные для отыскания закономерностей в многомерных данных.

К основным методам разведочного статистического анализа относится процедура анализа распределений переменных (например, чтобы выявить переменные с несимметричным или негауссовым распределением, в том числе и бимодальные), просмотр корреляционных матриц с целью поиска коэффициентов, превосходящих по величине определенные пороговые значения, или анализ многовходовых таблиц частот (например, "послойный" последовательный просмотр комбинаций уровней управляющих переменных)

* 1. Сессия 2 – Архитектура ЭВМ и систем; Методы анализа данных

|  |  |
| --- | --- |
| Вопросы | Ответы |
| В чём заключается анализ данных? | Это представление категориальных данных в числовой форме |
| Это процесс формирования структурированного набора данных в цифровой форме |
| **Это процесс поиска скрытых закономерностей и генерации новых знаний** |
| В каком порядке этапов идёт анализ данных? | Сбор – очистка – обработка |
| **Сбор – подготовка – обработка** |
| Сбор – обработка – подготовка к представлению |
| Что такое нормализация данных? | Приведение данных к табличному (матричному) виду |
| Исключение записей с отсутствующими или некорректными значениями |
| **Приведение числовых значений к определенному диапазону, например к диапазону 0...1** |
| На какие виды можно разделить данные? | **Числовые, категориальные** |
| Числовые, логические |
| Числовые, категориальные, логические |
| В каких случаях используется разведочный анализ данных? | **Когда отсутствуют (или недостаточны) априорные представления о природе этих связей** |
| При проверке статистических гипотез о регрессионной связи между параметрами |
| Выявление групп объектов в выборке данных |

|  |  |
| --- | --- |
| В чём разница терминов “Вычислительная машина” и “Вычислительная система”? | Это одно и то же |
| **Грань между ВМ и ВС часто бывает весьма расплывчатой, формально отличие ВС от ВМ выражается в количестве вычислителей** |
| Вычислительная машина включает в себя комплекс из вычислительных систем |
| Что понимается под архитектурой в широком толковании? | Под архитектурой вычислительной машины обычно понимается логическое построение ВМ, т.е. то, какой машина представляется программисту |
| Состав устройств, число регистров процессора, емкость памяти, наличие специального блока для обработки вещественных чисел, тактовая частота центрального процессора и т.д. |
| **Всё вместе** |
| Что является мозгом ПК? | Видеокарта |
| Жёсткий диск |
| **Процессор** |
| Где хранится вся информация на компьютере? | **На жёстком диске** |
| В оперативной памяти |
| В БИОСе |
| Может ли компьютер выводить картинку без дискретной видеокарты? | **Да, если в центральный процессор встроен графический процессор** |
| Нет |
| Да, в любом случае |

* 1. Учебная практика

Студент находится в компьютерном кабинете со своей группой и преподавателем. Преподаватель выдаёт студентам индивидуальные задания по учебной практике. После выдачи заданий студенты начинают работу за компьютерами.

Необходимо расположить блоки с кусками кода в правильном порядке, чтобы программа заработала. После этого необходимо разместить в правильном порядке пункты отчёта.

Индивидуальное задание: в холле ИРНИТУ установлено множество (больше 2) Wi-Fi точек доступа (с координатами X и Y) необходимо найти пару, которая отстоит друг от друга на наибольшем расстоянии.

Код:

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <time.h>

#include <math.h>

double Distance1(double X, double Y, double X2, double Y2)

{

double dis = sqrt(pow((X - X2), 2) + pow((Y - Y2), 2));

return dis;

}

int main() {

float m;

printf("Введите количество точек доступа Wi-Fi m = ");

while ((scanf("%f", &m) != 1) || (m < 3))

{

printf("Некорректный ввод элемента m. Повторите ввод, m = " );

while(getchar() != '\n');

}

int n = m;

double X[n];

double Y[n];

for (int i = 0; i < n; i++)

{

printf("Введите координату X устройства %d: ", i + 1);

while (scanf("%lf", &X[i])!= 1)

{

printf("Некорректный ввод координаты X %d устройства! Повторите ввод ", i + 1);

while(getchar() != '\n');

}

printf("Введите координату Y устройства %d: ", i + 1);

while (scanf("%lf", &Y[i])!= 1)

{

printf("Некорректный ввод координаты Y %d устройства! Повторите ввод ", i + 1);

while(getchar() != '\n');

}

}

double max = 0;

unsigned int XXX = 0;

unsigned int YYY = 0;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (X[i] != X[j] || Y[i] != Y[j]){

double distance = Distance1(X[i], Y[i], X[j], Y[j]);

if (distance > max)

{

max = distance;

XXX = i;

YYY = j;

}

}

}

}

if (XXX != YYY){

printf("Дальше всего друг от друга стоят устройства %d и %d", XXX+1, YYY+1);}

else {

printf("Все устройства имеют одинаковые координаты");

}

}

Так как программа пишется на языке C, то перед началом игроку даётся краткая теория по данному языку программирования.

Теория:

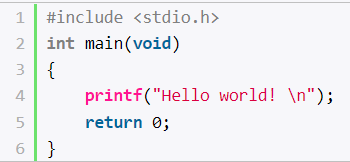
На предмете “Программирование на языке высокого уровня” вы уже познакомились с языком C++. Теперь же давайте вернёмся назад в прошлое и рассмотрим его предка, язык C.

Язык программирования С (си) является одним из самых популярных и распространенных языков. Он представляет компилируемый язык программирования общего назначения со статической типизацией, разработанный в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи (Dennis Ritchie).

Основные особенности Си:

* Универсальность - один и тот же код может быть скомпилирован на почти каждой платформе (при наличии для нее компилятора)
* Высокая скорость выполнения
* Компактность, небольшой размер выходных скомпилированных файлов

Программа на языке Си состоит из набора директив препроцессора, определений функций и глобальных объектов. Директивы препроцессора управляют преобразованием текста до его компиляции. Глобальные объекты определяют используемые данные или состояние программы. А функции определяют поведение или действия программы. Простейшая программа на Си:



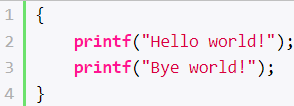
1. Простейшая программа на C

Простейшим строительным элементом программы на Си являются инструкции (statements). Каждая инструкция выполняет определенное действие. В конце инструкций в языке Си ставится точка с запятой (;). Данный знак указывает компилятору на завершение инструкции.

Например: printf("Hello world!");

Вызов функции printf, которая выводит на консоль строку "Hello world!" является инструкцией и завершается точкой с запятой.

Набор инструкций может представлять блок кода. Блок кода оформляется фигурными скобками, инструкции, составляющие тело этого блока, помещаются между открывающей и закрывающей фигурными скобками:

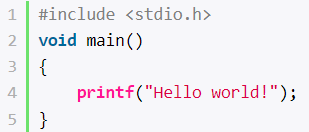
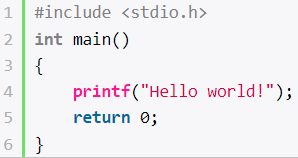


1. Набор инструкций на C

Стартовой точкой в любую программу на языке Си является функция main(). Именно с этой функции начинается выполнение приложения. Ее имя main фиксировано и для всех программ на Си всегда одинаково.

Функция также является блоком кода, поэтому ее тело обрамляется фигурными скобками, между которыми идет набор инструкций.

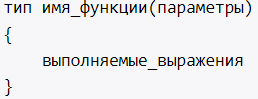
Стоит отметить, что в разной литературе и примерах можно встретить модификации этой функции. В частности, вместо определения выше мы могли бы написать по другому:

 или 

Использование этих определений не было бы ошибкой, и программа также вывела бы строку "Hello world" на консоль. И для большинства компиляторов это было бы нормально.

Если переменные и константы хранят состояние программы, то функции определяют ее действия.

Формальное определение функции выглядит следующим образом:



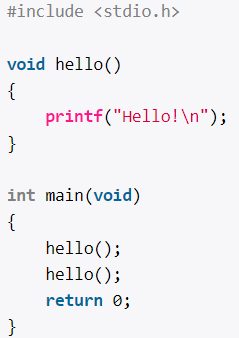
1. Функция в C

Первая строка представляет заголовок или сигнатуру функции. Вначале указывается возвращаемый тип функции. Если функция не возвращает никакого значения, то используется тип void.

Затем идет имя функции, которое представляет произвольный идентификатор, к которому применяются те же правила, что и к именованию переменных. После имени функции в скобках идет перечисление параметров. Функция может не иметь параметров, в этом случае указываются пустые скобки. После заголовка функции в фигурных скобках идет тело функции, которое содержит выполняемые выражения.

Для возвращения результата функция применяет оператор return, которые имеет следующие способы использования: 

В любой программе должна быть как минимум одна функция, которая имеет фиксированное имя main(). Именно с функции main начинается выполнение. Поэтому если мы хотим выполнить функцию hello, то нам надо ее вызвать в функции main.



1. Пример простой программы на C
2. Семестр 3
   1. Инфокоммуникационные системы и сети

На сегодняшний день компьютеры любого предприятия, независимо от его размеров, как правило, объединены в единую сеть. Целью такого объединения является предоставление совместного доступа к различным ресурсам предприятия. В качестве примера можно привести сетевой принтер, доступ к которому может осуществляться с любого компьютера в сети. Однако более важным, нежели совместное использование различных устройств, является совместное использование данных. Несмотря на тот факт, что пользователь может быть удален от данных на несколько сотен или тысяч километров, что никак не ограничивает его возможности доступа к этим данным.

Вторая цель объединения компьютеров предприятия в сеть связана в большей степени с людьми. Следует отметить, что сеть — это замечательная коммуникационная среда для работников предприятия. Большинство людей, для общения на работе, предпочитают использовать электронную почту (electronic mail, e-mail).

Еще одним способом связи между сотрудниками является видеоконференция. Используя эту технологию, можно устраивать встречи, на которых собеседники, находящиеся за тысячи километров друг от друга, будут не только слышать, но и видеть друг друга.

Третья цель создания сети на предприятии — это возможность совместной работы с документами. При помощи сети удаленные друг от друга сотрудники могут легко составить совместный отчет. Если один из сотрудников изменяет документ в режиме реального времени, то остальные сотрудники могут немедленно увидеть эти изменения, а не ждать. Подобное ускорение передачи информации делает возможным сотрудничество удаленных друг от друга групп людей.

В основном информационные сети классифицируют по признакам структурной и функциональной организации сети. Рассмотрим некоторые из них.

**По размеру сети:**

Первым признаком классификации сетей является их размер. По размеру сети можно разделить на персональные, локальные, муниципальные и глобальные сети.

**Персональная сеть** (Personal Area Network, PAN) предназначена для одного человека. Примером может служить сеть, соединяющая персональный компьютер и различные устройства связи, например мобильный телефон.

**Локальной сетью** (Local Area Network, LAN) называют сеть, которая объединяет компьютеры, как правило, в одном офисе, здании или на территории какой-либо организации. Их часто используют для обмена информацией и совместного доступа к ресурсам внутри организации.

**Муниципальные сети** (Metropolitan Area Network, MAN) объединяют компьютеры в пределах города. Самым распространенным примером муниципальной сети является система предоставления высокоскоростного доступа в сеть Интернет. Дома абонентов подключены к специальным распределительным узлам, которые в свою очередь соединены с входным узлом, имеющим доступ в Интернет.

**Глобальная сеть** (World Area Network, WAN) охватывает значительную географическую территорию и включает в себя большое число компьютеров. Крупнейшей глобальной сетью является Интернет.

**По топологии сети:**

Под топологией сети понимают способ описания конфигурации сети в виде графа, вершинам (узлам) которого соответствуют различные сетевые устройства (например, компьютеры, принтеры или коммуникационное оборудование), а ребрам линии связи (физические, логические или информационные) между ними.

От выбора топологии существенно зависят характеристики сети. Например, наличие между узлами нескольких путей повышаем надежность сети. Выделяют следующие топологии: полносвязная, ячеистая, общая шина, кольцо, звезда, дерево и смешанная.

**Полносвязная** топология соответствует сети, в которой каждый узел непосредственно связан со всеми остальными.

**Ячеистая** топология получается из полносвязной путем удаления некоторых связей. Эта топология, как правило, характерна для крупных сетей.

Топология **общая** **шина** предполагает использование одного кабеля, к которому подключаются все компьютеры сети. Отправляемое какой-либо рабочей станцией сообщение распространяется на все компьютеры сети. Каждая машина проверяет кому адресовано сообщение, — если сообщение адресовано ей, то обрабатывает его.

**Кольцо** — это топология, в которой каждый узел соединен только с двумя соседними: от одного он только принимает данные, а другому только передает. В сети с кольцевой топологией каждый узел получает данные от предыдущего узла и передает их далее, если они адресованы не ему.

**Звезда** — это топология, в которой все узлы подключаются непосредственно к общему центральному узлу. В функции центрального узла входит перенаправление передаваемой информации одному или нескольким подключенным к нему узлам.

Иногда имеет смысл проектировать сеть с использованием нескольких центральных узлов, иерархически соединенных между собой связями типа звезда. В результате получается топология, называемая **деревом**. В настоящее время дерево является самой распространенной топологией.

В то время как небольшие сети, как правило, имеют типовую топологию (звезда, кольцо или общая шина), для крупных сетей характерно наличие произвольных связей между сетевыми устройствами. В таких сетях можно выделить отдельные подсети, имеющие типовую топологию, поэтому их называют сетями со **смешанной** топологией.

**По типу функционального взаимодействии:**

Если некое сетевое устройство предоставляет свои ресурсы всем пользователям сети, то оно выполняет функции **сервера** (server). При этом устройство (обычно это компьютер), которое обращается к сетевым ресурсам, является клиентом (client). В зависимости от того, как распределены функции между устройствами сети, информационные сети делятся на два класса: **одноранговые** **сети** (peer-to-peer) и **сети с выделенным сервером** (server based).

Если основным назначением сетевого устройства является выполнение серверных функций (исключая функции клиента), то такое устройство называется **выделенным сервером**. Очевидно, что на выделенных серверах желательно устанавливать серверные операционные системы, специально оптимизированные для выполнения тех или иных серверных функций. Кроме того, для выделенных серверов используются более мощное аппаратное обеспечение.

В **одноранговых сетях** все устройства равны в правах доступа к ресурсам друг друга. В такой сети отсутствуют выделенные серверы, а каждый узел является как клиентом, так и сервером.

Одна из областей применения одноранговых сетей — это обмен файлами. Пользователи файлообменной сети вносят информацию о файлах в специальную базу данных, хранящуюся на своем компьютере. Если кто-то хочет скачать какой-нибудь файл, он посылает запрос на поиск этого файла в сети. После этого пользователь может скачать файл у найденных источников. В современных файлообменных сетях файлы загружается сразу с нескольких источников.

* 1. Курсовая работа 3 – Базы данных

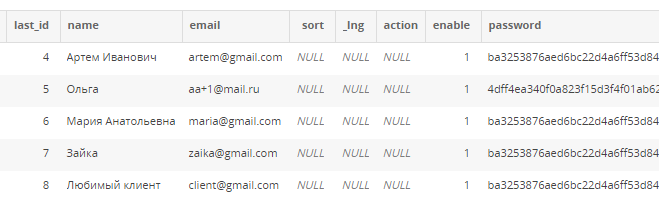
На паре по базам данных в компьютерном классе игроку рассказывают про то, что такое БД, как они устроены и зачем нужны. После теории идёт задание, в котором нужно соотнести блоки со словами так, чтобы получились корректная таблица с данными в БД.

Теория:

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. Говоря простым языком, это хранилище, куда приложение складывает свои данные. Если приложение небольшое, отдельная база не нужна. Но потом это становится удобнее и выгоднее с точки зрения памяти. Если приложение маленькое, то все данные можно хранить в памяти. Но учтите, что это память на вашем компьютере, вашем телефоне. И чем больше данных туда пихать, тем медленнее будет работать программа.

Место в памяти ограничено. Поэтому, когда данных много, их нужно куда-то сложить. Можно писать в файлики, а можно сохранять информацию в базу данных (сокращенно БД). Выбор за вами. А точнее, за вашим разработчиком.

Как она выглядит? Да, примерно, как Excel-табличка! Есть колонки с заголовками, и информация внутри:



Это называется реляционная база данных — набор таблиц, хранящихся в одном пространстве.

Базы данных и электронные таблицы (в частности, Microsoft Excel) имеют ряд отличий, несмотря на их внешнюю схожесть. Основные различия между ними заключаются в следующем:

* Способ хранения и обработки данных
* Полномочия доступа к данным
* Объем хранения данных

Электронные таблицы изначально разрабатывались для одного пользователя, и их свойства отражают это. Они отлично подходят для одного пользователя или небольшого числа пользователей, которым не нужно производить сложные операции с данными. С другой стороны, базы данных предназначены для хранения гораздо больших наборов упорядоченной информации иногда огромных объемов. Базы данных дают возможность множеству пользователей в одно и то же время быстро и безопасно получать доступ к данным и запрашивать их, используя развитую логику и язык запросов.

Для базы данных обычно требуется комплексное программное обеспечение, которое называется системой управления базами данных (СУБД). СУБД служит интерфейсом между базой данных и пользователями или программами, предоставляя пользователям возможность получать и обновлять информацию, а также управлять ее упорядочением и оптимизацией. СУБД обеспечивает контроль и управление данными, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление.

В качестве примеров популярного программного обеспечения для управления базами данных, или СУБД, можно назвать MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, СУБД Oracle Database и dBASE.

Но нельзя говорить про базы данных, не затронув SQL. Аббревиатура SQL расшифровывается как «Structured Query Language» — язык структурированных запросов. С помощью этого языка вы можете работать с записями в базах данных.

SQL состоит из команд и декларативных ключевых слов, которые являются как бы инструкциями для базы данных.

При помощи команд SQL можно создавать и удалять таблицы в базах данных, добавлять в них данные или вносить изменения, искать и быстро находить нужные сведения.

Все операции, которые можно осуществлять с данными, входят в понятие «CRUD». CRUD расшифровывается как Create, Read, Update и Delete (создать, прочесть, обновить, удалить). Это четыре основных операции, которые мы осуществляем, делая запросы к базе данных.

Мы создаем информацию в базе (CREATE), мы читаем, получаем информацию из базы (READ), мы обновляем данные или осуществляем какие-то манипуляции с ними (UPDATE) и, при желании, можем удалять данные (DELETE).

Для того, чтобы создать таблицу в SQL, используется выражение CREATE TABLE. Он принимает в качестве параметров все колонки, которые мы хотим внести, а также их типы данных. Например, CREATE TABLE months (id int, name varchar(10), days int);

Также, когда создаются таблицы, принято добавлять так называемый primary key. Это колонка, значения в которой уникальны. Чаще всего primary key колонкой является id, но в нашем случае это может быть и name, так как имена всех месяцев уникальны.

Теперь давайте добавим пару месяцев в нашу табличку. Сделать это можно с помощью команды INSERT. Есть два разных способа использовать INSERT:

Первый способ не подразумевает указания названий колонок, а лишь принимает значения в том порядке, в котором они указаны в таблице: INSERT INTO months VALUES (1,'January',31);

Первый способ короче второго, однако если в будущем мы захотим добавить дополнительные колонки, все предыдущие запросы работать не будут. Для решения данной проблемы следует использовать второй способ. Его суть в том, что перед вводом данных мы указываем названия колонок: INSERT INTO months (id,name,days) VALUES (2,'February',29);

SELECT запрос используется в случае, если нам нужно показать данные в таблице: SELECT \* FROM months

Результатом данного запроса будет таблица со всеми данными в таблице characters. Знак звёздочки (\*) означает то, что мы хотим показать все столбцы из таблицы без исключений. Так как в базе данных обычно больше одной таблицы, нам необходимо указывать название таблицы, данные из которой мы хотим посмотреть. Сделать это мы можем, используя ключевое слово FROM.

Когда вам нужны лишь некоторые столбцы из таблицы, то вы можете указать их имена через запятую вместо звёздочки: SELECT name, days FROM months

Зачастую нам нужно изменить данные в таблице. В SQL это делается с помощью UPDATE:

UPDATE months

SET name = 'new\_month'

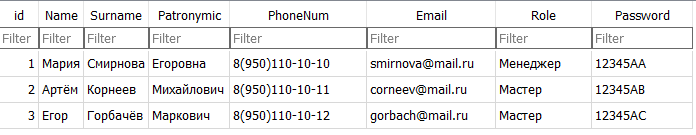
WHERE id = 1;

Удаление записи из таблицы через SQL - очень простая операция. Всё, что нужно — это обозначить, что именно мы хотим удалить:

DELETE FROM months

WHERE id = 4;

Практика:



Будут представлены блоки с данными, которые необходимо будет соотнести названиям столбцов.

После этого необходимо заполнить отчёт по курсовой, разместив в правильном порядке пункты отчёта.

Вторым заданием будет написать четыре запроса на языке SQL:

1. Создать таблицу books с полями id, name, genre;
2. Добавить запись в таблицу books со значениями (1, ‘Game of Torches’, ‘Sci-Fi’);
3. Обновить запись с id = 1, исправив name на ‘Game of Roses’;
4. Удалить запись с id = 1.
   1. Сессия 3 – Инфокоммуникационные системы и сети; Базы данных

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Зачем объединять устройства предприятия в единую сеть? | **Для предоставления совместного доступа к различным ресурсам предприятия** |
| Чтобы следит за сотрудниками |
| Для того, чтобы системные администраторы не бегали по всем отделам |
| На какие виды делятся сети по размеру? | Локальные, глобальные сети |
| **Персональные, локальные, муниципальные и глобальные сети** |
| Персональные, локальные, муниципальные, глобальные и специальные сети |
| Что такое топология сети? | Функциональное назначение сети |
| Уникальный числовой идентификатор сети |
| **Способ описания конфигурации сети в виде графа, вершинам (узлам) которого соответствуют различные сетевые устройства (например, компьютеры, принтеры или коммуникационное оборудование), а ребрам линии связи (физические, логические или информационные) между ними** |
| Какая топология на данный момент наиболее распространена? | Звезда |
| **Дерево** |
| Общая шина |
| В какой сети каждый узел является как клиентом, так и сервером? | **В одноранговой сети** |
| В одноранговой сети и сети с выделенным сервером |
| В сети с выделенным сервером |
| Что такое база данных? | **Упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе** |
| Основной объект для хранения информации |
| Отсортированный набор артефактов информации |
| Что такое СУБД? | **Программное обеспечение, которое требуется, чтобы создавать базы данных, изменять их, получать из них информацию и контролировать версии** |
| Упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе |
| Набор данных с предопределенными связями между ними. |

|  |  |
| --- | --- |
| Что является одним из основных отличий базы данных от электронной таблицы? | **Способ хранения и обработки данных** |
| Внешний вид |
| Наличие строк и столбцов |
| Какие есть командные слова в SQL? | SELECT, DISCARD, COPY, DELETE |
| SELECT, INSERT, UPDATE, DELETE, SHOW |
| **SELECT, INSERT, UPDATE, DELETE** |
| Что делает данный запрос на SQL?  “UPDATE engineers  SET country = 'Spain'  WHERE employee\_id = 1” | Добавляет новую строку в таблицу engineers |
| Удаляет строку с employee\_id = 1 и вместо неё записывает новую строку |
| **Изменяет значение поля country в строке таблицы engineers, где employee\_id=1** |

1. Семестр 4
   1. Анализ бизнес-процессов

* 1. Курсовой проект 4 – ООП

Теория:

Объектно-ориентированное программирование (сокр. ООП) — методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

**Объявление переменных.**

Переменные предназначены для того, чтобы хранить данные, с которыми работает программа в процессе выполнения.

**Объявление переменной** — это указание типа данных, которые можно в ней хранить, а также назначение некоторого идентификатора, по которому к ним можно будет обращаться. Иногда этот идентификатор неформально называют именем переменной.

Формула объявления переменной выглядит следующим образом:

**<тип данных> <имя\_переменной>;**

**# double x, y;**

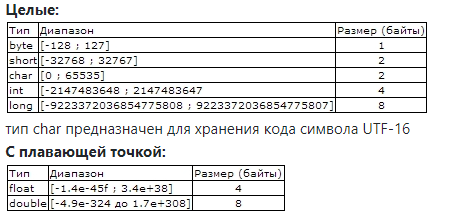
**# int a, b, A, c;**

Прежде чем перейти к рассмотрению типов данных, важно отметить, что все языки программирования регистрозависимы. Это означает что к переменной с идентификатором "name" нельзя обращаться как к "Name" (c заглавной буквы), т.е. при использовании переменной её имя должно полностью совпадать с данным ей при объявлении.

При выборе названия переменной следует знать о следующих ограничениях: имена переменных не должны начинаться с цифры, содержать пробелы и совпадать с зарезервированными словами

**Примитивные типы данных**

Всего в Java существует 8 базовых типов, еще их называют примитивными типами данных. Они делятся на три группы:

****

**Логический:**

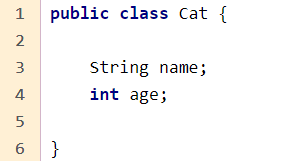
boolean - возможные значения true или false

Все остальные типы, включая строковые, относятся к ссылочным типам данных.

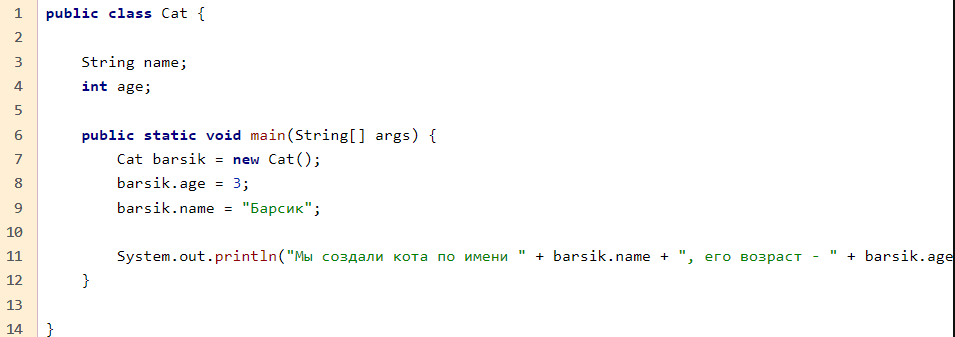
Класс в Java – это описание или прототип объекта. Класс содержит свойства и методы объекта. В свойствах хранится состояние объекта, а методы описывают поведение объекта.

Все программы состоят из объектов, которые как-то связываются между собой.

Класс — это, по сути, шаблон для объекта. Он определяет, как объект будет выглядеть и какими функциями обладать. Каждый объект является объектом какого-то класса.



Допустим, мы пишем программу, и в этой программе нам для чего-то нужны кошки (например, у нас ветеринарная клиника с возможностью онлайн-записи). Мы создали класс Cat, и указали для него две переменные — строку name, и число age. Такие переменные классов называются **полями**. По сути — это шаблон для всех кошек, которых мы создадим в будущем. У каждой кошки (объекта класса Cat) будет две переменные — имя и возраст.



Вот так это работает! Мы создали кота, присвоили ему имя и возраст и вывели все это в консоль. Ничего сложного :)

Принципы ООП

Объектно-ориентированное программирование опирается на четыре принципа:

Наследование — это передача всех свойств и поведения от одного класса другому, более конкретному. У карася и ерша, как и у всех рыб, есть плавники, хвосты, жабры и чешуя, они живут в воде и плавают.

Абстракция — это сокрытие подробностей и предоставление пользователю лишь самых важных характеристик объекта. Например, в адресе здания важны такие данные, как почтовый индекс, страна, населенный пункт, улица и номер дома. Его этажность и материал стен в таком случае не имеют значения.

Инкапсуляция — это размещение данных и методов для их обработки в одном объекте, а также сокрытие деталей его реализации. Мы знаем, как включать и выключать телевизор, переключать программы и регулировать громкость. Для этого не обязательно знать, как он устроен.

Полиморфизм — это проявление одного поведения разными способами. Животные могут издавать звуки, при этом кошка мяукает, а собака лает.

Практика:

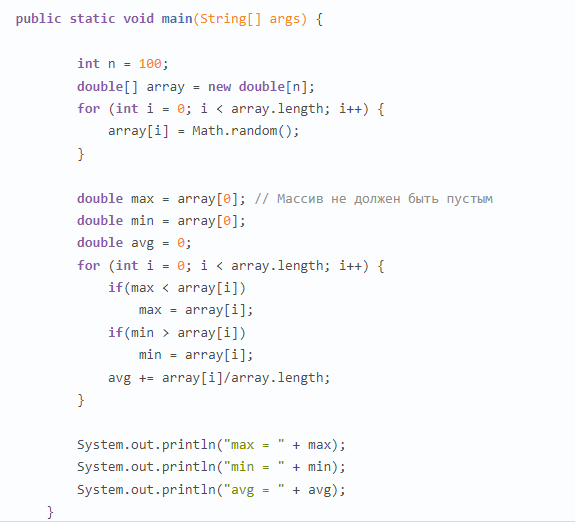
Дана формулировка задача. Необходимо расположить блоки кода в правильной последовательности

**Задача:**

Заполните массив случайным числами и выведите максимальное, минимальное и среднее значение.

Для генерации случайного числа используйте метод [Math.random()](https://docs.oracle.com/javase/10/docs/api/java/lang/Math.html" \l "random()), который возвращает значение в промежутке [0, 1].

**Решение:**



* 1. Сессия 4 – Анализ бизнес-процессов; ООП

|  |  |
| --- | --- |
| Вопросы | Ответы |
|  |  |
|  |
|  |
|  |  |
|  |
|  |
|  |  |
|  |
|  |
|  |  |
|  |
|  |
|  |  |
|  |
|  |
| Что такое ООП? | **Объектно-ориентированное программирование – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования** |
| Объектно-ориентированное программирование – так называют любой тип программирования, в котором используются понятия высокого уровня и в котором не работают напрямую с ячейками памяти ПК |
| Объектно-ориентированное программирование – просто красивое понятие. Если вдуматься, оно не несет дополнительной смысловой нагрузки, просто программисты любят аббревиатуры, так области их знаний выглядят сложнее |
| Что такое класс в java? | **Базовый элемент объектно-ориентированного программирования в языке** **java** |
| Уровень сложности программы. Все операторы делятся на классы в зависимости от сложности их использования |
| Просто одно из возможных названий переменной |

|  |  |
| --- | --- |
| Как объявить класс в коде? | **class MyClass {}** |
| new class MyClass {} |
| MyClass extends class {} |
| Что выведется в результате этого кода? | a > b |
| a = b |
| **a < b** |
| Какие принципы ООП существуют? | **Абстракция, наследование, инкапсуляция, полиморфизм** |
| Наследование, инкапсуляция, полиморфизм |
| Их не существует |

* 1. Производственная практика

**Цель освоения дисциплины:** Получение опыта профессиональной деятельности в выполнения первичного анализа предметной области и моделирования бизнес-процессов организации.

**Формируемые компетенции:** Способность выполнять анализ и проектирование бизнес-процессов с использованием современных методологических и программно-инструментальных средств

Игрок находится в фойе вуза, ему необходимо добраться до дирекции В210 и выбрать там из списка организацию, в которой он хочет проходить практику.

- ИРНИТУ (проблема: ?, возможное решение: ?)

- Яндекс (проблема: ?, возможное решение: ?)

- Дом.ру (проблема: ?, возможное решение: ?)

После выбора он попадает в аудиторию, где руководитель организации объясняет суть прохождения практики и выдает индивидуальное задание. Начинается мини-игра. Суть мини-игры: игрок попадает в организацию. У организации имеется проблема. Игроку необходимо предложить решение этой проблемы (выбрать из списка). Параллельно с прохождением практики игроку необходимо составить отчет. У игрока имеется несколько листов отчёта, необходимо расположить их в правильном порядке.

Теория:

Прохождение практики состоит из трех этапов:

* организационного этапа;
* этапа прохождения практики;
* отчетного этапа.

В соответствии с целями и задачами практики студентам необходимо осуществить углубленное обследование функций организации (подразделения), выполнить индивидуальное задание и оформить отчет.

Деятельность любого предприятия заключается в реализации соответствующих функций (т.е. решения конкретных задач). Совокупность логически связанных функций осуществляется в рамках определенных управленческих или производственных подразделений, образующих организационную структуру предприятия. В качестве объекта исследования следует выбрать либо предприятие в целом (собственно предприятие, фирму, компанию, организацию, корпорацию, открытое акционерное общество, закрытое акционерное общество и т.п.), определенное как место прохождения практики, либо одно из его подразделений в случае, когда речь идет о крупном предприятии или организации.

Содержание этапов прохождения практики.

|  |  |  |
| --- | --- | --- |
| № п/п | Этап | Содержание работ |
| 1 | Организационный | 1. Оформление документов для прохождения производственной практики.  2. Прибытие на базу практики, согласование подразделения, в котором будет организовано рабочее место.  3. Прохождение вводного инструктажа |
| 2 | Прохождение практики | 4. Изучение деятельности организации и его подразделения - конкретного места прохождения практики).  5. Исследование информационной системы и бизнес-процессов организации.  6. Самостоятельное выполнение отдельных видов работ в рамках обязанностей исполнителя или стажера (по заданию руководителя практикой от предприятия).  7. Сбор материала для написания отчета по практике |
| 3 | Отчетный | 8. Обработка и систематизация собранного материала, интерпретация полученных результатов исследования.  9. Оформление отчета о прохождении практики и его защита. |

По результатам прохождения практики обучающийся должен предоставить:

* дневник прохождения практики обучающимся;
* письменный отчет о прохождении практики;
* характеристика с места прохождения практики.

Требования к структуре отчёта о прохождении практики:

* Титульный лист.
* Индивидуальное задание на практику.
* Содержание. Объект обследования.
* Организация обследования и состав исполнителей.
* Результаты обследования и выполнения индивидуального задания.
* Заключение.
* Список использованных источников.

1. Семестр 5
   1. Нейросетевые технологии
   2. Курсовая работа 5 – Web-программирование
   3. Сессия 5 – Нейросетевые технологии; Web-программирование

1. Семестр 6
   1. Основы мобильной разработки

Теория

*Android* – открытая *операционная система* для мобильных телефонов, смартфонов, коммуникаторов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов, нетбуков и смартбуков, основанная на ядре Linux и поддерживающая различные аппаратные платформы.

Инструментарий

Как правило, разработка *Android*-приложений осуществляется на языке *Java*. Поэтому, в первую *очередь*, необходимо установить *Java Development Kit* (*JDK*).

**JDK** – это бесплатно распространяемый комплект разработчика приложений на языке *Java*, включающий в себя *компилятор* *Java*, стандартные библиотеки классов *Java*, примеры, документацию, различные утилиты и исполнительную систему *Java* Runtime *Environment* (*JRE*). В состав *JDK* не входит интегрированная *среда разработки* (Integrated *Development Environment*). Поэтому после того, как будет установлен *JDK*, следует установить *IDE*.

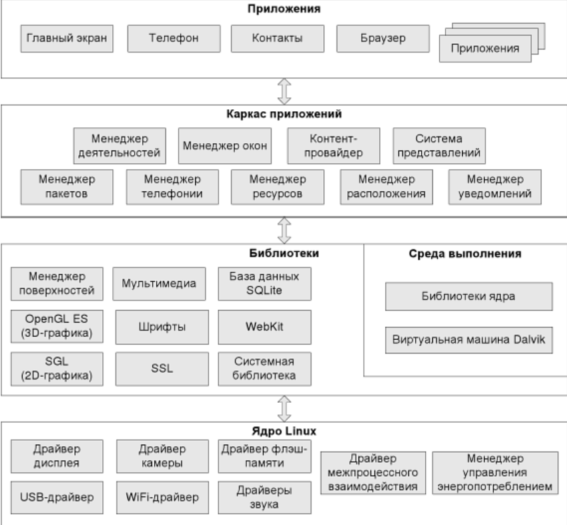
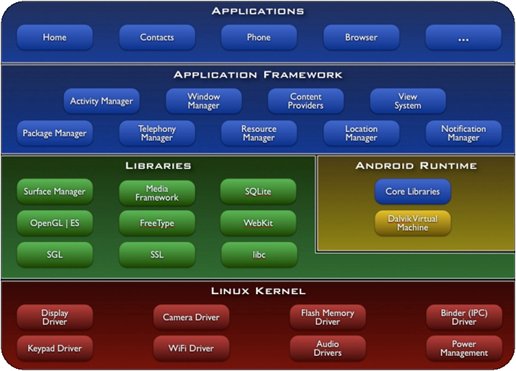
**Android Software Development Kit** (*SDK*) содержит множество инструментов и утилит для создания и тестирования приложений.

**Архитектура Android**

Рассмотрим основные компоненты операционной системы *Android*

**Applications.** *Android* поставляется с набором основных приложений, включающим календарь, карты, *браузер*, *менеджер* контактов и другие.

**Application Framework.**Предоставляя открытую платформу разработки, *Android* дает разработчикам возможность создавать гибкие и инновационные приложения.



**Libraries.** *Android* включает в себя набор C/C++ библиотек, используемых различными компонентами системы.

**Android Runtime.** Каждое *приложение* *Android* работает в своем собственном процессе, со своим собственным экземпляром виртуальной машины Dalvik. Dalvik проектировалась специально под платформу *Android*. *Виртуальная машина* оптимизирована для низкого потребления памяти и работы на мобильном аппаратном обеспечении.

**Linux Kernel.** *Android* основан на Linux версии 2.6 с основными системными службами – *безопасность*, *управление памятью*, *управление процессами* и модель драйверов.

**Структура Android-приложения**

*Android*-приложения могут быть простыми и сложными, но строение приложений всегда будет одинаковым. Есть обязательные элементы приложений, а есть опциональные, которые используются по мере необходимости.

Следующая *таблица* демонстрирует обязательные и возможные составляющие структуры *Android*-приложения:

|  |  |  |
| --- | --- | --- |
| **Название** | **Описание** | **Необходимость** |
| gen | Файлы, сгенерированные самой Java. Здесь находится такой важный файл как R.java | Да |
| AndroidManifest.xml | Файл манифеста AndroidManifest.xml предоставляет системе основную информацию о программе. Каждое приложение должно иметь свой файл манифеста | Да |
| src | Каталог, в котором содержится исходный код приложения | Да |
| assets | Произвольное собрание каталогов и файлов | Нет |
| res | Каталог, содержащий ресурсы приложения. В данном каталоге могут находиться подпапки drawable, anim, layout, menu, values, xml и raw | Да |

**Компоненты Android-приложения**

Каждое *Android*-*приложение* запускается в своем собственном процессе. Поэтому *приложение* изолировано от других запущенных приложений, и неправильно работающее *приложение* не может беспрепятственно навредить другим *Android*-приложениям.

В *Android* существует четыре типа компонентов: Activities, Services, Broadcast receivers и Content providers.

**Activities**

Activity представляет собой пользовательский интерфейс для одного действия, которое пользователь может совершить. Например, приложение для обмена текстовыми сообщениями может иметь одно Activity для отображения списка контактов, другое – для написания сообщения выбранному контакту, третье – для просмотра сообщений и еще одно для изменения настроек. Все эти Activities формируют единый пользовательский интерфейс, но не зависят друг от друга.

Приложение может состоять из одного Activity или из нескольких. Это зависит от типа приложения и его дизайна. Одно Activity может вызвать другое. Каждое Activity задает окно для отображения, которое, обычно, занимает весь экран, но может быть меньше и плавать поверх других окон. Activity может использовать дополнительные окна, например, всплывающий диалог, который требует промежуточного ответа пользователя, или окно, которое отображает пользователям важную информацию при выборе элемента, заслуживающего особого внимания.

Визуальный интерфейс строится на основе иерархии визуальных компонентов, производных от базового класса View. Android имеет ряд готовых к использованию компонентов, а также кнопки, текстовые поля, полосы прокрутки, меню, флажки и многие другие.

Activity может находиться в одном из трех состояний:

* Active или Running – находится на переднем плане и имеет фокус для взаимодействия с пользователем.
* Paused – потеряло фокус, но все еще видно пользователю. Сверху находится другое Activity, которое или прозрачно или закрывает не весь экран. Приостановленное Activity полностью "живое" (его состояние сохранено), но может быть уничтожено системой в случае нехватки памяти.
* Stopped – полностью перекрыто другим Activity. Оно больше не видно пользователю и будет уничтожено системой, когда понадобится память.

**Services**

Service – это некий процесс, который запускается в фоновом режиме. Как пример, Service может получать данные по сети, выполнять какие-либо длительные вычисления. Хорошим примером Service служит проигрыватель музыки. Пользователь может выбрать любую песню в проигрывателе, включить ее и закрыть плеер занявшись чем-нибудь другим. Музыка будет проигрываться в фоновом процессе. Service проигрывания музыки будет работать, даже если Activity плеера закрыта.

**Broadcast receivers**

Broadcast receiver– это компонент, который ничего не делает, кроме того, что рассылает и реагирует на широковещательные сообщения. Примером широковещательных компонентов могут быть: сообщения об переходе на летнее/зимнее время, сообщения об минимальном заряде батареи и так далее.

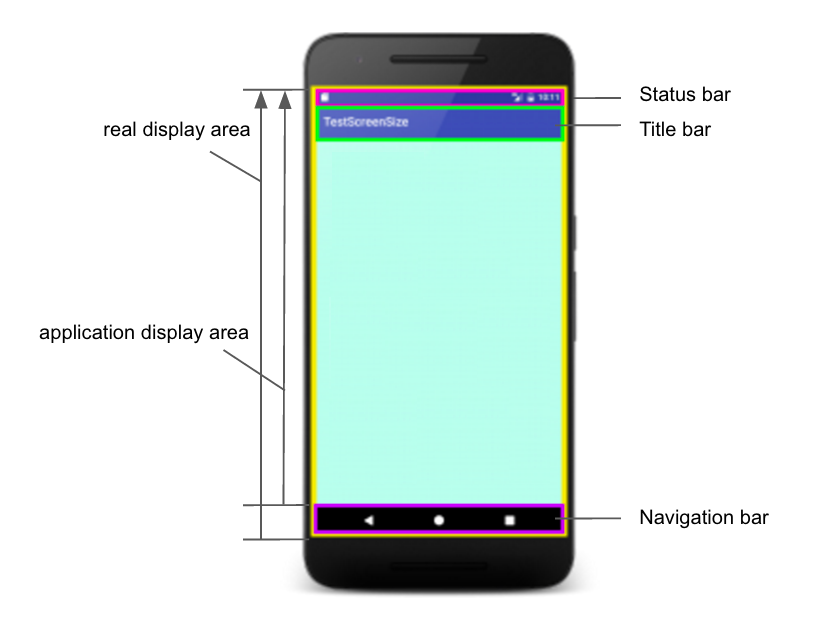
Broadcast receiver не отображает пользовательский интерфейс, но может запустить Activity на полученное сообщение или использовать NotificationManager для привлечения внимания пользователя. Привлечь внимание пользователя можно, например, вибрацией устройства, проигрыванием звука или миганием вспышки.

**Content providers**

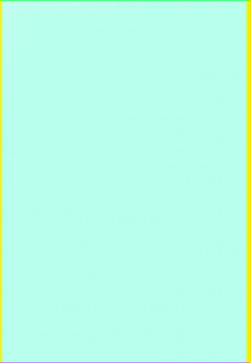
Content providers предоставляют доступ к данным (чтение, добавление, обновление). Content provider может предоставлять доступ к данным не только своему приложению, но и другим.

Практика

Переместить элементы на нужные места.

* 1. Курсовая работа 6 – Технология разработки программных комплексов

На занятии игроку рассказывают про жизненный цикл ПО, технологию разработки ПО и ее этапы.

Теория:

*Программное обеспечение (ПО)* – организованная совокупность обрабатывающих программ и обрабатываемых данных, реализованная на ЭВМ.

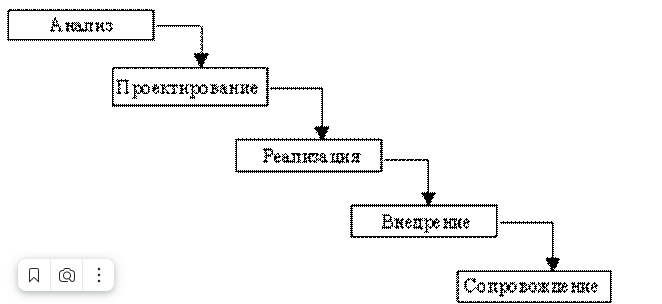
*ПО делится на две группы*

*1. Общее ПО – предназначено для обеспечения функционирования компьютера и эффективной работы на нём. Этим ПО пользуется каждый пользователь. В состав ПО входит: операционная система (ОС) и специальный комплекс программ технического обслуживания (КПТО).*

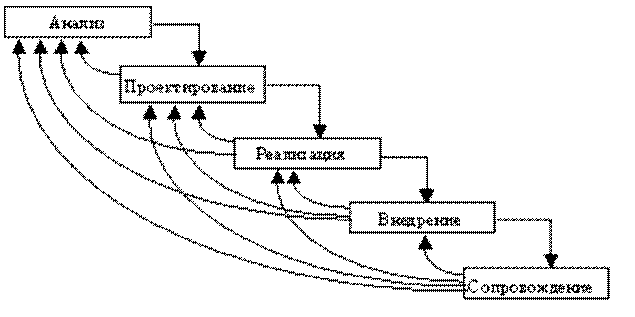
*2. Специальное (или прикладное) ПО – предназначено для решения специальных прикладных задач. С ним работают пользователи-специалисты какой-либо прикладной области (математики, экономисты, художники, программисты и др.).*

**Жизненный цикл программного обеспечения** —время от начального момента создания какого-либо программного продукта, до конца его разработки и внедрения.

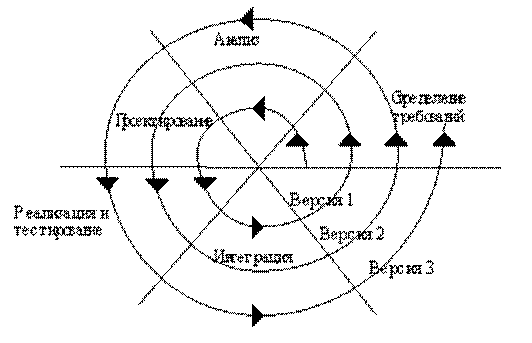
**Каскадная модель** (до 70-х годов XX в) определяет последовательный переход на следующий этап после завершения предыдущего.



**Итерационная модель** (70-80-е годы XX в.) соответствует технологии проектирования «снизу — вверх». Допускает итерационные возвраты на предыдущие этапы после выполнения очередного этапа, при этом возникает потребность в пересмотре ранее сформулированных требований.



**Спиральная модель** (80-90-е годы XX в.) соответствует технологии проектирования «сверху — вниз». Предполагает использование программного прототипа, допускающего программное расширение. Проект системы циклически повторяет путь от детализации требований к детализации программного кода.



**Технология разработки ПО** – совокупность приёмов, позволяющих создать безошибочную программу в течение заданного времени. Состоит из четырёх этапов:

**1)**формулировка задачи на естественном языке и создание математической модели;

**2)**разработка нового или выбор существующего метода численного решения математической задачи (алгоритма);

**3)**написание программы на языке программирования;

**4)**тестирование и отладка программ.

Практика:

Задание для игрока – соотнести этапы разработки ПО с тем, что происходит на этих этапах. Второе задание - соотнести модель типа с ее определением.

***1)****формулировка задачи на естественном языке и создание математической модели;*

***2)****разработка нового или выбор существующего метода численного решения математической задачи (алгоритма);*

***3)****написание программы на языке программирования;*

***4)****тестирование и отладка программ.*

*На первом этапе* необходимо наиболее глубоко исследовать предметную область (процесс, объект, явление), а также разработать наиболее полную математическую модель, учитывающую основные особенности предметной области.

*На втором этапе* при разработке алгоритма необходимо использовать приёмы *структурного программирования*, позволяющие создавать надёжно работающие программы.

*На третьем этапе* при выборе языка программирования необходимо учитывать тип решаемой задачи.

Для создания безошибочной программы за приемлемое время используются основные приемы *структурного программирования.*

Суть его заключается в следующем:

Исходная сложная задача условно разбивается на более простые подзадачи, которые являются относительно независимыми друг от друга. Каждая из этих задач программируются в отдельной программе-модуле. В результате получается структурированная иерархическая система – программа, представляющая собой композицию из последовательных или вложенных друг в друга модулей.

Тестирование и отладка написанной программы являются содержанием четвёртого этапа разработки ПО.

*Тестирование* – выполнение программы с целью обнаружения наличия ошибок.

*Тест* – совокупность специально подобранных исходных данных и соответствующих им результатов расчетов (как промежуточных, так и окончательных).

Отладка – выполнение программы с целью локализации, диагностики и исправления ошибок.

* 1. Сессия 6 – Основы мобильной разработки; Технология разработки программных комплексов

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Что такое Android? | Операционная система для ПК |
| **Открытая операционная система для мобильных телефонов, смартфонов, коммуникаторов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов, нетбуков и смартбуков, основанная на ядре Linux и поддерживающая различные аппаратные платформы** |
| Мобильное приложение |
| Из чего состоит архитектура Android? | Application, Application Framework, Libraries, Android Runtime |
| Application, Libraries, Android Runtime |
| **Application, Application Framework, Libraries, Android Runtime, Linux Kernel** |
| Сколько типов компонентов существует в Android-приложении? | Зависит от типа приложения |
| 5 |
| **4** |
| Как называется каталог, в котором содержится исходный код приложения | assets |
| **src** |
| gen |
| Сколько Activity может иметь мобильное приложение? | **Одно или несколько, зависит от типа приложения и его дизайна** |
| Только одно |
| Обязательно несколько |
| Что такое ПО? | **Программное обеспечение (ПО) – организованная совокупность обрабатывающих программ и обрабатываемых данных, реализованная на ЭВМ** |
| Программное обеспечение (ПО) – универсальное устройство для передачи информации |
| Программное обеспечение (ПО*)* – список имеющихся на компьютере программ |
| На какие группы делится ПО? | **Общее и специальное (прикладное)** |
| Личное и общественное |
| Персональное и совместное |

|  |  |
| --- | --- |
| Для чего предназначено специальное ПО? | **Для решения специальных прикладных задач. С ним работают пользователи-специалисты какой-либо прикладной области** |
| Для использования обычным пользователем |
| Для работы с важными документами |
| Какие существуют модели жизненного цикла ПО? | Каскадная, индивидуальная, спиральная |
| Круглая, квадратная, треугольная |
| **Каскадная, итерационная, спиральная** |
| Сколько существует этапов разработки ПО? | 7 |
| 4 |
| Всегда по-разному |

* 1. Научно-исследовательская практика

Игрок находится в кабинете, где руководитель по практике рассказывает о том, что включает в себя научно-исследовательская производственная практика, как она проходит и что должен включать в себя отчёт. После этого затемняется экран, и начинается мини-игра.

Прошло 4 недели практики, и студенту нужно составить отчёт.

У игрока имеется несколько листов отчёта, необходимо расположить их в правильном порядке.

Теория:

Прохождение практики состоит из трех этапов:

− организационного этапа;

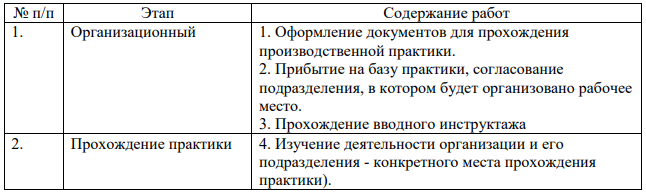
− этапа прохождения практики;

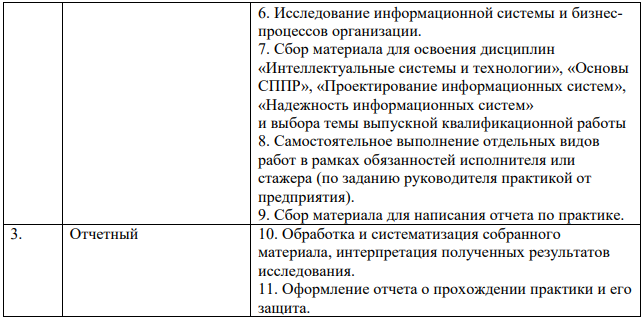
− отчетного этапа.

В соответствии с целями и задачами практики студентам необходимо осуществить углубленное обследование функций организации (подразделения), выполнить индивидуальное задание и оформить отчет.

Деятельность любого предприятия заключается в реализации соответствующих функций (т.е. решения конкретных задач). Совокупность логически связанных функций осуществляется в рамках определенных управленческих или производственных подразделений, образующих организационную структуру предприятия. В качестве объекта исследования следует выбрать либо предприятие в целом (собственно предприятие, фирму, компанию, организацию, корпорацию, открытое акционерное общество, закрытое акционерное общество и т.п.), определенное как место прохождения практики, либо одно из его подразделений в случае, когда речь идет о крупном предприятии или организации.

Содержание этапов прохождения практики приведено в таблице ниже.





По результатам прохождения практики обучающийся должен предоставить:

− дневник прохождения практики обучающимся;

− письменный отчет о прохождении практики;

− характеристика с места прохождения практики.

Требования к структуре отчёта о прохождении практики:

− Титульный лист.

− Индивидуальное задание на практику.

− Содержание. − Объект обследования.

− Организация обследования и состав исполнителей.

− Результаты обследования и выполнения индивидуального задания.

− Заключение.

− Список использованных источников.

1. Семестр 7
   1. Проектирование информационных систем

В данном семестры были изучены следующие дисциплины: ….Более углубленно игрок изучал проектирование информационных систем. (Ко всем предметам добавить описание во вкладке Расписание).

Игрок находится возле аудитории В208 со своей группой. В аудитории находится преподаватель. Зайти в аудиторию = начать игру.

Игрок, зайдя в аудиторию, подходит к преподавателю и садится за стол.

Игроку выдаётся краткий материал по типам ИС, после ознакомления необходимо сопоставить название ИС с карточкой-определением. Из всего списка использовать будем не все.

****

Типы ИС (по назначению)

1. САПР (Системы автоматизации проектных работ)
2. CAD-система (сomputer-aided design компьютерная поддержка проектирования)
3. CAE (Computer-aided engineering)
4. СAM (Computer-aided manufacturing) (например, для генерации программ обработки деталей для станков с ЧПУ)
5. PDM-системы (Product Data Management)
6. PLM-система (англ. product lifecycle management) — прикладное программное обеспечение для управления жизненным циклом продукции
7. АСУ ТП (Автоматизированная система управления технологическим процессом)
8. SCADA (Supervisory Control And Data Acquisition System)
9. MRPII (Manufacturing Resource Planning)
10. ERP (Enterprise(-wide) Resource Planning)
11. ERPII – это стратегия разработки и внедрения приложения, которая распространяется за пределы ERP-функций, чтобы обеспечить интеграцию ключевой для предприятия специфики, внутреннего и внешнего сотрудничества, операционных и финансовых процессов.
12. CRM (Customer Relationship Management System)
13. MRP (material requirement planning)
14. ECM (Enterprise Content Management) (СЭД)
15. HRM (Human Resource Management – управление персоналом)
16. **EAM** (Enterprise Asset Management) (управление ремонтами)
17. **MES**-система (Manufacturing Execution System)
18. **WMS** (Warehouse Management System) (управление складом)
19. **SCM** (Supply Chain Management) Система управления цепочками поставок
20. **CSRP** (Customer Synchronized Resources Planning) — управление, ориентированное на взаимодействие с клиентами, — планирование ресурсов, синхронизованное с покупателем.
21. **BI** (Business Intelligence) (системы бизнес-аналитики)
    1. Курсовая работа 7 – Интеллектуальные системы и технологии
    2. Сессия 7 – Проектирование ИС; Интеллектуальные системы и технологии
22. Преддипломная практика

Краткий бриф для обсуждения и обоснования выбора предполагаемой темы курсового проекта (выпускной квалификационной работы)

Игрок находится на предфинальном этапе игры (и обучения), впереди ВКР, данная курсовая выступает как моральная:D подготовка к диплому. Игроку предстоит ознакомиться с базовой информацией и некоторыми нюансами по ВКР, возможно что-то выбрать, например наставника и тему.

Это, конечно, еще не диплом, но что-то уже близкое и достаточно ответственное. На выбор будут предложены темы, игрок выбирает, например, первую из предложенных, и наставник(преподаватель) сообщает насколько тема подходит по компетенциям студента. То есть, студент ИИТиАД не сможет взять тему “Технология установки дождевой канализации в многоэтажных домах ”

1. ВКР – Создание ПЗ, создание проекта, защита ВКР

Игрок начинает в главном холле. Даётся на выбор несколько преподавателей-руководителей для написания вкр. При выборе одного из преподавателей до него необходимо будет дойти.

После нахождения преподавателя он рассказывает о том, что такое вкр, в какие сроки она пишется, как происходит защита работы.

Прочитав теорию, игрок начинает мини-игру по написанию вкр.

Теория:

Выпускная квалификационная работа (ВКР) — это комплексная самостоятельная исследовательская работа, в ходе которой студент решает конкретные практические задачи, соответствующие профилю деятельности и уровню образования, развивает практические навыки в реальных условиях в период прохождения преддипломной практики.

При написании выпускной квалификационной работы преследуются следующие цели:

- систематизация, закрепление и расширение полученных теоретических знаний по специальности и их применение для постановки и решения практических задач;

- углубление навыков ведения самостоятельной работы, овладение современными методами постановки и анализа проблем;

- развитие умения проводить обзор литературы.

Тема работы — это наиболее краткая форма отражения содержимого работы, отображающая ее сущность. Тема формулируется и указывается в виде заглавия работы.

Тема должна выбираться с учетом ее актуальности, то есть в соответствии современным представлениям, а также востребованности ее в рамках данной профессии, профессионального модуля и учебного заведения. Тема ВКР согласуется с работодателем.

Выбор темы необходимо соотносить со временем, которое предполагается отвести на ее создание. Выбрав слишком глобальную тему, обучающийся может не успеть ее рассмотреть в положенный срок.

Написание выпускной квалификационной работы состоит из следующих этапов:

1. Поиск, отбор и изучение литературы по исследуемой теме. Литературу для выпускной квалификационной работы можно найти в общественных библиотеках и в библиотеках учебных заведений; в электронных библиотеках интернета.

2. Составление плана выпускной квалификационной работы и согласование его с руководителем.

3. Написание введения к выпускной квалификационной работе. Обычно введение и заключение удобнее писать на основе уже готовой работы. Поэтому этот шаг можно оставить на потом и переходить к следующему.

4. Написание основной части выпускной квалификационной работы, которая состоит из двух частей – теоретической и исследовательской (практической). Основная часть должна всесторонне раскрывать тему выпускной квалификационной работы. Выполнение исследовательской (практической) части и ее описание во второй части выпускной квалификационной работы.

5. Написание заключения выпускной квалификационной работы.

6. Оформление списка литературы выпускной квалификационной работы.

7. Оформление приложений.

8. Получение отзыва на выпускную квалификационную работу.

9. Подготовка доклада и презентации к защите выпускной квалификационной работы.

10. Предзащита выпускной квалификационной работы (представление выпускной квалификационной работы, отзыва, доклада) в ЦМК.

11. Корректировка, получение рецензии на выпускную квалификационную работу и допуска к защите и, непосредственно, защита ВКР.

В составлении оглавления и написании следует ориентироваться на общепринятую структуру выпускной квалификационной работы.

Структура выпускной квалификационной работы и выглядит следующим образом:

* Титульный лист;
* Оглавление;
* Введение;
* Основная часть;
* Заключение;
* Список литературы;
* Приложение.

В рамках каждой структурной части может быть несколько глав/разделов, и называться они должны по-разному (в зависимости от того, на какую тему пишется работа).

Согласно нормам, объем дипломной работы должен составлять 25-40 листов.

Мини-игра:

После победы в мини-игре показывается сценка, как студент успешно защищает вкр и получает диплом.

1. Пересдача

Начинается новый семестр – у игрока появляется активный квест закрыть долг по предыдущему семестру.

Ему необходимо зайти в деканат за бегунком, после чего найти нужного преподавателя и перепройти мини-игру.