Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра казахского и русского языков

ПРОФЕССИОНАЛЬНЫЙ РУССКИЙ ЯЗЫК

Методические указания и варианты семестровых заданий для студентов специальности 5В060200 – Информатика

Алматы 2016

СОСТАВИТЕЛЬ: Ю.Г. Смирнова. Профессиональный русский язык. Методические указания и варианты семестровых заданий для студентов специальности 5В060200 – Информатика. – Алматы: АУЭС, 2016. – 40 с.

В методические указания по дисциплине «Профессиональный русский язык» включены задания семестровых работ № 1 и № 2, представлено 25 вариантов специальных текстов к семестровой работе № 2 для специальности 5В060200 – Информатика.

Библиогр. – 17 назв.

Рецензенты: канд. филол. н., доцент Р.А. Досмаханова
доцент Е.О. Елеукулов

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2016 г.

Введение

Целью изучения дисциплины «Профессиональный русский язык» является освоение языка специальности (англ. language for special purposes). Это принятое в современной лингвистике и лингводидактике обозначение функциональной разновидности литературного языка, обслуживающего профессиональное общение [1].

Изучение специальных текстов (в том числе в сопоставительном аспекте) вводит студентов в реальную жизнь языка профессии, показывает его действительное функционирование, требует творческого (а не механического) использования знаний – как языковых, так и специальных.

В разработке использовались не только новые тексты, но и апробированные в различных по уровню владения языком группах.

Впервые студентам предлагается работа с казахстанской патентной базой (www.kzpatents.com) и единой электронной терминологической базой (www.termincom.kz).

Составитель выражает благодарность Толганай Сагыновне Курманбаевой за помощь в подготовке текстов на казахском языке.

1 Семестровая работа № 1. Составление словаря терминов специальности

Цель: проявить навыки работы с терминологией по специальности в сопоставительном аспекте.

Задачи:

1) Составить картотеку словарей и справочников по специальности (переводные, терминологические, энциклопедические, на твердом носителе и онлайн-версии).

2) Выбрать текст-патент по специальности в казахстанской патентной базе (www.kzpatents.com).

3) Из выбранного текста патента выписать примерно 30-35 терминов или терминосочетаний. Дать толкование, пользуясь словарями и справочниками из составленной картотеки.

Требования: оформление – по стандарту АУЭС (СТ НАО 56023-1910-04-2014 Учебно-методические и учебные работы. Общие требования к изложению, оформлению и содержанию учебно-методических и учебных работ. – Алматы: АУЭС, 2014).

2 Семестровая работа № 2. Изучение текста по специальности в сопоставительном аспекте (перевод)

Цель: изучить особенности текста по специальности на всех уровнях (от лексики до синтакисиса и стилистики) и передать содержание текста на другом языке. Возможны следующие варианты направления перевода: с русского на казахский, с казахского на русский.

Задачи:

1) Выбрать научно-технический текст в специальной литературе (специализированные журналы, учебники, монографии, патенты и т.п.) или из предложенных в данной методической разработке вариантов.

2) Пользуясь словарями и справочниками из составленной картотеки (задание 1 СРС № 1), а также двуязычным отраслевым словарем по информатике и вычислительной технике (https://ru.termincom.kz/terminder/koldanystagy-salayk-terminder-turaly/informatika/), сделать перевод выбранного текста.

3) Сделать проверку перевода терминов и терминосочетаний на соответствие утвержденным терминам в единой электронной терминологической базе (https://ru.termincom.kz/terminder/bekitilgen-terminder-turaly/).

Требования: оформление – по стандарту АУЭС (СТ НАО 56023-1910-04-2014 Учебно-методические и учебные работы. Общие требования к изложению, оформлению и содержанию учебно-методических и учебных работ. – Алматы: АУЭС, 2014).

Варианты текстов к СРС № 2

В а р и а н т 1

Информация, неопределенность и возможность выбора

Количеством информации называют числовую характеристику сигнала, характеризующую степень неопределенности (неполноту знаний), которая исчезает после получения сообщения в виде данного сигнала. Эту меру неопределенности в теории информации называют энтропией. Если в результате получения сообщения достигается полная ясность в каком-то вопросе, говорят, что была получена полная или исчерпывающая, информация и необходимости в получении дополнительных сведений нет. И наоборот, если после получения сообщения неопределенность осталась прежней, значит, информации получено не было (нулевая информация).

Приведенные рассуждения показывают, что между понятиями «информация», «неопределенность» и «возможность выбора» существует тесная связь. Чем больше неопределенность, тем больше выбор. Чем меньше неопределенность из-за наличия информации, тем меньше возможностей выбора. При полной информации выбора нет. Частичная информация уменьшает число вариантов выбора, сокращая тем самым неопределенность.

Представим себе ситуацию, в которой человек бросает монету и наблюдает, какой стороной она упадет. Обе стороны монеты равноправны, поэтому одинаково вероятно, что выпадет одна или другая сторона. Такой ситуации приписывается начальная неопределенность, характеризуемая двумя возможностями. После того как монета упадет, достигается полная ясность и неопределенность исчезает (становится равной нулю).

Приведенный пример относится к группе событий, применительно к которым может быть поставлен вопрос типа «да – нет». Количество информации, которое можно получить при ответе на вопрос типа «да – нет», называется битом (англ. bit – сокращенное от binary digit – двоичная единица). Это количество и принимают за единицу информации. Бит –минимальная единица количества информации, и поэтому получить информацию меньшую, чем 1 бит, невозможно. При получении информации в 1 бит неопределенность уменьшается в 2 раза. Таким образом, каждое подбрасывание монеты дает нам информацию в 1 бит.

В качестве других моделей получения такого же количества информации может выступать электрическая лампочка, двухпозиционный выключатель, магнитный сердечник, диод и т. п. Включенное состояние этих объектов обычно обозначают цифрой 1, а выключенное – цифрой 0.

Рассмотрим систему из двух электрических лампочек, которые независимо друг от друга могут быть включены или выключены. Для такой системы возможны следующие состояния:

Лампа А	0	0	1	1
Лампа В	0	1	0	1

Чтобы получить полную информацию о состоянии системы, необходимо задать два вопроса типа «да –нет» − по лампочке А и лампочке В соответственно. В этом случае количество информации, содержащейся в данной системе, определяется уже в 2 бита, а число возможных состояний системы – 4. Если взять три лампочки, то необходимо задать уже три вопроса и получить, как следствие, 3 бита информации. Количество состояний такой системы равно 8 и т. д.

Связь между количеством информации и числом состояний системы устанавливается формулой Р. Хартли:

$$I = \log_2 N,$$

где I – количество информации в битах;

N – число возможных состояний.

Группа из 8 битов информации называется байтом. Если бит – минимальная единица количества информации, то байт принят в качестве ее основной единицы. Существуют производные единицы количества информации: килобайт (Кбайт), мегабайт (Мбайт), гигабайт (Гбайт), терабайт (Тбайт) и т. д. Так, в одном килобайте содержится 1024 байта.

1 Кбайт = 1024 байта = 2^{10} байта.

1 Мбайт = 1024 Кбайта = 2^{20} байта, (1024 × 1024).

1 Гбайт = 1024 Мбайта = 2^{30} байта, (1024 ×1024 × 1024).

1 Тбайт = 1024 Гбайта = 2^{40} байта, (1024 × 1024 × 1024 × 1024).

Эти единицы чаще всего используются для указания объема памяти ЭВМ. Имеются и более крупные единицы количества информации, однако в них пока нет практической надобности [8].

В а р и а н т 2

Из истории компании Intel

Американская корпорация Intel (сокр. от Integrated Electronics Technologies Incorporated) – крупнейший производитель микропроцессоров и оборудования для персональных компьютеров, компьютерных систем и средств связи. Корпорация основана в 1968 году Робертом Нойсом и Гордоном Муром. Тогда же к ним присоединился Эндрю Гроув. Целью нового предприятия стала разработка на базе полупроводниковых технологий дешевой альтернативы запоминающим устройствам на магнитных носителях.

В конце 1970 года при выполнении заказа японской фирмы Busicom инженер компании Тед Хофф сконструировал объединенную микросхему – универсальное логическое устройство, которое вызывало прикладные команды из полупроводниковой памяти. Являясь ядром набора из четырех микросхем, этот центральный вычислительный блок не только соответствовал

требованиям заказа, но и мог найти применение для выполнения разнообразных задач. Так появился микропроцессор марки 4004.

Вскоре была представлена микросхема 8008, которая единовременно обрабатывала 8 битов данных. Оба вычислительных устройства стали доступны разработчикам разнообразной продукции, предоставив широкие возможности для творчества и новаторской деятельности. В продовольственных магазинах появились первые цифровые весы – микросхема преобразовывала вес продуктов в цены и считывала этикетки с покупаемых товаров. Светофоры стали более эффективно управлять дорожным движением. Новый микропроцессор внес революционные изменения во все сферы жизни – от медицинских инструментов до кассовых систем ресторанов быстрого питания, от бронирования авиабилетов до заправки топливом на бензоколонках.

В 1981 году продукция Intel привлекла внимание гиганта американской электроники IBM, который вынашивал планы создания своего первого персонального компьютера. В 1982 году Intel разработала микросхему марки 286, состоявшую из 134 тысяч транзисторов. 286-й процессор имел производительность втрое большую, чем другие 16-разрядные процессоры того времени. Оснащенный встроенным устройством управления памятью, он стал первым микропроцессором, совместимым со своими предшественниками. Эта микросхема была применена в продукции IBM – персональном компьютере PC AT.

В 1985 году был разработан процессор Intel 386, имевший 32-разрядную архитектуру и оснащенный 275 тыс. транзисторами. Этой микросхемой, выполнявшей более пяти миллионов операций в секунду, был оснащен компьютер Deskpro 386 компании Compaq. В 1989 году был создан процессор Intel 486. Новая микросхема с 1,2 млн транзисторов была впервые оснащена встроенным математическим сопроцессором. Ее быстродействие примерно в 50 раз превышало показатель модели 4004, а рабочие характеристики были сравнимы с производительностью мощных стационарных электронно-вычислительных машин.

В 1993 году Intel выпустила процессор Pentium (пятого поколения), в 5 раз превосходящий по производительности процессор Intel 486. В нем задействованы 3,1 млн транзисторов, обеспечивающих быстродействие в 90 млн операций в секунду. В 1995 году появился процессор Pentium Pro – первый представитель семейства процессоров Intel на основе архитектуры P6. Объединивший 5,5 млн транзисторов, этот процессор был оснащен вторым кристаллом высокоскоростной кэш-памяти для повышения быстродействия.

В 1997 году Intel представила технологию MMX – новый набор команд, специально разработанный для повышения производительности мультимедийных средств. Эта технология применялась в процессорах последующих поколений. В том же году Intel представила процессор нового поколения – Pentium II. Процессоры Pentium II, оснащенные 7,5 млн

транзисторов, обеспечивали высокую производительность коммерческих приложений. Pentium II поддерживал технологию DVD и графические средства на шине AGP, что обеспечило более широкие возможности для домашних компьютеров.

В 1998 году был представлен процессор Celeron для персональных компьютеров начального уровня. Они обеспечивали возможность пользования стандартными бизнес-программами и приложениями на домашних компьютерах. Модель семейства Pentium II Xeon, появившаяся в 1998 году, была специально разработана для серверов среднего и высокого уровня, а также для рабочих станций. Процессор Pentium II Xeon был снабжен встроенной в корпус быстродействующей кэш-памятью второго уровня емкостью 512 Кбайт или 1 Мбайт, работающей на тактовой частоте процессорного ядра 400 МГц. Дальнейшие планы Intel были связаны с введением нового набора команд, который позволял бы ускорить обработку трехмерной графики и видеоданных, а также научных и инженерных приложений [13].

В а р и а н т 3

Алгоритм

Алгоритмом называется последовательность действий, которые выполняются для достижения определенного результата за конечное число шагов. Алгоритм служит для решения типовых задач.

Алгоритм обладает следующими свойствами:

1) Понятность – это свойство, которое означает, что все команды должны быть понятны исполнителю.

2) Дискретность – свойство, означающее, что каждый алгоритм можно разделить на составные части, которые выполняются как отдельный алгоритм.

3) Массовость – это возможность применения алгоритма для решения однотипных задач.

4) Конечность – это особенность, заключающаяся в том, что результат выполнения алгоритма достигается за конечное число шагов.

5) Однозначность – черта, предполагающая, что действия алгоритма и порядок их выполнения должны быть истолкованы однозначно.

6) Результативность – это получение требуемого результата за конечное число шагов.

Для записи алгоритма используют блок-схему. Блок схема – это набор графических элементов (блоков), соединенных друг с другом стрелками, каждый блок обозначает определенное действие.

Вычислительные процессы, выполняемые на ЭВМ, можно разделить на три вида: линейные, разветвляющиеся, циклические. Соответственно различают три основных типа алгоритмов – линейный, разветвляющийся, циклический.

Линейным алгоритмом называется алгоритм, в котором все действия выполняются последовательно друг за другом. Например, для того чтобы отправить SMS, необходимо:

- начало;
- включить телефон;
- выбрать команду «отправить SMS»;
- написать текст сообщения;
- ввести номер телефона абонента;
- нажать кнопку отправить;
- конец.

Любой алгоритм можно записать с помощью:

- словесно-формульной записи;
- псевдокода;
- блок-схемы;
- программы.

Линейные алгоритмы очень часто встречаются в математике.

Разветвляющийся алгоритм – это алгоритм, в котором выбирается одна или другая последовательность действий. В некоторых случаях требуется выполнить одни действия, в других случаях – другие действия. Например, если сегодня воскресенье, то занятий в университете не будет и можно пойти погулять, иначе нужно идти в класс.

В некоторых математических действиях необходимо выполнение специального условия, при которых это действие совершается. Для записи таких алгоритмов используют элемент блок-схемы «условие». Если условие выполняется, говорят «условие принимает значение – истина». Если условие не выполняется, можно сказать «условие принимает значение – ложь».

Циклический алгоритм – это алгоритм, в котором определенные действия повторяются несколько раз. Такие действия называются телом цикла. При повторяющихся действиях должно изменяться значение одной или нескольких переменных, такие переменные называются параметрами цикла. Если цикл повторяется бесконечное количество раз, то такая алгоритмическая ошибка называется зацикливанием.

Существует 3 типа цикла:

1) Цикл с параметром – цикл с заданным числом повторений. В таком цикле параметр изменяет свое значение от начального до конечного значения с определенным шагом. Если начальное значение параметра i = 0, конечное значение i = 20, а шаг h = 3, то количество повторений будет 7 (I = 0, 3, 6, 9, 12, 15, 18).

2) Цикл с предусловием начинается с проверки условия выхода из цикла. Если логическое выражение истинно, то выполняется тело цикла. В противном случае, т.е. если логическое выражение ложно, этот цикл прекращает свои действия.

3) Цикл с постусловием функционирует иначе, чем цикл с предусловием. Сначала выполняется один раз тело цикла, затем проверяется логическое выражение, определяющее условие выхода из цикла, если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае – происходит повторение тела цикла. В общем виде цикл с постусловием выглядит следующим образом. Цикл повторяется до тех пор, пока условие ложь.

Цикл с постусловием и цикл с предусловием взаимозаменяемые, но есть определенные отличия:

- в цикле с предусловием условие проверяется до тела цикла, в цикле с постусловием – после тела цикла;
- в цикле с постусловием тело цикла выполняется хотя бы один раз, в цикле с предусловием тело цикла может не выполниться ни разу;
- в цикле с предусловием проверяется условие продолжения цикла, в цикле с постусловием – условие выхода из цикла. [16]

В а р и а н т 4

Безопасность вычислительных сетей

Комплексное решение вопросов безопасности вычислительных сетей принято именовать архитектурой безопасности, где выделяются угрозы безопасности, службы безопасности и механизмы обеспечения безопасности.

Под угрозой безопасности понимается действие или событие, которое может привести к разрушению, искажению или несанкционированному использованию ресурсов сети, включая хранимую и обрабатываемую информацию, а также программные и аппаратные средства.

Угрозы подразделяются на случайные (непреднамеренные) и умышленные. Источником первых могут быть ошибки в ПО, неправильные (ошибочные) действия пользователей, выход из строя аппаратных средств и др. Умышленные угрозы преследуют цель нанесения ущерба пользователям (абонентам) вычислительных сетей и подразделяются на активные и пассивные. Пассивные угрозы не разрушают информационные ресурсы и не оказывают влияния на функционирование вычислительных сетей. Их задача – несанкционированно получать информацию. Активные угрозы преследуют цель нарушать нормальный процесс функционирования вычислительных сетей путем разрушения или радиоэлектронного подавления линий связи вычислительной сети, вывод из строя компьютера или его операционной системы, искажение баз данных и т.д.

Источниками активных угроз могут быть непосредственные действия физических лиц, злоумышленников, компьютерные вирусы и т.д.

К основным угрозам безопасности относятся: раскрытие конфиденциальной информации, несанкционированное использование ресурсов вычислительной сети, отказ от информации.

К механизмам обеспечения безопасности относятся: идентификация пользователей, шифрование данных, электронная подпись, управление маршрутизацией и др.

Для защиты вычислительной сети от несанкционированного доступа применяется идентификация пользователей (сообщений), позволяющая установить конкретного пользователя, работающего за терминалом и принимающего либо отправляющего сообщения. Право доступа к определенным вычислительным и информационным ресурсам, программам и наборам данных, а также вычислительная сеть в целом предоставляется ограниченному контингенту лиц, и система должна идентифицировать пользователей, работающих за терминалами. Идентификация пользователей чаще всего производится с помощью паролей. Пароль – совокупность символов, известных подключенному к сети абоненту, − вводится им в начале сеанса взаимодействия с сетью, а иногда и в конце сеанса (в особо ответственных случаях пароль нормального выхода из сети может отличаться от входного). Наконец, система может предусматривать ввод пароля для подтверждения правомочности через определенные кванты времени. Вычислительная система определяет подлинность пароля и тем самым – пользователя.

Для защиты средств идентификации пользователей от неправомочного использования пароли передаются и сравниваются в зашифрованном виде, а таблицы паролей также хранятся в зашифрованном виде, что исключает возможность прочтения паролей без знания ключей.

Для идентификации пользователей могут использоваться и физические методы, например, карточка с магнитным покрытием, на которой записывается персональный идентификатор пользователя, карточки с встроенным чипом. Для уменьшения риска злоупотреблений карточки, как правило, используются с каким-либо другим способом идентификации пользователя, например, с коротким паролем.

Наиболее надежным (хотя и наиболее сложным) является способ идентификации пользователя на основе анализа его индивидуальных параметров: отпечатков пальцев, рисунка линий руки, радужной оболочки глаз и др.

Секретность данных обеспечивается методами криптографии, т.е. методами преобразования данных из общепринятой формы в кодированную (шифрование) и обратного преобразования (дешифрование) на основе правил, известных только взаимодействующим абонентам сети. Криптография применяется для защиты передаваемых данных, а также информации, хранимой в базах данных.

Шифрование данных производится по алгоритму, определяющему порядок преобразования исходного текста в шифрованный текст, а дешифрование – по алгоритму, реализующему обратное преобразование. К криптографическим средствам предъявляются требования сохранения

секретности, даже когда известна сущность алгоритмов шифрования-дешифрования. Секретность обеспечивается введением в алгоритмы специальных ключей (кодов). Зашифрованный текст превращается в исходный только в том случае, когда и в процессе шифрования и дешифрования использовался один и тот же ключ. Область значений ключа выбирается столь большой, что практически исключается возможность его определения путем простого перебора возможных значений [11].

В а р и а н т 5

Объектно-ориентированная модель данных
(начало)

Объектно-ориентированная модель представляет собой структуру, которую можно изобразить графически в виде дерева, узлами которого являются объекты. Между записями базы данных и функциями их обработки устанавливаются связи с помощью механизмов, подобных тем, которые имеются в объектно-ориентированных языках программирования. Такая модель позволяет идентифицировать отдельные записи базы. Определяемый пользователем объект называют объектом-целью. Поиск в объектно-ориентированной базе состоит в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в базе.

Базовыми понятиями этой модели являются следующие: объекты, классы, методы, инкапсуляция, наследование, полиморфизм.

Понятие объекта взято из объектно-ориентированного программирования. В этой среде все состоит из объектов. Объект обладает следующими свойствами: идентифицируется уникальным неизменным образом, принадлежит к определенному классу, может посылать сообщения другим объектам, имеет внутреннее состояние.

Объектно-ориентированная база данных состоит из объектов, каждый из которых должен принадлежать к определенному классу, т.е. каждый объект – экземпляр класса. Объектно-ориентированная база данных состоит из коллекции классов. Структура и поведение объектов в объектной среде полностью определяется его классом. Класс, в свою очередь, является коллекцией объектов, при этом структура и поведение объектов одного класса одинаковы.

Класс объекта состоит из его интерфейса и закрытой области. Интерфейс класса – это то, что видно другим объектам. Он, в свою очередь, состоит из двух частей: свойства класса и методов класса. Аналогом свойств являются атрибуты отношений. Например, клиент может иметь следующие свойства: номер, Ф.И.О., адрес, телефон. К свойствам относятся также связи с другими объектами. Свойства сами могут быть объектами, что позволяет

создавать составные объекты, например, свойство Ф.И.О. может состоять из свойств: фамилия, имя, отчество.

Доступ к значениям свойств и манипулирование ими можно осуществлять только посредством методов класса. Поведение объекта задается с помощью методов его класса. Обычно они имеют форму операций и функций, которые могут содержать параметры. На уровне интерфейса видимым является только имя каждого метода и требуемые параметры. Методы служат для передачи объектам сообщений. Другими словами, метод представляет то, что, по мнению пользователя, должен делать объект. Например, клиент может сделать заказ, оплатить счет и т.п. Для каждого из этих видов деятельности должен быть соответствующий метод.

Закрытая область – это та часть определения класса, которая не видна другим объектам. Пользователю объекта предоставляется информация только о том, как работать с объектом при помощи его методов. Сама же работа объекта скрыта от пользователя. Например, могут существовать дополнительные свойства с закрытыми значениями, а также скрытые связи и сообщения другим объектам.

Свойства объектов описываются либо одним из стандартных типов, заложенных в системе, например, строковым типом String, либо типом, который конструирует сам пользователь. Этот тип определяется словом Class. Значением свойства типа Class является объект, являющийся экземпляром соответствующего класса. Каждый объект – экземпляр класса – считается потомком объекта, в котором он определен как свойства. Объект – экземпляр класса, принадлежит своему классу и имеет одного родителя. Родовые отношения в базе образуют связанную иерархию объектов.

Важным достоинством объектно-ориентированной базы является то, что пользователю не нужно знать о взаимодействии объектов: он просто обращается к конкретному объекту и использует конкретный метод. То, что при этом осуществляется воздействие на другие объекты базы, скрыто от пользователя.

Различные правила, руководящие использованием объектов, также могут быть скрыты от пользователя. Например, выбранный метод может, в свою очередь, обращаться к другим методам, например, методу проверки кредитоспособности выбранного клиента.

Понятие «класс объекта» во многом аналогично понятию «тип». Поэтому при проектировании объектно-ориентированной базы данных нужно прежде всего осуществить процесс классификации, т.е. выявить объекты с аналогичными свойствами и поведением и объединить их в классы.

Используя объекты и методы, можно хранить и неоднократно использовать не только структуру объекта базы данных, но и его поведение [17].

В а р и а н т 6

Объектно-ориентированная модель данных
(продолжение)

Объектно-ориентированная база данных – это попытка применить идеологию объектно-ориентированного программирования к технологии баз данных. Объектно-ориентированная база данных состоит из объектов, причем каждый объект принадлежит к определенному классу. Поведение объекта полностью определяется его принадлежностью к определенному классу. Процесс проектирования объектно-ориентированной базы основан на выявлении классов объектов.

Используя объекты и методы, можно хранить и неоднократно использовать не только структуру объекта базы данных, но и его поведение.

Инкапсуляция означает объединение в единое целое данных и алгоритмов (функций и методов) их обработки, а также скрытие данных внутри объектов, что повышает надежность разрабатываемого программного обеспечения. Вся информация об объекте заключается в определении его класса. Доступ к объекту может осуществляться только через его интерфейс. Поведение объекта полностью определяется принадлежностью к определенному классу.

Наследование распространяет множество свойств и методов на потомков объекта. Аналогом наследования можно считать разбиение на подтипы. Например, можно определить классы Мужчина и Женщина как наследующие класс Человек. Все эти классы будут иметь общие свойства и методы. Однако в определение новых классов можно добавить дополнительные свойства и методы.

Полиморфизм допускает в объектах разных типов иметь методы (процедуры и функции) с одинаковыми именами, что означает способность одного и того же программного кода работать с разнотипными данными.

Создание объектной модели начинается с классификации – выявления объектов с аналогичными свойствами и поведением и объединения их в классы. Например, в базе данных, содержащей диаграммы, можно классификацию начать с выделения объектов диаграмм, имеющих дату создания. Процесс классификации позволяет выделить объекты с общими свойствами и методами. Если некоторые их свойства и методы различны, то производят генерализацию и специализацию.

Генерализация выявляет классы объектов с аналогичными свойствами и образует на основе этих свойств абстрактный суперкласс. Например, в базе данных, содержащей описание геометрических фигур, можно начать проектирование с выделения классов треугольников, прямоугольников, окружностей, а затем образовать из них абстрактный суперкласс Фигуры, состоящий из свойств, общих для всех фигур.

Специализация – процесс обратный генерализации. При использовании этих процессов создается иерархия классов. Иерархии указывают цепочку наследования.

Важным процессом в объектно-ориентированной базе является агрегация. С помощью агрегации классы объектов могут связываться друг с другом, образуя класс агрегатов. Например, банковская база может содержать информацию о клиентах, счетах, филиалах, а также связи между ними. В объектно-ориентированной базе всю эту информацию можно инкапсулировать в одном агрегированном классе объектов.

Создание объектно-ориентированной базы данных основано на процессах классификации, генерализации, специализации и агрегации, которые проводятся параллельно.

Основным достоинством объектно-ориентированной модели данных по сравнению с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель позволяет также идентифицировать отдельные записи в базе и определять функции их обработки. Учитывая эти достоинства, сегодня уже некоторые реляционные системы управления базами данных дополняют функциями, позволяющими воспользоваться преимуществами объектной технологии.

Основной недостаток объектно-ориентированной модели состоит в сложности понимания ее сути и низкой скорости выполнения запросов. В настоящее время объектно-ориентированные базы данных достаточно сложны, и поэтому их коммерческое использование идет медленно. Но у этих моделей есть потенциал, а, следовательно, и будущее. Поэтому исследования в области объектной ориентации становятся главным направлением в теории систем управления базами данных.

Сегодня уже разработаны и успешно функционируют такие системы управления базами данных, как Iris, Orion и др., обслуживающие эти модели [17].

В а р и а н т 7

Linux

Linux – свободная UNIX-подобная операционная система. Она основана на системных программах, разработанных в рамках проекта GNU, и на ядре Linux. Обычно (по историческим причинам и для краткости) эта система называется просто Linux.

К операционной системе GNU/Linux также часто относят программы, дополняющие эту операционную систему, и прикладные программы, делающие ее полноценной многофункциональной операционной средой.

В отличие от большинства других операционных систем, GNU/Linux не имеет единой «официальной» комплектации. Вместо этого GNU/Linux поставляется в большом количестве так называемых дистрибутивов, в

которых программы GNU соединяются с ядром Linux и другими программами. Наиболее известными дистрибутивами GNU/Linux являются Slakeware, Red Hat, Fedora, Mandriva, SuSE, Debian, Gentoo, Ubuntu.

Большинство пользователей для установки GNU/Linux используют дистрибутивы. Дистрибутив – это не просто набор программ, а ряд решений для разных задач пользователей, едиными системами установки, управления и обновления пакетов, настройки и поддержки. Самые распространенные в мире дистрибутивы:

- американский Red Hat и его наследник Fedora;
- немецкий SuSE;
- французский Mandriva (бывший Mandrake);
- не имеющий национальной принадлежности международный дистрибутив Debian GNU/Linux;
- один из самых старых дистрибутивов Slackware;
- сравнительно молодой и активно развивающийся дистрибутив Gentoo;
- очень молодой, но перспективный дистрибутив Ubuntu Linux и его форк Kubuntu Linux, использующий KDE вместо Gnome.

Помимо перечисленных, существует множество других дистрибутивов, как базирующихся на перечисленных, так и созданных с нуля и зачастую предназначенных для выполнения ограниченного количества задач. Каждый из них имеет свою концепцию, свой набор пакетов, свои достоинства и недостатки. Не один из них не может удовлетворить всех пользователей, а потому рядом с лидерами благополучно существуют другие фирмы и объединения программистов, предлагающие свои решения, свои дистрибутивы, свои услуги. Существует множество LiveCD, построенных на основе GNU/Linux, например, Knoppix. LiveCD позволяет запускать GNU/Linux непосредственно с компакт-диска без установки на жесткий диск.

Для желающих досконально разобраться с GNU/Linux подойдет любой из дистрибутивов, однако довольно часто для этой цели используются так называемые дистрибутивы source-based, то есть предполагающие самостоятельную сборку всех (или части) компонентов из исходных кодов, такие как LFS, Gentoo или CRUX [2].

В а р и а н т 8

Из истории криптографии

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом, возникла в глубокой древности.

При раскопках в Месопотамии был найден один из самых древних зашифрованных текстов. Он был написан клинописью на глиняной табличке и содержал рецепт глазури для покрытия гончарных изделий.

Древние греки использовали для шифрования длинную узкую ленту, намотанную на посох – скиталь, и писали вдоль посоха. Прочесть текст можно было только после намотки ленты на цилиндр такого же диаметра. Метод дешифрования придумал Аристотель, который наматывал ленту на конус, и то место, где появлялось читаемое слово или его часть, определяло неизвестный диаметр цилиндра.

Юлий Цезарь использовал замену одних букв на другие, но такой шифр считается нестойким.

Грек Полибий придумал другой метод, получивший название «квадрат Полибия». Буквы были вписаны в квадрат, и каждую букву текста заменяли на пару чисел – номер строки и номер столбца. Идея Полибия была реализована в более сложных шифрах, применявшихся во время Первой мировой войны.

Более сложный шифр замены создал К.Гаусс.

Итальянский математик XVI века Джироламо Кардано изобрел систему шифрования, которая применялась в военном флоте Великобритании во время Второй мировой войны: в куске картона с размеченной решеткой случайным образом прорезали отверстия, решетку клали на бумагу и писали по отверстиям текст. После снятия картона промежутки бессмысленного набора букв дописывали до псевдосмысловых фраз, что позволяло скрыть сам факт передачи секретного сообщения.

Первую в мире шифрслужбу создал в XVII веке во Франции кардинал Ришелье.

Шифрование и дешифрование оставались уделом узкого круга умельцев до 1949 года, когда появилась работа Клода Шеннона «Теория связи в секретных системах», в которой проведено фундаментальное научное исследование шифров и важнейших вопросов их стойкости.

Использование криптографических методов стала особенно актуальным в настоящее время в связи с передачей в Интернете больших объемов информации государственного, военного, коммерческого и частного характера. В связи с высокой стоимостью ущерба от потерь, разглашения и искажения информации, хранящейся в базах данных и передаваемых по локальным сетям, в современных информационных системах рекомендуется хранить и передавать информацию в зашифрованном виде [10].

В а р и а н т 9

Компьютерлердің буындары

Компьютерлердің бірінші буыны – 1959 жылға дейін шығарылған электронды лампалық машиналар, разрядтылығы 31-43 бит, жедел жадыларының көлемі 1-4 кб. операциялардың жұмыс тәртібі тек қана тізбекті, яғни келесі жасалатын операция ағымдағы амал орындалып біткеннен соң ғана басталады, енгізу/шығару амалдары орындалып тұрғанда орталық

процессор тоқтап тұрады. Программа негізінен машиналық тілде қолмен жазылып орындалды. Жұмыс істеу режимі ашық болады, яғни, әрбір программалаушы басқару тетігінде өзі отырып программасын енгізіп жұмыс істетті. Негізінде сандық шамалармен байланысты есептер символдық шамаларды пайдаланбай шығарылды. Стандартты программалар жасала бастады.

Компьютердің екінші буыны – 1968 жылға дейін шығарылған транзисторлық компьютерлер, разрядтылығы 31-48 бит, жедел жадыларының көлемі 8-128 Кб. Процессордың жұмысын тоқтату және оны өңдеу жүйесі пайда болды (ол негізінен, енгізу/шығару амалдарын орындау кезінде іске қосылады). Алгоритмдік тілдерден машиналық тілге автоматты аударылатын программалар- трансляторлар шықты, яғни, программа құру үшін деңгейлері жоғары программалау (Fortran, Algol, Cobol және т.б.) тілдері қолданылды, стандартты программалардың қоры үлкейді.

Компьютердің үшінші буыны – 1970 жылдан бастап интегралды микросұлбалар арқылы жасалынған компьютерлер мен компьютерлер кешені, разрядтылығы 32-64 бит, жедел жадыларының көлемі 64-1024 Кб. Дамыған тоқтату жүйесі бар, енгізу/шығару амалдарының орындалуы орталық процессордың жұмысымен параллель жүргізетін қосымша процессорлар (арналар) қолданылды. Бұрын программалар атқаратын көп жұмыстар, соның ішінде жұмысты тоқтатуды ұйымдастыру және өңдейтін аппарат арқылы жүзеге асатын болды. Компьютердің сыртқы ортаны қабылдап және оған әсер ете алатын сенсорлық қондырмалары пайда бола бастады. Осылар компьютерді алдын ала енгізілген деректерді детерминді өңдейтін құрылғыдан сыртқы ортада туатын жағдайға қарай жұмыс істей алатын зерделі құрылғыға айналды. Жедел жадыны қорғау және динамикалық бөлу іске асты. Көптеген жоғары деңгейлі, солардың ішінде символдық есептерге бағытталған программалау тілдері қолданылды, символдық есептер мен логикалық есептер үлесі көбейді. Программалардың жұмысын бастан аяқ басқаратын (сыртқы және ішкі ортадағы жағдайларға мақсатты жауап бере алатын) дамыған операциялық жүйелер жұмыс істеді. Осы буынның негізгі ерекшелігі программалары төменнен жоғары қарай ұйқас болатындай біртекті және мүмкіншіліктері өспелі компьютерлердің бірнеше модулдерінен тұратын машиналар кешенінің пайда болуы (мысалы социалистік елдерде ЕС ЭВМ 1020-1050, ал АҚШ-та IBM/ 360-370 сияқты компьютерлердің бірыңғай жүйелері).

Компьютердің төртінші буыны – 1975 жылдан бастап үлкен немесе өте үлкен интегралды микросұлбалар арқылы жасалынған көп процессорлы суперкомпьютерлер мен микрокомпьютерлер (кейін оларды дербес компьютер деп атап кетті). Жалпы осы буындағы компьютерлер арқылы байланыс әдістері одан әрі дамып телефон, телеграф желілеріне қосылып компьютерлік глобалды (мысалы интернет), корпоративтік және жергілікті желілер құрылды, өте үлкен деректер архиві жиналды, деректердің визуалдық

(бейнелік) түрдегі берілуі және өңделуі дамыды, нақты уақыт масштабында жұмыс істей алатын жүйелер кеңінен жүзеге асты.

Компьютердің бесінші буыны – 1980 жылы Жапония жариялаған он жылдық жобадан басталды, онда компьютердің машиналық тілі ретінде логикалық программалау тілі PROLOG-ты апаратты түрде жүзеге асырып, жасанды зерде (интелект) жүйесін құру көзделді. Бұл жоба нәтижелі аяқталды, қазір өзінің жасанды зердесі бар, яғни, белгілі есептің берілгені бойынша оның шешуін табатын, тұжырымдарды жасап және оны дәлелдей алатын, белгілі тақырыпқа өлең немесе музыка шығара алатын және т.с.с интелектуалды жұмыстарды өздігінен жасай алатын компьютерлер бар. Бірақ олар кең таралмаған, себебі олардың бағасы өте қымбат және олармен жұмыс істеу аса біліктілікті талап етеді.

Компьютердің алтыншы буыны – өткен ғасырдың 90-шы жылдарының ортасынан бастап қолға алына бастады. Ол жасанды нейрон желісіне, көп мәнді логикаға және кванттық есептеу теориясына негізделіп жасалды. Бұл компьютерлердің дамыған жасанды зердесі болады: олардың өзін-өзі оқытатын қабілеті және өздігінен кейбір мәселені түсініп (образды танып), жобалап, оны шешу немесе жүзеге асыру үшін керекті программаны немесе құрылғыны құрастыра алатын мүмкіншілігі болады [6].

Вариант 10

Дербес компьютерлер

Компьютерлердің үшінші буыны транзистерден интегралды электрондық микросұлбаларға өтумен байланысты болды. Электрондық сұлбалардың көлемдерінің көп кішірейгені компьютелердің негізгі құрылғыларының көлемдерін кішірейтуге алып келді де, өте жылдам жұмыс істейтін процессорлардың шығуына себеп болды. Өте жылдам жұмыс істейтін негізгі құрылғылар мен ақырын жұмыс істейтін енгізу құрылғыларының арасында үлкен қарама-қайшылық туды. Сыртқы құрылғылардың жұмысын басқаратын процессор көп уақытта сырттан келетін деректерді немесе нәтижені сыртқа шығаруды күтіп қарап тұрды. Бұл проблеманы шешу үшін процессорды сыртқы дүниемен қатынас жасауды басқару міндетінен босату ұйғарылды, нәтижесінде ол жұмыс сыртқы құралдың процессоры деген арнаулы жасалынған құрылғыларға тапсырылды. Бұл құрылғылардың бірнеше атаулары бар: байланыс арнасы, енгізу/шығару процессоры контроллер.

Контроллердің өзінің командалар жүйесі болады және өзіне берілген сыртқы құрылғылардың жұмысын осы командалар арқылы құрылған белгілі программа бойынша басқарады. Осындай программа нәтижесі орталық процессор оқи алатындай етіліп контроллердің ішкі регистріне жазылады. Берілген есепті шешетін программа орындалып жатқанда сыртқы дүниемен алмасу қажет болса, орталық процессор ол туралы контроллерге тек тапсырма

береді де, программаны орындауға байланысты өзінің басқа жұмыстарымен айналыса береді, ал сыртқы дүниемен алмасу жұмысын контроллердің өзі басқарады.

Компьютердегі барлық байланыс орталық шина арқылы жүзеге асады. Шинаның үш бөлігі болады: деректер берілетін шина, адрестер берілетін шина, басқару берілетін шина.

Жоғарыда көрсетілген архитектураны ашық деп атайды. Себебі, оған өзіңізге керек қана құрылғыларды алып ықшамдауға да болады немесе жаңа құрылғыларды қосып мүмкіншілігін күшейте беруге болады.

Деректерді шығаруға арнаулы құрылғы дисплей жасалынды, оның жұмыс істеу принципі кәдімгі телевизор сияқты. Дисплей өте жылдам жұмыс істейтін болғандықтан оған арнаулы жады керек болды.

Қазіргі кезде дербес компьютерлерде деректер ағымының өсуіне байланысты шинаның бірнешеуі қажет бола бастады. Оның бір түрі өте жылдам жұмыс істейтін құрылғылар үшін, ал екіншісі ақырын істейтін құрылғылар үшін өндірілді.

Сонымен дербес компьютерлердің ашық архитектурасы сыртқы құрылғылардың саны мен санасының тұрақты дамып тұруына және мүмкіншіліктері жоғары жаңа орталық құрылғылардың шығуына себеп болды. Мұндай компьютерлерде орталық процессордан басқа бірнеше арнаулы (математикалық видео) процессорлар жұмыс істей бастады және компьютер аралық байланыстар дамып неше түрлі (локалдық, корпорациялық, глобалды) компьютерлік желілер пайда болды.

Дербес компьютерлердің интерфейстері дамыған болғандықтан олармен жұмыс істеу ыңғайлы және тиімді. Ал дербес компьютерлер кең көлемде шығарылғандықтан олардың бағасы тұрақты төмендей береді. Сондықтан оларды қолдану экономикалық жағынан да тиімді.

Дербес компьютерлердің типтері өте көп. Солардың ішінде біздің елімізде кең тараған IBM PC типтес дербес компьютерлерді қарастырамыз. Мұнда IBM (International Business Machine) – АҚШ-тағы фирма аты, ал PC (Personal Computer) – дербес компьтерлер дегенді білдіреді [6].

В а р и а н т 11

Дербес компьютер архитектурасы

Дербес компьютерлер тұрмыстық, оқу жүйесіне арналған және кәсіби (әмбебап) деп аталатын үш түрге бөлінеді. Тұрмыстық компьютерлер – компьютерлердің қарапайым түрлері (1970-жылдары дайындалған, қазіргі кезде өндірістен алып тасталған БК-0010, Ириша, Apple, т.б.). Олардың жадтарының сыйымдылығы үлкен емес, атқаратын жұмыстары да жеткіліксіз болатын.

80-90 жылдар арасында оқу орындарында «Информатика мен есептеуіш техника негіздері» пәнін оқып үйрену үшін жад көлемі 64,128 Кбайт шамасындағы Корвет, УКНЦ, Ямаха MSX-1, Ямаха MSX-2, т.б оқулық компьютерлері пайдаланылып келеді.

Халық шаруашылығының түрлі салаларында пайдалануға болатын, қазіргі кездегі оқу орындарын толық қамтамасыз етілген, жұмыс істеу мүмкіндігі жоғары және әрекет жылдамдығы үлкен дербес компьютерлер әмбебап не кәсіби дербес компьютерлер деп аталады. Мұндай машиналарды және олардың негізгі құрылғысы – процессорларды шығарумен алғашқы рет дүние жүзінде Apple, IBM, Intel, т.б фирмалар айналысқан болатын. ДК-де процессор үшін ҮИС, АҮИС пайдалатындықтан, оны микропроцессор деп атайды. Бұл жұмыста, әсіресе, IBM, Intel фирмаларының еңбегі көп. Intel фирмасының жасап шығарған түрлі үлгілі (модельді) микропроцессорларын пайдаланып, IBM фирмасы компьютердің түрлі сапалы типтерін дүниеге келтірді. Фирманың компьютерлеріне, типтеріне қарай ретімен IBM PC, IBM PC XT, IBM PC AT, IBM PS/2, Pentium (Пентиум), т.б. аттар берілген (PC (Personal Computer) — дербес компьютер, XT(extended technology) — кеңейтілген технология, PS (Personal System) — дербес жүйе, AT (Advanced Technology) — алдыңғы технология). Олар, әсіресе, IBM PC AT типтес компьютерлер кезінде дүние жүзіне кең таралды. Қазіргі кезде мұндай компьютерлердің Пентиум сияқты жаңа жетілдірілген нұсқалары пайдалануда.

Пентиум сияқты кәсіби компьютерлер күрделі әр алуан математикалық, физикалық есептерді шешіп қана қоймай, логикалық операцияларды да орындай алады (логика – ойлау, қисын; логикалық операция – «ақылмен» орындалатын әрекеттер. Мысалы, автоматты түрде технологиялық процесстерді басқарады, шахмат ойнайды, күрделі график сызады). Оның үстіне, олар экономикалық ғылымдарда, жоспарлауда, медицинада, биологияда, ауа райын болжауда, мекеменің айлық ақпар тізімін жасауда, кітапхана жұмысын ұйымдастыруда және т.б. оларда ойдағыдай пайдаланып келеді. Құрылысы бойынша компьютерлер бірнеше түрлі болады: үстел үстінде тұратын (DeskTop), ұтқыр (NoteBook), қалтаға салуға келетін (Pocket PC), алақанға сыятын (Palm PC) т.б. (төменде берілген суретке қараңыз) [6].

В а р и а н т 12

Особенности языков программирования

Языки программирования занимают некоторое промежуточное положение между формальными и естественными языками. С формальными языками их объединяют строгие синтаксические правила, на основе которых строятся предложения языка. От языков естественного общения в языки программирования перешли лексические единицы, представляющие собой

основные ключевые слова (чаще всего это слова английского языка, но существуют языки программирования, чьи ключевые слова заимствованы из русского и других языков). Кроме того, из алгебры языки программирования переняли основные обозначения математических операций, что также делает их более понятными человеку.

Для задания языка программирования необходимо решить три вопроса:

- определить множество допустимых символов языка;
- определить множество правильных программ языка;
- задать смысл для каждой правильной программы.

Только первые два вопроса полностью или частично удается решить с помощью теории формальных языков. Для решения остальных вопросов приходится прибегать к другим, неформальным методам.

Первый вопрос решается легко. Определяя алфавит языка, мы автоматически определяем множество допустимых символов. Для языков программирования алфавит – это чаще всего тот набор символов, которые можно ввести с клавиатуры. Основу его составляет младшая половина таблицы международной кодировки символов (таблицы ASCII), к которой добавляются символы национальных алфавитов.

Второй вопрос решается в теории формальных языков только частично. Для всех языков программирования существуют правила, определяющие синтаксис языка, но, как уже было сказано, их недостаточно для того, чтобы строго определить все допустимые предложения языков программирования. Дополнительные ограничения накладываются семантикой языка. Эти ограничения оговариваются в неформальном виде для каждого отдельного языка программирования. К таким ограничениям можно отнести необходимость предварительного описания переменных и функций, необходимость соответствия типов переменных и констант в выражениях, формальных и фактических параметров в вызовах функций и др.

Отсюда следует, что практически все языки программирования, строго говоря, не являются формальными языками. И именно поэтому во всех трансляторах, кроме синтаксического разбора и анализа предложений языка, дополнительно предусмотрен семантический анализ.

Третий вопрос в принципе не относится к теории формальных языков, поскольку, как уже было сказано, такие языки лишены какого-либо смысла. Для ответа на него нужно использовать другие подходы [9].

В а р и а н т 13

Формальное определение транслятора

Транслятор – это программа, которая переводит программу на исходном (входном) языке в эквивалентную ей программу на результирующем (выходном) языке.

В этом определении слово «программа» встречается три раза, и это не ошибка и не тавтология. В работе транслятора действительно участвуют три программы.

Во-первых, сам транслятор является программой. То есть транслятор – это часть программного обеспечения (ПО), он представляет собой набор машинных команд и данных и выполняется компьютером, как и все прочие программы в рамках операционной системы (ОС). Все составные части транслятора представляют собой динамически загружаемые библиотеки или модули этой программы со своими входными и выходными данными.

Во-вторых, исходными данными для работы транслятора служит программа на исходном языке программирования – некоторая последовательность предложений входного языка. Эта программа называется входной, или исходной программой. Обычно это символьный файл, но этот файл должен содержать текст программы, удовлетворяющий синтаксическим и семантическим требованиям входного языка. Кроме того, этот файл несет в себе некоторый смысл, определяемый семантикой входного языка. Часто файл, содержащий текст исходной программы, называют исходным файлом.

В-третьих, выходными данными транслятора является программа на результирующем языке. Эта программа называется результирующей программой. Результирующая программа строится по синтаксическим правилам выходного языка транслятора, а ее смысл определяется семантикой выходного языка.

Важным пунктом в определении транслятора является эквивалентность исходной и результирующей программ. Эквивалентность этих двух программ означает совпадение их смысла с точки зрения семантики входного языка (для исходной программы) и семантики выходного языка (для результирующей программы). Без выполнения этого требования сам транслятор теряет всякий практический смысл.

Итак, чтобы создать транслятор, необходимо прежде всего выбрать входной и выходной языки. С точки зрения преобразования предложений входного языка в эквивалентные им предложения выходного языка транслятор выступает как переводчик. Например, трансляция программы с языка С в язык ассемблера, по сути, ничем не отличается от перевода, скажем, с русского языка на английский, с той только разницей, что сложность языков несколько иная. Поэтому и само слово «транслятор» (английское «translator») означает «переводчик».

Результатом работы транслятора будет результирующая программа, но только в том случае, если текст исходной программы является правильным – не содержит ошибок с точки зрения синтаксиса и семантики входного языка. Если исходная программа неправильная (содержит хотя бы одну ошибку), то результатом работы транслятора будет сообщение об ошибке (с дополнительными пояснениями и указанием места ошибки в исходной

программе). В этом смысле транслятор сродни переводчику, например, с английского, которому подсунули неверный текст [9].

В а р и а н т 14

Ішкі жад бөлімдері

Жоғарғы жадтың құрамында компьютерді жасап шығарған зауытта DOS-тың бір бөлігі (BIOS) қойылған бөлім бар. Оны тұрақты есте сақтау құрылғысы (ТЕСҚ) деп атайды. ТЕСҚ-ның көлемі үлкен емес 32-64 Кб. Пайдаланушының ТЕСҚ-ға берілгендерін не өз программасын сақтау мүмкін емес. Ондағы бейнежад – монитор адаптері (бейнеадаптер) пайдаланатын жад.

8088 үлгі ТВМ ХТ компьютерде жоғарғы және қосымша жадтардың сыйымдылығы 384 Кбайтан, ал негізгі және қосымша жад бөлімдері оперативті есте сақтау құрылғысын құрайды. Қосымша және негізгі жад бірдей микропропроцессолардан тұратын бөлек тақталардан тұрады. Сондықтан бұл компьютердегі ОЕСК-ның көлемі 1 Мбайтқа тең. Компьютер электр желісіне қосылған кезде ОЕСК-ға DOS-тың BIOS-тан басқа бөлімдері және түрлі жүйелік программалар да енгізіледі.

80386 SX-тен бастап шыға бастаған кәсіби компьютерлер жоғарғы жағы (ТЕСҚ-дан басқа бөлімдері) машинаның сыртқы құрылғыларының драйверлері мен деректер енгізілетін ОЕСК-ның бір бөлігі ретінде де пайдаланылатын болды. Оған қосымша, 80286 АТ және одан жоғарғы типті кәсіби компьютерлердің үлкен көлемді информациямен жұмыс істеуі үшін жадты динамикалық және виртуалды тәсілдер бойынша пайдалану режимдері бар.

Динамикалық тәсіл – дегеніміз жоғарғы жадтың бір бөлігінде 64 Кбайттық не одан да үлкен программа бөліктерін (беттерін) кезегімен бір-бірлеп бірден орындап, орындау аяқталған соң осыған бөлінген орынды автоматты түрде тазалап қоюды айтамыз.

Виртуалды тәсілмен құрылған жад (виртуалды жад) арнайы программалар бойынша ішкі жадтың ТЕСҚ-дан басқа бөліктерінен және қатты жадтан (винчестерден) құрылатын уақытша жылдам жад болып табылады.

Үлкен ақпаратымен динамикалық тәсіл бойынша жұмыс істеу кезінде компьютер жадының кеңістігін нақты мөлшеріндегі беттерге не сегменттерге бөліп, адрестеп қояды да, кезекті мәндерді осы орындарға енгізіп, оларды бірден оқып шығарады. Мұндай тәсілмен компьютердің уақытша құрған жадын виртуалды жад деп атайды. Компьютер электр желісінен ажыратылған кезде ол жойылып кетеді (birtualis- мүмкін болатын).

Соңғы кәсіби компьютерлерде ОЕСҚ-ны кеңейту үшін жүйелік блокқа қосымша жад орнатылған басқа тақшаларды да енгізіп қою мүмкіндігі бар.

Жалпы, процессор төрт бөлімнен: арифметикалық логикалық құрылғы (АЛУ), басқару құрылғысы (БҚ), жалпы міндет атқаратын регистірлер және

өте жылдам жұмыс істейтін, аумағы үлкен емес кэш-жадтан (cache-жасырып қойма) тұрады. АЛУ берілген арифметикалық логикалық амалдарды орындайды. БҚ-ға жалпы басқару программалары енгізіледі; регистірлерде негізгі жадтан түсетін аралық мөрлер сақталады. Кэш-жад қойма сияқты, оған аралық деректер мен командалар келіп түседі. Көп мәліметтерді кэш жадтан алу әрекеті процессордың мәліметтерді алуына жеңілдік келтіріп, оның жылдамдығын арттыруға жәрдемдеседі. Яғни кэш-жад компьютердің жалпы жұмыс өнімділігін арттыруға арналған. Кэш-жад екі деңгейлі. Біріншісі процессор ішінде, екіншісі процессордың сыртында (тақшада) орналастырылады, көлемі 256-512 Кбайт шамасында [6].

В а р и а н т 15

Компьютерлік желілердің жіктелуі

Коммуникациялық желі – генерация функцияларын жүзеге асыратын объектілерден тұратын жүйе, желінің пункті (тораптар) деп аталатын өнімді өзгерту, сақтау және тұтыну, сондай-ақ пунктілер арасында өнімді тапсыруды жүзеге асыратын тапсыру сызықтары.

Ақпараттық желі – бұл коммуникациялық желі, мұнда ақпарат өнімді генерациялаушы, өңдеуші, сақтаушы және қолданушы ретінде болады.

Есептеу желісі – құрамына есептегіш жабдық кіретін ақпараттық желі. Деректердің қайнар көзі және қабылдағышы болатын ЭЕМ, шеттегі құрылғылардың есептеу желісінің компоненттері болуы мүмкін. Бұл компоненттер деректердің шетті жабдықтауын құрастырады (ДШЖ немесе Data Terminal Equipment). ДШЖ ретінде ЭЕМ, принтерлер, плоттерлер, сондай-ақ автоматты және автоматтандырылған жүйелердің басқа есептеу, өлшеу және атқару жабдықтары алға шыға алады. Мәліметтерді тарату, ортақ деректерді беру ортасы деп аталатын орталардың және құралдардың арқасында жүзеге асырады.

Есептеу желісі – бұл өзара байланысқан және келістіріле істелетін бағдарламалық және аппараттық компоненттердің - компьютерлердің, коммуникациялы жабдықтаудың, операциялық жүйелердің, желілі қосымшалардың күрделі кешені.

Кез келген желінің негізінде стандартталған компьютерлік платформалардың аппараттық қабаты жатыр. Қазіргі уақытта желілерде әр түрлі кластағы компьютерлер кең және табысты қолданылуда - арнайы компьютерлерден басталып мэйнфреймдер мен супер ЭЕМ-ге дейін.

Есептеу желілері мына белгілер қатарымен топтастырылады.

Есептеу желілерін байланатын тораптар арасының қашықтығына байланысты былай бөледі:

а) аумақтық — маңызды географиялық кеңістікті қамтиды; аумақтық желілердің ішінде, сәйкесінше, аймақтың немесе глобалды масштабтарға ие

болатын аймақтық және глобалдық желілерді ерекшелеуге болады; аймақтық желілерді кейде MAN (MetropolitanAreaNetwork) желілері деп атайды, ал аумақтық желілер үшін жалпы ағылшынша атауы − WAN (WideAreaNetwork); ерекше белгіленетін жалғыз глобалды желі Internet (осыда іске асырылған WORLD WIDE WEB (WWW) ақпараттық қызметі қазақ тілінде «дүниежүзілік өрмек» деген мағынаны білдіреді); бұл өз технологиясы бар желілердің желісі. Internet-те интражелі ұғымы бар (Intranet) – Internet аймағында бірлескен желілер;

ә) жергілікті (ЖЕЖ) – шек қойылған аумақты қамтиды (әдетте, станциялардың қашық шектерінде бір-бірінен бірнеше он немесе жүздеген метрден көп емес, сирек 1..2 км); жергілікті желілер LAN (Local Area Network) деп белгілейді.

Желілерді желі жұмыс істейтін кәсіпорындардың масштабына байланысты былай бөледі:

а) бөлімдер (жұмысшы топтардың) – жалпылық жұмысты шешетін кішкене қызметкерлердің тобымен қолданылады. Бөлімдік желінің басты мақсаты – қосымша, деректер сияқты жергілікті қорларды бөлу. Әдетте, қандайда бір желілік технологияның негізінде жасалады, желі астына бөлінбейді.

ә) кампустар – бөлек ғимарат немесе бір кәсіпорын аумағының ішінде бөлімдердің бірнеше желілерін қосады, бірнеше шаршы шақырым аумақты жаба алады. Жергілікті болатын басты желі және желі асты болады, әдетте әр текті компьютерлік жүйелерді қосады;

б) бірлестік (масштабты кәсіпорындар) – бір ғана кәсіпорын немесе мекеме орналасқан бір немесе бірнеше таяу ғимараттар алатын аумақты қамтитын, бір-бірімен байланысқан ЖЕЖ жиынтығы.

Қызмет көрсететін түрлеріне байланысты былай бөледі:

а) деректерді тарату желілері (ДТЖ) – тек кана компьютерлік трафикті беру қызметімен айналысады;

ә) қызметті интеграциялаумен алынған желі – компьютерлік трафикті таратумен қатар, дауыс, бейне тапсыратын сервистерге сүйенеді.

Желі қандай стандартты технология негізінде құрылғанына байланысты былай бөлінеді: Ethernet, Fas Ethernet, Gigabit Ethernet, ArcNet, Token Ring, Token Bus, 100 VG AnyLan:(Integrated Services Digital Network).

Тапсыру ортасына байланысты былай бөлінеді:

- желілік (коаксильді, өрілген будағы, оптикалық-талшықты);

- өткізгішсіз (инфрақызыл диапазонындағы радио-арналар арқылы хабар таратумен).

Қорларға ашық рұқсат етілген деңгей бойынша былай бөлінеді:

- көпшілік (қоғамдық) – бәріне кіруге рұқсат етілген желілер;

- жеке (бірлескен) – желіге ие кәсіпорын қызметкерлеріне қызмет ету немесе пайдаланушыларга байланысты, желі иелері ұсынатын белгілерді толықтыратын пайдаланушылар [6].

В а р и а н т 16

Біррангілі және сервер негізіндегі желілер

Желілік компьютерлер арасындағы байланыс желілік адаптерлер мен байланыс арналары арқылы хабарламаны жеткізу әсерінен болады. Осы хабарламалар арқасында, әдетте, бір компьютер басқа компьютердің жергілікті қорына кіруге қол жеткізуді сұрайды. Мұндай қорларға бағдарламалар, дискте сақталған деректер, әр түрлі шеттегі құрылғылар – принтерлер, модемдер, факс-аппараттары, т.б. кіреді. Әрбір компьютердегі жергілікті қорларды барлық желіні пайдаланушылар арасында бөлу-есептеу желісін құрудың басты мақсаты.

Желідегі компьютерлер арасында функцияларды тарату көзқарасы тұрғысында барлық желілерді екі топқа бөлуге болады:

Біррангілі желілер – тең дәрежелі (желіге қол жеткізу көзқарасынан) компьютерлерден тұратын желі.

Серверлер негізіндегі желілер, онда тек қана желілік функциялармен шұғылданатын белгіленген (dedicated) серверлер бар. Белгіленген сервер жалғыз немесе олар бірнеше болуы мүмкін.

Біррангілі желілер және лайықты бағдарламалық құралдар, ереже бойынша, сан жағынан аз компьютерлерді біріктіру қажет кезінде қолданылады. Осындай желінің әрбір компьютері бір уақытта желінің сервері де, клиенті де бола алады, бірақ қайсыбір компьютер тек қана сервер больп, ал тағы біреуі тек қана клиент болып тағайындалуы әбден мүмкін.

Біррангілі желілердің қасиеті – олардың аса үлкен иілгіштігінде: мұндай жағдайда желі өте белсенді қолданыла алады немесе мүлдем қолданылмауы мүмкін, ол нақты міндетке тәуелді. Компьютерлердің үлкен дербестігінен мұндай желілерде желіні артық жүктеу жағдайы сирек болады (сондай-ақ, компьютерлердің саны әдетте аз болады),

Біррангілі желілерде желілік қорларға қол жеткізуі бойынша әр түрлі құқылы қолданушыларды анықтау рұқсат етіледі, бірақ құқықтарға шек қою жүйесі аса дамымаған. Сонымен қатар, біррангілі желілердің кемшілігі бар: желі жұмысын бақылаудың және хаттамалардың нашар жүйеленуі. Сондай-ақ, кез келген компьютер сервердің істен шығуы жалпы ақпараттың жоғалуына әкеп соғады, яғни барлық осындай компьютерлер мүмкіндігінше сенімді болуы тиіс, бірақ әрқашан олай мүмкін бола бермейді.

Сервер негізіндегі желілер желіге көп пайдаланушылар бірігетін жағдайда ғана қолданылады. Сондықтан желіге мамандандырылған компьютер – сервер қосылады. Ол тек кана желіге қызмет етеді, басқа

міндеттерді орындамайды. Мұндай сервер бөлінген деп аталады. Желілік сауалдарды бөлінген қорларға жылдам өңдеу үшін және файлдар мен каталогтерді қорғауды басқару үшін серверлер арнайы ықшамдалған. Желінің қуаттылығының үлкен мөлшерінде бір сервер жеткіліксіз болуы мүмкін, сол кезде желіге бірнеше серверлерді қосады. Серверлер басқа да міндеттерді атқара алады: желілі баспа, глобалды желіге шығу, басқа жергілікті желімен байланыс, электрондық поштаға қызмет көрсету т.с.с. Сервердің негізінде желіні пайдаланушылардың саны бірнеше мыңдаған бола алады.

Сервер негізінде желіде компьютерлерді айқын, клиенттерге (немесе жұмысшы станцияларына) және серверлерге бөлу бар. Клиенттер серверлер сияқты жұмыс істей алмайды, ал серверлер клиенттер мен автономиялық компьютерлер сияқты жұмыс істей алады. Барлық желілі дискілік қорлар тек қана серверде орналасуы мүмкін, ал клиенттер тек қана серверге сүйене алады, бірақ бір-біріне сүйене алмайды. Бұл олар бір-бірімен хабарласа алмайды деген сөз емес, тек бір клиенттен екіншісіне ақпаратты тарату сервер арқылы мүмкін болады, мысалы, барлық клиенттер қол жеткізе алатын файл арқылы дегенді білдіреді.

Сервер негізіндегі желінің кемшілігіне аз шамадағы компьютерлер саны мен компьютер клиенттердің серверге тәуелділігі, қымбат серверді қолданған кездегі желінің аса қымбат құны жатады.

Бұл жағдайда серверде желілік операциялық жүйе орналастырылады. Бұл желілік ОЖ ерекше операцияларды, желілік айырбасты ұйымдастыруда нәтижелі орындау үшін арнайы ықшамдалған. Ол жұмысшылар стансасында (клиенттерде) жұмыстың желілік тәртібін қолдайтын желілік қабық сияқты да, операциялық жүйе сияқты да орналастыра алады [6].

В а р и а н т 17

Локалды желілердің топологиясы

Компьютерлік желінің топологиясымен (құрастырумен, кескін үйлесімімен, құрылыммен), әдетте, желілік компьютерлердің бір-бірімен салыстырғанда физикалық орналастырылуы және байланыс сызықтарымен қосу тәсілі түсіндірледі.

Әдебиетте желінің топологиясы туралы еске алсақ, әр түрлі деңгейдегі желілік архитектураға қатысты төрт түрлі ұғым:

физикалық топология – яғни, компьютерлердің және кабельдердің салуларының орналасу сызбасы;

қисынды топология – яғни, байланыстардың құрылымы, желі бойынша сигналдарды тарату өзгешелігі;

айырбастауды басқару топологиясы – яғни, жеке компьютерлер арасында желіні басып алу құқығын тапсырудың принципі және жүйелілігі;

ақпараттық топология – яғни, желі арқылы жіберілетін ақпарат ағынының бағыты қалыптасады.

Мүмкін болатын топологиялар кескін үйлесімдерінің жиыны арасында: толық байлаулы және толық емес байлаулы болып бөлінеді.

Толық байлаулы топологиялар ірі желілерде сирек қолданылады, өйткені N торапты байланыс үшін N(N-1)/2 физикалық дуплексті байланыс сызықтары қолданылады.

Екі компьютер арасында деректермен алмасу үшін желінің басқа тораптары арқылы аралық деректерді тарату керек болса, барлық басқа нұсқалар толық емес байлаулы топогияларға негізделеді.

Ұялық топоголия (mesh) толық байланыстың кейбірі мүмкін болатын байланыстарды жою арқылы реттеледі. Ол үлкен көлемді компьютерлерді қосуға рұқсат етеді және ірі желілерге де байланысты.

Жергілікті желілерде желілік топологияның үш негізгі түрін ерекшелейді:

- шина (bus), мұнда барлық компьютерлер параллельді түрде бір байланыс сызығына қосылады және әр компьютерден жіберілген хабарлама бір уақытта басқа қалған компютерлерге беріледі;
- жұлдызша (star), желінің бір орталық элементіне шеттегі компьютерлер қосылады және де олардың әрқайсысы өзінің бөлек байланыс сызығын қолданады;
- сақина (ring), әр компьютер әрқашан тек бір тізбектегі келесі компьютерге ақпаратты тасымалдайды, ал ақпаратты тізбектегі алдыңғы компьютерден алады және де бұл тізбек «сақинамен» тұйықталған.

Тәжірибеде көбінесе негіздік топологиялардың қиыстыруын қолданады: жұлдыз – сақина, жұлдыз – шина.

«Шина» топологиясында (немесе оны «жалпы шина», «bus» деп те атайды) барлық ақпарат жіберілетін орталық абонент болмайды. Шинаға жаңа абоненттерді қосу оңай және әдетте, тіпті желі жұмыс істеп тұрғанда да мүмкін болады. Көп жағдайда шинаны қолдану кезінде қосқыш кабельдер басқа топологиялармен салыстырғанда аз қажеттілік етеді.

«Жұлдызша» – бұл барлық абоненттер қосылатын айқын бөлінген ортасы бар топология. Барлық ақпаратпен айырбас тек қана орталық арқылы жүреді. Орталық абоненттің желілік жабдықтауы шеттегі абоненттердің жабдықтауына қарағанда аса күрделі болуы тиіс. Желіде «жұлдызша» топологиясымен ешқандай қақтығыстар болуы мүмкін емес, өйткені басқару толығымен орталықтандырылған, ештеңеге талас жоқ. «Жұлдызша» топологиясында кез келген кабельдің үзілуі немесе қысқа тұйықталуы тек қана бір компьютермен айырбас жасауды бұзады, ал қалған компьютерлер жұмысын қалыпты түрде жалғастыра алады.

«Жұлдызша» топологиясының маңызды кемшілігі – оның абоненттердің санын қатаң түрде шектейді. Әдетте, орталық абонент 8-16-дан көп емес

шеттегі абоненттерге қызмет көрсете алады. Егер осы шамада жаңа абоненттерді қосу оңай болса, олардың санын арттырғанда қосу мүмкін емес.

Орталығы орталық компьютер болатын жұлдыз белсенді немесе нағыз жұлдыз деп аталады. Сонымен қатар, жұлдызға тек қана сырттай ұқсас белсенді емес жұлдыз деп аталатын топология бар. Тап осы топологиямен берілген желінің орталығында компьютер емес, концентратор немесе репитер атқаратын функцияны орындайтын хаб (hub) орналастырылады. Ол келуші сигналдарды қалпына келтіреді және оларды басқа байланыс сызықтарына жібереді.

Сонымен қатар, белсенді және белсенді емес жұлдыз арасында аралық типті топологияны ерекшелеуге болады. Бұл жағдайда концентратор өзіне түсетін сигналдарды тек қана ретрансляциялап қоймай, сонымен қатар айырбастауды және басқаруды жасап шығарады, бірақ өзі айырбасқа қатыспайды.

Жұлдыздың үлкен артықшылығы барлық қосу нүктелері бір орында жиналғандығы болып табылады. Бұл – желінің жұмысын жеңіл бақылауға көмектеседі, желінің ақауын орталықтан кез келген абонентті сөндіріп тастау тәсілімен таратылуын тоқтатады.

Барлық «жұлдызша» топологияларының кемшілігі ретінде, олардың басқа топологияларға қарағанда кабельді аса көп шығындауын айтуға болады.

«Сақина» – бұл әр компьютердің екі басқа байланыс сызықтарымен қосылған топологиясы: біреуінен ол тек қана ақпарат алады, ал екіншісіне таратады. Әрбір байланыс сызығында тек қана бір хабарлағыш және бір қабылдағыш жұмыс істейді. Бұл сыртқы терминаторларды қолданудан бас тартуға көмектеседі. Сақинаның маңызды ерекшелігі – әр компьютер өзіне келуші сигналды ретрансляциялауында (қалпына келтіреді), яғни репитердің рөлін атқаруында, сондықтан барлық сақинада сигналдың сөнуінің ешқандай маңызы жоқ, тек қана сақинаның көршілес компьютерлер арасындағы сөнуі маңызды болады.

Әдетте, «сақинаға» жаңа абоненттерді қосу мүлде күйзеліссіз өтеді, бірақ барлық желінің жұмысын міндетті түрде тоқтатуды талап етеді. Әдетте, сақиналық топологияны жүктеу тұрақты болады, яғни желі арқылы берілетін ең үлкен ағымды ақпараттармен сенімді жұмыс істеуді қамтамасыз етеді, өйткені онда ереже бойынша қақтығыстар болмайды, сонымен қатар жоқ орталық абонент болмайды.

Сақинадағы сигнал барлық желілік компьютерлер арқылы өткендіктен, біреуінің (немесе оның желілік жабдықтауының) істен шығуы бүтіндей барлық желінің жұмысын бұзады. Дәл осылай сақинаның кез келген кабельдерінің үзілісі немесе қысқа тұйықталуы, барлық желінің жұмысын құртады. Сақина кабельге зақым келтіре алмайды, сондықтан бұл топологияда екі (немесе одан да көп) параллель байланыс сызықтарын салуды, әдетте, алдын ала ескереді, олардың біреуі резервте болады.

Сондай-ақ, сақинаның аса ірі артықшылығы сигналдардың ретрансляциясы әр абонентке бүтіндей барлық жүйенің өлшемін (уақытпен бірнеше он дана километрлердің) үлкейтуге мүмкіндік береді.

Сақинаның кемшілігіне әр желілік компьютерге екі кабельден жалғау керектігін жатқызуға болады [6].

В а р и а н т 18

Компьютерді программамен қамтамасыз ету

Программамен қамтамсыз ету дегеніміз – компьютерде белгілі бір үлгілерді орындайтын, оларды орындауға септігін тигізетін техникалық программалық құжаттамалар жиынын қоса алғандағы программалардың жиынтығын айтуға болады.

Барлық программалық құралдардың жиынтығы мен олардың қажет ететін, компьютерде қолданылатын деректерін, әдетте (Software-програмалық қаматамасыз ету, Soft-жұмсақ, ware – өнім, яғни жұмсақ өнім), аппараттық құралдардың жиынтығы- hardware (hardware- аппараттық құрал, hard-қатты) Software деп жиі атайды.

Кез келген нақты компьютерде аппараттық және программалық құралдардардың нақты жиыны, сондай-ақ олардың ресурстарын қалыптастыратын әр түрлі деректер болады.

Программалық қамтамасыз ету әр түрлі қызмет салаларындағы есептеуіш жүйені пайдалануға бағдарланған әрі ол алға қойылған міндеттерді уақтылы және баламалы шешуді қамтамысыз етуі керек. Бұл программалық етудің компоненттерін әзірлеуге қойылатын талаптарды сақтаудың қажеттілігін тудырады. Бұл талаптардың негізгілеріне келесілер: модульдігі, өсуі мен дамуы, сенімділігі, болжанатыны (предсказуемость), қолайлы болуы, эргономикалығы, икемділігі, тиімділігі, үйлесімділігі жатады.

Пайдаланушыға берілетін мүмкіндіктердің жүзеге асырылуы үшін програмалық жүйе модульдік құрылымға ие болуы керек. Үлкен жүйені шолып талдауға келетіндей етіп жекелеген бөліктерге бөлуді оңайлатады, бірақ жұмыстың өте дәл ұйымдастырылуын талап етеді. Оған қоса, басқа да қалған талаптарды сапалы жүзеге асыру көп жағынан модульдік талабының дұрыс сақталуына байланысты болады.

Әдетте, программадағы деректерді өңдеу бойынша функцияның орындалуы үшін белгілі бір көмекші информация қажет. Іліктес (родственных функции) функциялардың орындалуы үшін әдетте бір емес, бірнеше өзара байланысты программалар әзірленеді.

Информацияны өңдеудің іліктес функцияларының орындалуын қаматамасыз ететін өзара байланысты программалар тобын және осыған қажетті көмекші декректер жиынын программа дестесі немесе программалық жүйе деп аталады.

Жаппай тираждау мақсатында жасалатын программа немесе программа дестесі, программалық өнім деп аталады.

Дестені немесе программаны жұмыс жағдайына келтіру үшін орындау процедурасын инсталяциялау (install – орнату) қажет, яғни программалар деректері сол сәттегі құжаттамаларда баяндалған ереже бойынша жұмыс істей алатын жағдайларға ойластыру үшін (для их развертывания) дестенің тек заңды иесі ғана өзінің орнатқанын орындауға мүмкіндік беретін арнайы кодты білуі қажет.

Іс жүзінде программалық өнімдердің пайдаланылуына қарай олардың кемшіліктері, пайдаланылмаған мүмкіндіктері, содан соң, әзірлеу кезінде байқаусызда кеткен қателіктер анықталады. Өндіруші фирмалар өзіне келіп түскен мұндай информацияны ескере отырып, шамалары келгенше елге танылып үлгерген өнімге тиісті өзгертулер енгізеді. Бұл өзгертулер қолданыстағы программалық өнімнің модификациялары мен нұсқаларының пайда болуы түрінде көрінеді [6].

В а р и а н т 19

Понятие о системе программирования

Любой компилятор не существует сам по себе, а решает свои задачи в рамках всего системного программного обеспечения. Основная цель компиляторов – обеспечивать разработку новых прикладных и системных программ с помощью языков высокого уровня. Всякая программа, как системная, так и прикладная, проходит этапы жизненного цикла, начиная от проектирования и вплоть до внедрения и сопровождения. При этом компиляторы – это средства, служащие для создания программного обеспечения на этапах кодирования, тестирования и отладки. Однако сам по себе компилятор не решает полностью всех задач, связанных с разработкой новой программы. И хотя компилятор является главной составляющей средств разработки, одного только лишь компилятора недостаточно для того, чтобы обеспечить прохождение программой всех указанных этапов жизненного цикла. Поэтому компиляторы – это программное обеспечение, которое функционирует в тесном взаимодействии с другими техническими средствами, применяемыми для разработки программного обеспечения.

Основные технические средства, используемые в комплексе с компиляторами, включают в себя следующие программные модули:

- текстовые редакторы, служащие для создания текстов исходных программ;
- компоновщики, позволяющие объединять несколько объектных модулей, порождаемых компилятором, в единое целое;
- библиотеки прикладных программ , содержащие в себе наиболее часто используемые функции и подпрограммы в виде готовых объектных модулей;

- загрузчики, обеспечивающие подготовку готовой программы к выполнению;

- отладчики, выполняющие программу в заданном режиме с целью поиска, обнаружения и локализации ошибок;

- другие программные средства, служащие для разработки программ и их компонентов.

Все эти средства разработки функционируют не отдельно, каждое само по себе, а в тесном взаимодействии друг с другом. Данные, полученные одним модулем, поступают на вход другого и наоборот. В современных средствах разработки интеграция модулей столь высока, что пользователь часто даже и не представляет, что он работает с несколькими программными средствами – для него вся система разработки представляет собой единое целое. Весь этот комплекс программно-технических средств составляет новое понятие, которое здесь названо «системой программирования» [9].

В а р и а н т 20

Появление интегрированных сред разработки

Командная строка – эффективное, но не всегда удобное средство управления компиляцией (хотя среди программистов под ОС типа UNIX обязательно найдутся желающие поспорить с этим). Фактически для каждого сложного проекта разработчикам приходилось писать еще одну дополнительную программу, описывающую на языке Makefile, как следует строить (собирать, компилировать) этот проект. А программирование всегда есть потенциальный источник ошибок. Поэтому развитие систем программирования на этом не завершилось.

Следующим шагом в развитии систем программирования стало появление так называемой «интегрированной среды разработки». Интегрированная среда объединила в себе возможности текстовых редакторов и командный язык компиляции. Пользователь (разработчик исходной программы) теперь не должен был отвечать за передачу данных с выхода одного средства разработки на вход другого, от него также не требовалось описывать этот процесс с помощью языка Makefile. Теперь ему было достаточно только указать в удобной интерфейсной форме состав необходимых для создания программы исходных модулей и библиотек. Команды, библиотеки и «ключи», необходимые компилятору и другим средствам разработки, также задавались в виде интерфейсных форм настройки.

После этого интегрированная среда разработки сама автоматически готовила всю необходимую последовательность команд Makefile, выполняла их, получала результат и сообщала о возникших ошибках при их наличии. Но главным преимуществом интегрированной среды разработки было то, что все

действия пользователь мог выполнять непосредственно в окне редактирования исходного текста программы. Он мог исправлять текст исходных модулей, получать информацию об ошибках непосредственно в исходном тексте программы, не прерывая работу с интегрированной средой. Кроме удобства работы, объединение текстовых редакторов, компиляторов и компоновщиков в единую среду дало системам программирования целый ряд достоинств, о которых будет сказано далее.

Создание интегрированных сред разработки можно назвать вторым шагом в развитии систем программирования. Оно стало возможным благодаря бурному развитию персональных компьютеров и появлению развитых средств интерфейса пользователя (сначала текстовых, а потом и графических). Пожалуй, первой удачной средой такого рода можно признать интегрированную среду программирования Turbo Pascal на основе языка Pascal производства фирмы Borland. Ее популярность на рынке определила тот факт, что со временем все разработчики компиляторов обратились к созданию интегрированных средств разработки для своих продуктов.

Развитие интегрированных сред снизило требования к профессиональным навыкам разработчиков исходных программ. Теперь в простейшем случае от разработчика требовалось только знание исходного языка. Разработчик программы, работающий в интегрированной среде, мог даже не иметь представления о структуре системы программирования, которой он пользуется.

Дальнейшее развитие средств разработки также тесно связано с повсеместным распространением развитых средств графического интерфейса пользователя (GUI – Graphical User Interface). Такой интерфейс стал неотъемлемой составной частью многих современных ОС и так называемых графических оболочек. Со временем он стал стандартом де факто во всех современных прикладных программах. Это не могло не сказаться на требованиях, предъявляемых к средствам разработки программного обеспечения. В их состав были сначала включены соответствующие библиотеки, обеспечивающие поддержку GUI и взаимодействие с функциями ционных систем. А затем для работы с ними потребовались дополнительные средства, обеспечивающие разработку внешнего вида интерфейсных модулей. Такая работа была характерна уже более для дизайнера, чем для программиста [9].

В а р и а н т 21

Ақпаратты қорғау және бақылау

Ақпараттық жүйенің тиімділігі ондағы өңделінетін ақпараттың қауіпсіздігімен (қорғалғандығымен) байланысты. Себебі кез келген мемлекеттік органның, кәсіпорынның, мекеменің, жеке адамның қызметінде жасыратын құпия мәліметтері немесе деректері болады. Олардың жария

болуы мемлекет қауіпсіздігіне, кәсіпорындар мен мекемелердің қаржы-экономикалық жағдайына, жеке адамдардың жұмысына кері әсерін тигізетіні сөзсіз. Әрбір мемлекетте ақпаратты қорғау туралы заңдары қабылданған. Мысалы, Қазақстан Республикасының ақпараттандыру туралы заңының жаңа редакциясы 2007 жылдың қаңтар айының 11- жұлдызында қабылданған. Ақпараттың қауіпсіздігі дегеніміз оған амалдар орындалғанда немесе оны қабылдағанда, өңдегенде, сақтағанда, беріліс кезінде және пайдаланғанда әртүрлі қауіп-қатерден қорғалғандығын айтамыз.

Есептеу техникасында қауіпсіздік туралы сөз болғанда алдымен келесі мәселелер қарастырылады:

- компьютер мен оның құрылғыларының жұмысының сенімділігі;
- құнды ақпараттың бұзылмай сақталынуы;
- ақпаратқа рұқсатсыз өзгерістер енгізілмеуі;
- электрондық байланыс кезінде құпияның сақталынуы.

Қауіп-қатердің пайда болу себептері адамдардың әрекеттеріне, автоматтандырылған жүйелерде қолданылатын аппараттық және программалық құралдарға, сонымен қатар сыртқы факторларға байланысты болуы мүмкін. Аталынған себептерден туатын қауіп-қатерді екі сыныпқа бөлуге болады:

- қасақана емес;
- қасақана.

Әдейі жасалатын қатер көбінесе табиғи апаттан және зіл-заладан, техникалық құралдардың жұмысындағы ақаулық пен олардың істен шығып қалуынан, алгоритмдер мен программалардың кешендерінде жіберілген қателерден, сонымен бірге пайдаланушылар мен қызмет етуші адамдардың қатесінен болуы мүмкін. Табиғи апат, өрт, су тасқыны, жер сілкінісі және т.б. жатады. Осылардың себептерінің нәтижесінде ақпарат сақтаушы құрылғылардың бұзылуына немесе жойылуына алып келеді. Техникалық құралдардың жұмысындағы ақаулықтар немесе кейде олардың мүлдем жұмыс істемей қалуы ЭЕМ-нің аппараттық құралдарының бұзылуы мен жаңылуынан, электр тогының немесе кәбелдің үзілуінен және т.б. себептерден болуы мүмкін. Осындай себептер программалар мен деректердің жойылуына, құрылғылардың жұмыс істеу алгоритмдерінің бұзылуына, сонымен бірге ақпараттың құпия болмауына (конфиденциалдығының бұзылғандығына) келтіреді. Алгоритмдер мен программаларда жіберілген қателер де осыған келтіруі мүмкін. Ал қауіпсіздікке ең көп нұқсан келтіретін себептер – пайдаланушылар мен қызмет етуші адамдардың жіберетін қателері.

Әдейі жасалатын қатерлердің алдын алу үшін қазіргі кездегі аппараттық және программалық құралдар мен автоматтандырылған жүйелерді тиімді пайдалану, ақпараттың көшірмелерін жасау сияқты іс-шаралар қолданылады [7].

В а р и а н т 22

Информатика

Информатика – бұл деректерді есептеу техникасы құралдарымен құру, сақтау, қайталау, өңдеу және беру тәсілдерін және осы құралдардың қызмет ету принциптері мен оларды басқару әдістерін жүйелейтін кешендік техникалық ғылым. «Информатика» термині түбірін Informatigue деген француз сөзінен алады және ол екі: ақпарат, автоматика деген сөзден тұрады. Бұл термин алғаш рет Францияда XX ғасырдың 60 жылдары, есептеу техникасы кеңінен етек жайып дами бастағанда енгізілген. Сол кезде ағылшын тілді елдерде «Computer Science» термині қолданысқа ене бастады. Бұл термин есептеу техникасын пайдалануға негізделген ақпаратты түрлендіретін ғылымды белгілеу үшін қолданыла бастады.

Информатиканың ғылым ретіндегі заттық мағынасына келесілер жатады:

1) есептеу техникасы құралдарының аппараттық қамтамасы;

2) есептеу техникасы құралдарының программалық қамтамасы;

3) аппараттық және программалық қамтамалардың өзара әрекеттесу құралдары;

4) аппараттық және программалық құралдардың адаммен өзара әрекеттесу құралдары.

Өзара әрекеттесу құралдары информатикада интерфейс деп аталады. Сондықтан аппараттық және программалық қамтамалардың өзара әрекеттесу құралдарын – программалық-ақпараттық интерфейс деп, ал аппараттық және программалық құралдардың адаммен өзара әрекеттесу құралдарын пайдаланушы интерфейсі деп атайды.

Информатиканың ғылым ретіндегі негізгі міндеті – есептеу техникасының аппараттық және программалық құралдарымен жұмыс істейтін әдістері мен тәсілдерін жүйелеу. Жүйелеу мақсаты деректермен жұмыс істеу кезеңдерін автоматтандыратын тиімді және алдыңғы қатардағы технологияларды ерекшелеу, ендіру және дамытумен қатар, жаңа технологиялық зерттеулерді әдістемелік түрде қамтамасыз ету [5].

В а р и а н т 23

Программалау тілдері

Программаларды ЭЕМ-де тікелей орындауға арналған алгоритмдерді жазу тәсілі, яғни алгоритмдерді ЭЕМ-ге түсінікті мәтін ретінде жазуға арналған қарапайым жасанды тіл программалау тілі деп аталады. Әрбір ЭЕМ-нің өзінің машиналық программалау тілі болады, оны командалар тілі немесе кодтар (арнайы таңбалау) тілі дейді.

Программа жазуды жеңілдету үшін математикалық формулаларды кеңінен қолданатын, ағылшын тілінің негізінде жасалған алгоритмдік тілдер Бейсик, Паскаль, Фортран, Си, т.б. кеңінен қолданылады.

1. Кобол – өткен ғасырдың 60 жылдары басында жасалған, экономикалық есептер шығаруда қолданылатын, машиналық тілге аударылатын тіл. Коболда сыртқы мәлімет сақтау құрылғыларында сақталатын көлемді мәліметтерді өңдеудің қуатты мүмкіндіктері жүзеге асырылды.

2. Паскаль – 70-ші жылдар аяғында швейцар математигі Никлаус Вирттің программалауды үйрету мақсатында қолдану үшін арнайы жасаған тілі. Ол адамның алгоритмдік ойлау қабілетін қалыптастыратын, көпшілікке түсінікті қысқа программа жазуға қолайлы, алгоритмдеудің негізгі тәсілдерін көрсетуге және де ірі жобаларды да жүзеге асыра алатын тіл болды.

3. Бейсик – бұл да 60-шы жылдары программалауды үйренушілерге арналып жасалған қарапайым тіл болды. Бұл көпшілікке кең тараған, игеруге жеңіл программалау тілдерінің бірі болды.

4. Си – 70-ші жылдары шықты, ол алдымен көпшілікке арналып жасалмаған тіл еді. Си нақты процессорға тәуелді емес, қысқа, әрі тиімді программалар жасайтын ассемблер орнына пайдалану үшін жоспарланған тіл болатын. Си тілінің көптеген қасиеттері Паскаль тіліне ұқсас болғанымен, оның компьютер жадымен тікелей жұмыс істей алатын мүмкіндіктері бар еді. Осы тілде көптеген қолданбалы және жүйелік программалармен қатар Unix операциялық жүйесі де жазылып шыққан болатын.

5. Си ++ - обьектіге бағытталған Си тілінің кеңейтілген түрі, оны 1980 жылы Бьярн Страуструп жасап шығарды.

6. Java тілін 90-шы жылдар басында Си ++ тілін негізге ала отырып, Sun компаниясы шығарды. Ол Си ++ тілінен төменгі деңгейдегі мүмкіндіктерді алып тастап, қолданбалы программалар жасауды оңайлату мақсатында жасалды [5].

В а р и а н т 24

Қосалқы программалар

Компьютерлерде шығарылатын есептердің күрделенуіне байланысты программалардың көлемдері өсіп, оларды жазу, оқу, түзету істері күннен-күнге қиындап келеді. Белгілі бір өндірістік мәселенің, мезгіл-мезгіл қайталанып отыратын есептеулердің программалары ұзақ уақыт пайдаланылады, олар күнделікті өмір талабына сәйкес өзгертіліп, түзетіліп отыруы тиіс. Осыларға байланысты программаларды құрастыруды, түсінуді, өзгертуді жеңілдететін тәсілдер жасалған, олар құрылымды (структуралық) программалау жолдары деп аталады. Программаларды адамның түсіну мен қабылдауын ыңғайлы түрде жүргізуге бағытталған тәсілдер жиынын құрылымды программалау деп атайды. Әрбір жасалған программа бөліктері бір-бірімен тығыз логикалық байланыста болуы тиіс.

Белгілі бір ат қойылып, жеке программа түрінде бөлек жазылған, қажет кезінде оны қайталап пайдаланып отыруға болатын негізгі программаның арнайы бөлігін қосалқы программа (подпрограмма) деп атайды. Қосалқы программаларда бірсыпыра операциялардың біріге отырып толық орындалуын

қадағалап, программаның негізгі бөлігінде оның тек атын өзгерту арқылы бір рет орындаумен шектелуге болады.

Қосалқы программаны пайдалану:

- негізгі программаның көлемін кішірейтеді;

негізгі программада пайдаланылған айнымалыларды қосалқы программада да пайдалануға болады;

- қосалқы программаға берілген компьютердің жедел жады көлемін ол орындамай тұрғанда, бос ұялар ретінде басқа мақсаттарға пайдалануға болады; қосалқы программаны пайдалану тәсілдері құрылымдық программалау талаптарына сай келеді.

Құрамында қосалқы программалар тәрізді құрылымды программалау жабдығы бар Турбо Паскаль тілі процедураға бағытталған тіл деп аталып жүр [5].

В а р и а н т 25

Паскаль тілі

Информатикада студенттердің алгоритмдік әдіспен ойлау қабілеті басты рөл атқарады. Ол жалпы дамуға және жаңа ақпараттық технологиялармен таныстырады. Адамның ЭЕМ-мен байланысу үшін қарым-қатынастық тіл керек, сондықтан арнайы бағдарламалау тілдері құрылып және кең қолданылады.

Паскаль тілі XX ғасырдың 70-нші жылдарында швейцар профессоры Никлаус Виртпен бағдарламалауды жүйелік оқыту құралы ретінде шығарды. Бұл тіл құрылымдық бағдарламалау элементтерінен (тізбекті, тармақты, циклдық құрылымдар) және мәліметтер құрылымынан (массивтер, жазбалар, файлдар және т.б.) тұрады. 1983 жылы француз математигі Филипп Кан Турбо Паскаль деп аталатын тез жұмыс істейтін, жинақы компилятор шығарды. 1992 жылы Borland International фирмасы интерфейсі жақсартылған және одан да тез компиляторлы Турбо Паскаль 7.0 тілінің кезекті түрін шығарды.

Паскальдің бағдарлама құру негіздері Ада, Модула-2, Си және т.б. тілдерде дамыды. Турбо Паскаль логикалық құрылымын сақтай отырып, үлкен бағдарлама құруға мүмкіндік береді. Оған қатар Паскаль қарапайым болғанмен инженерлік есептерді шығаруда тиімді құрал.

Турбо Паскальдің ерекшелігі – ол бағдарлама құрылымына қатысты. Турбо Паскаль тілінде бағдарлама символдар тізбегі ретінде жүреді. Ол символдар: латын әріптері, араб сандары, амалдар және тыныс белгілері. Бастапқы берілгендерді, есеп шешімін, аралық берілгендерді белгілеу үшін айнымалылар қолданылады, олардың аттары тек a, b, X,Y және т.б. әріптерден ғана емес және де символдар тізбегінен тұруы мүмкін (мысалы: x1, time, alfa және т.б., бірақ олар тек әріптен басталуы керек). Кілттік сөздер дегеніміз – осы тілде операторларды жазуда қолданылатын аттар. Айнымалыларға берілетін аттар кілттік сөздермен сәйкес болмауы керек [5].

Список литературы

1 Азимов Э.Г., Щукин А.Н. Новый словарь методических терминов и понятий (теория и практика обучения языкам). – М.: Издательство ИКАР, 2009.

2 Антоненко М.В., Пономарев В.В., Куприянова А.В. «Толстый» самоучитель работы на компьютере. – СПб.: Наука и Техника, 2007.

3 Единая электронная терминологическая база Республики Казахстан. Сайт www.termincom.kz. (дата обращения: 05.10.2016).

4 Казахско-русский, русско-казахский терминологический словарь / под общей ред. М.Б.Касымбекова, научный руководитель проекта А.К.Кусаинов. – Алматы: Издательская корпорация «КАЗакпарат», 2014. URL: https://ru.termincom.kz/terminder/koldanystagy-salayk-terminder-turaly/informatika/ (дата обращения: 05.10.2016).

5 Кәсіби қазақ тілі 5В060200 – Информатика мамандығының студенттеріне арналған әдістемелік нұсқаулықтар / К.Т. Төлеубаева. – Алматы: АЭжБУ, 2014. URL: http://lib.aipet.kz/aies/facultet/eef/kaf_rkj/1/umm/ki_47.pdf (дата обращения: 01.09.2016).

6 Кәсіби қазақ тілі. 5В070400 – Есептеу техникасы және бағдарламалық қамтамассыз ету мамандығының студенттеріне арналған әдістемелік нұсқаулықтар / Л.А. Асылханова. – Алматы: АЭжБУ, 2014. URL: http://lib.aipet.kz/aies/facultet/eef/kaf_rkj/1/umm/ki_51.pdf (дата обращения: 01.09.2016).

7 Қазақ тілі-1 пәні. 5В100200 – «Ақпараттық қауіпсіздік жүйелері» бағыты студенттеріне арналған әдістемелік нұсқаулар мен өздік тапсырмалар топтамасы / Г.С. Әлмұхаметова, Э.Т. Ахметова. – Алматы: АЭжБУ, 2012. URL: http://lib.aipet.kz/aies/facultet/eef/kaf_rkj/1/umm/ki_24.htm (дата обращения: 01.09.2016).

8 Миллер Л., Политова Л. Политехнический русский: Учебник. – СПб.: Питер, 2013.

9 Молчанов А.Ю. Системное программное обеспечение: Учебник для вузов. – 3-е изд. – СПб.: Питер, 2010.

10 Ожиганов А.А. Криптография: Учебное пособие. – СПб.: Университет ИТМО, 2016.

11 Острейковский В.А. Информатика: Учебное пособие. – М.: Высшая школа, 2001.

12 Патентная база Республики Казахстан. Сайт www.kzpatents.com. (дата обращения: 01.09.2016).

13 Смирнова Ю.Г. Профессиональный русский язык: Учебное пособие. – Edinburgh: Lennex Corp. − М.: Нобель Пресс, 2013.

14 СТ НАО 56023-1910-04-2014 Учебно-методические и учебные работы. Общие требования к построению, изложению, оформлению и содержанию учебно-методических и учебных работ. – Алматы: АУЭС, 2014.

15 Термины, утвержденные государственной терминологической комиссией. URL: https://ru.termincom.kz/terminder/bekitilgen-terminder-turaly/ (дата обращения: 01.10.2016).

16 Толстяков Р.Р., Забавникова Т.Ю., Попова Т.В. Информатика: Учебное пособие. – Тамбов: ТГТУ, 2011.

17 Якунина М.В. Основы информационных систем и базы данных: Учебное пособие. – Тюмень: Издательство Тюменского государственного университета, 2013.

Содержание

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
«Алматинский университет энергетики и связи»

УТВЕРЖДАЮ_____
Проректор по учебно- методической работе
_____ С. В. Коньшин
“_____” _____2016 г.

ПРОФЕССИОНАЛЬНЫЙ РУССКИЙ ЯЗЫК

Методические указания и варианты семестровых заданий для студентов специальности 5В060200 – Информатика

СОГЛАСОВАНО
Вр.и.о. нач. УМО
_____Р.Р. Мухамеджанова
“____”_____2016 г.

Председатель по ОУМК
“____” _____ 2016 г.
_____ Б.К. Курпенов

Редактор
_____ Л.Т.Сластихина
“_____” _____2016 г.

Специалист
по стандартизации
_____ Н. К. Молдабекова
“_____” _____2016 г.

Рассмотрено и одобрено на заседании кафедры КиРЯ

Протокол № ____
«_____» _____ 2016 г.
Зав.кафедрой:
Р.К.Букейханова _____

Составитель:
_____ Ю.Г.Смирнова

Зав. кафедрой ИС
_____ Ш.И.Имангалиев
“_____” _____2016 г.

Алматы, 2016

Дополнительный план 2016 г., позиция 26

Юлия Георгиевна Смирнова

ПРОФЕССИОНАЛЬНЫЙ РУССКИЙ ЯЗЫК

Методические указания и варианты семестровых заданий для студентов специальности 5В060200 – Информатика

Редактор Л.Т. Сластихина
Специалист по стандартизации Н. К. Молдабекова

Подписано в печать _ _ _
Тираж 20 экз.
Объем 2,5 уч.-изд. л.

Формат 60х84 1/16
Бумага типографская №1
Заказ _____. Цена 1250 тенге.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, Байтурсынова,126