Вопросы к экзамену:

1)Предмет информатики.
2)Информация и сообщения.
3)Интерпретация сообщений.
4)Знаки и символы.
5)Кодирование.
6)Системы счисления.
7)Обработка сообщений.
8)Обработка информации.
9)Автоматизация обработки информации.
10)Конструктивное описание процесса обработки дискретных сообщений.
11)Свойства алгоритмов.
12)Сложность алгоритмов.
13)Интерпретация дискретных сообщений.
14)Необходимость формального определения алгоритма.
15)Машина Тьюринга.
16)Нормальные алгоритмы Маркова.
17)Диаграммы машин Тьюринга.
18)Моделирование машин Тьюринга.
19)Эквивалентность программ и диаграмм.
20)Эквивалентность диаграмм и программ.
21)I теорема Шеннона. Доказательство*.
22)II теорема Шеннона. Доказательство*.
23)Вычислимые функции.
24)Нормированные вычисления. Теорема о нормированной вычислимости. Доказательство*.
25)Теорема о композиции.
26)Теорема о ветвлении.
27)Теорема о цикле.
28)Обобщенная теорема о цикле.
29)Схема машин Тьюринга. Нисходящаяя разработка.
30)Теорема Бойма-Джакопини-Миллса.
31)Универсальная машина Тьюринга. Мостроение*.
32)Линейная запись схем машин Тьюринга.
33)Критика модели вычислений Тьюринга.
34)Модель фон Неймана.
35)Построение процессора фон Неймана.
36)Машина фон Неймана.
37)Структура программ для машины фон Неймана.
38)Нотация программ Э. Дейстры. Обобщенная инструкция присваивания и композиции.
39)Обобщенная инструкция ветвления.
40)Обобщенная инструкция цикла.
41)Понятия типа данных.
42)Тип логический.
43)Тип целый.
45)Тип вещественный.
46)Согласование типов.
47)Небазовые типы данных (диапазон, перечисление, множество).
48)Понятия о структурном типе данных.
49)Тип массив.
50)Понятия о файлах.
51)Понятия о файлах.
52)Блочная структура программ. Локальные и глобальные переменные.
53)Процедуры и функции.
54)Способы передачи параметров.
55)Критика языков программирования Паскаль и Си.
56)Критика алгоритмической модели фон Неймана.

Ответы:

Глава 1:

1) Предметом информатики как научной дисциплины являются структура, свойства и закономерности информации, процессы ее сбора, переработки, хранения, поиска, распространения (передачи) и использования.
   Предмет информатики как науки составляют:
   * аппаратное обеспечение средств вычислительной техники;
   * программное обеспечение средств вычислительной техники;
   * средства взаимодействия аппаратного и программного обеспечения;
   * средства взаимодействия человека с аппаратными и программными средствами.

2) Информация и сообщение – основные (неопределяемые) понятия информатики. Информация передается посредством сообщения и наоборот, сообщение – то, что несет информацию. Информация может существовать только в форме некоторого сообщения. Соответствие между информацией и несущим ее сообщением не является взаимно-однозначным: 1)Одна и та же информация может передаваться с помощью различных сообщений; 2) одно и то же сообщение может передавать различную информацию.

3) Интерпретация сообщений: информация i, которая передается сообщением n, устанавливается с помощью правила интерпретации, которое представляет собой отображение рассматриваемого множества сообщений B в множество сведений. (Сведения – множество информации). $\varphi = N \to I$
   Есть конечные языки и бесконечные языки. Пример конечного языка – светофор. Зеленый, желтый и красный. Интерпретация этих сообщений как «Едь», «внимание» и «Стой» соответственно. Это φ конечного языка. Пример бесконечного языка – наш родной язык. Бесконечный он только теоретически, слов в этом языке «бесконечно» много, он постоянно расширяется и всегда можно придумать новые слова с новыми значениями.

4) Письменные языковые сообщения представляют из себя последовательности знаков. Мы будем различать атомарные и составные знаки. Атомарным знаком или буквой называется «неделимый» символ 1-ого уровня. Пример: буквы (А, Б, В….). Они складываются в слова второго уровня (слова). Знаки второго уровня в знаки третьего (предложения) и т.д. (абзацы и пр.). Все знаки большего уровня называются составными знаками. Есть специальный знак, чтобы разделять одни составные знаки от других. Для слов это пробел, для предложений – точка. С каждым знаком связывается его смысл или семантика. Знак вместе с сопоставленной ему семантикой называется символом. (А – буква, пи – символ).

5) Кодом называется правило, описывающее отображение $C = A \to A`$ алфавита (набора знаков) A на другой набор знаков A`. Например, отображение восьмеричного числа $0_8 \to 000_2, 7_8 \to 111_2$. При кодировании сообщения n получается новое сообщение C(n), которое содержит ту же информацию, что и n, но изменяется правило интерпретации сообщений. (Например – кодирование по Цезарю). Кодирование, при котором каждый образ является отдельным знаком, называется шифрованием. В симметричных системах шифрования операции кодирования и декодирования являются взаимно обратными функциями. Т.е. зная способ шифрования, действуя наоборот, расшифровать сообщения. Ассиметричное кодирование, когда один ключ используется для кодирования, второй для декодирования. Одностороннее шифрование, используемое для хранения паролей, не является кодированием. Для каждого шифрованного пароля получается множество вариантов дешифрованных.

6) Система счисления – это способ числовой интерпретации цифровых сообщений.
Непозиционные:
а) Натуральная система счисления. Число принадлежит целым от единицы до бесконечности. Пример – пересчет предметов. II яблока.
б) Кардинальная система счисления. Для нуля отдельная палочка. Недостатки при мат. Операциях, (после сложения надо убрать одну палочку)
Проблемы: для числа 1000 потребуется половина экрана терминала.
Позиционные:
$A_n = a_k * n^{k-1} + ... + a_2 * n^1 + a_1 * n^0$. Для отрицательных чисел вся эта сумма домножается на -1. Пример (десятичная) $349_{10} = 3 * 10^2 + 4 * 10^1 + 9 * 10^0$. Перевод из двоичной в десятичную. $101_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0$. Как мы видим, позиция влияет на значение числа. Поэтому эта система счисления называется позиционной.

7) Обработка сообщений. Примеры: кодирование, перевод с одного языка на другой, перепечатывание, чтение вслух и пр. Т.е. обработка сообщений состоит в выделении в исходном сообщений знаков некоторого уровня, составляющих это сообщение, и замене каждого выделенного знака другим знаком. При чтении вслух каждый знак заменяется звуков, фонемой. Следовательно любая обработка сообщений может рассматриваться как кодирование в широком смысле.

8) Во всех случаях, когда правило обработки σ является отображением, правило обработки сообщений v называется сохраняющим информацию. Если v сохраняет информацию, то диаграмма коммутативна: $\nu^{\circ}\varphi` = \varphi^{\circ}\sigma$. Отображение σ называется правилом обработки информации. Если σ обратимое отображение, т.е. информация при обработке не теряется, то соответствующая обработка называется дешифровкой. Пусть v – обратимая перешифровка. Тогда обработка обратимой перешифровкой называется перекодировкой. Иначе она называются сжимающей. Т.е. в результате необратимой перешифровки сообщений их кол-во уменьшается, а информация может либо сохраняться, либо теряться.

$$\begin{array}{ccc} N & \xrightarrow{\varphi} & I \\ \nu \downarrow & & \Downarrow \sigma \\ N' & \xrightarrow{\varphi'} & I' \end{array}$$

(рис. 1).

$$\begin{array}{ccccc} & D & \xleftarrow{C} & N & \xleftarrow{\psi} & I \\ P \downarrow & & & \downarrow \nu & & \downarrow \sigma \\ & D' & \xrightarrow{Q} & N' & \xrightarrow{\varphi'} & I' \end{array}$$

(рис. 2).

9) Автоматизация обработки информации.
Чтобы выполнить автоматическую обработку информации, нужно располагать тремя физ. Представлениями
1)D – множество исходных данных
2)$D`$ - Множество результирующих данных
3)Обработка из D в $D`$ будет P и оно будет выполняться на физ. Устройстве.
Таким образом, для того, чтобы выполнить автоматическую обработку сообщений, необходимо автоматизировать выполнение трех отображений:
1)Отображения C, которое позволит представить сообщение N в множестве данных D
2)Отображения P, которое переводит элемент D в $D`$
3)Отображения декодирования Q, которое позволяет интерпретировать результат в элемент N ` (рис. 2)

10) Конструктивное описание процесса обработки дискретных сообщений. Обработка дискретных сообщений состоит в применении отображения v или в последовательном применении отображений C P Q. Чтобы отображение P могло служить основой для автоматизации обработки, оно должно задавать некоторый способ построения сообщения,

исходя из сообщения D. Для этого нужно представить P в виде последовательности элементарных шагов обработки или тактов, каждый из которых достаточно прост для выполнения физ. Объектом., такие шаги называются операциями. Например – переписывание слова: Переписываем последовательно каждую букву. Подобные алгоритмы обладают следующими свойствами:

11) Свойства алгоритмов.
1)Массовость, т.е. потенциальная бесконечность исходных сообщений для обработки
2)Детерминированность – заключается в последовательном применении шагов, однозначно определяемых результатами предыдущего шага
3)Элементарность каждого шага
4)Результативность
5)Сложность алгоритма определяется кол-вом простых операций, которые нужно совершить для обработки сообщения. От него зависит время выполнения алгоритма.

12) Сложность алгоритмов. Сложность алгоритма определяет, как долго придется ждать, пока будет получен результат его работы. Всего выделяют 7 классов сложности:
1)O(1) – алгоритмы с постоянным временем выполнения, например доступ к элементу массива. При увеличении размера задачи вдвое время выполнения не меняется
2)O(log n) – Алгоритмы с логарифмическим временем выполнения, например, бинарный поиск. Время выполнения удваивается при увеличении размера задачи в n раз
O(n) – Алгоритмы с линейным временем выполнения, например, последовательный поиск или схема Горнера. При увеличении размера задачи вдвое, время выполнения также удваивается.
O(n*logn) - Алгоритмы с линеарифмическим временем выполнения, например, быстрая сортировка. При увеличении размера вдвое, время выполнения увеличивается немного больше, чем вдвое.
$O(n^2)$ - Алгоритмы с квадратичным временем выполнения, например, простые алгоритмы сортировки. При увеличении размера задачи вдвое, время выполнения увеличивается в 4 раза.
O(n^3) – Алгоритмы с кубическим временем выполнения, например, умножение матриц. Время выполнения увеличивается в 8 раз при увеличении размера задачи в 2 раза.
O(2^n) - Алгоритмы с экспоненциальным временем выполнения, например, задача о составлении расписания. Время выполнения увеличивается в 2^n при увеличении размера задачи в 2 раза.

13) Интерпретация дискретных сообщений. Чтобы информация в сообщениях не была воспринята по разному разными субъектами, мы должны зафиксировать представление о мире в виде его модели. У нас должно быть формальное и строгое изложение данного материала. Как правило, это математические модели. Моделью называется некоторое множество M с заданным на нем наоборот отношений. Иначе говоря, это упорядоченная пара объектов. Любая задача обработки информации, которую мы хотим автоматизировать, должна быть поставлена на соответствующей модели. Прежде чем разрабатывать алгоритм решения задачи обработки, необходимо формализовать задачу. Т.е. четко описать модель, на которой ставится задача и попытаться сформулировать теорию для этой модели.

Глава 2:

14) Основной недостаток неформального определения алгоритма является его расплывчатость, т.к. мы не знаем, что значит «Понимать и выполнять действия одинаково» и что значит «всем понятные и легко выполнимые действия». Поэтому нам нужно формализовать алгоритмы и его понятие. Формализация алгоритма реализуется с помощью построения

алгоритмических моделей. Можно выделить три основных типа универсальных алгоритмических моделей: рекурсивные функции (понятия алгоритма связывается с вычислениями и числовыми функциями), машины Тьюринга (алгоритм представляется как описание процесса работы некоторой машины, способной выполнять лишь небольшое число весьма простых операций), нормальные алгоритмы Маркова (алгоритмы описываются как преобразования слов в произвольных алфавитах). Этими моделями мы формализуем понятие алгоритма и у нас остается гораздо меньше вариантов интерпретации одной модели.

15) Машина Тьюринга. Машина Тьюринга состоит из ограниченной с одного конца бесконечной ленты, разделенной на ячейки, и комбинированной читающей и пишущей головки, которая может перемещаться вдоль ленты от ячейки к ячейке. В каждой ячейке ленты может быть записан один знак рабочего алфавита A, либо пробел. Головка МТ в каждый момент времени располагается над одной из ячеек ленты, называемой рабочей ячейкой, и воспринимает знак, записанный в этой ячейке, при этом головка находится в одном из множества состояний, среди которых есть одно начальное состояние и в зависимости от состояния головки и от буквы, записанной в ячейке, выполняет одну из команд, составляющих ее программу. Эта головка может двигаться влево на одну ячейку, вправо на одну ячейку и производить запись буквы или пробела.
Перед началом работы МТ на ее ленту записывается так, что в каждую ячейку ленты записывается либо одна буква сообщения, либо пробел. Все пустые ячейки справа от сообщения считаются пустыми. Строка команды МТ в четверичном представлении состоит из 4 сообщений: состояние данной команды, какой символ головка видит, что делает при этом символе, следующее состояние. Например, строка 03,0,1,04. В состоянии 03, если головка увидит 0, то сотрет этот 0 и напишет 1, после чего перейдет в состояние 04. Также можно возвращаться в это же состояние, например, 03,0,>,03, данная строка будет идти вправо, пропуская все нули.

16) Нормальные алгоритмы Маркова. В этой алгоритмической модели преобразуются текстовые сообщения. Элементарными тактами обработки алгоритмов Маркова являются замены подслов исходного сообщения на некоторые другие слова. Нормальные алгоритмы Маркова (НАМ) по существу являются детерминистическими текстовыми зменами, которые для каждого выходного слова однозначно задают вычисления и, тем самым, в случае их завершения, порождают определенный результат.
Марковская стратегия применения правил заключается в следующем:
1)Если применено несколько правил, то берется правило, которое встречается в описании алгоритма первым;
2)Если правило применимо в нескольких местах обрабатываемого слова, то выбирается самое левое из этих мест.
НАМ представляет собой упорядоченный набор правил-продукций – пар слов, соединенных знаком ->. Слева от этого знака стоит слово, которое заменяется, а справа от этого знака стоит слово, на которое заменяется. Например строка az->z заменяет все слова az на z. Пример работы такой программы: на входе aaazazaz, на выходе zzz. НАМ не является нормируемым.

17) Диаграммы машин Тьюринга (ДМТ). ДМТ могут быть представлены через МТ. В ДМТ есть:
1) Машина копирования. Копирует слово слева от головки в конец сообщения, была создана для удобства нормирования программы.
2) Машина L – Двигает голову МТ влево до первого попавшегося пробела
3) Машина R – Двигает голову МТ вправо до первого попавшегося пробела
4) Элементарная l – двигает головку МТ влево на одну ячейку

5) Элементарная r – двигает головку МТ вправо на одну ячейку
6) Элементарная λ – производит запись буквы или пробела
Надо заметить, что все машины – группа трех элементарных команд, которые были созданы для удобства программиста.

18) Моделирование машин Тьюринга. Рассмотрим две МТ T $= (A, Q, P, q_o)$ и $T` = (A`, Q`, P`, q_o`)$. Будем говорить что машина $T`$ моделирует машину T и обозначать это $T` \cong$ T, если выполняются следующие условия.
1)Указан способ кодирования знаков алфавита A знаками A`
2)Каждому состоянию $q \ni Q$ машины T поставлено в соответствие некоторое состояние $q` \ni Q`$ машины T`, т.е. определено отображение $Q \to Q`$
3)Если C_0– начальная конфигурация машины T, то ее образ $C_0`$ - Начальная конфигурация машины T`
4)Если машина E из начальной конфигурации C_0 после конечного числа тактов останавливается на конфигурации C_k, то машина T` из начальной конфигурации $C_0`$, являющейся образом C_0, после конечного числа тактов также останавливается на конфигурации $C_k`$, являющейся образом C_k
5)Если из начальной конфигурации C_0 машина T пробегает последовательность конфигураций $C_0, C_1, \dots, C_k, \dots$, то каждая последовательность конфигураций, пробегаемая машиной (как же я заколебался) T` из начальной конфигурации $C_0`$ содержит в качесте подпоследовательности конфигураций $C_i` \; i = (0,1, \dots, k, \dots) -$ образы конфигураций C_i машины T.
T.e. машина T` описывает тот же алгоритм, что и T, но, возможно, проходит при выполнении алгоритма большее число промежуточных конфигураций. Таким образом, понятие моделирования вводит бинарное отношение алгоритмического равенства между машинами Тьюринга.

19) Эквивалентность программ и диаграмм. Каждой программе P, задающей МТ T $= (A, Q, P, q_o)$, можно эффективным образом сопоставить диаграмму D, образованную символами элементарных МТ так, чтобы МТ, определяемая этой диаграммой, моделировала бы машину T. T.e. нам нужно указать способ эффективного построения диаграммы по программе P, а потом убедиться. Что для исходной МТ и МТ, определяемой диаграммой D, выполняются все пять условий моделирования МТ.
Построение: Каждой строке A, Q, P, q_o сопоставим элемент диаграммы .v., при этом нам не нужно отображать пустые действия программы, а окончанием работы диаграммы будет последняя точка программы.
Док-во, что оно эквивалентно. Все 5 условий моделирования должны быть выполнены. Условие 1 выполняется потому, что алфавиты МТ и ДМТ совпадают. Условие 2 выполняется потому, что состояниям МТ соответствуют точки на ДМТ. Где A – левая точка, q – правая точка. Условие 3-5 выполняются потому, что ДМТ определяет ту же последовательность элементарных действий, что и программа P.

20) Эквивалентность диаграмм и программ. Каждой ДМТ, задающей программу P, можно эффективным образом сопоставить МТ T $= (A, Q, P, q_o)$,, образованную строками команд МТ так, чтобы МТ, определяющая эту диаграмму, смоделировала бы машину T. T.e. нам нужно указать способ эффективного построения программы по ДМТ программы P, а потом убедиться. Что для исходной ДМТ и МТ, выполняются все пять условий моделирования МТ.
Построение: Каждому элементу диаграммы .v. сопоставим строку команды A, Q, P, q_o ,при этом нам нужно будет поставить вместо последней точки ДМТ заключающую строку программы МТ.

Док-во, что оно эквивалентно. Все 5 условий моделирования должны быть выполнены. Условие 1 выполняется потому, что алфавиты МТ и ДМТ совпадают. Условие 2 выполняется потому, что точкам ДМТ соответствуют состояния МТ. Условие 3-5 выполняются потому, что МТ определяет ту же последовательность элементарных действий, что и программа Р.

21) I теорема Шеннона. Доказательство*. Формулировка: Для любой МТ Т $= (A, Q, P, q_o)$ с множеством состояний $Q = \{q_0, q_1, \dots, q_s\}$ можно эффективным образом построить МТ $T` = (A_r, \{a, b\}, P`, q_0`)$, моделируя машину Т и имеющую всего два состояния, a и b. Рабочий алфавит $A_r$ машины $T`$ содержит r=(3p+1)(s+1)+p букв.

22) II теорема Шеннона. Доказательство*. Для любой МТ Т $= (A, Q, P, q_o)$ можно эффективным образом построить МТ $T`` = (A_1, Q``, P``, q_0``)$, моделирующую машину Т и имеющую всего однобуквенный алфавит $A_1 = \{a_1\} = \{|\}$

23) Вычислимые функции.(а или б)
а)Функция называется вычислимой, если f(n) определено и программа завершает свою работу на входе n и возвращает значение f(n)
Если f(n) не определено, то программа зависает. Если выполняются оба условия, то функция вычислима.
б) Вычислимые функции — это множество функций вида, $f: N \to N$, которые могут быть реализованы на машине Тьюринга. Задачу вычисления функции f называют алгоритмически разрешимой или алгоритмически неразрешимой, в зависимости от того, возможно ли написать алгоритм, вычисляющий эту функцию.

24) Нормированные вычисления. Теорема о нормированной вычислимости. Доказательство*. Смысл нормированного вычисления состоит в том, что исходные данные сохраняются. Т.е. данные копируются и машина работает уже с откопированными данными, не меняя исходные, при этом головка после копирования не должна заходить левее последнего символа исходных данных.
Теорема: Предикат p называется нормировано вычислимым по Тьюрингу (НВТ-предикатом), если существует МТ с рабочим алфавитом А = {И, Л}, которая вычисляет p и удовлетворяет следующим условиям:
1)Если функция не определена, то программа зависает
2)Если функция определена и p – истинно, то все символы принадлежат алфвиту(?)
3) Если функция определена и p – ложно, то все символы принадлежат алфвиту(?)

25) Теорема о композиции. Пусть n-местная функция $f: (A_s^*) \to A_t^*$ определяется равенством $f(u_1, u_2, \dots, u_n) = h\big(g_1(u_1, u_2, \dots, u_n), g_2(u_1, u_2, \dots, u_n), \dots, g_m(u_1, u_2, \dots, u_n)\big)$, где g и h ВТ-фукцнии. Тогда f является ВТ-функцией, причем МТ, вычисляющую функцию f может быть эффективно построено из МТ, вычисляющих функцию g (i=1…m) и h

26) Теорема о ветвлении.
$$f(u_1, u_2, \dots, u_n) = \begin{cases} g_1(u_1, u_2, \dots, u_n), \text{если } p_1(u_1, u_2, \dots, u_n) = \text{И} \\ g_2(u_1, u_2, \dots, u_n), \text{если } p_2(u_1, u_2, \dots, u_n) = \text{И} \\ \dots \\ g_m(u_1, u_2, \dots, u_n), \text{если } p_m(u_1, u_2, \dots, u_n) = \text{И} \end{cases}$$
Если все g – ВТ функции, а все p – ВТ-предикаты, то тогда f - ВТ функция

27) Теорема о цикле. Пусть g – ВТ функция, вычисляемая функцией Е, а p – ВТ предикат, вычисляемый машиной Т, Тогда функция f, определеяемая вычислительным процессом: $g = g_0, g_0 \in A_8^*; f(u_1, u_2, \dots, u_n) = g\big(u_1, u_2, \dots, u_{j-1}, g, u_{j+1}, \dots, u_n\big)$ пока

$p(u_1, u_2, \dots, g, \dots, u_n) =$ И, тоже является ВТ-функцией.

28) Обобщенная теорема о цикле. Пусть $g_i: (A_8^n)^n \to A_8^*(i = 1,2, \dots, m) -$ ВТ $-$ функции, вычисляемые МТ T_{gi} , а $p_i: (A_8^n)^n \to \{И, Л\}$ $(i = 1,2, \dots, m)$ – ВТ предикаты, вычисляемые МТ T_{pi} $(i = 1,2, \dots, m)$ соответственно, тогда функция, определяемая соотношениями: $g_i: (A_8^n)^n \to A_8^*(i = 1,2, \dots, m)$, $f(u_1, u_2, \dots, u_n) =$

$$\begin{cases} g_1(u_1, u_2, \dots, u_{j-1}, g, u_{j+1}, \dots, u_n), \text{если } p_1(u_1, \dots, g_1, \dots, u_n) = И \\ g_2(u_1, u_2, \dots, u_{j-1}, g, u_{j+1}, \dots, u_n), \text{если } p_2(u_1, \dots, g_1, \dots, u_n) = И \\ \dots \\ g_m(u_1, u_2, \dots, u_{j-1}, g, u_{j+1}, \dots, u_n), \text{если } p_m(u_1, \dots, g_1, \dots, u_n) = И \end{cases} \text{ пока } \bigvee_{j=1}^{m} p_j = И$$

Также является ВТ-функцией, причем МТ, вычисляющая функцию, может быть эффективно построена из машин T_{gi} и T_{pi} $(i = 1,2, \dots, m)$

29) Схема машин Тьюринга. Нисходящаяя разработка.
Нисходящая разработка состоит в следующем. Мы функцию f выражаем через f1,f2,..,fk и составляет диаграмму машины T, включающую символы машины Ti (i=1…k). Каждую из функций fi выражаем через новые, более простые. И так до тех пор, пока на каком-то уровне не получатся диаграммы, включающие только символы элементарных МТ

30) Теорема Бойма-Джакопини-Миллса. Для любой машины Тьюринга T можно эффективно построить машину Тьюринга S, которая является структурной (т.е. диаграмма является схемой) и которая моделирует машину T

31) Универсальная машина Тьюринга. Построение*.
Для начала на ее ленту записывается последовательность четверок, представляющую программу машины, работу которой необходимо выполнить и начальные данные. Универсальная МТ (УМТ) просматривает программу, записанную на ее ленте, и выполняет команду за командой этой программы, получая на ленте, вслед за аргументами, результат вычислений.
Док-во

32) Линейная запись схем машин Тьюринга. Ветвление диаграммы представляется в виде: IF a1?S1 | a2?S2 | … | ak?Sk FI. Где a – условие, S – действие. IF – FI – начало и конец ветвления. Цикл представляется как DO a1?S1 | a2?S2 | … | ak?Sk OD. Все то же самое, только DO и OD – начало и конец цикла. Это позволит нам преобразовать двумерную диаграмму в одномерную строку

33) Критика модели вычислений Тьюринга. Надостатки МТ довольно существенны.
1)Описание программы, выполняющая алгоритм сложения двух дробей занял целую страницу, поэтому для более сложных математических операций понадобится большая трудоемкость программиста, даже при составленном алгоритме. 2)Необходимость многочисленных копирований. При описании нашего алгоритма около половины всех действий были именно действиями копирования. Из-за этого поиск нужных слов среди всего это громоздкого кода становится довольно затруднительным. 3)Необходимость при составлении программы выписывать ситуации на ленте, иначе можно легко запутаться. МТ удобно для построения красивой теории алгоритмов, но не для практических задач.

34) Модель фон Неймана.

35) Построение процессора фон Неймана. Для построения процессора необходимо:
1) Выбрать и зафиксировать рабочий алфавит процессора. Обычно это первые p чисел, включая ноль. Т.е. от 0 до p-1.

2)Зафиксировать конечное или бесконечное множество допустимых слов над алфавитом. Если множество допустимых слов конечно и включает лишь слова фиксированной длины к (либо слова, имеющие длину не больше, чем к(, то удобно добавить к множеству допустимых слов еще одно слово, называемое «переполнением».
3)Составить и записать на ленту программы МТ, выполняющие операции процессора, включая операции РИМ и АДР
4)Сообщить обозначения операций универсальной МТ, точнее, управляющей программе универсальной МТ, на основе которой строится процессор

36) Машина фон Неймана. Машиной фон Неймана называется аппаратная реализация процессора фон Неймана. Машина состоит из управляющего устройства, устройства памяти и одного или нескольких регистров. Память машины фон Неймана содержит конечное число ячеек ограниченного размера. В состав машины должны входить устройства записи сообщений в память и вывода данных из памяти на носитель, доступный для восприятия органами чувств человека.
В результате аппаратной реализации процессора фон Неймена мы перешли от сообщений к данным, потеряв абсолютную вычислимость и лишившись возможности непосредственно созерцать процесс обработки данных в ЭВМ.

37) Структура программ для машины фон Неймана. /*Программа и данные помещаются в одном и том же устройстве памяти, но на регистрах процессора слова с командами программы и данными существенно различаются. Таблица имен, программа и данные могут быть записаны в виде, непосредственно переносимом в память машины. Процесс составления таблицы имен и размещения данных в памяти машины может бать автоматизирован.
Каждый объект, используемый для получения значений других объектов при выполнении программы обработки данных должен иметь определенное значение. Начальные значения таких объектов могут быть заданы либо при размещении в их памяти машины (константы), либо путем выполнения инструкции ввода данных*/
А теперь к сути. Программа для машины фон Неймана представляет собой текст, обязательно включающий в себя инструкции описания объектов (данных, которые обрабатываает программа) инструкции ввода данных, инструкции обработки данных и инструкции вывода значений объектов-результатов.
Начало => инструкции описания объектов программы => инструкции ввода => инструкции обработки данных => инструкции вывода => КОНЕЦ

38) Нотация программ Э. Дейстры. Обобщенная инструкция присваивания и композиции.

39) Обобщенная инструкция ветвления. Правила ветвления:
1)одновременно и независимо вычисляются все предохранители.
2)Среди вычисленных предохранителей инструкции ветвления должен быть хотя бы один, принимающий значение И.
Если среди предохранителей инструкции ветвления нет ни одного, принимающего значение И, то происходит ОТКАЗ – выполнение программы прекращается аварийно
3)Допускается, чтобы среди предохранителей инструкции ветвления было более одного предохранителя, принимающего значение И.
Инструкции могут быть не упорядочены

40) Обобщенная инструкция цикла.
1)Количество предохранителей, принимающих значения И может быть более одного
2)Если среди предохранителей инструкции нет ни одного, принимающего значение И, то выполнение инструкции заканчивается естественным образом без прекращения работы программы

3)Выполнение каждой охраняемой инструкции должно приводить к изменению аргументов предохранителей, в противном случае выполнение инструкции цикла может никогда не закончиться

41) Понятия типа данных. Тип данных – множество изображений(Сам объект), для которых определено правило их интерпретации, позволяющее каждому изображению сопоставить его значение, и множество атрибутов(Что с этим объектом делать), которые позволяют одному или нескольким элементам типа данных сопоставить либо изображения данных того же типа, либо изображения данных другого типа.

42) Тип логический. Boolean. Тип данных, который может принимать всего два значения, 0 или 1, True или False, T или F. Занимает всего 1 бит памяти.
Популярные операции: Конъюнкция, дизъюнкция, тождественность и отрицание.
Операции логического типа обладают следующими свойствами: (по-моему они из пальца высосаны)
1)Если хотя бы один операнд имеет значение NULL, то и результат имеет значение NULL
2)X&X=X, XVX=X при X=И, Л
3)X = И, Y=любое значение кроме NULL, то, XVY=И
4) X = Л, Y=любое значение кроме NULL, то, X&Y=Л
5)Дизъюнкция и конъюнкция коммутативны.
6)XV-X=И, кроме X= NULL
7)X&-X=Л, кроме X= NULL
8)-(-X)=X

43) Тип литерный (char) Литерный тип используется для ввода-вывода и обработки текстовых данных. Множество значений определяется кодировкой: ASCII, KOI-8, и т.д.. Эта кодировка задает порядковый номер литеры, определяемой функцией преобразования, связывающей литерный тип с диапазоном целого числа (обычно 0…255) и предопределяет упорядоченность множества значений так, что имеет смысл понятия, как следующая литера и предыдущая литера. Одномерный массив литер – строка.

44) Тип целый. (integer, long int и пр). Значениями машинного целого типа являются все математические целые числа, заключенные между двумя фиксированными значениями Min и Max, являющимися атрибутами конкретного целого типа (разные границы у integer и long long). Формульно: $Z = \{z \in Z | MIN \leq z \leq MAX\} \cup$
$\{Null,$ переопределенное значение$\}$
Над такими типами можно проводить следующие операции: +,-,*,/,%.
(Свойства на странице 147. Это идиотизм, выделю важные)
4)Если один из операндов Null или inf (переполнение), то результат будет тоже Null или inf соответственно

45) Тип вещественный. (real). Рациональные числа используются для более точного измерения отрезков и промежутков времени. Число с плавающей точкой (вспоминаем мантису, да-да), а числа с фиксированной точкой никто не любит. Бит – знак мантисы, 7 бит – мантиса, бит – знак числа, оставшиеся 23 – само число

46) Согласование типов. Согласование целого и вещественного типа, как близких с математической точки зрения, осуществляется с помощью округления или отбрасывания дробной части (round() или trunc()). Обратное преобразование (из целого в вещественный) производится по умолчанию в случае необходимости приведения смешанного выражения к более сложному вещественному типу.

47) Небазовые типы данных (диапазон, перечисление, множество).
1)Отрезок type diap = (0.0 .. 2.0; real; real;);
2)Перечисление type StreetLight ((Red,Yellow,Green) ; := ; integer);
3)*ERROR*

48) Понятия о структурном типе данных. Структурные значения – упорядоченные систематически организованные совокупности других значений, рассматриваемых как единое целое. Компоненты структурных значений называются полями или элементами. Пример структурного типа – комплексный тип, состоящий из двух частей, вещественная и мнимая. Существует несколько методов структурирования:
1)Регулярный содержит только компоненты базового типа.
2)Комбинированный содержит компоненты различных типов. Его различают на:
2а)Индексированный (все компоненты имеют свой индекс)
2б)Квалифицированный (компонент идентифицируется именем)
2в)Секвенциальный (все последовательно)

49) Тип массив. Массив – регулярный структурный тип с индексированным методом доступа. Т.е. все элементы одного типа и все они имеют индекс. Для массивов одного типа определена операция присваивания (A=B) без поиндексированного копирования каждой переменной. Обработка массива в основном осуществляется покомпонентно, с помощью циклов.

50) Тип запись. Это комбинированный структурный тип с квалифицированным методом доступа. Комбинированность означает, что поля записи имеют различные типы. Квалифицированный доступ гибок и невычислим, т.к. требует явного указания. С помощью него можно делать удобные и понятные, в частности погромисту, структуры. (Пример С, стр. 178)

51) Понятия о файлах. Структура файла является обобщением понятия последовательности. Поэтому файлы следует считать «массивами на диске». Компоненты все одного тип и они доступны только путем последовательного прочтения, движения «назад» нет ввиду инерционности электромеханических устройств. Аналог – магнитная лента. Внешние файлы обычно перечисляются в заголовке программы. Они существуют до начала работы программы и/или сохраняются после окончания ее работы. Внутренние файлы, также как и внешние, описываются в программе как файловые переменные. Их время жизни совпадает с временем работы программы. Текстовые – в них только символы, юзается спец. Знак для конца строки. Нетекстовые файлы не предназначены для ввода-вывода и хранят данные непосредственно во внутримашинном представлении.

52) Блочная структура программ. Локальные и глобальные переменные.
Пример:

```
Block 1
{
   Block 2
   {
        Block 3{ }
   }
}
```

Такая структура эффективна по управлению и по организации самой программы по данным. Перед работой блока данные заносятся в памяти, а после отработки убираются, что позволяют сэкономить количество занимаемой программой памяти. Типичный пример – for.

53) Процедуры и функции. По сути – выделенный блок с названием (ни в коем случае это не пишите). Их значение в том, что они. Как и блоки, являются средством представления программ в виде логически связанных компонентов (это пишите). Процедуры помогают избегать повторения кода, в некоторых случаях ускоряют работу программы (в сравнении с тем случаем, где процедуры не использованы) и упрощают понимание программистом самой программы. В заголовке процедуры пишется имя, количество и виды параметров на вход, тип результата, а также некоторые необязательные атрибуты программы, которых довольно много. В теле находится сам код этой процедуры в виде блока.

54) Способы передачи параметров.
1)Передача по значению может быть охарактеризована как read-only. При передаче сначала выделяется память под значение, потом значение каждого параметра вычисляется в точке вызова подпрограммы и пересылаются в выделенную область памяти.
2)По результату. Вычисляется результат функции, а потом передается уже сам результат, а не как в передаче по значению (передается значение, а потом вычисляется результат).
3)По ссылке. При передаче параметров по ссылке передается адрес объекта и все обращения к параметру в подпрограмме происходят по этому адресу. При этом всякое присваивание значения внутри программы приводит к одновременному изменению значения во всей программе.
4)По имени. Он откладывает обработку параметров ровно до того момента, когда они действительно потребуются. Этот метод является самой мощной формой передачи параметров, наиболее опасной и неэффективной.

55) Критика языков программирования Паскаль и Си.

56) Критика алгоритмической модели фон Неймана. Модель фон Неймана пришла на замену модели МТ, дабы избавиться от недостатков предыдущей модели. К модели фон Неймана отнсят как ппаратные компьютеры, так и традиционные языки программирования. Их математические основания считаются сложными, громоздкими и концептуально бесполезными. Модель фон Неймана использует память и чувствительна к предыстории. Семантика также заключается в переходах из состояния в состояние, только они более сложные. Ясность программ гораздо выше. Наибольшая проблема модели фон Неймана – шина, которая за один раз может передавать только один определенный элемент памяти и является «Узким горлышком». Таким же горлышком в программировании является оператор присваивания, именно он вынуждает нас программировать на уровне «слово за слово».