

- 1 **С/к Введение в совр. программирование (v.5.5)**
 2 Физфак МГУ. 2006/7 уч. год.
 3 **Лекция 4**
- 4 **Напоминание** Формальная логика.
 5 Символизм: первый оператор **ASSERT(bool. expr)**
 6 Метафизика; Математика; Ремесло (меню)
- 7 **Метафизика: структура мышления ☺**
 8 2 Критический
 9 1 Комбинаторный
 10 0 Базовый
- 11 **0 Базовый: распознавание образов (pattern matching)**
 12 "Здорово, Евлампий!"
 13 Конфигурация сигналов --> символ
- 14 **Образец (pattern, схема)** -- конфигурация "сигналов", порождающая "клик" -
 15 - электрохим. возбуждение = "сигнал узнавания" + соотв. "символ".
 16 Сей "символ" -- ярлык для действия, правила и т.п., которое и применяется, в
 17 т.ч. для дальнейшего распознавания.
 18 Левая часть логич. или алгебраич. подстановки.
 19 Условие применимости преобразования, теоремы.
 20 Pattern matching = из ключевых пунктов деятельности по искусств. интеллекту
 21 (AI=artificial intelligence).
- 22 **NB** Неоднозначность ("лучи холода").
- 23 В мозгу "защиты": зрительный анализатор; слуховой а.
 24 Именно они используются. *Никакой мистики — только биохимия и физиология.*
 25 Чаще всего мышление на базе з.а. (*"Всякая мысль сопровождается зрительным*
 26 *образом"*)
 27 Зрительный исторически вроде бы старше, мощнее.
 28 *Но не у всех он доминирует!* (опыт преподавания)
- 29 Гены, воспитание, обучение, образование, опыт закладывают в нас огромное
 30 количество образов.
 31 Словарь, терминология; привычки, знания, предрассудки, мифы.
- 32 **Чаще всего усваиваются некритически** — просто не успеваем обдумывать!
 33 *Человеки = рабы привычек (all caps), предрассудков и мифов.*
- 34 **1 Комбинаторный уровень ("соображение")**
 35 *"палка+ящик --> банан"*
 36 Комбинирование видимых 3D объектов.
 37 Комбинирование символов в синтаксических конструкциях.
 38 ...
 39 Отличие человека от обезьяны: способность комбинировать гораздо больше
 40 двух объектов, чтобы достать "банан".
 41 Промежуточное комбинирование — "клики", электрохим. возбуждение центра
 42 удовольствия в мозгу (**иначе белковое тело ничего делать не будет**).
 43 --> Возможность искусств. стимуляции. Фантазирование.
 44 **NB** Источник избыточн. сложности в ИТ.
- 45 Нужна хорошая, быстрая память + энергия/работоспособность.

- 46 **2 Критический уровень ("мышление")**
 47 "Внимательное рассматривание", нахождение новых, переделка старых,
 48 организация самих образов.
 49 Это некая обратная связь — как любая обратная связь, переводит систему в
 50 **новое состояние**.
 51 Здесь важен момент интраверсии.
 52 *Далеко не у всех сей уровень развит.*
- 53 **Сложные задачи:** аморфность, отсутствие готовых образов.
 54 Либо имманентно сложные, либо сложные в силу количества деталей,
 55 переходящего в качество.
- 56 При сильном уровне 1 человек проходит через экзамены и проч. с песнями, не
 57 имея нужды (и в упор не видя, не понимая необходимости) развивать уровень 2
 58 (примеры...).
- 59 *Первую задачу А решил за минуту, Б за полчаса.*
 60 *Вторую задачу А решил за полчаса, Б на следующий день.*
 61 *Третью задачу А на следующий день, Б за неделю.*
 62 *Четвертую А не решил, Б принес решение через месяц.*
- 63 В реально сложных задачах, где комбинаторика "затыкается", уровень 2 бьет.
 64 Обычно в новой проблемной области:
 65 прикладывают старые паттерны ("теплород"),
 66 как попало придумывают новые на основе поверхностных корреляций,
 67 привыкают к ним --> по 100 лет не видят "правильных".
- 68 **Совершенно новые паттерны находить |вводить дьявольски трудно.**
 69
- 70 **Thomas Kuhn. The structure of scientific revolutions. Chicago U. 1964.**
 71 Томас Кун. Структура научных революций, Благовещенск, 1999
 72 "нормальная" наука, научные "революции" (термин технический)
 73 "революция" = смена доминирующего паттерна
 74 Умение взглянуть "сверху", "глобально", "абстрактно" (пример: законы сохр-я).
 75 Умение разглядеть то, что в упор не видят другие.
- 76 Программирование: "программистов много, архитекторов очень мало"
 77 *Эту разницу можно видеть всюду.*
 78 *В обсуждении образования: "практически-ориентированное" <-->*
 79 *"фундаментальное"*
 80 При чем даже те, кто вроде говорит о "фундаментальном", чаще всего не вполне
 81 понимают, о чем они говорят (просто повторяют заученные "паттерны":
 82 *"Фундаментальность — это хорошо! ДА!"*)
 83
- 84 Математика и точные науки подчеркивают комбинаторный уровень ("западня").
 85 В гум. науках его мало -- там прямо 2 ("выявление смыслов").
 86
- 87 Факты и данные --> интерпретации=паттерны --> мета-паттерны
- 88 **Башня интерпретаций над месивом фактов**
- 89 Понимание = организация данных, «сокращенные описания»
 90 (не вполне верно: «понимание = перевод на другой язык»)

131 Теория множеств (элементарная)

132 **Georg Ferdinand Ludwig Philipp Cantor** (1845 – 1918) was a mathematician
133 who was born in Russia and lived in Germany for most of his life. He is best known
134 as **the creator of modern set theory**. He is recognized by mathematicians for
135 having extended set theory to the concept of transfinite numbers, including the
136 cardinal and ordinal number classes. ...
137 Cantor's innovative mathematics faced significant resistance, especially by Leopold
138 Kronecker, Hermann Weyl, L.E.J. Brouwer, Henri Poincare and Ludwig Wittgenstein.
139 The hostile attitude of many contemporaries <"по полной программе для
140 **особо одаренных**"> is believed to have severely aggravated Cantor's emotional
141 ailments and to have caused several nervous breakdowns.

142 Today, the vast majority of mathematicians accept Cantor's work on transfinite sets
143 and recognize it as a paradigm shift of major importance. ..."

144 Теория множеств это:

145 (а) Язык, упорядочение терминологии — важнеющая вещь.
146 + "элементарная теория множеств" -- конечных и счетных.

147 (б) Теория несчетных множеств (континуум и т.п.), основания математики
148 (анализа).

149 Пример применения: открытие "аксиомы детерминированности" (Steinhaus-
150 Muczeliski, 1966) вместо одиозной "аксиомы выбора" Цермело.

151 АД = "аксиома отсутствия патологических контрпримеров" (все множества и все
152 функции измеримы по Лебегу; все обобщенные функции непрерывны в слабой
153 топологии ...)

154 По поводу (б) см. В.Кановой "Аксиома выбора и аксиома детерминированности",
155 М., Физматгиз, 1982.

156 У нас: только (а) = удобнейший и необходимейший инструмент ...

157 Понятия

158 **элемент, множество, пустое множество** \emptyset

159 элемент принадлежит множеству: $x \in A$

160 **мощность** — количество элементов. Может быть конечной или бесконечной.

161 Примеры

162 Дни года (1 января, ... 31 декабря).

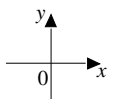
163 Дни недели (понедельник, вторник, ... воскресенье).

164 Буквы русского алфавита.

165 Натуральные числа.

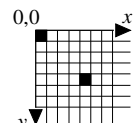
166 Точки x вещест. прямой.

167 Точки (x, y) — элементы вещественной плоскости. Идентифицируются парой



168 вещ. чисел.

169 Пиксели — элементы экрана. Идентифицируются парой целых, $(0,0)$ -- верхний



170 левый угол.

131 Соответственно, множества бывают **конечные, счетные** (можно
132 перенумеровать натуральными числами) или **несчетные** (напр., континуум
133 вещественных чисел).

134 Множество пикселей экрана конечно, его можно перенумеровать, но такая
135 нумерация смысла не имеет.

136 подмножество

137 множество A есть подмножество множества B (не обязательно строгое):
138 обозначается как $A \subset B$.

139 Примеры

140 Официальные праздничные дни.

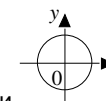
141 Рабочие дни.

142 **Q** Кстати, в каком отношении дни недели к дням года?

143 Буквы, обозначающие гласные.

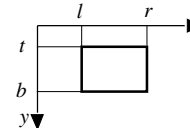
144 Сегменты $[n .. m]$ — подмножества целых чисел.

145 Сегменты (x, y) , $[x, y]$ и т.п. — подмножества вещественной прямой.



146 Внутренность единичного круга в плоскости.

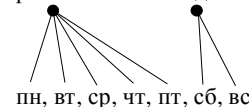
147 Пиксели, видимые внутри окна документа ББ — подмножество всех пикселей



148 экрана. left, top, right, bottom -- l, t, r, b = две точки (l, t) , (r, b)

149 **НВ** Главный инструмент абстракции:

150 рабочие выходные



151 пн, вт, ср, чт, пт, сб, вс

152 Как определить/описать (под)множество:

153 (0) прямым предъявлением всех элементов (не так редко);

154 (1) алгоритмом перечисления/порождения (связь с вычислимостью и т.п.);
155 когда большое конечное множество — "все клиенты" — то они прямо
156 предъявлены в файле,
157 но доступ к ним — через некий алгоритм чтения (ср. числа в "потоке
158 ввода", получаемые после $In.Real(x)$).

159 (2) косвенно, каким-либо свойством элементов (тут связь с логикой): $\{ i \mid$
160 $P(i) \}$.

161 Здесь надо перечислять все i , и фильтровать те, что удовлетворяют $P(i)$
162 — похоже на (1).

163 (3) Вербально. Выделяем этот пункт, т.к. надо еще проделать работу
164 перевода вербального описания на язык формул математики, языка прог-я и т.п.

165 **Упр** Перевести на язык логики описание подмножества -- прямоугольника
166 экрана l, t, r, b .

167 **НВ** Кстати, мы ненавязчиво ввели две схемы-паттерна:

168 (а) перечисление через спец механизм последовательного доступа;
169 (б) фильтрация.

171 **Множество всех подмножеств** в каком-либо множестве A : обозначается как
 172 $\{A\}$ (пустое и само A включаются сюда как элементы).
 173 если $x \in \{A\}$, то $x \subset A$.

174 Если мощность A есть n , то мощность $\{A\}$ есть 2^n .

175
 176 **НВ** Последовательность $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \dots$ моделирует целые числа, т.е.
 177 арифметику, а отсюда рукой подать до моделирования всей математики...
 178 <Ср. центральность понятия пустоты в буддизме.>

179
 180 **НВ** Понятия вроде "множество всех подмножеств" (т.е. всех вообще) приводят к
 181 логическим противоречиям.

183 Операции с подмножествами

184
 185 Заведомо всегда есть операции сравнения на (не)равенство.

186
 187 Уже появились две элементарных специфически теоретико-множественных
 188 операции над множествами и их элементами; значения операций — булевские
 189 (Т, F).
 190 Операция "быть подмножеством" напоминает (нестрогое!) сравнение чисел
 191 ($B \subset A \Leftrightarrow B \leq A$). Но подмножества лишь частично упорядочены.

192
 193 Теперь операции, дающие другие подмножества.
 194 Во-первых, преобразование элемента в подмножество, состоящее из этого
 195 элемента:

196 $x \rightarrow \{x\}$ при этом $x \in A, \{x\} \subset A$

197 Это — пример "преобразования типа".

198
 199 **Дополнение:** $x \in \bar{A} \Leftrightarrow \sim (x \in A)$ (еще пишут $\bar{C}A, A'$)

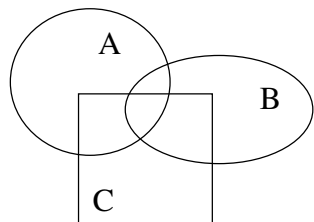
200 **НВ** Дополнение всегда в каком-то множестве. Если хотим подчеркнуть, то
 201 пишем $\bar{A} = X \setminus A$

202 Эту операцию можно использовать и для "вычитания" одного подмножества из
 203 другого (A не обязательно содержится в X).

204
 205 **Пересечение** $x \in (A \cap B) \Leftrightarrow (x \in A) \& (x \in B)$

206 **Объединение** $x \in (A \cup B) \Leftrightarrow (x \in A) \text{ OR } (x \in B)$

207 **Упр** Выписать формулы для дополнения пересечения и дополнения
 208 объединения.



210
 211 **Упр** Найти пересечение трех подмножеств: A , B и дополнения к C .

212 **НВ** Общие свойства множеств удобно иллюстрировать на таких картинках.

213 **Упр** Найти пересечение двух подмножеств экрана, заданных наборами чисел $l, t,$
 214 r, b и $l1, t1, r1, b1$.

215 **Упр** Записать логическое условие, когда это пересечение пусто.

216
 217 Постоянно встречается понятие:

218 **Непересекающиеся (под)множества** — которые не имеют общих элементов,
 219 т.е. пересечение пусто: $A \cap B = \emptyset$.

220 **Покрытие подмножества A подмножествами B_i** : объединение всех B_i
 221 содержит A : $A \subset \bigcup_i B_i$.

222 Важный вариант: покрытие *непересекающимися* множествами.

223
 224 Еще операции:

225 **Разность** $x \in (A \setminus B) \Leftrightarrow (x \in A) \& (x \in \bar{B})$

226 **Симметрическая разность** $A \Delta B \Leftrightarrow (A \setminus B) \cup (B \setminus A)$

227 **Упр** Написать логическое выражение, выражающее принадлежность x
 228 симметрической разности.

229 **Упр** Найти на картинке эти два подмножества.

230 Элементарный тип SET в О/КП

231 $\text{set} = \text{множество}$

232
 233 Встроенный в язык элементарный тип SET — прежде всего **средство для**
 234 **работы с битами**. Метафора теории множеств очень удобна (вместо выдумок
 235 вроде "битовых логических операций" в других языках).

236 **Важно различать** математические понятия и базовые понятия ЯП.

237 Например, процедуры-функции и SET-ы в КП — понятия уровня реализации, а
 238 не уровня проблемной области.

239 Т.е. общие множества могут реализовываться по-разному, чем-то более
 240 сложным, чем переменными типа SET.

241 Можно, конечно, с помощью SET непосредственно моделировать маленькие
 242 множества.

243
 244 **Значения типа SET** — подмножества множества целых чисел $\{0 \dots 31\}$.

245 **Полезные псевдо-константы** (из разд. 10.3):

246 $\text{MIN}(\text{SET}) = 0, \text{MAX}(\text{SET}) = 31$ (для гарантии переносимости программ на КП)

247
 248 Конкретные значения (константы) типа SET изображаются как $\{1, 3, 6 \dots 9, 17 \dots$
 249 $19, 30\}$.

250 Здесь $6 \dots 9$ — диапазон/range — все числа от 6 до 9 включительно.

251 $\text{VAR } a: \text{SET};$ — ячейка памяти в 32 бита (4 байта, одно маш. слово).

252 Простой пример

253 Объявим для элементов константы:

254 $\text{CONST } \text{понедельник} = 0; \text{вторник} = 2; \dots \text{воскресенье} = 6;$

255 Потом можно

256 CONST

257 $\text{неделя} = \{\text{понедельник} \dots \text{воскресенье}\};$

258 $\text{рабочие} = \{\text{понедельник} \dots \text{пятница}\};$

259 $\text{выходные} = \{\text{суббота} \dots \text{воскресенье}\};$

260 И всюду в программе оперировать этими константами.

261

262 **О машинном представлении.** Обычно если текущее значение s содержит
 263 число 5, то 5й бит установлен в 1, иначе в 0.
 264 Т.е. значение $\{ \}$ соответствует последовательности битов
 265 00000000000000000000000000000000,
 266 значение $\{0 \dots 31\}$ соответствует
 267 11111111111111111111111111111111,
 268 а значение $\{1, 3, 6 \dots 9, 17 \dots 19, 30\}$ соответствует
 269 01010011110000000111000000000010.
 270 **Проблема**, однако, в том, что такое представление предполагает некоторую
 271 группировку битов по байтам, а это вещь неочевидна (например, i386 и
 272 PowerPC ведут себя в этом отношении по-разному).
 273 **NB** Хорошо написанная программа на О/КП не предполагает этого знания: в
 274 О/КП есть все средства, чтобы абстрагироваться от этих деталей.
 275
 276 Логическое отношение:
 277 **x IN s** = "x является элементом s", $x \in s$
 278 x должен быть целым в диапазоне $0..MAX(SET)$,
 279 s должен иметь тип SET.
 280 Можно использовать в логических выражениях:
 281 $(x \bmod 7 = \text{понедельник}) \& (x \text{ IN } \text{праздничные})$
 282
 283 **Полезные процедуры** (из разд. 10.3):
 284 Пусть v : SET; x : целое, $0 \leq x \leq MAX(SET)$. Тогда:
 285 EXCL(v , x) — исключить x из v , или $v := v - \{x\}$
 286 INCL(v , x) — включить x в v , или $v := v + \{x\}$
 287
 288 **Упр** Пусть CONST январь = $\{1 \dots 31\}$.
 289 Пусть известно, что 1 января — среда.
 290 Написать последовательность операций, создающих в VAR s: SET значение,
 291 соответствующее всем рабочим дням января.
 292
 293 **Операции** над a, b : SET — смотрим Сообщение:
 294 **8.2.3 Операции над множествами**
 295
 296 + объединение
 297 - разность $(x - y = x * (-y))$
 298 * пересечение
 299 / симметричная разность $(x / y = (x - y) + (y - x))$
 300
 301 Операции над множествами применимы к операндам типа SET и дают результат
 302 типа SET. Одноместный минус обозначает дополнение множества x , т.е. $-x$
 303 обозначает множество целых от 0 до $MAX(SET)$, которые не являются
 304 элементами множества x . Операции над множествами не являются
 305 ассоциативными $((a+b)-c \neq a+(b-c))$.
 306 Конструктор множества определяет значение множества перечислением его
 307 элементов между фигурными скобками. Элементы должны быть целыми в
 308 диапазоне $0..MAX(SET)$. Диапазон $a..b$ обозначает все целые i такие, что
 309 $i \geq a$ и $i \leq b$.
 310 -----конец цитаты-----
 311
 312 **NB** Теор-множественные операции соответствуют побитовым логическим
 313 операциям след. образом.

314 Если выражать значения типа SET с помощью битов, устанавливаемых в 0, 1,
 315 как описано выше, и если интерпретировать эти 0, 1 как FALSE, TRUE, то для
 316 двух VAR a, b : SET будем иметь соответствия
 317 $a+b \rightarrow$ побитовое & (И)
 318 $a*b \rightarrow$ побитовое OR (ИЛИ)
 319 $a/b \rightarrow$ побитовое XOR
 320 **Упр** Выяснить побитовую интерпретацию операции - (разность множеств).
 321 **NB** Свои программы лучше строить на основе метафоры подмножеств, а не
 322 "побитовых операций".
 323 **Полезные функции** (из разд. 10.3):
 324 BITS(x) x : INTEGER результат: SET $\{i \mid ODD(x \text{ DIV } 2^i)\}$
 325 ORD(x) x : SET результат: INTEGER $(\sum i: i \text{ IN } x: 2^i)$
 326 **Упр** Обдумать, определена ли функция BITS для отрицательных аргументов?
 327 Определена ли ORD для x , содержащего 31? Прodelать тесты.
 328 **NB** Сообщение — очень серьезный (точный) документ. Из Введения:
 329 *"Большинство недоговоренностей оставлено намеренно, либо потому, что*
 330 *соответствующее уточнение может быть выведено из сформулированных*
 331 *правил языка, либо потому, что уточнение потребовало бы конкретизировать*
 332 *определения там, где универсальная конкретизация не кажется полезной."*
 333 "Неполезная" может значить и "затрудняющая написание эффективного
 334 переносимого компилятора без достаточных оснований".
 335
 336 **На практике** в прикладных программах SET не часто используется. Но для
 337 программирования "низкого уровня" (драйверы и т.п.) важен. В любом случае
 338 сложных конструкций не бывает.
 339
 340 Вернемся к математике.
 341 **Фундаментальный результат**
 342 **Эквивалентность булевых алгебр и алгебр подмножеств.**
 343 Прямая аналогия с булевой алгеброй: каждое подмножество можно трактовать
 344 как логическую переменную (вне-логический предикат $x \in A$). Тогда имеет
 345 место эквивалентность алгебры подмножеств некоторой булевой алгебре.
 346 Пересечение подмножеств \leftrightarrow конъюнкция (И) соотв. предикатов:
 347 $A \cap B \leftrightarrow (x \in A) \& (x \in B)$
 348 Объединение подмножеств \leftrightarrow дизъюнкция (ИЛИ) соотв. предикатов:
 349 $A \cup B \leftrightarrow (x \in A) \text{ OR } (x \in B)$
 350 Справедливо и обратное: каждую булеву алгебру можно интерпретировать как
 351 алгебру подмножеств некоторого множества.
 352
 353 **Обдумать** Интерпретация первой канонической формы в терминах
 354 "минимальных" непересекающихся подмножеств.
 355
 356 **На практике** сложные подмножества нередко описываются как пересечения
 357 или объединения других подмножеств попроще — точно так же, как сложные
 358 предикаты составляют из простых с помощью И и ИЛИ.
 359 **Упр** Написать предикат, соответствующий прямоугольнику экрана l,t,r,b,
 360 используя арифметические неравенства. Каким подмножествам соответствуют
 361 такие неравенства?
 362
 363 Квантор общности ("для всех") -- пересечение набора множеств.
 364 Квантор существования ("существует") -- объединение набора множеств.

361 **Характеристическая функция** подмножества: функция на элементах, равна
 362 нулю вне подмножества, единице внутри.
 363 Нередко удобнее брать F , T вместо 0 , 1 .
 364 **NB** Пример забавной инерции мышления у математиков (*кто-бы мог*
 365 *подумать...*):
 366 Тета-функцию Хевисайда (0 , 1) приятнее определять на логических значениях
 367 и использовать в виде $\theta(x > 0)$ вместо $\theta(x)$, тогда $\theta(x < y)$ вместо темного $\theta(x, y)$

368 Операции над множествами

369 **Почему это важно:** сложные множества всегда "делаются" из более простых.
 370 Полезно перечислить простейшие способы такого конструирования, из которых
 371 составляется все остальное. Это даст ориентацию в понимании набора средств
 372 конструирования "структур данных" в программировании.

373 **Множество всех подмножеств:** из одного множества A мощности n
 374 получаем другое, $\{A\}$ мощности 2^n . Несчетно для счетного A .
 375 **Важный вариант:** множество конечных подмножеств. Счетно для счетного A .

376 Декартово произведение

377 $A \times B$ — множество пар (a, b) . Мощность есть произведение мощностей A и B .
 378 Можно взять несколько "сомножителей" и применить конструкцию
 379 соответствующее число раз. Поскольку порядок в произведении фиксирован,
 380 имеется некая естественная ассоциативность: порядок попарной группировки
 381 *может быть* не важен. Но, строго говоря, $((a, b), c)$ и $(a, (b, c))$ различны.
 382 **Пример** Конструкция RECORD (запись) создает переменную, множество
 383 значений которой есть декартово произведение соотв. множеств для полей
 384 записи.
 385 **Упр** Пусть A и B — счетные (т.е. нумеруются натуральными числами).
 386 Перенумеровать $A \times B$, т.е. доказать его счетность.

387 Множество последовательностей

388 элементов A : $()$, (a) , (a, b) , (a, b, c) ...
 389 Например, множество цепочек литер.
 390 У каждой цепочки есть конечная **длина**.

391 Эту конструкцию можно выразить так: $\emptyset \cup A \cup (A \times A) \cup (A \times A \times A) \cup \dots$
 392 Здесь допущение бесконечной цепи операций — новый логический элемент,
 393 строго говоря.

394 **Упр** Найти мощность множества последовательностей длины не больше N .
 395 **Упр** Перенумеровать для счетного A .

396 Для этой конструкции в О/КП есть два б.м. прямых способа реализации:
 397 массивы (ARRAY) и списки. Будем подробно изучать.

398 Вариации

399 — Порядок важен или нет? Зависит от задачи. От порядка литер в цепочке
 400 смысл меняется, от порядка чисел в наборе измерений смысл (среднее,
 401 стандартное отклонение и т.п.) не меняется.
 402 — Допускаются ли последовательности любой длины или нет? Зависит.
 403 — Разрешена ли пустая последовательность? Зависит.
 404 **NB** Уже много возможностей.

405 Фактор-множество

406 Фактор-множество по отношению эквивалентности: простым языком:
 407 "огрубляем" рассмотрение, отождествляя некоторые элементы, различия между
 408 которыми несущественны в данный момент.
 409 Отношение эквивалентности = условие, при выполнении которого два элемента
 410 считаются эквивалентным ("неразличимыми" в данном контексте). Достаточно
 411 требовать, чтобы такое условие разбивало исходное множество на
 412 непересекающиеся части. Каждая часть — "класс эквивалентности", т.е.
 413 содержит все элементы исходного множества, которые считаются
 414 эквивалентными.

415 **Пример** Все целые \rightarrow четные (0) и нечетные (1). Здесь фокусируем внимание
 416 на делимости на 2 , абстрагируясь от всего остального.

417 **NB** Инструмент *абстракции* как развитие идеи "подмножества".

418 Идея особенно хороша, когда над "отождествленными" элементами можно
 419 выполнять операции, выполняя их на самом деле с представителями
 420 соответствующих подмножеств.

421 **Пример** Все целые разобьем на классы, объединяющие числа с одинаковым
 422 остатком от деления на $p > 1$. Таких классов p штук соответственно возможным
 423 остаткам $0, 1, \dots, p-1$. Эти остатки и можно взять в качестве представителей соотв.
 424 классов. Классы можно обозначить как $k(n)$, где $n=0, 1, \dots, p-1$.
 425 Тогда можно определить сложение классов:
 426 берем соответствующие остатки и их складываем, у результата находим остаток
 427 от деления на p (=складываем по модулю p). Соответствующий полученному
 428 остатку класс и есть результат сложения исходных классов:
 429 $k(a) + k(b) = k((a+b) \bmod p)$.

430 **NB** Суть конструкции в том, что для исходных классов можно взять любые
 431 числа из них в качестве представителей, и выполнить эту же операцию
 432 (сложение по модулю). Остаток не будет зависеть от выбора представителей.
 433 Только в таких случаях имеет смысл определять операции с классами.

434 Операций, наследуемых классами, может быть несколько. Не углубляемся -- так
 435 определяются фактор-группы, фактор-алгебры, фактор-пространства.

436 **Общий смысл всегда** — "огрубление" рассмотрения, "игнорирование" каких-
 437 то деталей — "абстракция".
 438 Например, фактор-пространства эквивалентны некоторым проекциям (проекции
 439 $(x, y) \rightarrow x$ и т.п.).
 440 Абстракция, согласованная с соотв. операциями.

441 **Пример** Фактор-кольцо кольца (есть полноценное сложение/вычитание, а
 442 также умножение, возможно, без деления) целых чисел по отношению
 443 эквивалентности "равный остаток от деления на p " оказывается полем (т.е.
 444 определено деление на ненулевой элемент), — только если p простое.

445 **NB** В компьютерной алгебре проверки -- головная боль. Полезный прием:
 446 использовать такие поля с разными p для проверки алгебраических алгоритмов
 447 вместо целых или рациональных чисел произвольной длины: числа "по модулю
 448 p " всегда меньше p , а соотв. арифметич. операции реализуются просто.

449 **Упр** (на будущее) Реализовать операции для такого поля в виде процедур.

450 **Множество неупорядоченных пар** множества A : (a,b) , где оба элемента из A ,
 451 причем (a,b) считается неотличимым от (b,a) .
 452 Тут — классический пример фактор-множества по отношению эквивалентности:
 453 взято декартово произведение $A \times A$, и в нем выбрано отношение
 454 эквивалентности " (a,b) и (b,a) считаются эквивалентными".
 455 **Упр** Найти мощность множества неупорядоченных пар.

456 Отображения (функции, mapping)

457 **НВ** Обсуждаем для полноты. Это понятие — несколько другого типа, чем
 458 рассмотренные выше конструкции, и по-другому используется (в т.ч. в
 459 раздумьях).

460
 461 Отображение (mapping, функция) A в B = множество пар (a,b) , в котором
 462 каждое a из A встречается в точности один раз. Это подмножество $A \times B$. Пишут
 463 $b=f(a)$.

464 Еще говорят про B -значную функцию, определенную на A .

465 Соответственно, множество всех отображений $A \rightarrow B$.

466 **Упр** Найти мощность для конечных A и B . Что для счетных — есть ли счетность?

467 **НВ** Эти понятия заведомо годятся для конечных и счетных множеств A , но
 468 обобщение на случай, когда A имеет более высокую мощность (континуум) —
 469 нетривиально: в общем случае понятие не имеет конструктивного смысла, и
 470 нужны ограничения (важнейшее — непрерывность функции).

471 Вариации

472 Функция f может быть **частичной**, т.е. определенной на подмножестве A ,
 473 называемом областью определения ($\text{domain: } \text{domain } f \subset A$). Можно говорить и об

474 области значений: $f(\text{domain } f) \subset B$

475 "Частичность" функции может состоять в том, что соответствующий алгоритм
 476 "обламывается" или "зацикливается" для некоторых значений аргумента
 477 (аргументов).

478
 479 Функция f определяет отображение *подмножеств* своего домена в подмножества
 480 B , т.е.

481 отображению $f: \text{domain } f \rightarrow B$ естественно и однозначно сопоставляется
 482 отображение $f': \{\text{domain } f\} \rightarrow \{B\}$.

483 Раз однозначно, то используют вольность речи и пишут f вместо f' , а также
 484 говорят про то, как f отображает какие-то подмножества в какие-то другие.

485
 486 Говорят об отображения на B , если $f(\text{domain } f) = B$, т.е. для каждого b из B
 487 можно решить уравнение $b=f(a)$.

488
 489 Отображение может быть или не быть **взаимно однозначным**. Если оно таким
 490 имеет место быть, то решение уравнения $b=f(a)$ единственно.

491

492 Метафизика

493 Все конструктивные объекты суть элементы счетных множеств.

494 (**Q** Что значит "суть"? фр. sont, лат. sunt, нем. sind)

495 Т.е. могут быть представлены натуральными числами. Т.е. любые манипуляции
 496 с ними, в т.ч. любое вычисление, программа, алгоритм -- это манипуляции с
 497 целыми числами.

498

499 Богатство смыслов появляется, когда мы добавляем определения (вводим
 500 паттерны).

501 Ср. начало лекции + рассуждения насчет того, добавляется ли смысл
 502 (информация) в доказательствах и теоремах, в кн.:

503 В.Я.Перминов, Развитие представлений о надежности математического
 504 доказательства. МГУ, 1986.

505 (Частично-) Упорядоченные множества

506 Порядок — исключит. важное понятие в комбинаторике и
 507 вообще в алгоритмах (на сортировки уходит большая доля времени CPU).

508 Множество + "старшинство" эл-тов (операции \leq и \geq , и, конечно, $<$ и $>$).
 509 Естественные ограничения ($a \leq a$, $a \leq b$ и $b \leq c \Rightarrow a \leq c$ и т.п.).

510 Часто важное понятие сравнения есть, но определено не для всех пар (как для
 511 включения подмножеств).

512 Поэтому даже если некоторое сравнение есть, не обязательно любая пара эл-
 513 тов сравнимы.

514 Если любая — то имеем **линейно упорядоченное множество** — элементы
 515 можно расположить в один ряд, перенумеровать и сравнивать номера.

516 **Пример** Цепочки литер с **лексикографическим упорядочением**.

517

518 **Обобщение** цепочки литер: (a_0, a_1, \dots, a_n) , где каждый a_i — элемент
 519 своего множества A_i (не обязательно конечного). Если внутри каждого A_i можно
 520 сравнить любую пару, то можно лексикографически сравнить и любую пару
 521 таких цепочек (a_0, a_1, \dots, a_n) и (b_0, b_1, \dots, b_m) — сначала сравним a_0 и b_0 ... и
 522 т.д.

523 Часто какие-нить объекты комбинаторной природы кодируются такими
 524 обобщенными цепочками (обычно выбор кодировки неоднозначен, важно
 525 выбрать хорошо, зависит от задачи). Соответствующая лексикографическая
 526 упорядоченность "наводит порядок".

527

528 **Пример** "Спаривания" при вычислении шпуров дираковских гамма-матриц в
 529 квантовой теории поля. На человеческом языке речь идет о наборе $2n$ объектов
 530 (можно их считать числами $0 \dots 2n-1$), и требуется перебрать все их разбиения
 531 на n пар, причем ни порядок чисел внутри пар, ни порядок пар не важны.

532 Будем представлять такие "спаривания" как последовательности пар
 533 $((a_1, b_1), (a_2, b_2), \dots, (a_n, b_n))$,

534 причем есть ограничение:

535 $\{a_1, b_1, \dots, a_n, b_n\} = \{0 \dots 2n-1\}$

536 и доп. условия:

537 для любого i : $a_i < b_i$

538 и для любых $i < j$: $a_i < a_j$. (**Q** Можно ли здесь брать b ?)

539 Упорядочить можно, сравнивая сначала a_1 , потом b_1 , потом a_2 ...

540

541 По образцу лексикогр. порядка можно упорядочивать самые разные множества,
 542 выбирая разные представления для элементов. Разные представления -- разные
 543 порядки.

544 **Упр** Придумать три существенно разные кодировки последовательностями
 545 чисел для вещественных полиномов степени не выше N .

546

547

548 В общем случае нужно найти (что всегда возможно — **Q** Почему?)
 549 представление типа последовательности (обычно будут какие-то
 550 дополнительные ограничения). Такое представление будет похоже на запись в
 551 виде слова из некоторых "букв". Затем ввести (квази-) лексикографический
 552 порядок для таких "слов".

553 Обычно важной задачей при работе с любым множеством является возможность
 554 перебрать его элементы. (Например, в приведенном примере "спариваний" их
 555 нужно перечислить, чтобы вычислить соответствующий шпур.) Если они уже
 556 есть (заданы извне, как измерения, клиенты, ...) -- хорошо. Если они имеют
 557 комбинаторный характер, то нужен алгоритм их построения.

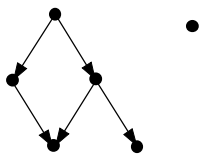
558 Если найдено представление элементов с квази-лексикографическим порядком,
 559 как обсуждалось выше, то порождать их можно, генерируя соответствующие
 560 "слова" в лексикографическом порядке, и отсеивая те, что не удовлетворяют
 561 нужным ограничениям. Здесь важно как можно раньше распознавать и
 562 отсеивать то, что заведомо не нужно. Это дело техники. Пример будет
 563 рассмотрен.

564

565 Короче говоря, **подход, связанный с упорядоченностью, — один из самых**
 566 **важных** в задачах этого типа. См. по этому поводу М.Айгнер, Комбинаторная
 567 теория, М., Мир, 1982.

568

569 В общем случае — **"ориентированный ациклический граф"**
 570 Т.е. элементы — точки, отношения < — стрелочки.



571

572 **Пример** Модули, упорядоченные по импорту.

573

574 Вообще графы (граф = набор точек, некоторые из которых соединены
 575 линиями/стрелочками) — целая отдельная алгоритмическая песня ("теория
 576 графов", масса книг: Оре, Харари, Кауфман, ...).

577 **Конечные автоматы (automaton, *p/* automata)**

578 Тема важнее для проектировщиков железа (им нужен целый формализм), чем
 579 для программистов (нам нужна только главная идея).

580 Неформально:

581 Конечный автомат = устройство, могущее принимать состояния из некоторого
 582 конечного множества, а также менять свое состояние детерминированным
 583 образом под действием каких-то хорошо определенных операций.

584 Два конечных множества: состояния, операции + для каждого состояния
 585 подмножество допустимых операций + правило перехода от состояния к
 586 состоянию под действием допустимых операций.

587 Важнейшее понятие: текущее **состояние** / current **state**.

588 **Пример:** Состояния: свет есть/света нет; операции: вкл./выкл.

589 **NB** Не все операции могут быть применимы в каком-то состоянии.

590 **Другой пример:** переменная. Состояния = значения. Операция перехода =
 591 присваивание, результат не зависит от предыдущего состояния.

592 **Третий пример:** компьютер. (Частный случай: машина Тьюринга).

593 Состояние = состояние его памяти (включая регистры, которые помнят, какую
 594 маш. команду только что выполнили и т.п.).

595 Переходы = все, что мы делаем (нажатия клавиш, клики и т.п.).

596 **Четвертый пример:** модуль In.

597 Состояние: пара In.Done и текущее положение во входном потоке.

598 Переходы — вызов процедур In.Real и т.п. Допустимые операции зависят от
 599 входного потока и текущего положения в нем.

600

601 Это — фундаментальное понятие для описания взаимодействия человек-
 602 компьютер.

603 **NB** В частности, отсюда видно, что *"функциональная парадигма"* (где нет
 604 переменных, а есть только функции и параметры) неадекватна для
 605 описания таких взаимодействий — если не прибегать к логическим
 606 извращениям (*"функции, принимающие значения во множестве всех*
 607 *состояний компьютера; тогда каждый акт взаимодействия есть некоторая*
 608 *такая функция ..."* — но результат-то — состояние компьютера как целого
 609 — все равно хранится между отдельными вызовами функций...)

610

611 Важное понятие: **инициализация**.

612 Некая операция, гарантирующее у данного автомата некое стандартное
 613 начальное состояние.

614 Пример: при входе в процедуру мы *не знаем* состояние (значения локальных
 615 переменных).

616

617 **"Состояния управления"** Когда состояний много, но они как-то
 618 классифицируются (на непересекающиеся подмножества), и мы в данный
 619 момент не интересуемся различиями между состояниями одного семейства.

620 Например, не интересуемся, в каком месте входного потока стоит In, только
 621 In.Done: TRUE или FALSE.

622 А.Шальто в этом случае говорит о "состоянии управления" -- на языке теории
 623 множеств тут будет что-то вроде фактор-автомата.

624 Второй пример: можно объединять группу операций в одну процедуру — тоже
 625 упрощаем автомат.

626 Для нас тут главное: понятие о некоем устройстве с фиксированным набором
 627 операций и (возможно, частично скрытом) наборе состояний.

628 Ремесленная часть **Как менять меню (в т.ч.** 629 **контекстное [по правому клику])**

630 Info —> Menu, откроется окошко (ББ-текст с гиперссылками)
Menu Files:

[System/Rsrc/Menu](#)

[Dev/Rsrc/Menu](#)

[Epse21/Rsrc/Menu](#)

[Form/Rsrc/Menu](#)

[Obx/Rsrc/Menu](#)

[Plot/Rsrc/Menu](#)

[Sql/Rsrc/Menu](#)

[Text/Rsrc/Menu](#)

631
632 Скажем, хотите изменить контекстное меню для текстов (т.е. меню,
633 появляющееся при правом клике в окне с ББ-текстом): кликнуть по самой
634 первой гиперссылке (System) — откроется главный документ, описывающий
635 меню -- просто ББ-текст. Увидите след. (показана часть содержимого окна):

```

MENU"File"
    "&New"           "N"      "StdCmds.New"      ""
    "&Open..."      "O"      "HostCmds.Open"    ""
    "&Open Stationery..." ""    "HostCmds.OpenCopyOf" ""
    "&Save"           "S"      "HostCmds.Save"    "HostCmds.SaveGuard"
636    "Save &As..."  "**S"   "HostCmds.SaveAs"  "StdCmds.WindowGuard"

```

637 Измените в заголовке File --> МойFile, спасите документ (Ctrl+S — ББ иногда
638 может выдать предупреждение о создании новой папки, это нормально,
639 кликните по ОК).
640 Закройте его (Ctrl+F4). Пока меню старые.

641 Теперь выполните Info, Update Menu. Voila!

642 **NB:** по этой команде ББ считывает тот менюшный файл, который на диске —
643 независимо от того, что в памяти. Поэтому закрывать файл не обязательно, но
644 спасти — обязательно!

646 **Упр** Поменяйте обратно МойFile —> File.

647 Откройте главное меню снова (та же картинка, что и выше).

648 **Первая колонка:** название пункта меню. Амперсанд — подчеркнутая буква
649 (кто не знает: Alt+F,N = Ctrl+N и т.п.;

650 **NB** полезно запоминать часто встречающиеся 2-клавишные комбинации;
651 например, Alt+A, R = Attr, Red = покрасить в красный/начать печатать
652 красным).

653 **Вторая колонка:** между двойными кавычками описывает горячую клавишу для
654 данного пункта меню:

```

    Если буква X, то Ctrl+X,
    если *X, то Ctrl+Shift+X,
    если F5, то функц. клавиша F5,
    если *F5, то Shift+F5.

```

659 Если "", то никакой гор. клавиши не предусмотрено, но присутствовать должны.

660 **Третья колонка:** команда ББ, которая должна вызываться. Можно цепочку
661 команд через ;
662 Иногда можно и параметры, но сейчас обсуждать не будем.

663 **Четвертая колонка:** "охрана" или пусто — процедура с определенным
664 интерфейсом, которая проверяет, применим ли данный пункт меню в данный
665 момент (см. в меню серым "выключенные" пункты).
666 Часто нужно разрешить команду только если что-то выделено. Для этого есть
667 StdCmds.SelectionGuard (указать между кавычек в 4й колонке).

668 **Упр** Поменяйте какие-нибудь горячие клавиши.

669 Подсистемы и меню

670 Каждая подсистема обычно задает свое меню в стандартном файле
671 */Rsrc/Menu.odc.

672 Строка:

673 **INCLUDE** "*"

674 заставляет ББ включить все меню, описанные в документах */Rsrc/Menu.odc

675 Контекстное меню для текстов

676 Info, Menu

677 кликнуть по последнему пункту (Text/Rsrc/Menu),

678 увидим окно, в котором первая строка:

679 **MENU** "Text" ("TextViews.View")

680 Выражение в скобках означает, что это меню будет видно только в том случае,
681 если в фокусе ("переднее" окошко в ББ) находится текст.

682

683 Ниже будет строка

684 **MENU** "*" ("TextViews.View")

685 Именно это меню (со звездочкой) и появится в качестве контекстного, если в
686 фокусе — текст.

687 **Упр** Посмотрите, как выглядит контекстное меню. Поменяйте порядок пунктов.
688 Обновите меню. Откройте контекстное меню правым кликом, проверьте, что
689 порядок изменился. Верните все в начальное положение.

690 Суммируем

691 Меню в Блэкбоксе (как и сообщения компилятора, все диалоги и т.п.) —
692 обновляются "на лету".

693 Причем заметьте: все пункты меню вызываются определенными процедурами
694 из определенных модулей (см. менюшные документы).

695 Но все эти команды — включая Info, Update Menu и т.п. — **можно вызывать из наших собственных программ!**

697 Если уметь открывать/менять/сохранять документы ББ, то возможны забавные
698 игры.

699 **Упр*** Сделать новое меню с единственным пунктом (горячая клавиша
700 Ctrl+Shift+G), который появлялся бы в верхней линейке меню, только когда в
701 фокусе выделен фрагмент текста. Действие при выполнении пункта меню
702 должно состоять в том, чтобы печатать в Log среднее арифметическое
703 последовательности чисел в выделенном фрагменте.

704 **Hint.** Проверить, как работает In, когда в фокусе есть выделенный фрагмент
705 текста (см. файл System/Docu/In_rus.odc)

706 **NB Интерфейсы = мосты через пропасть между мозгом и камнем**