

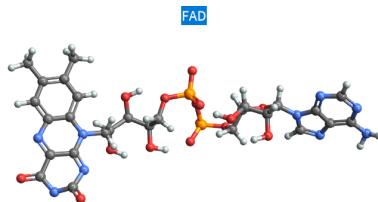
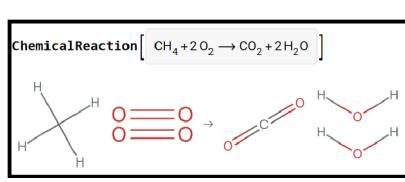
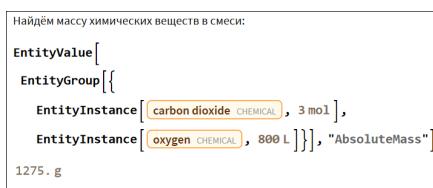
```
In[1]:= Remove["Global`*"];
Needs["JLink`"];
ReinstallJava[JVMArguments -> {"-Xmx16000m"}];
```

Виртуальная лаборатория

Введение.

Современная вычислительная математика предоставляет огромный арсенал инструментов, которые могут быть использованы в виртуальной лаборатории.

Они позволяют исследователю не только упростить вычисления, но и *анализировать* молекулярные объекты, их химический состав, определять значения свойств, изучать химические реакции. Виртуально раскрывая механизмы протекающих процессов, учёные получают возможность находить причинно-следственные связи и элементы, ответственные за такие связи .



Возможность визуализации структуры молекул посредством пространственных диаграмм особенно важна для биологически активных веществ, поскольку именно специфичность 3D-структур определяет их активность. Так, взаимодействие с биополимером может происходить благодаря определённому расположению в пространстве некоторых групп, которое не отражается 2D-графикой.

При разработке лекарств, изучении сворачивания белков и молекулярном распознавании ДНК используют функции, позволяющие классифицировать молекулярные формы и топологию структур, вычисляя длины связей, двугранные углы, моменты инерции и т.д.

Для прогнозирования химического поведения и определения химического синтеза находят и выделяют шаблоны для функциональных групп, выполняют замены подструктур. С помощью алгоритмов теории графов проводят поиск кратчайших путей при анализе сетей цепных реакций.

При построении химических реакторов, решении уравнений реакций или моделировании массопереноса в условиях сложной геометрии используют моделирование PDE.

Разработка промышленных биотехнологических устройств связана с анализом дезактивации катализатора, вычислениями поглощение газа, выполнением циклической вольтамперометрии, моделированием центрифugирования растворенного вещества и т.п.

Атомы, молекулы, высокомолекулярные последовательности, химические формулы или химические реакции для виртуальной работы могут представляться не только посредством символов, но и через их химические названия, строки SMILES, вырезанные изображения и др.

Биомолекулярные последовательности исследуются с помощью строкового представления, интегрированного с распознаванием, сравнением, транслитерацией и другими функциями. С помощью встроенной системы сущностей можно проанализировать последовательности генов и белков и их поведение .

Встроенная в *UK* поддержка органической и неорганической номенклатуры позволяет быстро создавать объекты атомов и молекул. Используемые специализированные *UK*-функции тесно интегрированы с общедоступными данными.

Встроенная в *UK* поддержка органической и неорганической номенклатуры позволяет быстро создавать объекты атомов и молекул. Используемые специализированные *UK*-функции тесно интегрированы с общедоступными данными.

Проводимые исследования могут дополняться различными научными сведениями (о биологических объектах, их строении и функциях, медицинскими данными и т.п.), информация о которых содержится в базах данных, накопленных различными научными сообществами.

Entity	Сущности
--------	----------

Иойрийэ уюйиргожпржукмвгкжлч лмпргз

Под *лсч лмпръ* понимают любой однозначно идентифицируемый конкретный или абстрактный объект, обычно относящийся к какой-либо предметной или тематической области. Это “набор данных, который структурирован таким образом, чтобы обеспечить установление отношений между отдельными элементами данных в соответствии с различными потребностями пользователей” [W5].

Сущности помогают управлять уникальными данными. Они обеспечивают их целостность в программе или базе данных.

Сущность является основным понятием *CP*-модели (*cl rgийжиргнл фглк нбсј*, модель “сущность-связь”), предложенной П. Ченом [C1]. Модель основывается на семантической информации о реальном мире (отдельная сущность опирается на смысл представляемых данных) и предназначена для логического представления данных. Она определяет значения данных в контексте их взаимосвязи с другими данными. В сущности, согласно этой модели, представлено множество атрибутов, которые описывают свойства всех членов данного набора сущностей (типа сущностей). На этой основе выделяют ключевые сущности и обозначают связи, которые могут устанавливаться между этими сущностями.

CP-модель является наиболее общей, служа основой для всех существующих моделей данных (иерархическая, сетевая, реляционная, объектная).

Для визуализации *CP*-модели используют различные графические нотации (нотация П. Чена, Crow’s Foot, UML, DFD, IDEF0 и IDEF1x, нотация Мартина, нотация Бахмана и др.).

В настоящее время в программировании применяется *CCP*-модель (*cl f YI acb cl rgийжиргнл фглк нбсј*, расширенная *CP*-модель). Это *CP*-модель, дополненная понятиями подкласса, суперкласса и относящимися к ним понятиями специализации, обобщения и категоризации.

ОУУмрүллсч лмпрэжж

С помощью специальных функций сущности включают в сложные вычисления. Поскольку сущности обладают различными свойствами, то изменения в значениях их свойств автоматически приведёт к изменениям результатов вычислений, в которых они участвуют. Это позволяет широко использовать сущности в моделировании различных процессов.

UJ-база знаний включает физические, математические и другие сущности (*лк лCl rgийжиргнсј*), в том числе свойства и отношения. В различных предметных областях значение термина сущность может обозначать специфические понятия.

Для работы с сущностями *UJ*-база знаний используют встроенные символы языка Wolfram Language.

Объект сущности может иметь следующие формы:

 ethanol CHEMICAL  , ethanol

сущность молекулы этанола,

deoxyribonucleic acid bases CHEMICALS , deoxyribonucleic acid bases

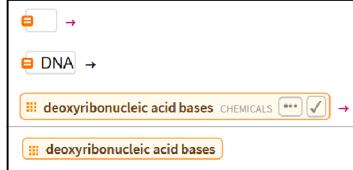
сущность класса “DNA” из типа “Chemical”, содержащего известные молекулы ДНК;

k ljasjYpu cfr

сущность свойства “Молекулярный вес”.

Для того, чтобы использовать объект сущности в вычислениях, сущность можно указать:

- используя ввод на естественном языке с помощью команды `ctrl =`:



- используя функцию `Entity [...]`.

Для обозначения сущности могут использоваться разные имена, большинство из которых являются атрибутами объекта. Свойства одного и того же атрибута могут различаться для различных сущностей с одним и тем же именем. Для атрибутов существуют функции, которые вычисляют значение имени из других сущностей. Кроме того, некоторые сущности являются неоднозначными: они могут содержаться в различных типах базы данных. При этом они представляют различные объекты, имея различную структуру в разных доменах.

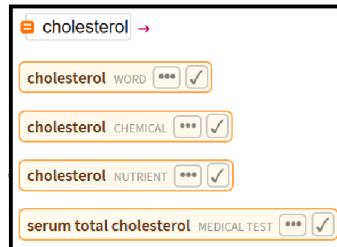


Рис. Сущность “Холестерол” представлена в типах сущностей: “Chemical”, “Medical Test”, “Nutrient”, “Word”.

Сущность обычно определяется как `Entity[“имя сущности”]`.

Для манипулирования встроенными сущностями необходимо иметь возможность привести используемое имя к определённому виду. Для этого используют функции `CanonicalName`, `CommonName`.

```
EntityList[  6 phosphofructokinase complex PROTEINS ] // CanonicalName
(*наименования сущностей класса "Комплекс 6-фософруктокиназы"*)
EntityList[  6 phosphofructokinase complex PROTEINS ] // CommonName
```

```
Out[=] =
{PFLIsoformA, PFKM, PFKP}

Out[=] =
{liver phosphofructokinase isoform a,
phosphofructokinase, muscle, phosphofructokinase, platelet}
```

Для работы с группой выбранных сущностей можно использовать функцию `EntityGroup`.

Встроенные *WJ*-базы сущностей являются тематическими (лк. **Босиницкіч лмпрг з жок УЛІУжнц жпУлж д**). Некоторые из них рассмотрены подробно (“`Gene`”, “`Chemical`”, “`Protein`” и др.).

Отдельные типы сущностей содержат классы, в которые вложены соответствующие объекты сущностей.

Для работы с классами сущностей используют функцию `EntityClass`.

Определить, к какому типу относится объект сущности можно с помощью `EntityType`.

Информация `Information` о выражении `EntityClass` и `Entity` будет иметь различные свойства в зависимости от `EntityType`, связанного с выражением.

```
In[4]:= {Information[Entity["Element", "Oxygen"]], Information[Entity["Chemical", "Oxygen"]]}
```

Out[4]=

<p>Entity</p> <p>oxygen ELEMENT</p> <p>Canonical Name Oxygen</p> <p>Image</p> <p>Atomic Mass 15.999 u</p> <p>Atomic Number 8</p> <p>Boiling Point -182.9 °C</p> <p>Mass Density 0.001429 g/cm³</p> <p>Melting Point -218.3 °C</p> <p>Phase gas</p> <p>Short Electronic Configuration [He]2s²2p⁴</p> <p>Full Dataset...</p>	<p>Entity</p> <p>oxygen CHEMICAL</p> <p>Canonical Name Oxygen</p> <p>Phase Gas</p> <p>Melting Point -218. °C</p> <p>Boiling Point -183. °C</p> <p>Mass Density 0.001429 g/cm³</p> <p>Formula O₂</p> <p>Molar Mass 31.998 g/mol</p> <p>C A S Number CAS7782-44-7</p> <p>Dielectric Constant 1.00049</p> <p>M D L Number MFCD00011434</p> <p>Full Dataset...</p>
--	--

```
In[5]:= {Information[EntityClass["Element", "NobleGas"]], Information[EntityClass["Disease", "CoronavirusDiseases"]]}
```

Out[5]=

<p>Entity Class</p> <p>noble gases ELEMENT</p> <p>Entity Count 7</p>	<p>Entity Class</p> <p>coronavirus diseases DISEASE</p> <p>Entity Count 4</p>
--	---

Список доступных классов сущностей в типе можно найти с помощью [EntityclassList](#).

```
EntityclassList["CrystallographicSpaceGroup"]
(*тип сущностей "CrystallographicSpaceGroup"*)
```

```
Out[n] =
{ dihexagonal pyramidal , dihexagonal dipyramidal , diploidal ,
ditetragonal dipyramidal , ditetragonal pyramidal , ditrigonal dipyramidal ,
ditrigonal pyramidal , ditrigonal scalahedral , gyroidal , hexagonal dipyramidal ,
hexagonal pyramidal , hexagonal trapezoidal , hexoctahedral ,
monoclinic domatic , monoclinic prismatic , monoclinic sphenoidal ,
orthorhombic bipyramidal , orthorhombic pyramidal , orthorhombic sphenoidal ,
rhombohedral , tetartoidal , tetragonal dipyramidal , tetragonal disphenoidal ,
tetragonal pyramidal , tetragonal scalenoidal , tetragonal trapezoidal ,
tetrahedral , triclinic pedial , triclinic pinacoidal , trigonal dipyramidal ,
trigonal pyramidal , trigonal trapezoidal , crystallographic space groups }
```

Для получения списка всех доступных сущностей - членов в данном классе используют [EntityList](#).

```
EntityList[ CDC-like kinases GENES ]
(*список сущностей "циклинзависимых киназ"-ферментов,
участвующих в смене фаз клеточного цикла*)
```

```
Out[n] =
{ CDC-like kinase 1 , CDC-like kinase 2 ,
CDC-like kinase 2, pseudogene , CDC-like kinase 3 , CDC-like kinase 4 }
```

Для ознакомления с сущностями определённого типа/класса можно воспользоваться функцией [RandomEntity](#) или [SampledEntityClass](#).

Объекты [Entity](#) могут возвращаться функцией [Interpreter](#):

```
In[n] := Interpreter["Isotope"] ["polonium-212"]
Out[n] =
polonium-212

In[n] := Interpreter["Isotope"] ["polonium-212"] === Entity["Isotope", "Polonium212"]
Out[n] =
True
```

Если база сущностей содержит иерархическую структуру, то взаимосвязи объектов в заданной структуре можно выявить с помощью функции [EntityType](#).

Пользователь может конструировать собственные сущности и классы сущностей из них (*пк\Asqmk iBcd\ cb Cl rgw Rnscq*, создавать динамические типы сущностей (*пк\Bw Yk gYjjwAm\ qpsarcb Cl rgwA\Yqscq*) из внешних хранилищ сущностей и реляционных баз данных, группировать и сортировать сущности из баз сущностей по установленному свойству.

ИоУрийЭ уйоийргожжий огкэфмилшу Уйжвүллшү

Огкэфмилшү Уйжвүллшү – это набор данных, имеющие изначально заданные взаимосвязи.

Она составлена по реляционной модели, предложенной Э.Коддом [C2; C3], который сформулировал законы, единые для любой

реляционной базы данных, и заложил основы построения запросов для работы с данными. Модель основывается на теории множеств.

Данные объединяются в таблицы, в которых:

- каждая строка представляет собой отдельную запись,
- каждый столбец состоит из атрибутов, содержащих значения.

Табличный формат позволяет легко устанавливать связи между отдельными данными и получать доступ к информации, не реорганизовывая данные.

Разные таблицы в базе данных также соотносятся друг с другом строго определенным образом.

Реляционные базы данных используют обширный комплекс инструментов, обеспечивающих целостность данных - их точность, полноту и единство.

Для взаимодействия с реляционными базами данных используется SQL (Structured Query Language) — язык структурированных запросов, служащий основой интерфейса систем управления базами данных. SQL позволяет работать со строками таблиц, извлекать нужные блоки информации и производить транзакции.

Сформированный класс, в котором все сущности удовлетворяют указанным ограничениям на значения свойств, [EntityClass\[*rgw*, {*пртсри*→*tYsc₆*, *пртсри*→*tYsc_y*,...}\]](#) называется [лгэалммногвгкгллшк икүтпмк псч лмпргз](#).

Нев явно определенные классы сущностей могут быть преобразованы в канонически именованные сущности с помощью [EntityList\[EntityClass\[*rgw*, {*пртсри*→*tYsc₆*, *пртсри*→*tYsc_y*,...}\]\]](#).

Памз праў псч лмпргз

Отдельные объекты баз сущностей обладают определенными свойствами. Сущности свойств “*пртсри*” можно получить:

- для отдельной сущности с помощью функции [EntityProperty](#);
- для целого типа сущностей с помощью [EntityProperties](#) [*rgw*], что

эквивалентно [EntityValue\[*rgw*, “Properties”\]](#).

Свойства как сущности в некоторых типах также могут группироваться по классам. Список доступных классов свойств для определенного типа сущностей можно найти с помощью [EntityPropertyClass](#).

Некоторые свойства доступны для типа сущности “*rgw*” в целом. Их можно использовать в качестве аргумента [пртсри](#) для:

- [Entity](#) [“*rgw*”] [*пртсри*]
- [EntityValue](#) [“Protein”, *пртсри*].

Возможные [пртсри](#) включают (вкэ многвгкяллшк уржна псч лмпргз лгимрмошг памз праў к мбср ўшцрълг вмпрслш):

“Properties”	список доступных свойств
“PropertyCanonicalNames”	стандартные названия доступных свойств
“SampleEntities”	пример списка доступных сущностей (обычно длиной 10)
“SampleEntityClasses”	пример списка доступных классов сущностей (обычно длиной 10)
“EntityCount”	количество доступных сущностей
“Entities”	список доступных сущностей
“EntityCanonicalNames”	стандартные имена доступных сущностей
“EntityClasses”	список доступных классов сущностей
“EntityClassCanonicalNames”	стандартные имена доступных классов сущностей
“PropertyClasses”	список доступных классов свойств
“PropertyClassCanonicalNames”	стандартные названия доступных классов свойств
“PropertyCount”	количество доступных свойств

Жүхглэ памз пра псч лмпргз

Значения свойств сущностей находят с помощью следующих команд:

<i>cl rgw[<i>пртсри</i>] ==</i>	
<i>EntityValue</i> [...]	вычисление значения указанного свойства <i>пртсри</i> для сущности <i>cl rgw</i>

<code>cl rgw [{nprm_g ...}]</code>	вычисление значений для списка свойств
<code>cl rgw [nprm, { os Yj → t Yj_g, ...}]</code>	возвращает значение указанного свойства,
измененного правилами квалификатора $os Yj_g \rightarrow t Yj_g$	
<code>EntityClass["UJrwc", {nprm_g → spca_g ...}]</code>	представляет класс сущностей со значениями $nprm_g$
определенными спецификацией $spca_g$	

<code>cl rgw [Dated [nprm, bYrc]]</code>	дает значение свойства,
связанного с конкретной спецификацией даты (вкэ многвгкяллшу ржнма пч лмпргз ж памзпра)	

Если значение свойства в базе отсутствует, в качестве результата выдаётся {};
если свойство для данной сущности отсутствует, в качестве результата выдаётся Missing["NotAvailable"].

Форму представления результатов вычислений можно определить, используя следующие команды:

- `EntityValue[cl rgw EntityProperties[cl rgw, YI / mYrgt]]` вкэ памзпрау пч лмпржà
- `EntityValue[cl rgw YI / mYrgt]` вкэ алгу памзпра пч лмпржà

Могут использоваться следующие `YI / mYrgt` (вкэ многвгкяллшу ржнма пч лмпргз лгимрмошг памзпрау к мбср ўщръ лгвмпрслш):

“EntityAssociation”	ассоциация сущностей и значений свойств
“PropertyAssociation”	ассоциация свойств и их значений
“EntityPropertyAssociation”	ассоциация, в которой указанные сущности являются ключами, а значения - это вложенная ассоциация свойств и их значений
“PropertyEntityAssociation”	ассоциация, в которой указанные свойства являются ключами, а значения - это вложенная ассоциация сущностей и значений свойства
“Dataset”	набор данных, в котором указанные сущности являются ключами, а значения - это ассоциация свойств и их значений
“Association”	вложенная ассоциация, в которой сущности являются ключами на первом уровне и свойства - на втором уровне
“NonMissingPropertyAssociation”	ассоциация свойств и их значений, за исключением тех, для которых значения отсутствуют
“NonMissingEntityAssociation”	ассоциация сущностей и значений их свойства, за исключением тех, для которых значения отсутствуют

“Source”	исходная информация, связанная со значением свойства
“Date”	дата, связанная со значением свойства

Метаданные свойств определяют с помощью команды `EntityValue [nprmg YI / mYrgt]`.

Свойства метаданных включают (вкэ многвгкяллшу ржнма пч лмпргз лгимрмошг памзпрау к мбср ўщръ лгвмпрслш):

“Qualifiers”	список возможных квалификаторов для свойства
“QualifierValues”	список возможных значений, которые можно присвоить каждому квалификатору
“DefaultQualifierValues”	список значений по умолчанию для квалификаторов свойства
“Description”	краткое текстовое описание свойства
“Definition”	подробное текстовое определение свойства
“Label”	метка свойства
“Source”	источник информации о свойстве
“PhysicalQuantity”	физическая величина, связанная со значением свойства
“Unit”	единица измерения свойства

```
In[]:= EntityValue[acjjsjYpark nm cl rq, "Description"]
Out[=]= cellular components

In[]:= EntityValue[jmasq, "Label"]
Out[=]= locus
```

Значение квалификатора [Automatic](#) указывает на использование применимого формата значений: например, для квалификатора "Date" - это конкретная дата или диапазон дат.

При проведении расчётов может быть полезна работа с сущностью, у которой определены некоторые значения свойств. Для этого используют функцию [EntityInstance](#).

Мплмалшг т слифюквкэ оўїмрщплсч лмпрэкж

Символическое представление сущности Entity

Entity [<i>типа</i> , <i>по</i> <i>именованию</i>]	представляет сущность указанного типа <i>типа</i> , идентифицированную по наименованию <i>по</i> <i>именованию</i>
Entity [<i>агрега</i> , <i>по</i> <i>именованию</i>]	представляет сущность из вычисляемого типа сущностей, указанного <i>агрега</i>

Вгрўкжнїфкэ памз пра

типа может быть:

- либо встроенным типом [Entity](#),
- либо типом, указанным в созданном пользователем [EntityStore](#), зарегистрированном с помощью [EntityRegister](#).

Возможные формы *агрега*

<i>"типа"</i>		
EntityClass [...]	AggregatedEntityClass [...]	CombinedEntity-
Class [...]	ExtendedEntityClass [...]	
FilteredEntityClass [...]	SampledEntityClass [...]	SortedEntityClass
[...]		

Значения свойств сущности можно вычислить с помощью [Entity](#) [...] [*примеси*], где *примеси*:

- строка,
- выражение [EntityProperty](#);
- выражение [EntityFunction](#).

[Entity](#) [*типа*] иногда используется для представления общей сущности указанного типа.

Entity ["*AnimalUse*"] ["*Properties*"] (*свойства *WL* -типа "*AnimalUse*"*)

```
Out[=]= {cl rgwajYqqcq, cl rgwrlncjgr, l Yk c}
```

Выражения сущностей могут форматироваться в **StandardForm** и **TraditionalForm** с использованием печатной формы имени сущности.

```
In[1]:= beta actin PROTEIN // StandardForm
beta actin PROTEIN // TraditionalForm
Out[1]//StandardForm=
beta actin
Out[1]//TraditionalForm=
beta actin
```

Нажмите

Найдём сущность из встроенного типа «**Protein**», представляющая β -актин – важнейший белок цитоскелета:

```
In[2]:= Entity["Protein", "ACTB"]
Out[2]=
beta actin
```

Определим его молекулярную массу:

```
In[3]:= %[ "MolecularWeight"]
Out[3]=
41.6060000 kDa
```

Найдём сущность из встроенного типа «**Gene**», представляющая PDE7B человека — ген млекопитающих, кодирующий 3'5'-циклическую нуклеотидфосфодиэстеразу (PDE), которая преобразует цAMP в 5'-AMP, важнейший белок цитоскелета:

Используем для указания сущности естественный язык **ctrl + =**:

```
In[4]:= PDE7B
Out[4]=
phosphodiesterase 7B (human)
```

Посмотрим базовую структуру найденной сущности:

```
In[5]:= InputForm[%]
Out[5]//InputForm=
Entity["Gene", {"PDE7B", {"Species" -> "HomoSapiens"}}]
```

Множественное предложение

`rwic` в `Entity` может быть динамически сформированным классом:

```
ent = Entity[
  AggregatedEntityClass["Country", "Population" → Total, "Continent"],
  EntityClass["Country", "Europe"]]
```

`Out[#] =`

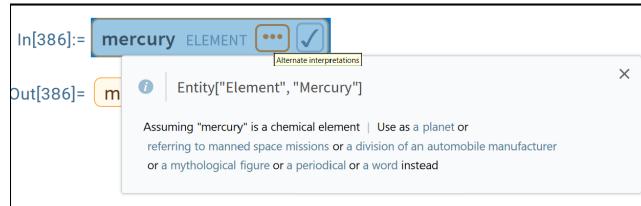
Найдём количество населения всех стран Европы:

```
In[#] := ent["Population"]
```

`Out[#] =`

597 817 459 people

Для неоднозначных сущностей используют механизм предположений (*ожиданий*), чтобы выбрать желаемую альтернативу:



```
In[#] := mercury ELEMENT
```

`Out[#] =`

```
In[#] := Mercury PLANET
```

`Out[#] =`

```
In[#] := EntityTypeName /@ {█████████████████████, ██████████████████████}
```

`Out[#] =`

{Element, Planet}

Для поиска элемента «Entities» для заданного типа сущности можно использовать `EntityValue`:

```
In[#] := EntityValue[fever PROTEINS, "Entities"]
```

`Out[#] =`

{interleukin 1, alpha proprotein, interleukin 1, beta proprotein}

CanonicalName и **CommonName** сущностей

Найдём наименования отдельной сущности:

CanonicalName [**caffeine** CHEMICAL] (*каноническое имя кофеина*)

Out[*n*] =

Caffeine

CommonName [**polonium-212** ISOTOPe] (*общеупотребительные наименование полония 212*)

Out[*n*] =

polonium-212

Найдём наименования сущностей в классе “Азотистые основания ДНК”:

EntityList [**deoxyribonucleic acid bases** CHEMICALS] // **CanonicalName**

EntityList [**deoxyribonucleic acid bases** CHEMICALS] // **CommonName**

Out[*n*] =

{Adenine, Cytosine, Guanine, Thymine}

Out[*n*] =

{adenine, cytosine, guanine, thymine}

Значение свойства объекта **Entity**:

beta actin PROTEIN ["Properties"] (*все свойства сущности*)

Out[*n*] =

{ "Ybbggnt Yj Yrnk rnncq", "ZYjj Yl b qrgi bgeprk", "ZgjnegeYj prnacqccq", "acjjsjYpank nnclrq",
 "af Yg jYZcjq", "bnk Yg Bq", "bnk Yg nnaggntq", "bnk Yg q", "cl rgwajYqccq", "cl rgwrwncjgr", "ecl c",
 "ecl c B", "ewprgnl pYbgsq", "k ck Zcpfngq", "k njcasjYpablargtq", "k njcasjYpucgef", "l Yk c",
 "LA Graaccqgnq", "NB Bqjgr", "njk YpnNB B", "njk YpnNB SPJ", "qcaml bYpnqrsarspcpsjcq",
 "qcoscl ac", "pxdpxl ac qcoscl acjcl erf", "qnYacqijg e bgeprk", "pzZml bgeprk", "ugcphk c bgeprk" }

beta actin PROTEIN [**ZgjnegeYj prnacqccq**] (*одно свойство*)

Out[*n*] =

{CellMotility, SensoryPerceptionOfSound}

polonium-212 ISOTOPe ["Radioactivity"]

Out[*n*] =

2.32×10^6 Bq

Символическое представление класса сущностей
EntityClass

EntityClass[*rnnc*, *l Yk c*]

иентифицированных по наименованию *l Yk c*

представляет класс сущностей *rnnc*,

EntityClass[*rwnc*, {*пртсрг* → *tqnsa_g*, *пртсрг* → *tqnsa_y*, ...}] предоставляет неявно определенный класс сущностей, содержащий сущности указанного типа *rwnc*

для которых свойства *пртсрг* соответствуют указанным значениям *tqnsa_g*

EntityClass[*афсаупс*] предоставляет класс сущностей с членами, указанными *афса*, выбранными по свойству *лцж*

В格外кожыфю памзра

Встроенные объекты **EntityClass** можно указывать на естественном языке с помощью **ctrl =**:

In[1]:=  DNA

 deoxyribonucleic acid bases CHEMICALS (*представляет базу данных ДНК*)

In[2]:= **InputForm**[ deoxyribonucleic acid bases CHEMICALS] (*базовая структура найденной сущности*)
Out[2]:= EntityClass["Chemical", "DNABases"]

rwnc может быть:

- либо встроенным типом **Entity**,
- либо типом, указанным в созданном пользователем **EntityStore**, зарегистрированном с помощью **EntityRegister**.

rwnc иметь форму:

- “*rwnc*”,
- “*rwnc*” → “*афѓб_g*” → “*афѓб_y*” →

Для задания значений свойств *tqnsa_g* в неявно определенных классах сущностей могут использоваться:

- **Quantity** и интервалы **Quantity** для размерных значений,
- **Between** для числовых значений,
- **DateObject** для дат,
- **TakeLargest** и **TakeSmallest** для порядковых выборок,
- **ContainsAll**, **ContainsExactly**, **ContainsAny**, **ContainsOnly**, **ContainsNone** для сущностей. Списки сущностей интерпретируются как **ContainsAll**, одна сущность - как **ContainsAny** [{*c1* | *гүй*}].

Ножкгощ

Получим из *UJ*-базы список химических веществ, являющихся спиртами:

```
EntityList[
  EntityClass["Chemical", "Alcohols"]] // Shallow
Out[1]:= {methanol, ethanol, propargyl alcohol, allyl alcohol, N-propanol, isopropanol,
  2-aminoethanol, 2-fluoroethanol, (R)-(+)-3-butyn-2-ol, (S)-(-)-3-butyn-2-ol, <<1180>>}
```

Получим список молекулярных масс для найденного списка:

```
EntityValue[
  EntityClass["Chemical", "Alcohols"], k nicasjYpk Yqq] // Shallow
Out[=]//Shallow=
{ 32.042 u, 46.07 u, 56.064 u, 58.08 u, 60.10 u,
  60.10 u, 61.08 u, 64.059 u, 70.09 u, 70.09 u, <<1180>> }
```

Создадим неявно определенный класс сущностей, состоящий из всех элементов, плотность которых попадает в указанный диапазон значений:

```
In[=]:= EntityClass["Element",
  {"Density" → Quantity[Interval[{5, 8}], "Grams" / "Centimeters" ^ 3]]]
Out[=]= EntityClass[Element, {Density → (5 to 8) g/cm^3}]
```

Перечислим эти элементы:

```
In[=]:= EntityList[%]
Out[=]= {gadolinium, iron, manganese, samarium, indium, tin,
  promethium, chromium, zinc, tennessine, neodymium, antimony,
  cerium, praseodymium, ytterbium, zirconium, astatine, tellurium,
  lanthanum, vanadium, gallium, arsenic, germanium, europium, radium}
```

Используя неявно определенный класс сущностей, выберем из группы молекул ацил-СоА тиоэстераз пять с наибольшей длиной кодирующих их генов и получим соответствующий список:

```
EntityValue[
  EntityClass[{"acyl-CoA thioesterases" GENES, {ecl c jcl erf} → TakeLargest[5]}, 
  {ecl c jcl erf}, "EntityAssociation"]
Out[=]= {acyl-CoA thioesterase 11(human) → 86 611 bp,
  acyl-CoA thioesterase 12(human) → 64 042 bp, acyl-CoA thioesterase 13(human) → 38 035 bp,
  acyl-CoA thioesterase 7(human) → 129 495 bp, acyl-CoA thioesterase 9(human) → 39 631 bp}
```

МпмУглмпржножглглжэ

Если неявно определенный класс сущностей не содержит объекты сущностей, необходимо проверить корректность выбранных ограничений:

```
In[1]:= EntityClass[{"interferons", "GENES"}, 
  "ecl/cjcl/erf"] → Quantity[Interval[{40000, 50000}], "BasePairs"];
EntityList[%]

Out[1]= {}

In[2]:= EntityValue[EntityClass[{"interferons", "GENES"}, 
  "ecl/cjcl/erf"] → TakeLargest[5], 
  "ecl/cjcl/erf", "EntityAssociation"] (*оценим наибольшую длину гена*)

Out[2]= {{"interferon (alpha, beta and omega) receptor 1 (human)" → 34916 bp, 
  "interferon (alpha, beta and omega) receptor 2 (human)" → 34597 bp, 
  "interferon gamma receptor 1 (human)" → 21947 bp, 
  "interferon gamma receptor 2 (interferon gamma transducer 1) (human)" → 34627 bp, 
  "interferon, lambda receptor 1 (human)" → 33119 bp}}
```



```
In[3]:= EntityClass[{"interferons", "GENES"}, 
  "ecl/cjcl/erf"] → Quantity[Interval[{30000, 35000}], "BasePairs"];
EntityList[%]

Out[3]= {"interferon (alpha, beta and omega) receptor 1 (human)", "interferon (alpha, beta and omega) receptor 2 (human)", 
  "interferon gamma receptor 2 (interferon gamma transducer 1) (human)", "interferon, lambda receptor 1 (human)"}
```

Неявно определенный класс сущностей может быть указан с использованием любого количества значений свойств.

Зададим интервалы значений для длины кодирующих генов и молекулярной массы и получим список интерферонов, удовлетворяющих указанным критериям:

```
In[=]:= EntityClass[{"interferons", "GENES"}, 
  {{"ecl cjc1 erf"} → Quantity[Interval[{1000, 2000}], "BasePairs"], 
   {"nprmcg k njcasjYpu cgef r"} → Quantity[Interval[{15, 25}], "Kilodaltons"]}]]

EntityList[%]

Out[=]= EntityClass[{"interferons"}, 
  {{"ecl cjc1 erf"} → (1000 to 2000) bp, {"nprmcg k njcasjYpu cgef r"} → (15 to 25) kDa}]

Out[=]= {"interferon, alpha 2(human)", "interferon, alpha 21(human)", 
  "interferon, alpha 8(human)", "interferon, lambda 2(human)"}
```

Перечислим сущности, которые одновременно удовлетворяют комбинации наибольшего и наименьшего порядковых условий вместе с явным значением свойства:

```
In[=]:= EntityClass[{"chemical carcinogens", "CHEMICALS"}, 
  {{"bwI Yk ggtgamgw"} → TakeLargest[4], 
   {"k Yqqbcl qgw"} → TakeSmallest[4], {"l crgt graf Ypec"} → 0}][["Entities"]]

Out[=]= {"benzene"}
```

Рассчитаем среднее значение и стандартное отклонение максимального веса для всех пород собак, выбранных по минимальному росту самца:

```
EntityValue[
  EntityClass["DogBreed", {EntityProperty["DogBreed", "HeightMaleMinimum"] →
    Quantity[Interval[{15, 18}], "Inches"]}], 
  EntityProperty["DogBreed", "WeightMaleMaximum"], {Mean, StandardDeviation}]

Out[=]= {21.3661 kg, 6.77924 kg}
```

Определим, какие измерительные приборы являются термометрами:

```
In[=]:= EntityClass["MeasurementDevice",
  {"InstanceOf" → ContainsAll[{ thermometer MEASUREMENT DEVICE }]}] // EntityList

Out[=]= {aethrioscope, bimetallic strip thermometer, exhaust gas temperature gauge, infrared thermometer,
permanent downhole gauge, pyrometer, silicon bandgap temperature sensor, thermistor,
thermocouple, thermograph, microwave thermograph, thermohydrometer,
alcohol thermometer, Beckmann thermometer, Breguet's thermometer, candy thermometer,
fiber optical thermometer, Galileo thermometer, maximum-minimum thermometer,
meat thermometer, medical thermometer, mercury thermometer, pill thermometer,
quartz thermometer, resistance thermometer, vapor pressure thermometer}
```

Перечислим все физические эффекты, открытые в 1976 году:

```
In[=]:= EntityClass["PhysicalEffect", {"DiscoveryDate" → DateObject[{1976}]}] // EntityList

Out[=]= {Faber-Jackson law, Gravity Probe A, Kumakhov radiation effect, supergravity, Unruh effect}
```

Ножкамд глжэ

Используем неявно определенный класс сущностей, чтобы получить информацию о пяти самых длинных пищеварительных органов:

```
In[]:= organ =  
EntityClass["AnatomicalStructure", {cl rgwajYqqcq →  digestive organs ANATOMICAL STRUCTURES, jcl erf + average of all STATISTICAL GROUP → TakeLargest[5]}];  
EntityValue[organ, {"Length", "Image"}, "Dataset"]
```

Out[=]



Используем неявно определенный класс сущностей, чтобы найти шесть лучших пород овец, отсортированных по количеству производимой ими шерсти:

```
In[6]:= sheep = EntityClass["SheepBreed", {gccacucgefr → TakeLargest[6]}];
EntityValue[sheep, {"FleeceWeight", "Image"}, "Dataset"]
```

Out[6]=



Создадим коллаж из изображений динозавров с определённой массой тела:

```
In[7]:= din = EntityClass["Dinosaur", {clrgwajYqqq →  dinosaur species DINOSAURS ,
ucgefr → Between[{Quantity[100, "Kilograms"], Quantity[200, "Kilograms"]}] }];
```

In[1]:= `ImageCollage[Rule @@ EntityValue[din, {"Weight", "Image"}], Background -> Red]`

Out[1]=



Найдём математические функции:

Найдём математические функции, связанные с функцией Abs

In[2]:= `EntityClass["MathematicalFunction", {"RelatedFunctions" -> ContainsAny[{absolute value function |x| MATHEMATICAL FUNCTION}]}] // EntityList`

Out[2]=

{absolute value function $|x|$, argument function $\arg(x)$, ceiling function $\lceil x \rceil$,
 complex conjugate function x^* , floor function $\lfloor x \rfloor$, fractional-part function $\text{frac}(x)$,
 imaginary-part function $\text{Im}(x)$, integer-part function $\text{int}(x)$, maximum function $\text{Max}(v, w)$,
 minimum function $\text{Min}(v, w)$, congruence function $k \bmod l$, integer quotient function $\text{quotient}(k, l)$,
 real-part function $\text{Re}(x)$, nearest-integer function $\lfloor x \rfloor$, sawtooth wave function $\text{SawtoothWave}(v)$,
 sign function $\text{sgn}(x)$, square wave function $\text{SquareWave}(v)$, triangle wave function $\text{TriangleWave}(v)$,
 unit box function $\Pi(v)$, unit step function $\theta(v)$, unit triangle function $\Lambda(v)$ }

Найдём математические функции с более чем 10 именованными тождествами:

In[3]:= `EntityClass["MathematicalFunction", {"NamedIdentities" -> (Length[#] > 10 &) }] // EntityList`

Out[3]=

{modified Bessel function $G(x)$, Bessel function $H(x)$, modified Bessel function $I_v(x)$,
 binomial coefficient $\binom{l}{i}$, cosine function $\cos(x)$, exponential function e^x ,
 gamma function $\Gamma(x)$, harmonic number F_x , hypergeometric function ${}_3D_2(Y_1, Y_2, Y_3; Z_1, Z_2; x)$,
 natural logarithm $\log(x)$, power function x^y , sine function $\sin(x)$ }

Получим случайную физическую систему с двумя степенями свободы:

```
In[=]:= RandomEntity[EntityClass["PhysicalSystem",
  {EntityProperty["PhysicalSystem", "DegreesOfFreedom"] \[Rule] 2}]]
```

Out[=]=

particle in a box in 2D

Список сущностей из класса сущностей EntityList

<code>EntityList[<i>ajYqq</i>]</code>	представляет список сущностей из класса сущностей <i>ajYqq</i>
<code>EntityList["<i>rnnc</i>"]</code>	представляет список сущностей указанного типа <i>rnnc</i>
<code>EntityList[<i>ajYqq</i> & <i>njgdr</i>]</code>	составляет список сущностей из класса <i>ajYqq</i> при этом упрощение <i>ajYqq</i> определит, следует ли сводить сущности к максимально простому типу
<i>В гүлкүйфө памз пра</i>	

ajYqq может быть: EntityClass FilteredEntityClass SortedEntityClass
 SampledEntityClass ExtendedEntityClass AggregatedEntityClass CombinedEntityClass

`EntityList["rnnc"]` систематически поддерживается только для типов, допускающих фиксированное количество сущностей.

`EntityList[ajYqq]` фактически эквивалентен `EntityList[ajYqqTrue]`.

По умолчанию *ajYqq*True.

При приведении сущностей к их базовому типу внешние слои `EntityClass`, `FilteredEntityClass`, `SortedEntityClass` и `SampledEntityClass` можно удалить; для остальных функций они сохраняются.

Амжк мәдлишг номүкгкүш

В общем случае лучше избегать вызова `EntityList` перед `EntityValue`, иначе придется выполнить два отдельных вызова. Это повлияет на производительность и на атомарность при работе с внешними базами данных.

Когда `EntityList` возвращает список объектов `Entity` со сложными первыми аргументами, они могут перестать существовать, если данные изменятся.

При работе с внешними базами данных можно столкнуться с такой, в которой для некоторых таблиц не установлен первичный ключ, о чём сообщит `EntityStore`. При этом некоторые функции, связанные с отдельными сущностями, будут отключены, хотя `EntityValue`, как правило, будет продолжать работать. Единственный способ обойти эту проблему — установить ограничение первичного ключа во внешней базе данных.

Ножкгош

Получим из *У*-базы список химических веществ, являющихся нуклеозидами:

```
In[=]:= EntityList[
  EntityClass["Chemical", "Nucleosides"]]
```

Out[=]=

{ 2'-deoxycytidine, 2'-deoxyuridine, thymidine, cytidine, uridine, adenosine, guanosine }

Получим список представленных в WL-базе рационов питания:

```
In[1]:= EntityList["SpeciesDiet"]
Out[1]= {algae, amphibians, aquatic crustaceans, aquatic or marine worms, birds, blood,
body fluids, bryophytes, carrion, cnidarians, detritus, dung, echinoderms,
eggs, eucalyptus leaves, fish, flowers, fruits, fungus, grasses, insects,
invertebrates, leaves, lichens, macroalgae, macrophytes, mammals,
marine invertebrates, microbes, molluscs, nectar, phytoplankton, pollen, reptiles,
roots and tubers, sap or other plant fluids, seagrasses, seeds, grains, and nuts, sponges,
terrestrial non-insect arthropods, terrestrial worms, wood, bark, or stems, zooplankton}
```

МпмУглмпржноожглглжэ

Используем `EntityList` со вторым аргументом, чтобы понять, упрощается ли запись сущности:

```
In[2]:= EntityList[
  FilteredEntityClass[药物 CHEMICALS,
    EntityFunction[c, c["Solubility"] < Quantity[.01, "Percent"]]]]
Out[2]= {N-pentane, 1,2-dimethylbenzene, 1,4-dimethylbenzene, guanine, uric acid}

In[3]:= EntityList[
  FilteredEntityClass[药物 CHEMICALS,
    EntityFunction[c, c["Solubility"] < Quantity[.01, "Percent"]]], False]
Out[3]= {guiN N-pentane, guiN 1,2-dimethylbenzene, guiN 1,4-dimethylbenzene, guiN guanine, guiN uric acid}
```

Сущность `guiN guanine` не эквивалентна `guanine CHEMICAL`, потому что их формы записи не совпадают:

```
In[4]:= guiN guanine === guanine CHEMICAL
Out[4]= False

guanine CHEMICAL // FullForm
guiN guanine // FullForm (*внешние слои не удаляются-получаются новые сущности*)

Out[5]//FullForm=
Entity["Chemical", "Guanine"]

Out[6]//FullForm=
Entity[FilteredEntityClass[EntityClass["Chemical", "Drugs"],
  EntityFunction[c, Less[c["Solubility"], Quantity[0.01, "Percent"]]]], "Guanine"]
```

Некоторые классы сущностей не могут быть упрощены:

```
In[1]:= el1 = EntityList[ExtendedEntityClass[{"chemical carcinogens CHEMICALS", "vol" \[Rule] EntityFunction[p, p["k nJYpk Yqq"] * p["k Yqqbcl qgw"]]]]
Out[1]= {ethylene oxide, 1,3-butadiene, nickel, vinyl chloride, benzene, chloromethyl methyl ether, bis(chloromethyl)ether, 2-naphthylamine, mustard gas, 4-aminodiphenyl, phenacetin, thiopeta, radon, cyclophosphamide, thorium(IV) oxide, diethylstilbestrol, azathioprine, chlorambucil, melphalan, aflatoxin, tamoxifen, cyclosporin A}
```



```
In[2]:= el2 = EntityList[ExtendedEntityClass[{"chemical carcinogens CHEMICALS", "vol" \[Rule] EntityFunction[p, p["k nJYpk Yqq"] * p["k Yqqbcl qgw"]]], False]
Out[2]= {ethylene oxide, 1,3-butadiene, nickel, vinyl chloride, benzene, chloromethyl methyl ether, bis(chloromethyl)ether, 2-naphthylamine, mustard gas, 4-aminodiphenyl, phenacetin, thiopeta, radon, cyclophosphamide, thorium(IV) oxide, diethylstilbestrol, azathioprine, chlorambucil, melphalan, aflatoxin, tamoxifen, cyclosporin A}
```

Проверим, являются ли элементы из el1 и el2, одинаковыми (на примере 1,3-butadiene):

```
In[3]:= el1[[2]] == el2[[2]]
Out[3]= True

In[4]:= el1[[2]] // FullForm
el1[[2]] // FullForm
Out[4]//FullForm=
Entity[ExtendedEntityClass[EntityClass["Chemical", "Carcinogens"], Rule["vol", EntityFunction[p, Times[p[EntityProperty["Chemical", "MolarMass"]], p[EntityProperty["Chemical", "MassDensity"]]]]]], "1,3Butadiene"]

Out[5]//FullForm=
Entity[ExtendedEntityClass[EntityClass["Chemical", "Carcinogens"], Rule["vol", EntityFunction[p, Times[p[EntityProperty["Chemical", "MolarMass"]], p[EntityProperty["Chemical", "MassDensity"]]]]]], "1,3Butadiene"]
```

[EntityList](#) можно использовать с [FilteredEntityClass](#):

Используя отфильтрованный класс молекул, получим список липидов с числом ароматических атомов больше 5:

```

EntityList[
  FilteredEntityClass[ lipids CHEMICALS ,
    EntityFunction[c, c[ Ypink Yrgg Ymk ans/r ] > 5] ][ { "Name", Ypink Yrgg Ymk ans/r } ] ]
Out[=]= {{menadione, 6}, {octanoic acid-7-13 C, 6}, {\alpha-tocopherol, 6}}

```

EntityList можно использовать с SampledEntityClass :

```

In[=]:= EntityList[SampledEntityClass["AnatomicalFunctionalConcept", 2]]
Out[=]= { abduction , abductive reasoning }

```

EntityList можно использовать с SortedEntityClass :

```

In[=]:= EntityList[
  SortedEntityClass["Dinosaur", EntityProperty["Dinosaur", "Weight"] \rightarrow "Descending", 3]]
Out[=]= { Amphicoelias , Paralititan , Argentinosaurus }

```

EntityList можно использовать с ExtendedEntityClass , однако получаемые сущности не упрощаются:

```

EntityList[
  ExtendedEntityClass["Dinosaur",
    "BMI" \rightarrow EntityFunction[d, d["Weight"] / d["Length"]^2]] // Short
Out[=]/.Short=>
{ Sauropsida , Abelisauria , Abelisauridae , Abelisauroidea ,
  <<1562>> , Sauropodomorpha , Theropoda , Thyreophora , Dinosauria }

```

При вложении ExtendedEntityClass в другие функции заголовки могут упроститься :

```

In[=]:= EntityList[
  SortedEntityClass[
    ExtendedEntityClass["Dinosaur",
      "BMI" \rightarrow EntityFunction[d, d["Weight"] / d["Length"]^2]], "BMI" \rightarrow "Descending", 3]]
Out[=]= { Liaoixornis , Juravenator , Archaeopteryx }

```

`EntityList` можно использовать с `CombinedEntityClass`, однако получаемые сущности не упрощаются:

```
In[]:= EntityList[CombinedEntityClass[
  EntityClass["Element", "Period1"], "Isotope", "Entity" → "Element"]]
Out[=]={{Hydrogen, Hydrogen1}, {Hydrogen, Hydrogen2}, {Hydrogen, Hydrogen3},
{Hydrogen, Hydrogen4}, {Hydrogen, Hydrogen5}, {Hydrogen, Hydrogen6},
{Hydrogen, Hydrogen7}, {Helium, Helium2}, {Helium, Helium3},
{Helium, Helium4}, {Helium, Helium5}, {Helium, Helium6}, {Helium, Helium7},
{Helium, Helium8}, {Helium, Helium9}, {Helium, Helium10}}
```

Список классов сущностей в типе сущностей `EntityclassList`

`EntityclassList["тип"]` предоставляет список классов сущностей для типа сущности "тип"

Берүкжөнүфө памзра

`EntityclassList` возвращает список объектов `EntityClass`.

Идентификатор "тип" — это строка, представляющая канонический тип сущности, связанный с конкретной сущностью.

Ножкгощ

Найдём какие классы сущностей составляют указанный тип сущностей:

```
In[]:= EntityclassList["MedicalTest"]
Out[=]={{NHANES Demographic}, {arterial blood gas}, {cardiac screens}, {CBC with differential},
{CBC without differential}, {comprehensive metabolic panel}, {basic metabolic panel},
{electrolyte profile}, {hemoglobin and hematocrit}, {iron test}, {lipid panel},
{liver function test}, {osmolality test}, {anemia screening}, {arsenic screening},
{diabetes screening}, {heavy metal screening}, {nutrient screening}, {phenol screening},
{PSA screening}, {toxoplasma gondii screening}, {tobacco screening},
{chlamydia screening}, {physical examination}, {hepatitis A screening},
{hepatitis B screening}, {hepatitis C screening}, {hepatitis screening}, {HIV testing},
{iodine screening}, {mercury screening}, {pregnancy test}, {medical tests}}
```

```
In[=]:= EntityclassList["AnatomicalStructure"]

Out[=]= {arteries, bilateral structures, blood vessels, body parts,
body segments, bones, cardiac chambers, cardiovascular organs,
cell types, cervical vertebrae, coccygeal vertebrae, digestive organs,
digits, endocrine organs, extremities, female genital organs, fingers,
glands, integumentary organs, internal organs, joints, lumbar vertebrae,
lymphatic organs, lymphatic vessels, lymph nodes, male genital organs,
muscles, muscular system organs, nerves, nervous system organs, organs,
organ systems, respiratory organs, ribs, sacral vertebrae, sensory organs,
skeletal system organs, spinal nerves, teeth, thoracic vertebrae, toes,
urinary organs, veins, vertebrae, vestigial structures, anatomical structures}
```

Сущность свойства EntityProperty

EntityProperty [*имя* *Yk c*]

представляет свойство,

идентифицированное *l Yk* для использования в **EntityValue**

представляет свойство,

EntityProperty [*аж* *Yqfynl Yk c*]

введенное вычисляемым классом сущности *аж*

представляет свойство,

EntityProperty [*имя* *Yk c*, *{os Yj_g → t Yj_g1, os Yj_g → t Yj_g2, ...}*]

представляет свойство,

измененное правилами квалификатора *os Yj_g → t Yj_g*

В грүкжүйфкә памз пра

При ссылке на свойство без квалификаторов применение **EntityProperty**, как правило, не обязательно.

common cold average temperature →

```
In[=]:= common cold DISEASE [ YtcpYec rck ncpYrspx ] ✓
```

Out[=]=

36.7661 °C

```
In[=]:= COVID-19 DISEASE [ bcd#gg1 ]
```

Out[=]=

a viral respiratory disease caused by a novel coronavirus
(SARS-CoV-2) first identified in Wuhan, Hubei Province, China

В **EntityProperty** [*имя*, *l Yk c*] *имя* может быть:

- либо встроенный типом сущности,
- либо типом, указанным в **EntityStore**, зарегистрированном с помощью **EntityRegister**;
- либо функции **ExtendedEntityClass** и **AggregatedEntityClass**, использование которых позволяет вводить новые свойства.

имя может иметь форму:

- "runc",
- "runc" → "afgb_x" → "afgb_y" → ...

EntityProperty [имя] возвращает сущность свойства указанного типа.

```
In[1]:= EntityProperty["Disease", "Definition"]
```

Out[1]=

bcd_{fggt}

Значения свойств для объекта **Entity** можно получить с помощью:

- или **Entity** [...][**EntityProperty** [...]];
- или **EntityProperty** [...][**Entity** [...]] .

```
In[2]:= Entity["Disease", "COVID19"]["Definition"] ===
```

```
Entity["Disease", "COVID19"][EntityProperty["Disease", "Definition"]]
```

Out[2]=

True

Свойства, получаемые для выражения **EntityProperty** при применении **Information**, различаются в зависимости от **EntityType**, связанного с выражением.

Амъкмдлыш номӯкгүц

Вызов сущностей с одним свойством за раз выполняется медленнее, чем вызов сущностей с одним списком свойств:

```
In[3]:= AbsoluteTiming[α-tocopherol CHEMICAL [ Ymk ans/r ] ]
```

Out[3]=

{1.54808, 81 atoms}

```
In[4]:= Quit
```

```
In[5]:= AbsoluteTiming[α-tocopherol CHEMICAL [ { Ymk ans/r, npZgYjf nZpbfgYrgn } ] ]
```

(*форма работает быстрее, поскольку свойства сущности запрашиваются одним вызовом*)

Out[5]=

{0.582416,
 { 81 atoms, {{0, sp²}, {0, sp²}, {C, sp³}, {C, sp²}, {C, sp³}, {C, sp³}, {C, sp³}, {C, sp²}, {C, sp³}, {C, sp²}, {C, sp³}, {C, sp³}, {C, sp²}, {C, sp³}, {C, sp³}, {C, sp³}, {C, sp³}, {C, sp³}, {C, sp³}}}}

```
Entity["Country", "Chile"] [ Table[EntityProperty["Country", "Population", {"Date" → DateObject[{year}]}], {year, Range[1900, 2010, 10]}] ] // AbsoluteTiming
```

Ymk ans/r npZgYjf nZpbfgYrgn

Сущности и их свойства кэшируются, поэтому последующие вызовы выполняются намного быстрее:

```
In[1]:= AbsoluteTiming[α-tocopherol CHEMICAL] [ {Yrnk anslr, npzgYjf wZpfgyrgt} ] ]
(*форма работает быстрее, поскольку свойства сущности запрашиваются одним вызовом*)

Out[1]= {0.0225325,
{81 atoms, {{0, sp2}, {0, sp2}, {C, sp3}, {C, sp3}}}]]

Entity["Country", "Chile"] [
Table[EntityProperty["Country", "Population", {"Date" → DateObject[{year}]}], {year, Range[1900, 2010, 10]}]] // AbsoluteTiming
```

In[2]:= Quit

Ножглощ

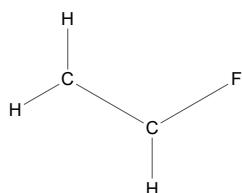
Определим сущность свойства типа «Chemical», представляющую структурный график, и найдём её значение для определённой молекулы:

```
In[3]:= EntityProperty["Chemical", "StructureGraph"]
```

```
Out[3]= qpsarspx epYnf
```

```
In[4]:= vinyl fluoride CHEMICAL [ qpsarspx epYnf ]
```

Out[4]=



МпмÜгллмпржножглглжэ

Использование доступных квалификаторов "Qualifiers" для определения значений свойств "QualifierValues":

Найдём доступные квалифиликаторы и их значения:

```
In[5]:= pcegnl Yj jnaYrgt gk Yec ["Qualifiers"]
```

Out[5]=

{View}

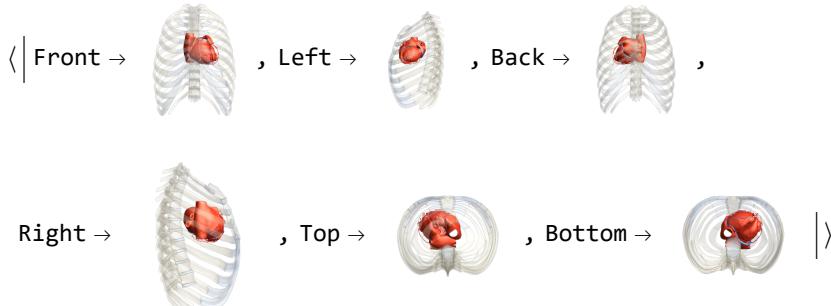
```
In[6]:= Lookup [ pcegnl Yj jnaYrgt gk Yec ["QualifierValues"], "View" ]
```

Out[6]=

{All, Back, Bottom, Front, Left, Right, Top}

```
In[1]:= EntityValue[heart ANATOMICAL STRUCTURE, EntityProperty["AnatomicalStructure", "RegionalLocationImage", {"View" → "All"}]]
```

Out[1]=



EntityProperty может иметь встроенные квалификаторы для более конкретного описания свойства:

```
In[2]:= population = Hold[ ]
```

(*квалификатор "Data" указывает конкретный год*)

Out[2]=

```
Hold[Spain [ nmnsjYrgn + Year: 1960 ]]
```

```
In[3]:= InputForm[population]
```

Out[3]//InputForm=

```
Hold[Entity["Country", "Spain"] [EntityProperty["Country", "Population", {"Date" → Date[
```

Список свойств сущностей для типа сущности
EntityProperties

EntityProperties[*нмс*]

предоставляет список свойств,

связанных с типом сущности *нмс*

В групккжүфкә памзра

EntityProperties[*нмс*] эквивалентно **EntityValue**[*нмс*, "Properties"]

```
In[4]:= EntityProperties["Disease"] === EntityValue["Disease", "Properties"]
```

Out[4]=

True

Ножкгощ

Найдём свойства, связанные с типом сущности "Protein":

```
In[=]:= EntityProperties["Protein"]
```

```
Out[=]=
```

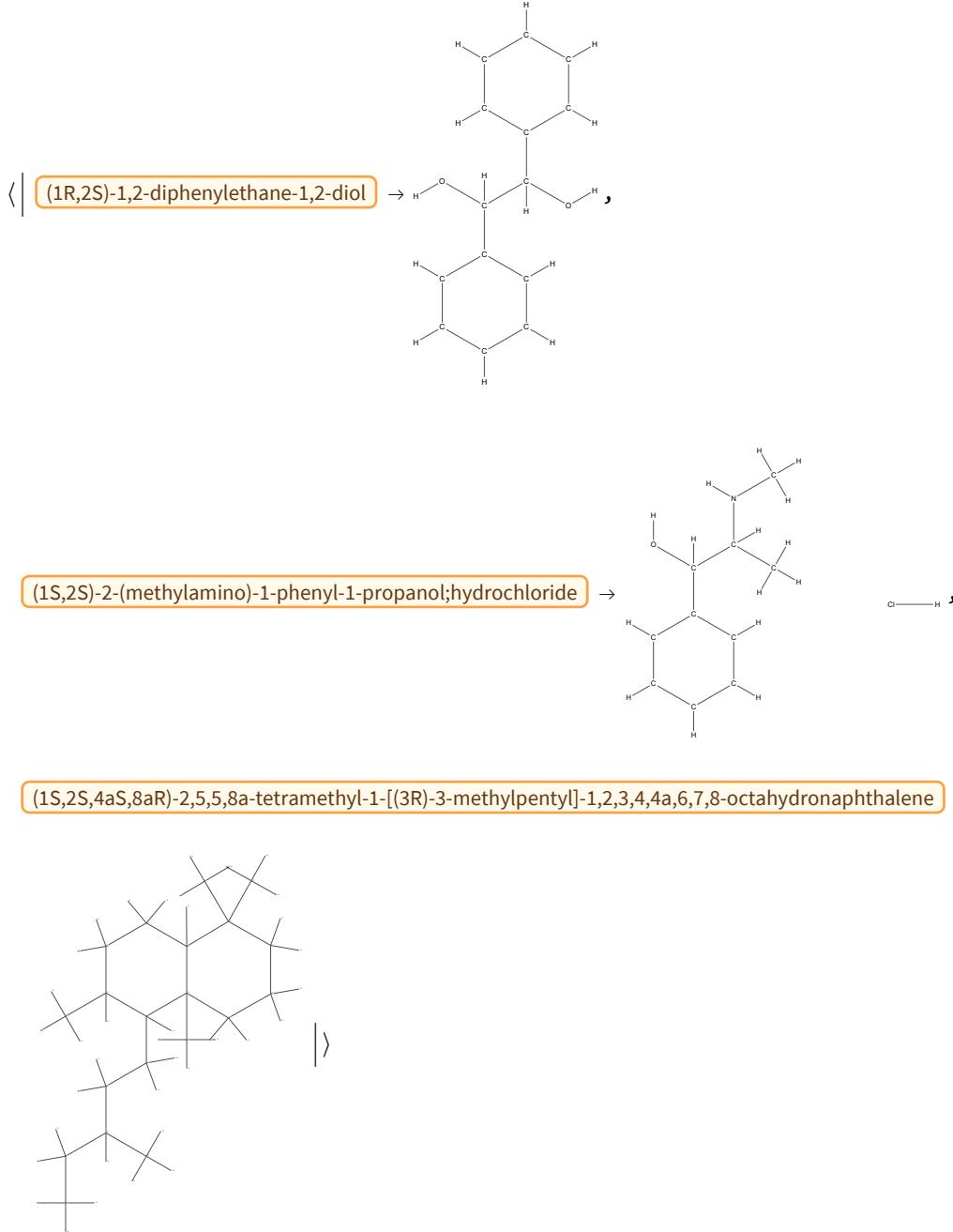
```
{Ybbggt Yj Yrnk rwncq, ZYjj Yl b qrgi bg'ep'k, ZgjnegaYj npnacqqcq, acjjsjYpank nnl cl rq,
afYg jYZcjq, bnk Yg Bq, bnk Yg nmaggnt q, bnk Yg q, cl rgwajYqqcq, cl rgwrwncjgr, ecl c,
ecl c B, euplrgt pYbgsq, k ck Zcpqfgnq, k njcasjYpdl argt q, k njcasjYpu cgef'r, l Yk c,
LA Graaccqgnl q, NB B jgr, npk YpnNB B, npk YpnNB SPJ, qcant bYpwqrsarspc pjcq,
qcoscl ac, pdpxl ac qcoscl ac jcl erf, qnYacdijg e bg'ep'k, pgZnt bg'ep'k, ugxcphyk c bg'ep'k}
```

Используя [EntityValue](#), чтобы получить значение определенной сущности для заданного [EntityProperty](#):

```

entities = SampledEntityClass["Chemical", 3];
(*выберем три образца заданного типа сущности*)
EntityValue[entities,
  EntityProperty["Chemical", "StructureGraph"], "EntityAssociation"]

```

Out[*o*] =

Класс свойств сущностей для типа сущности
EntityPropertyClass

EntityPropertyClass [*инспир/Ук с*]представляет класс свойств, идентифицированных *н/Ук с*

В групкожъфю памз пра

Форма стандартного имени класса свойств ***наименование***:

- или строка;
- или список со строками и правилами .

В **EntityPropertyClass** [*имя*, *наименование*] ***имя*** может быть:

- либо встроенный типом сущности,
- либо типом, указанным в **EntityStore**, зарегистрированном с помощью **EntityRegister**.

имя может иметь форму:

- “***имя***”,
- “***имя***”→”***атрибут***”→”***атрибут***”→....

Свойства, получаемые для выражения **EntityPropertyClass** при применении **Information**, различаются в зависимости от **EntityType**, связанного с выражением.

Нажмите

Определим свойства масс, связанные с физическими системами и найдём список объектов свойств, включенных в класс свойств:

```
In[1]:= EntityPropertyClass["PhysicalSystem", "MassProperties"]
```

```
Out[1]=
```

имя

```
In[2]:= EntityValue[EntityPropertyClass["PhysicalSystem", "MassProperties"], "Properties"]
```

```
Out[2]=
```

{***имя***, ***имя***, ***имя***, ***имя***, ***имя***, ***имя***}

Найдём список свойств, включенных в класс свойств “ToxicityProperties”:

```
In[3]:= EntityPropertyClass["Chemical", "ToxicityProperties"]
```

```
Out[3]=
```

имя

```
In[4]:= EntityValue[имя, "Properties"]
```

```
Out[4]=
```

{***имя***, ***имя***, ***имя***}

Значения свойств сущностей
[EntityValue](#)

EntityValue[*имя*]

предоставляет значение свойства *имя* для сущности *имя*

EntityValue[{*имя*, *имя*, ...}, *имя*]

предоставляет список значений свойства *имя* для каждой сущности *имя*

EntityValue[*имя*]

предоставляет

список значений свойства *имя* для всех сущностей класса *имя*

EntityValue[*имя*{*имя*, ...}]

представляет список значений свойства *пропсри* для сущности *cl rgw*
EntityValue [*cl rgw* {*пропсри* | *пропсри*, ...}] предstawляет
 список значений свойств *пропсри* для каждой из сущностей, представленных *cl rgw*

Верхний фокус памзы

Для поиска названий сущностей и их свойств можно использовать естественный язык с помощью **ctrl =**:

ethylene oxide atom count →
In[1]:= **ethylene oxide** CHEMICAL [*Ymk ans/r*]

Out[1]=
7 atoms

cl rgv [“*пропсри*”] эквивалентно **EntityValue** [*cl rgv* {*пропсри*}], когда *cl rgv* является допустимой сущностью или конструкцией класса сущности.

пропсри [*cl rgv*] эквивалентно **EntityValue** [*cl rgv* {*пропсри*}], когда свойство *пропсри* является допустимым функциями **EntityProperty**, **EntityPropertyClass** или **EntityFunction**.

пропсри указывается:

- либо стандартной строкой наименования свойства,
- либо **EntityProperty** [*rnsc*, *I Yk c*, *mt/gt q*], где опции *mt/gt q* могут включать квалификаторы свойств.

В **EntityValue** [*cl rgv* {*пропсри*}] можно использовать следующие формы для *cl rgw*

все сущности указанного типа	<i>type</i>		
объекты	Entity	EntityClass	
список сущностей	{ Entity [...], Entity [...], ...}		
функции	FilteredEntityClass	SortedEntityClass	SampledEntity-
Class			Com-
binedEntityClass	ExtendedEntityClass	AggregatedEntityClass	Com-

В **EntityValue** [*cl rgv* {*пропсри*}] можно использовать следующие формы для *пропсри*

“ <i>I Yk c</i> ”	именованное свойство для указанного типа сущности
EntityProperty [<i>rnsc</i> , “ <i>I Yk c</i> ”]	полностью определённое свойство
EntityFunction [...]	вычисляемое свойство
EntityPropertyClass [...]	именованный класс свойств
{ EntityProperty [...], ...}	список сущностей свойств

Некоторые свойства доступны в общем виде для типа сущности. Их можно использовать в качестве аргумента для **Entity**[“*rnsc*”] или просто “*rnsc*”:

“Properties”	список доступных свойств
“PropertyCanonicalNames”	стандартные названия доступных свойств
“SampleEntities”	пример списка доступных сущностей (обычно длиной 10)
“SampleEntityClasses”	пример списка доступных классов сущностей (обычно длиной 10)
“EntityCount”	количество доступных сущностей
“Entities”	список доступных сущностей
“EntityCanonicalNames”	стандартные имена доступных сущностей

“EntityClasses”	список доступных классов сущностей
“EntityClassCanonicalNames”	стандартные имена доступных классов сущностей
“PropertyClasses”	список доступных классов свойств
“PropertyClassCanonicalNames”	стандартные названия доступных классов свойств
“PropertyCount”	количество доступных свойств

In[1]:= EntityValue["MeasurementDevice", "Properties"]

Out[1]= {Yjrcd Yrc l Yk cq, bcqpgnrgt, cl rgwajYqqcq, cl rgwrwnc jgr, gqrYl ac nq, k cYqspcb osYl rggq, l Yk c, l npk Yj prcaggnt, l npk Yj pYl ec, qncagtg gqrYl acq}

In[2]:= EntityValue["Disease", "EntityClasses"]

Out[2]= {contagious diseases, coronavirus diseases, diseases}

EntityValue кэширует результаты между сеансами, когда это возможно.

Все кэшируемые значения свойств для сущностей указанного “rnc” можно извлечь с помощью EntityPrefetch [“rnc”]. При этом они будут сохранены в системе.

In[3]:= AbsoluteTiming[EntityValue["mustard gas", "Chemical"]]

Out[3]= {0.334216, C4H8Cl2S}

In[4]:= AbsoluteTiming[EntityValue["mustard gas", "dphk sjY qrgt e"]]

Out[4]= {0.0149353, C4H8Cl2S}

In[5]:= Quit

In[6]:= AbsoluteTiming[EntityValue["mustard gas", "Chemical"]]

Out[6]= {0.303107, C4H8Cl2S}

EntityValue извлекает связанные базы знаний в фоновом режиме на основе ранее запрошенных данных.

EntityValue [] возвращает список всех доступных типов сущностей (лк/Босннцпч лмпргз жк ИЛЮЖЩЖПУЛЖ).

EntityValue [] возвращает список всех доступных типов сущностей (лк/Босннцпч лмпргз жк ИЛЮЖЩЖПУЛЖ).

Амжмдлигк номүкгкщ

В общем случае лучше избегать вызова EntityList перед EntityValue, иначе придется выполнить два отдельных вызова. Это повлияет на производительность и на атомарность при работе с внешними базами данных.

Когда EntityList возвращает список объектов Entity со сложными первыми аргументами, они могут перестать существовать, если данные изменятся.

При работе с внешними базами данных можно столкнуться с такой, в которой для некоторых таблиц не установлен первичный ключ, о чём сообщит EntityStore . При этом некоторые функции, связанные с отдельными сущностями, будут отключены, хотя EntityValue , как правило, будет продолжать работать. Единственный способ обойти эту проблему – установить ограничение первичного ключа во внешней базе данных.

Ножкгощ

Получим свойство «Описание» в объекте «Бактериальная пятнистость листьев»:

```
In[1]:= bacterial leaf spot PLANT DISEASE [ bcqapgrgt ]
Out[1]=
a bacterial disease most commonly caused by Pseudomonas
or Xanthomonas bacteria, which can cause dark brown to black
spots with chlorotic halos on leaves that can turn into shot holes
```

Узнаем, сколько сущностей и сколько их свойств доступно в типе “Chemical”:

```
In[2]:= EntityValue["Chemical", "EntityCount"]
```

```
Out[2]=
44 096
```

```
In[3]:= EntityValue["Chemical", "PropertyCount"]
```

```
Out[3]=
119
```

Извлечём свойство “Молярная масса” из EntityClass “Наноматериалы”:

```
nanomaterials CHEMICALS [ k njYpk Yqq ]
(*молярная масса для всех сущностей, входящих в состав класса*)
```

```
Out[4]=
{ 58.319 g/mol, 126.689 g/mol, 149.625 g/mol, 150.802 g/mol, 153.22 g/mol,
  176.892 g/mol, 181.51 g/mol, 215.095 g/mol, 227.92 g/mol, 232.77 g/mol,
  234.38 g/mol, 239.23 g/mol, 593.61 g/mol, 737.93 g/mol, 1111.45 g/mol }
```

Памзпраў

Извлечём несколько свойств одновременно:

Найдём свойства “Масса” и “Количество клеток” в объекте «Головной мозг»:

```
In[5]:= EntityValue[ brain ANATOMICAL STRUCTURE, { k Yqq, acjj ans/r } ]
Out[5]=
{ 1.4 × 103 g, Interval[ { 1.48457 × 1011, 1.76152 × 1011 } ] }
```

Извлечём свойство из нескольких сущностей:

```
In[6]:= EntityValue[
  {Entity["Element", "Oxygen"], Entity["Element", "Beryllium"]}, "AtomicNumber"]
Out[6]=
{8, 4}
```

Применим [EntityFunction](#) к одной сущности:

```
In[]:= EntityValue[iridium ELEMENT ..., EntityFunction[i, i["BoilingPoint"] - i["MeltingPoint"]]]
```

Out[]= 1962. °C

Определим значение свойства из сложного класса сущности:

Найдём среднюю атомную массу для элементов III периода:

```
In[]:= EntityValue[AggregatedEntityClass[EntityClass["Element", "Period3"], "AtomicMass" → Mean], "AtomicMass"]
```

Out[]= {30.10 u}

Извлечём свойства из [EntityInstance](#):

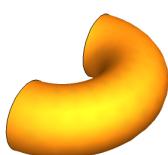
```
EntityValue[EntityInstance[half-torus SOLID ..., {a → 1, c → 2}], "SurfaceArea"]
```

(*найдём площадь поверхности объекта*)

Out[]= $4\pi(2 + \pi)$

```
In[]:= half-torus SOLID [g Yec]
```

Out[]= $a = \frac{1}{2}, c = 1$



Для извлечения значений свойства на определённую дату в аргументе используют [Dated](#):

Найдём численность населения Испании в указанные годы:

```
In[]:= EntityValue[Spain COUNTRY, Table[Dated["Population", i], {i, 1940, 2000, 10}]]
```

Out[]= $\left\{ 2.5878 \times 10^7 \text{ people}, 2.8009 \times 10^7 \text{ people}, 3.07769 \times 10^7 \text{ people}, 33\,638\,249 \text{ people}, 37\,342\,360 \text{ people}, 38\,874\,178 \text{ people}, 40\,633\,717 \text{ people} \right\}$

Для указания значения свойства на определённую дату используют квалификатор "Date":

Найдём численность населения Испании в указанные годы:

```
In[=]:= EntityValue[Norway COUNTRY,
EntityProperty["Country", "Population", {"Date" → DateObject[{2005}]}]]
```

Out[=]=
4 606 746 people

Получим список образцов сущностей для типа сущности с помощью “SampleEntities”:

```
In[=]:= EntityValue["MedicalTest", "SampleEntities"]
```

Out[=]=
{serum iron, serum rubella antibody test, serum LDL cholesterol,
serum retinyl palmitate, serum retinyl stearate, serum selenium,
serum total protein, systolic blood pressure, urine iodine, urine lead}

Мпмўгллмпржножжглглж

Результат в виде ассоциации можно получить с помощью различных модификаторов:

Получим ассоциацию для списка сущностей:

```
In[=]:= EntityValue[EntityValue[sensory organs ANATOMICAL STRUCTURES, "Entities"],
"EntityAssociation"]
```

Out[=]=
(| ear → {auditory perception}, eye → {visual perception},
nose → {olfaction}, skin → {somatosensation}, tongue → {gustatory perception} |)

Результат в виде ассоциации для всех свойств объекта :

```
EntityValue[Russian Blue CAT BREED , "PropertyAssociation"]
```

(*все свойства кошки породы "Русская голубая")

Out[**1**]=

```
<| Yjrcp Yrc l Yk cq → {Archangel Blue, Archangel Cat}, anjnpr → {blue, silver},  

bcqapprgnt → {Medium-sized cat whose short coat is always silver-tipped blue,  

Fur is dense and stands away from its body, appearing to shimmer in movement,  

Ears are proportionally large and flared, Appears to smile,  

Fine boned, with long legs}, cl rgwajYqqcq → {},  

cl rgwrwncjgr → {cat breed}, epmkk gte → {minimal}, fvgpjcl erf → {short},  

fgrnpv → {Breed brought from the area of Arkhangel in northwestern Russia  

to England in the 1860s, and shown at the Crystal Palace in 1875,  

Reduced in numbers after World War II when the breed was revived by  

mixing with Siamese or British Shorthairs with blue coloration},
```



gk Yec → , **k Yvgksk ucgef** → 5.4 kg ,

k cYl ucgef → 4.0 kg ,

k gk sk ucgef → 2.5 kg ,

l Yk c → Russian Blue,

mpqg → {Russia},

anYrnYrrcp → {marbled},

pccanlgccbzw → {{The Governing Council of the Cat Fancy, GCCF},
{The International Cat Association, TICA}, {Fédération Internationale Féline,
FIFe}, {The Cat Fanciers' Association, Inc., CFA}},
qfcbbge → {minimal}, **qgxc** → {medium}, **rck ncp/k clr** → {active, affectionate,
agile, calm, gentle, graceful, intelligent, loyal, playful, shy},
rwnctndZpxcb → {natural}, **ucgef** → (2.4 to 5.6) kg |>

Ассоциация для списка свойств:

```
In[2]:= EntityValue[Entity["Chemical", "Water"],  

{"Name", "HillFormulaString", "LewisDotStructureDiagram"}, "PropertyAssociation"]
```

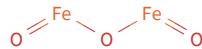
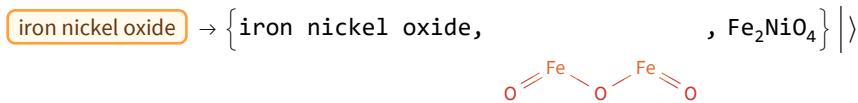
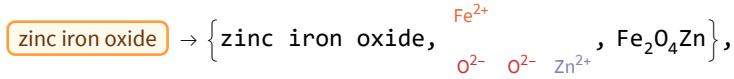
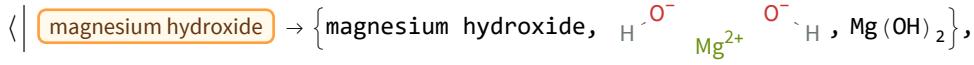
Out[**2**]=

```
<| Name → water, HillFormulaString → H2O, LewisDotStructureDiagram → H-O-H |>
```

Ассоциация для списков сущностей и/или свойств:

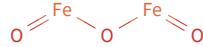
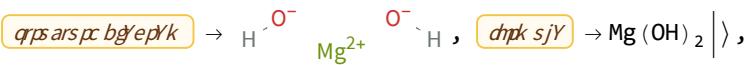
In[1]:= EntityValue[{magnesium hydroxide CHEMICAL, zinc iron oxide CHEMICAL, iron nickel oxide CHEMICAL}, {"EntityAssociation"}]

Out[1]=



In[2]:= EntityValue[{magnesium hydroxide CHEMICAL, zinc iron oxide CHEMICAL, iron nickel oxide CHEMICAL}, {"PropertyAssociation"}]

Out[2]=



Использование доступных квалификаторов "Qualifiers" для определения значений свойств "QualifierValues":

Найдём доступные квалифиликаторы:

In[3]:= EntityValue[EntityProperty["Country", "AgricultureProduction"], "Qualifiers"]

Out[3]=

{AgricultureProduct, Date}

Найдём значения для найденных квалифиликаторов:

In[4]:= EntityValue[EntityProperty["Country", "AgricultureProduction"], "QualifierValues"]

```
{"AgricultureProduct" →
 {"Apple", "AquaticPlant", "Banana", "Barley", "Bean", "Beer", "BovineMeat",
 "Cassava", "Cephalopod", "Cheese", "CoconutOil", "Coffee", "CottonseedOil",
 "Cream", "Crustacean", "Date", "DemersalFish", "EdibleOffal", "Egg",
 "FreshwaterFish", "Grape", "Grapefruit", "GroundnutOil", "Honey", "LemonAndLime",
 "Maize", "Milk", "Millet", "MuttonAndGoatMeat", "Nuts", "Oats", "Olive",
 "OliveOil", "Onion", "OrangeAndMandarine", "PalmkernelOil", "PalmOil", "Pea",
 "PelagicFish", "Pepper", "Pigmeat", "Pimento", "Pineapple", "Plantain",
 "Potato", "PoultryMeat", "RapeAndMustardOil", "RiceMilledEquivalent",
 "RicePaddyEquivalent", "Rye", "SesameSeedOil", "Soybean", "SoybeanOil",
 "Spices", "SugarAndSweeteners", "SunflowerseedOil", "SweetPotato",
 "Tea", "Tomato", "Wheat", "Whey", "Wine", "Yam"}, "Date" → {}}
```

Используя значения квалификаторов, определим значение для заданного свойства:

```
N@EntityValue[Entity["Country", "Russia"],
 EntityProperty["Country", "AgricultureProduction",
 {"AgricultureProduct" → "Apple"}]] (*производство яблок в России*)
```

Out[*]=

$2.2162 \times 10^6 \text{ t/yr}$

In[*]:= UnitConvert[%, "Kilograms" / "Years"]

Out[*]=

$2.2162 \times 10^9 \text{ kg/yr}$

Определим значение для свойства с помощью **ctrl =**:

In[*]:=  // N

Out[*]=

$2.2162 \times 10^6 \text{ t/yr}$

Когда результаты **EntityValue** содержат другие сущности, их можно вернуть в **EntityValue**:

In[*]:= EntityValue[]

Out[*]=

$\left\{ \left\{ \text{lithium-6}, \text{helium-4} \right\}, \left\{ \text{beryllium-10}, \text{beryllium-11}, \text{beryllium-9}, \text{beryllium-8}, \text{helium-6}, \text{lithium-8}, \text{lithium-9} \right\} \right\}$

```
In[6]:= EntityValue[%, {kηΥppΥbgΥarg gw, dΥj bcaΥwnpmbs ard}]

Out[6]= {{{{0 Bq/mol, {lithium-6 → 100%}}, {0 Bq/mol, {helium-4 → 100%}}}, {{8.77×109 Bq/mol, {boron-10 → 100.%}}, {{3.023×1022 Bq/mol, {boron-11 → 97.%, lithium-7 → 2.9%}}}, {{0 Bq/mol, {beryllium-9 → 100%}}, {{5.10×1039 Bq/mol, {helium-4 → 100.%}}}, {{5.174×1023 Bq/mol, {lithium-6 → 1.0×102%, helium-4 → 0.00028%}}, {{4.970×1023 Bq/mol, {helium-4 → 100.%}}, {{2.341×1024 Bq/mol, {helium-4 → 51.%, beryllium-9 → 49.%}}}}}}
```

Полученные данные временного ряда обычно можно напрямую передать в [DateListPlot](#):

```
In[7]:= DateListPlot[

FrameLabel → Automatic, Joined → True]

Out[7]=
```

Алнмк мөргкүлүг т слифжвкэ ойүмрүүлпсч лмпрэкж

связанных с [Entity](#)

Вычислительная функция, которая используется в функциях,
[EntityFunction](#)

[EntityFunction](#) [*v*, *Znbi*]

функция с одним формальным параметром *v*

[EntityFunction](#) [{*v₁*, *v₂*, ...}, *Znbi*]

[EntityFunction](#) со списком формальных параметров

Вгрүүкжүүлжээ замз пра

[EntityFunction](#) [*v*, *Znbi*] может использоваться непосредственно в [EntityValue](#).

обычно привязан к одной сущности за раз.

Иключение составляет использование [EntityFunction](#) в [AggregatedEntityClass](#), где формальный параметр будет привязан к каждой группе сущностей.

Следующие функции агрегации возможны при использовании [AggregatedEntityClass](#) для сущностей, поддерживаемых реляционной базой данных:

Min	минимум значений
Max	максимум значений
Length	количество значений
Total	сумма значений
Mean	среднее значение
StandardDeviation	стандартное отклонение выборки
Variance	дисперсия выборки

[EntityFunction](#) [{*aca,def1*}, *Znvi*] может быть использован в [CombinedEntityClass](#).

При этом *jcd* привязан к каждой [Entity](#) в первом аргументе, а *defr* — к каждой [Entity](#) во втором аргументе.

Только следующие формы [EntityFunction](#) могут быть скомпилированы в [SQL](#) при работе с сущностями, зарегистрированными в [EntityStore](#), поддерживаемых реляционной базой данных :

$a + b$	сложение чисел и значениями дат и времени
$a - b$	разница между числами или датами
$a \cdot b$	произведение чисел
a/b	отношение чисел
a^b	возвведение в степень чисел

Quotient[a,b]	целочисленное деление
Mod[a,b]	остаток от деления целых чисел
Sqrt[a]	квадратный корень
Exp[a]	экспонента
Log, Log10, Log2	логарифмы с разными основаниями
Abs[a]	модуль
OddQ[a], EvenQ[a]	проверка четности целых чисел

$a == b, a > b, a \neq b, \dots$	сравнение чисел, дат, строк
$a \&\& b, a b, !a, \dots$	Boolean алгебра
If [a,b,c]	“глж” для утверждения
Which[...]	заявление
MissingQ[a]	проверка, отсутствует ли элемент
Length[list]==0	проверка, является ли список пустым
OrderedQ[a,b]	проверка алфавитного порядка строк
MemberQ[<i>gr</i>,a]	проверка, находится ли элемент в списке или сущность в классе

Sin, Cos, Tan, ...	тригонометрические функции
ArcSin, ArcCos, ArcTan, ...	обратные тригонометрические функции
Sinh, Cosh, Tanh, ...	гиперболические функции
ArcSinh, ArcCosh, ArcTanh	обратные гиперболические функции

Round, Floor, Ceiling	функции округления
Min, Max	минимум и максимум (вне агрегаций)
N[a]	преобразовать целое число в действительное
IntegerPart[a]	извлечь целую часть действительного числа

Now, Today	получение текущей даты
AllTrue, AnyTrue	Булевы операции над списками

SelectFirst[<i>list</i> ,Not@*MissingQ] BitNot, BitAnd, BitOr, BitShiftLeft,...	выбор первого не пропущенного значения в списке побитовые операции
StringJoin[a,b,...] StringMatchQ[...],StringStartsQ[...],... StringTake, StringDrop	конкатенация строк (объединение нескольких строк в одну) основные операции сопоставления строк извлечь часть строки (включая поддержку UpTo)
ToString[a] FromDigits[a] Boole[a] Interpreter["StructuredDate"][[a], Interpreter["StructuredTime"][[a], Interpreter["StructuredDateTime"][[a] UnixTime[a] FromUnixTime[a] JulianDate[a] FromJulianDate[a] DateValue[a,"SecondsFromMidnight"] DateValue[UnixTime[a],"TimeObject"]	преобразовать число или дату в строку преобразовать строку в целое число преобразовать логическое значение в 0 или 1 преобразовать строку в объект даты или времени преобразовать объект даты со временем в число преобразовать число в объект даты со временем преобразовать объект даты в число преобразовать число в объект даты преобразовать объект времени в число преобразовать число в объект времени

При работе с сущностями, поддерживаемыми реляционной базой данных, функции могут иметь ограниченную функциональность (по сравнению с их аналогами в WL).

Например, в SQL отсутствует поддержка комплексных чисел. Поэтому домены числовых функций ограничены областями, где они имеют реальные значения. Как правило, недоступна символическая оценка, Семантика отслеживания точности также может различаться.

Ножкощ

Рассчитаем отношение между атомной массой и атомным номером для всех благородных газов:

```
In[]:= EntityValue[EntityClass["Element", "NobleGas"],  
EntityFunction[e, N[e["AtomicMass"]]/e["AtomicNumber"]]]  
Out[=] = {2.0013 u, 2.01797 u, 2.21944 u, 2.32772 u, 2.43135 u, 2.5814 u, 2.49153 u}
```

Применим EntityFunction непосредственно к одной сущности, например EntityProperty :

Используя UJ-базу "MedicalTest", рассчитаем среднее значение полного сывороточного холестерола на одного обследуемого:

```
In[]:= EntityFunction[c, c[EntityProperty["MedicalTest", "Median"]] /  
c[EntityProperty["MedicalTest", "SamplePopulationValue"]]] [  
serum total cholesterol MEDICAL TEST ]  
Out[=] = 0.00610611 mg / (person dL)
```

Используем EntityFunction как условие для FilteredEntityClass :

Используя UJ-базу "Element", найдём все элементы с заданным атомным радиусом:

```
In[1]:= EntityList[FilteredEntityClass["Element",
  EntityFunction[c, c["AtomicRadius"] < Quantity[100, "Picometers"]]]]

Out[1]= {hydrogen, helium, boron, carbon, nitrogen, oxygen, fluorine,
neon, phosphorus, sulfur, chlorine, argon, bromine, krypton}
```

Имена сущностей

[CanonicalName](#)

[CommonName](#)

[CanonicalName](#) [*c/rги*] предоставляет каноническое имя для сущности, указанной с помощью *c/rгw*.
[CanonicalName](#) [{*c/rги* *и* *c/rги*}] предоставляет канонические имена для сущностей из списка: от сущности *c/rги* до сущности *c/rги*.

В грифке памз пра

Для [CanonicalName](#) [*c/rги*] сущность может быть выражением [Entity](#), [EntityClass](#), [EntityProperty](#) или [EntityProperty-Class](#).

[CanonicalName](#) [*c/rги*] извлекает *aYl mгYjl Yk* с из [Entity](#) [*rwnc,aYl mгYjl Yk c*].

[CommonName](#) [*c/rги*] предоставляет общеупотребительное имя для сущности, указанной с помощью *c/rгw*.
[CommonName](#) [{*c/rги* *и* *c/rги*}] предоставляет общеупотребительное имена для сущностей из списка: от сущности *c/rги* до сущности *c/rги*.

В грифке памз пра

Для [CommonName](#) [*c/rги*] сущность может быть выражением [Entity](#), [EntityClass](#), [EntityProperty](#) или [EntityProperty-Class](#).

[CommonName](#) [*c/rги*] ищет общее имя сущности в онлайн *W*-базе знаний Wolfram или в системном кэше.

Ножкгощ

Получим общее и каноническое имена сущности:

```
In[2]:= CommonName[Entity["Isotope", "Lithium11"]]

Out[2]= lithium-11

In[3]:= lithium-11 ISOTYPE // CanonicalName

Out[3]= Lithium11
```

Некоторые общие и канонические имена существуют более чем одном типе сущностей, имея различную структуру в разных доменах:

cholesterol →

cholesterol CHEMICAL

In[1]:= cholesterol NUTRIENT

serum total cholesterol MEDICAL TEST

In[2]:= {CanonicalName[serum total cholesterol MEDICAL TEST CommonName[serum total cholesterol MEDICAL TEST

Out[2]= {SerumTotalCholesterol, serum total cholesterol}

In[3]:= {CanonicalName[cholesterol CHEMICAL

Out[3]= {Cholesterol, cholesterol}

Каноническое и общее имена класса сущности:

In[4]:= {CanonicalName[indicators CHEMICALS], CommonName[indicators CHEMICALS]}

Out[4]= {Indicators, indicators}

In[5]:= {CanonicalName[elements ELEMENTS], CommonName[elements ELEMENTS]}

Out[5]= {All, elements}

Общие имена классов сущностей в списке:

In[6]:= CommonName[{severe acute respiratory syndrome DISEASE,
common cold DISEASE, Middle East respiratory syndrome DISEASE, COVID-19 DISEASE}]

Out[6]= {severe acute respiratory syndrome,
common cold, Middle East respiratory syndrome, COVID-19}

Каноническое и общее имена свойства сущности:

In[7]:= {CanonicalName[Kbgrpgzsgt njm], CommonName[Kbgrpgzsgt njm]}

Out[7]= {BMIDistributionPlot, BMI distribution plot}

Каноническое и общее имена свойств типа сущности:

```
In[]:= CanonicalName[Take[EntityProperties["Gene"], 5]]
Out[=] = {AlternateNames, BiologicalProcesses, CellularComponents, Context, EntityClasses}

In[]:= CommonName[Take[EntityProperties["Gene"], 5]]
Out[=] = {alternate names, biological processes names,
cellular components names, context, entity classes}
```

Каноническое имя неявно определенного класса сущности:

```
In[]:= CanonicalName[
EntityClass["Isotope", {EntityProperty["Isotope", "HalfLife"] → TakeLargest[5]}]]
Out[=] = {fYjdjgt → TakeLargest[5]}
```

Общее имя неявно определенного класса сущности обычно не определено:

```
In[]:= CommonName[
EntityClass["Isotope", {EntityProperty["Isotope", "HalfLife"] → TakeLargest[5]}]]
Out[=] = Missing[NotAvailable]
```

Копии сущностей

EntityCopies

`EntityCopies[c/r/gw/l]` предоставляет *l* –копий сущности,
где *l* – любое положительное целое число

В гру́жка́ фжэ памз пра

`EntityCopies` позволяет выполнять операции с набором из *l*-сущностей, например, находить общее значение свойства.

В `EntityCopies[c/r/gw/l][пртсрц]` физическая величина, связанная со свойством *пртсрц* может быть:
- экстенсивной, и тогда результатом является общее значения для всего числа указанных копий;
- интенсивной, и тогда в качестве результата предоставляется список из *l*-повторяющихся значений свойства.

```
In[]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 3][k ccf tnjsk c]
Out[=] = 1.69937 × 106 mm3

In[]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 3][qfYnc]
Out[=] = {{hollow irregular ovoid}, {hollow irregular ovoid}, {hollow irregular ovoid}}
```

Количество копий **I** может быть любым положительным целым числом.

Для экстенсивных свойств при отсутствии копий возвращается **0**; для интенсивных - {}.

```
In[=]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 0] [k cqd tnjsk c]
```

Out[=]=

0

```
In[=]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 0] [qfYnc]
```

Out[=]=

{}

При указании в выражении **EntityCopies** одной копии сущность не формируется. Операции над таким выражением эквивалентны операциям над обычным **Entity**.

```
In[=]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 1]
```

Out[=]=

EntityCopies[stomach, 1]

```
In[=]:= EntityCopies[stomach ANATOMICAL STRUCTURE, 1] [k cqd tnjsk c] ==
```

stomach ANATOMICAL STRUCTURE [k cqd tnjsk c]

Out[=]=

True

Указание недействительного количества сущностей возвращает соответствующую форму **Missing**:

```
In[=]:= EntityCopies[stomach ANATOMICAL STRUCTURE, -1] [k cqd tnjsk c]
```

Out[=]=

Missing[InvalidEntityCopiesCount, -1]

Результаты **EntityCopies** могут возвращаться в единицах, отличных от аналогичных величин в системе "СИ":

```
In[=]:= EntityCopies[electron PARTICLE, 10] ["Mass"]
```

Out[=]=

5109.989461 keV/c²

```
In[=]:= UnitConvert[Quantity[10, "ElectronMass"], "SI"]
```

Out[=]=

0.00548579910 u

```
In[=]:= % == %%
```

Out[=]=

True

EntityGroup из / копий сущности эквивалентна EntityCopies[*c1 rgw1*]:

```
In[1]:= EntityValue[EntityGroup[Table[parathyroid gland ANATOMICAL STRUCTURE, {2}]], "Mass"] ===
EntityValue[EntityCopies[parathyroid gland ANATOMICAL STRUCTURE, 2], "Mass"]

Out[1]= True
```

Ножкгош

Найдём возможный объём трёх почек:

```
In[2]:= EntityCopies[kidney ANATOMICAL STRUCTURE, 3][trjisk c]

Out[2]= (3.0×102 to 1.01×103) mL
```

МплУгллмпржножкглглжэ

Выражения EntityCopies можно использовать вместе с обычными сущностями в списках:

```
In[3]:= EntityValue[{ovary ANATOMICAL STRUCTURE,
EntityCopies[testis ANATOMICAL STRUCTURE, 2],
pancreas ANATOMICAL STRUCTURE,
pineal body ANATOMICAL STRUCTURE}, k Yqq]

Out[3]= {11. g, (40. to 60.) g, (56. to 1.1×102) g, 0.10 g}
```

Использование EntityCopies для рассчёта количества используемых сущностей

Вычислим , какое количество сердец может соответствовать найденной массе:

```
In[4]:= EntityCopies[heart ANATOMICAL STRUCTURE, 2][k Yqq]

Out[4]= (4.8×102 to 6.5×102) g
```

```
In[5]:= heart ANATOMICAL STRUCTURE [k Yqq]

Out[5]= (2.4×102 to 3.3×102) g
```

```
In[6]:= EntityCopies[heart ANATOMICAL STRUCTURE, 2][k Yqq] / heart ANATOMICAL STRUCTURE [k Yqq]

Out[6]= Interval[{1.4, 2.8}]
```

Неэкстенсивные значения свойств **EntityCopies** повторяют свои значения для каждой копии:

Найдём молярную радиоактивность двух отдельных изотопов:

```
In[1]:= EntityCopies[lithium-11 ISOTOPE, 2][k ηΥρρΥbgνYarggw]
Out[1]= {4.86 × 1025 Bq/mol, 4.86 × 1025 Bq/mol}
```

Вычисленные значения нескольких копий сущности могут возвращать комбинацию общего значения и списка значений для свойств разного типа.

```
In[2]:= EntityCopies[serum C-reactive protein MEDICAL TEST, 10][{qΥk ηjc nmnsjYrgt, k cYI}]
Out[2]= {323 290 people,
{0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL,
0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL, 0.368762 mg/dL}}
```

Группа сущностей EntityGroup

EntityGroup [{*cl rgw*, *cl rgw*, ...}] представляет группу сущностей

В группах может быть заданы:

- объектом **Entity**,
 - объектом **EntityClass**,
 - явным типом сущности (“Chemical”, “Element” и т.п.).
- EntityGroup** может состоять из сущностей разных типов.

При определении группы, содержащей сущности разных классов и типов, необходимо учитывать возможность включение одного домена в другой:

```
In[3]:= EntityValue[{  
EntityGroup[cardiac screens MEDICAL TESTS], serum HDL cholesterol MEDICAL TEST},  
qΥk ηjc nmnsjYrgt] (*кардиологические тесты содержат анализ на HDL-холестерол*)  
Out[3]= {175 170 people, 7773 people}
```

```
In[=]:= EntityValue[
  EntityGroup[{ 
    EntityGroup[{"cardiac screens" MEDICAL TESTS, "serum HDL cholesterol" MEDICAL TEST}], 
    "q'k njc nmnsjYrgt"}]
]

Out[=]= 182 943 people
```

EntityGroup из / копий сущности эквивалентна EntityCopies[c/rgw/]:

```
In[=]:= EntityValue[EntityGroup[Table[{"parathyroid gland" ANATOMICAL STRUCTURE}, {2}]], "Mass"] ===
EntityValue[EntityCopies[{"parathyroid gland" ANATOMICAL STRUCTURE}, 2], "Mass"]

Out[=]= True
```

EntityGroup [ajYqđ] [нртсрги] эквивалентна EntityGroup [EntityList [ajYqđ]] [нртсрги].

```
In[=]:= EntityGroup[{"muscles" ANATOMICAL STRUCTURES}] [{"tnjsk c"}] ===
EntityGroup[EntityList[{"muscles" ANATOMICAL STRUCTURES}]] [{"tnjsk c"}]

Out[=]= True
```

Физическая величина, связанная со свойством нртсрги должна быть экстенсивной.
В противном случае результатом будет Missing["NotAvailable"].

```
In[=]:= EntityGroup[{"mitochondrion" PROTEINS}] [{"k njcasjYpu cgef r"}]

Out[=]= Missing[NotAvailable]
```

Ножкгош

Найдём общую площадь поверхности органов пищеварения человека :

```
In[=]:= EntityValue[
  EntityGroup[{ 
    "pharynx" ANATOMICAL STRUCTURE, "esophagus" ANATOMICAL STRUCTURE, "stomach" ANATOMICAL STRUCTURE, 
    "small intestine" ANATOMICAL STRUCTURE, "large intestine" ANATOMICAL STRUCTURE}], "SurfaceArea"]

Out[=]= 30. m2
```

Найдём общую численность пациентов, прошедших медицинское тестирование, результаты которого представлены в UJбазе :

```
EntityGroup[
  EntityClass["MedicalTest", All] ] [ qYk njc nmnsjYrgt ]
```

Out[*#*] =
3 389 806 people

Ножк глглжт

Найдём общую массу комбинации химических веществ:

Укажем количество химических веществ, используя различные физические величины:

```
In[#] := EntityValue[
  EntityGroup[{
    EntityInstance[ carbon dioxide CHEMICAL , Quantity[3, "Moles"] ],
    EntityInstance[ oxygen CHEMICAL , Quantity[800, "Liters"] ] }], "AbsoluteMass"]

Out[#] =
1275. g
```

Сущности могут быть заданы как класс сущностей:

```
In[#] := EntityGroup[ muscles ANATOMICAL STRUCTURES ] [ tnjsk c ]

Out[#] =
(10. to 11.) L
```

Результат будет таким же, если указать явный список сущностей:

```
In[#] := EntityGroup[ EntityList[ muscles ANATOMICAL STRUCTURES ] ] [ tnjsk c ]

Out[#] =
(10. to 11.) L
```

Классификация сущностей в иерархических базах сущностей EntityType

<pre>EntityType["<i>nnc</i>"] EntityType["<i>nnc</i>" → "afgb_e" → "afgb_y" → ...]</pre>	представляет тип сущности с указанным именем представляет собой дочерний тип сущности " <i>nnc</i> "
--	---

ВгрУкжжфю памз пра

Объект **EntityType** имеет вид: *gene*.

EntityType [...] [“*prop*”] можно использовать для извлечения следующих свойств:

“EntityTypeGraph”	граф, содержащий EntityType в вершине иерархии
“EntityTypeTree”	дерево всех дочерних типов сущности
“EntitySourceType”	тип, предоставляющий канонические имена для сущностей
“EntitySourceTypeTree”	дерево, корнем которого является исходный тип сущности

"ParentType"	прямой родительский тип в иерархии
"ChildTypes"	прямые дочерние типы в иерархии

Ножкгощ

Найдём заданный тип сущности и построим граф из составляющих его классов и объектов сущностей:

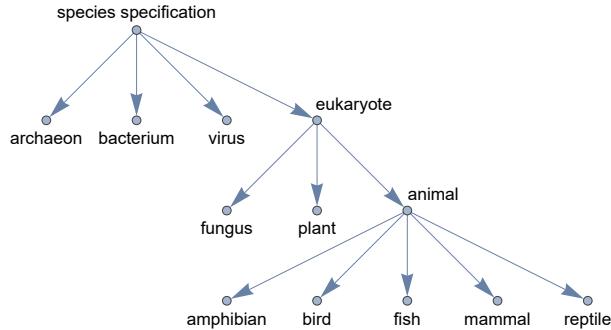
In[*#*] := EntityType ["TaxonomicSpecies"]

Out[*#*] =

species specification

In[*#*] := EntityType ["TaxonomicSpecies"] ["EntityTypeGraph"]

Out[*#*] =



Найдём дочерние классы сущностей в заданном типе и построим для них дерево, корнем которого является исходный тип сущности:

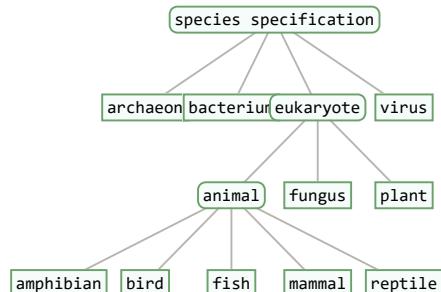
In[*#*] := EntityType ["TaxonomicSpecies"] ["ChildTypes"]

Out[*#*] =

{ species specification > archaeon , species specification > bacterium ,
 species specification > eukaryote , species specification > virus }

In[*#*] := EntityType ["TaxonomicSpecies"] ["EntityTypeTree"]

Out[*#*] =



Найдём родительский тип сущности для класса "Animal":

In[*#*] := EntityType ["TaxonomicSpecies" → "Eukaryote" → "Animal"]

Out[*#*] =

species specification > eukaryote > animal

In[*#*]:= species specification > eukaryote > animal ["ParentType"]

Out[*#*]=

species specification > eukaryote

Найдём главный источник для класса сущностей “Animal”:

In[*#*]:= species specification > eukaryote > animal ["EntitySourceType"]

Out[*#*]=

species specification

Найдём случайное животное в заданном классе:

In[*#*]:= RandomEntity[EntityType["TaxonomicSpecies" → "Eukaryote" → "Animal"]]

Out[*#*]=

Coryphaspiza

In[*#*]:= Coryphaspiza SPECIES SPECIFICATION ["Image"]

Out[*#*]=



Перечислим все типы сущностей, связанные с данной сущностью:

American alligator SPECIES SPECIFICATION ["EntityTypeList"] (*систематика аллигатора*)

Out[*#*]=

{, species specification > eukaryote, species specification > Eukaryote > animal}

16SrXII (Stolbur group) SPECIES SPECIFICATION ["EntityTypeList"]

(*систематика Stolbur фитоплазмы*)

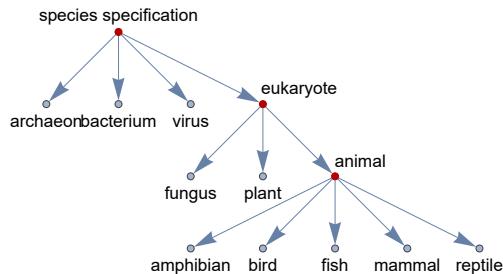
Out[*#*]=

{, species specification > bacterium}

Рассмотрим место заданной сущности в иерархии типов:

```
In[1]:= HighlightGraph[species specification ["EntityTypeGraph"],  
Coryphaspiza SPECIES SPECIFICATION ["EntityTypeList"]]
```

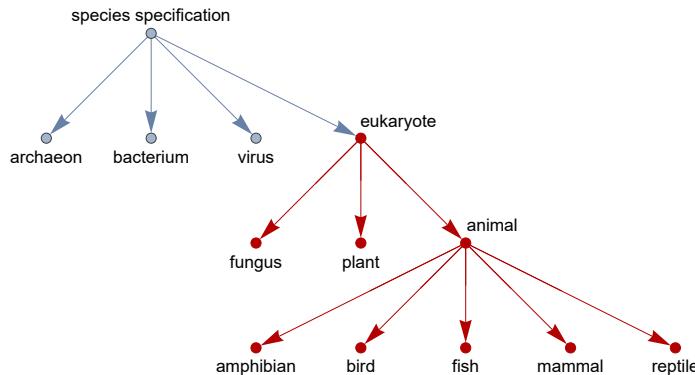
Out[1]=



Покажем встраивание дерева типов сущностей «Эукариот» в его граф типов:

```
In[2]:= HighlightGraph[species specification > eukaryote ["EntityTypeGraph"],  
VertexReplace[  
TreeGraph[species specification > eukaryote ["EntityTypeTree"], {t_, _} :> t]]]
```

Out[2]=



Памэз праў

Дочерние типы наследуют свойства родительского типа:

```
In[4]:= EntityProperties[物种 specification > Eukaryote > animal] // Short
Out[4]//Short= {Yjrcp Yrc ank k nt l Yk cq, Yjrcp Yrc qagl rgtl Yk cq, Yl l s Yj l sk Zcpndfsk Yl aYqf Yjggq, Yos Yrg Zgnk cq, Yqgnqg rcb Znbwqg cq, Yqgnqg rcb fsk Yl bgc Yqcq, Zgne cneplnf gpxegtl q, Zgnk Yqq, Zgnk cq, ajYqq, ajYqqansl r, ajYqqjgr, anfnpr, anjnpr, ank k nt l Yk c, ant qcpt Yrgt qYrsq, bg ggnl, ck ntg, cl rgwrwnc jgr, d'k gw, d'k gwansl r, d'k gwjgr, dtbcpljrf pYrc cb npcl bYl ecpcb qYrsq, dphk, ecl cpYrgt rk c, eclsq, eclsqansl r, fYZgYrbcnrf, fYZgYr cictYrgt, fYZgYr pcegtl q, fcgefr, fnrq, gk Yec, gqYajYqq, gqYnpbcp, igebnk, igebnk ansl r, igebnk jgr, jYrcqr vcYpnadnsZjgYrgt, jcl erf, jgtqnYl, k Yvg sk pcampcbcbjcl erf, k Yvg sk pcampcbcbjgtqnYl, k Yvg sk pcampcbcbucgfr, k Yvg sk rck ncprlrs pcmjcpYl ac, k gk sk rck ncprlrs pcmjcpYl ac, l Yk c, LA GRYvn nk wB, lsajcmgbc qcoscl acjcl erf, lsk Zcpndafgb rYVY, lsk Zcpndafpnk nqk cq, mZqcpt Yrgt ansl rpgq, mZqcpt Yrgt jnaYrgt q, mZqcpt Yrgt naYl q, npbcp, npbcpansl r, npbcpjgr, npqg Yj vcYpnadnsZjgYrgt, nYpcl rrYvn, nYpnt npbcp, nf vjcrq epmsnq, nf vjsk, nf vjsk ansl r, nf vjsk jgr, npk YpvrYvn nk gqcoscl ac, PcdQo Yqck ZjwYaacqgnt lsk Zcp, qagl rgtl Yk c, qcargt, <<36>>, Yl rgnpcbYrnpYbYnrYrgt q, ZYqYjk crYZnjg pYrc, ZcYi jcl erf, Zcf YtgpYjdYrs pcq, Zgrf rnncq, Zgrf ucgefr, Zgc dpmac, Znbwrck ncprlrs p, ZpYg ucgefr, Zpcbg eg rcpt Yj, Zpcbg e k nt rfq, Zpcbg e ncpgnb rnncq, ank k sl gYrgt af Yl l cj, bYgwYarg gwnYrcd q, bctcjmkk cl rYj dYrs pcq, bnpk Yl awrnncq, ceeqncpajraf, cee ucgefr, cjcarrp bgaf Ypec, dtcbg e fYZgq, dptqg Yrgt rnncq, k gprgnt dgfr bgr Yl ac, fcYb Yl b Znbwjcl erf, fcYp e pYl ec, k Yvg sk fnp jcl erf, fmnpqndbYgwojccn rnngYjjwl cccb, jcl erf nqf gZcp Yrgt, k Yg Znbwu grf, k Yrg e qmrcck, k Yvg sk dgf g e pcampbjcl erf, k Yvg sk dgf g e pcampbucgefr, k Yvg sk pcampcbfcYb Yl b Znbwjcl erf, k Yvg sk pcampcbqf cjjjcl erf, k Yvg sk pcampcb rYgjcl erf, k Yvg sk pcampcb u g eqnYl, k Yvg sk qnccb, k Yvg sk qnccb g rfc Yg, k Yvg sk qnccb g rfc u Yrcp, k Yvg sk qnccb nt jYl b, k crYZnjg pYrc ncpZnbwu cgefr, k gprgmpvncpgnb, k nbcqndank k sl gYrgt, mZqcpt Yrgt jnaYrgt qZwbYrc, ncprnrgt af Yl l cj, npcbYrnpq, bspYrgt ndnpxel Yl aw, npcw, npk Ypwbgr, npnpxbsargc cngnbc rnncq, npnpxbsargc dYrs pcq, npnpxbsargc npmacqpcq, pcosgxb bYgwdmb g rYic, qcYqnt Yjk ntck cl r qprYrcgq, qc l qnprwaf Yl l cj, qcvs Yj Yqgk l k cl rq, qcvs Yj bgk npnf gk, qcvs Yj npYqcv Yj pcpnpxbsargt, qcnsjbcpcfcgefr, lsk Zcpndbg bgt gfs Yjqg anjnt w, qnqgj Zcf Ytgp, rYgjcl erf, rcpqgnpawYpcY, rfcpk npcesjYrgt, rmrYj l sk Zcpndll cspnq, rmrYj l sk Zcpndrcrcf, ug eqnYl, ug e eqnYl}
```

```
In[1]:= CountsBy[%, EntityTypeName]  
Out[1]= <| TaxonomicSpecies → 103, Animal → 68 |>
```

Исходный тип сущности не имеет родительского типа:

```
In[1]:= EntityType["Chemical"]["ParentType"]  
Out[1]= Missing[NotApplicable]
```

При построении иерархии корнем «EntitySourceTypeTree» является тип источника сущности, корнем «EntityTypeTree» - заданный тип сущности:

```
In[1]:= species specification > eukaryote [{"EntitySourceTypeTree", "EntityTypeTree"}]
```

Out[¹] =

```

graph TD
    A[species specification] --> B[archae]
    A --> C[bacter]
    A --> D[eukaryot]
    A --> E[virus]
    B --- F[{{}}
    ]
    C --- F
    D --- G[{{}}
    ]
    E --- G
    D --> H[animal]
    D --> I[fungus]
    D --> J[plant]
    H --- K[{{}}
    ]
    I --- K
    J --- K
    H --> L[amphibian]
    H --> M[bird]
    H --> N[fish]
    H --> O[mammal]
    H --> P[reptile]
    I --> Q[amphibian]
    I --> R[bird]
    J --> S[fish]
    J --> T[mammal]
    J --> U[reptile]
    
```

Название типа для объекта сущности
EntityType

В гүрүк жок жүйе памз пра

Для `EntityType` [*сущность*] сущность может быть выражением `Entity` , `EntityClass` , `EntityProperty` или `EntityProperty-Class` .

`EntityType` [Entity[*тип*, ...]] возвращает тип *тип*.

Ножкгощ

Узнаем тип сущности для списка объектов из списка сущностей:

EntityType [**EntityList** [ generic chemicals CHEMICALS]]
(*для списка из определённого класса*)

```
EntityTypeName[{{ethylene oxide CHEMICAL}, {helium-6 ISOTOPE}, {lithium-11 ISOTOPE},
{vinyl chloride CHEMICAL}, {serum LDL cholesterol MEDICAL TEST}, {serum HDL cholesterol MEDICAL TEST},
{serum triglycerides MEDICAL TEST}}] (*для списка сущностей разных типов*)
```

Out[*#*] = {Chemical, Isotope, Isotope, Chemical, MedicalTest, MedicalTest, MedicalTest}

Некоторые канонические имена сущностей существуют в более чем одном типе сущностей:

```
In[#] := EntityTypeName[{{mercury WORD}, {mercury ELEMENT}, {Mercury PLANET}, {Mercury MYTHOLOGICAL FIGURE}}]
```

Out[*#*] = {Word, Element, Planet, Mythology}

Найдём, для каких типов сущностей относятся указанные свойства сущности:

```
In[#] := EntityTypeName[{{njYsqZjcprlec}, {Yjjmptmgk sjrgnjgq}, {qnaqjZcfYtgp}}]
```

Out[*#*] = {MedicalTest, Element, Dinosaur}

Найдём тип сущности неявно определенного класса сущности:

```
In[#] := entCL =
EntityClass["AnatomicalStructure", {ucgefrcncaclrYecndrmrYjZnbw} → TakeLargest[10]];
```

Out[*#*] = EntityTypeName[entCL]

AnatomicalStructure

Случайный объект сущности
RandomEntity

RandomEntity[*cls*] предоставляет псевдослучайную сущность с типом, определяемым спецификацией *cls*

RandomEntity[*cls*ar] предоставляет список из / псевдослучайных сущностей

В группах языка

RandomEntity[*clrgy*] извлекает образцы сущностей из онлайн УБазы знаний, возвращая объекты Entity.

cls может быть:

- объектом Entity,
- объектом EntityClass,
- неявные выражения EntityClass, основанные на условных значениях EntityProperty,
- явным типом сущности ("Chemical", "Gene" и т.п.).

Амжындалишгүй номын гүйцэтгэлийн төслийн талаар

Если размер выборки / превышает количество доступных сущностей, выдается сообщение об ошибке:

```
In[6]:= RandomEntity[{"group 1 elements", ELEMENTS}, 8]
```

RandomEntity: Requested 8 Entity objects, but only 7 are available.

```
Out[6]= RandomEntity[{"group 1 elements", 8}]
```

Указанный тип должен быть [Entity](#), [EntityClass](#) или известным типом сущности:

```
In[7]:= RandomEntity["Insect", 3]
```

RandomEntity: Insect is not a valid type of Entity or EntityClass.

```
Out[7]= RandomEntity[Insect, 3]
```

Выборка образцов осуществляется с использованием онлайн-базы знаний и не следует состоянию [SeedRandom](#).

Ножкгоощ

Получим образец сущности “Chemical”:

```
In[8]:= RandomEntity["Chemical"]
```

```
Out[8]= {2-fluoro-6-nitrophenol}
```

Найдём образец из 5 сущностей “Gene”:

```
In[9]:= RandomEntity["Gene", 5]
```

```
Out[9]= {{18543, RIKEN cDNA 8030444G23 gene (mouse), cyclin B2 (mouse), potassium inwardly rectifying channel, subfamily J, member 11 (mouse), matrinil 3 (mouse)}}
```

Найдём образцы из [EntityClass](#):

```
In[10]:= RandomEntity[{"general transcription factors", GENES}, 5]
```

```
Out[10]= {{general transcription factor IIIC, polypeptide 6, alpha 35kDa (human), general transcription factor IIA, 1, 19/37kDa (human), general transcription factor IIH, polypeptide 2B (pseudogene) (human), transcription elongation factor A (SII), 1 pseudogene 1 (human), general transcription factor IIB (human)}}
```

Найдём образцы из неявно определённого [EntityClass](#):

```
RandomEntity[EntityClass["Element",
  "Density" → Quantity[Interval[{5, 8}], ("Grams") / ("Centimeters")^3]], 3]
(*сущности типа"Element" с заданной плотностью*)

Out[*]=
{manganese, tellurium, indium}
```

Кэшируемые значения сущности
EntityPrefetch

<code>EntityPrefetch["<i>тип</i>"]</code>	извлекает кэшируемые значения, связанные со всеми сущностями указанного типа <i>тип</i>
<code>EntityPrefetch[EntityProperty["<i>тип</i>", "<i>имя</i>"]]</code>	извлекает все значения для указанного свойства

В гр҃кжжфж памз пра

`EntityPrefetch` извлекает все кэшируемые значения свойств для сущностей указанного “*тип*” и сохраняет их в системе.

Ножггощ

Извлечём все кэшируемые данные для сущности “Марганец”:

```
In[*]:= EntityPrefetch[manganese ELEMENT]
Out[*]= Success [ + ✓ Message: Prefetch resources are up to date for Element.
Type: Element ]
```

Предварительная выборка для определенного свойства:

```
In[*]:= EntityPrefetch[
EntityProperty["Element", "AtomicNumber"]]
Out[*]= Success [ + ✓ Message: Prefetch resources are up to date for Element.
Type: Element ]
```

`EntityPrefetch` используют для предварительной загрузки данных для заданного типа сущности :

```
In[*]:= AbsoluteTiming[EntityValue[manganese ELEMENT, "Density"]]
Out[*]= {1.62155, 7.470 g/cm³}
```

```
In[*]:= EntityPrefetch["Element"]
Out[*]= Success [ + ✓ Message: Prefetch resources are up to date for Element.
Type: Element ]
```

```
In[1]:= AbsoluteTiming[EntityValue[indium ELEMENT, "Density"]]

Out[1]= {1.28018, 7.310 g/cm3}
```

Если ресурсы предварительной выборки недоступны, сообщается об ошибке **Failure**:

```
In[2]:= EntityPrefetch["not an entity domain"]

Out[2]= Failure[ Message: Prefetch resources are unavailable for not an entity domain.
Tag: EntityFramework]
```

EntityPrefetch извлекает только те свойства, которые еще не заполнены в системе:

```
In[3]:= AbsoluteTiming[EntityPrefetch["Isotope"]]

Out[3]= {1.8751, Success[ Message: Prefetch resources are up to date for Isotope.
Type: Isotope]}
```

```
In[4]:= AbsoluteTiming[EntityPrefetch["Isotope"]]

Out[4]= {0.0143751, Success[ Message: Prefetch resources are up to date for Isotope.
Type: Isotope]}
```

Cl rgwRuncq

Боснинц псу лмпргз жок И йіўїжцжпўлж

Все встроенные *WJ*-типы сущностей можно найти с помощью **EntityValue** []:

```
In[5]:= EntityValue[]

Out[5]= {AdministrativeDivision, AHSPlantHeatZone, Aircraft, Airline, Airport,
AirportType, Alphabet, AmusementPark, AmusementParkRide, AmusementParkType,
AnatomicalConceptIdentifier, AnatomicalFunctionalConcept, AnatomicalSpatialConcept,
AnatomicalStructure, AnatomicalTemporalConcept, AnimalAnatomicalStructure,
AnimalCoatColorAndPattern, AnimalEggColor, AnimalFeatherColor, AnimalSkinColor,
AnimalUse, ArchitecturalStyle, Artwork, AstronomicalObservatory,
AstronomicalRadioSource, AtmosphericLayer, AtomicLevel, AtomicLine, BasicFoodGroup,
BatteryType, Beach, BiogeographicRegion, Biome, BioSequenceType, BlackHole, BoardGame,
Book, Bridge, BridgeType, BroadcastStation, BroadcastStationClassification,
Building, CalculusResult, Canal, Castle, CatBreed, CattleBreed, Cave, Cemetery,
Character, Chemical, City, ClimateStudy, ClimateType, Cloud, CognitiveTask,
Color, ColorSet, Comet, Company, ComputationalComplexityClass, Concept,
Constellation, ConstructionMaterial, ContinuedFraction, ContinuedFractionResult,
ContinuedFractionSource, Country, CrystalFamily, CrystallographicSpaceGroup,
CrystalSystem, CultivatedPlant, CultivatedPlantType, CurrencyDenomination,
CurrencyDenominationType, Dam, DeepSpaceProbe, Desert, Digimon, Dinosaur,
Disease, DisplayFormat, DistrictCourt, DogBreed, EarthImpact, Earthquake,
EclipseType, EggSize, ElectricalOutlet, Element, Emotion, EthnicGroup,
```

Exoplanet, FamousChemistryProblem, FamousGem, FamousMathGame, FamousMathProblem, FamousPhysicsProblem, FAOConservationStatus, FictionalCharacter, FictionalPlace, FictionalSpecies, FileFormat, Financial, FiniteGroup, Flight, FlightStatus, Food, FoodAge, FoodAlcoholLabel, FoodBeefGrade, FoodBoneContent, FoodBrandName, FoodCaffeineLabel, FoodCalorieLabel, FoodComposition, FoodConcentration, FoodCrustType, FoodCulture, FoodDataSource, FoodFatLabel, FoodFatType, FoodFiberLabel, FoodFlavor, FoodGeometryType, FoodGlutenContent, FoodIntendedUse, FoodIronLabel, FoodLocation, FoodMaltType, FoodManufacturer, FoodMeatCut, FoodMeatQuality, FoodMoistureLevel, FoodNutritionalSupplement, FoodNutritionalSupplementNotAdded, FoodPackaging, FoodPart, FoodPattyCount, FoodPeelingType, FoodPreparation, FoodProcessingType, FoodSeafoodVariety, FoodSeedContent, FoodServingType, FoodShape, FoodSize, FoodSkinContent, FoodSodiumLabel, FoodState, FoodStorageType, FoodSubBrandName, FoodSugarLabel, FoodSugarType, FoodTexture, FoodTrimmingLevel, FoodType, FoodTypeGroup, FoodVariety, FoodVegetablePart, FoodYeastType, Forest, FrequencyAllocation, FunctionalAnalysisSource, FunctionSpace, FungusCapType, FungusEdibility, FungusHymeniumAttachment, FungusHymeniumType, FungusStipeType, Galaxy, Gender, Gene, GeneticTranslationTable, GeographicRegion, GeologicalFormation, GeologicalLayer, GeologicalPeriod, GeometricScene, GivenName, Glacier, GoatBreed, GrammaticalUnit, Graph, HistoricalCountry, HistoricalEvent, HistoricalPeriod, HistoricalSite, HistoricalUSDAPlantHardinessZone, HorseBreed, HorseBreedConservationStatus, HorseCoatColorAndPattern, HorseType, HorseUse, ICDNine, ICDTen, Icon, IntegerSequence, InternetDomain, IPAddress, Island, ISO15924Group, Isotope, Knot, Lake, Lamina, Language, LanguageLocale, Laser, Lattice, LatticeSystem, LibraryBranch, LibrarySystem, LightColor, LunarInteriorLayer, MannedSpaceMission, MathematicalFunction, MathWorld, MatterPhase, MeasurementDevice, MedicalTest, MeteorShower, MetropolitanArea, MilitaryConflict, MilitaryConflictType, MilitaryFacilityType, Mine, Mineral, MinorPlanet, MoonPhase, Mountain, Movie, MPAARating, Museum, MusicAct, MusicAlbum, MusicAlbumRelease, MusicAlbumType, MusicalInstrument, MusicWork, MusicWorkRecording, MusicWorkRendition, Mythology, NaturalHazard, NaturalResource, NCBITranslationTableVersion, Nebula, Neighborhood, NetworkService, Neuron, NonperiodicTiling, NotableComputer, NuclearExplosion, NuclearExplosionType, NuclearReactor, NuclearReactorType, NuclearTestSite, Nutrient, ObjectStatus, Ocean, OilField, OperatingStatus, OrganizationType, Park, Particle, ParticleAccelerator, ParticulateMatter, PepperHeat, Periodical, PeriodicTiling, Person, PersonTitle, PhysicalActivity, PhysicalConstant, PhysicalEffect, PhysicalQuantity, PhysicalSystem, PigBreed, PigeonBreed, PilatesExercise, PlaneCurve, Planet, PlanetaryMoon, Plant, PlantDisease, PlantEarliness, PlantGrowthForm, PlantHeredity, PlantLeafRetention, PlantLifeCycle, PlantPhysiologicalResistance, PlantPollinator, PlantPropagationMethod, PlantSoilMoisturePreference, PlantStructure, PlantSunRequirement, PlantUse, Pokemon, PokemonAbilities, PokemonEggGroup, PokemonGeneration, PokemonPokedexColor, PokemonType, Polyhedron, PopularCurve, PoultryBreed, PoultryBreedConservationStatus, PreservationStatus, PrivateSchool, ProgrammingLanguage, Protein, PublicSchool, Pulsar, PyrometricCone, Reef, Religion, ReserveLand, RiemannHypothesisFormulation, River, Rocket, RocketFunction, Satellite, SchoolDistrict, SheepBreed, Ship, Shipwreck, SkinType, SNP, SolarSystemFeature, Solid, Sound, Source, SpaceCurve, Species, SpeciesAntipredatorAdaptation, SpeciesBehavioralFeature, SpeciesCaptivityStatus, SpeciesColor, SpeciesConservationStatus, SpeciesDevelopmentalFeature, SpeciesDiet, SpeciesFeedingBehavior, SpeciesHabitatStatus, SpeciesOxygenTolerance, SpeciesPhyleticGroup, SpeciesReproductiveFeature, SpeciesSensoryChannel,

SpeciesSexualDimorphism, SpeciesTemporalConcept, SpeciesThermalAdaptation, SpeciesThermoregulation, SpeciesTrophicGroup, Sport, SportMatch, SportObject, Stadium, Star, StarCluster, StatisticalGroup, StormType, Supernova, SupernovaType, Surface, Surname, SystemModel, TaxonomicRank, TaxonomicSpecies, TerrainType, TidalConstituent, TideStation, TimeZone, TopLevelDomain, TopologicalSpaceType, Transparency, TropicalStorm, Tunnel, UnderseaFeature, University, USCongressionalDistrict, USDAEggSize, USDAFoodGroup, USDAPlantHardinessZone, USDAPlantShadeTolerance, USDASoilTexture, USDAWetlandIndicatorStatus, USDAWetlandRegion, USFWSEndangeredSpeciesStatus, VisualArtsArtForm, VisualArtsArtGenre, Volcano, Waterfall, WeatherStation, WeightTrainingExercise, WolframDemonstration, WolframLanguageSymbol, Word, WritingDirection, WritingScript, WritingScriptBaseline, WritingScriptLigature, WritingScriptStatus, WritingScriptType, WritingScriptWhiteSpace, YogaPose, YogaPosition, YogaProp, YogaSequence, ZIPCode}

Выделяют следующие группы *UJ*-типов:

Activities & Hobbies	Astronomical Entities	Computing-Related Entities	Culture &
Entertainment	Dynamical Systems		
Engineering & Structures	Geographic Entities	History-Related	Linguistic
Entities	People & Personal Attributes		
Space-Related & Earth Science	Transportation-Related	Visual Entities	Weather

Сущности, которые могут быть полезны при решении биологических и медицинских проблем, могут содержаться в следующих группах *UJ*-типов:

Dumb g Ls rpggt

Food	BasicFoodGroup	FoodBoneContent	FoodAlcoholLabel	FoodType
FoodBrandName		FoodComposition		
FoodAge	FoodTypeGroup	FoodGlutenContent	FoodCaffeineLabel	FoodCrustType
FoodSubBrandName		FoodBeefGrade		
FoodConcentration	USDAFoodGroup	FoodSeedContent	FoodCalorieLabel	FoodFatType
		FoodFlavor		
FoodDataSource		FoodSkinContent	FoodFatLabel	FoodGeometryType
FoodCulture		FoodIntendedUse		
FoodMoistureLevel	FoodVariety		FoodFiberLabel	FoodMaltType
FoodLocation		FoodMeatCut		
FoodState			FoodIronLabel	FoodPeelingType
FoodManufacturer		FoodMeatQuality		
FoodTexture	FoodSize		FoodSodiumLabel	FoodProcessing-
Type	FoodPackaging	FoodNutritionalSupplement		
FoodTrimmingLevel	USDAEggSize		FoodSugarLabel	FoodServingType
FoodPreparation		FoodNutritionalSupplementNotAdded		FoodStorageType
		FoodPart		FoodSugarType
		FoodPattyCount		FoodYeastType
		FoodSeafoodVariety		
		FoodVegetablePart		

Jgt Qagl acq

Nutrient	Питательные вещества, включая витамины, минералы и жирные кислоты, содержащиеся в продуктах питания
-----------------	---

In[4]:= **RandomEntity["Nutrient", 5]**

Out[4]= {tryptophan, octadecenoic acid (р/ж), arsenic, 11:0, added vitamin B₁₂}

EntityValue[Entity["Nutrient", "VitaminC"], "DailyValue"]
(*рекомендуемая дневная норма*)

Out[4]= 0.09 g/day

BioSequenceType	Gene	GeneticTranslationTable	SNP
Protein			

KYrfck YrgYj g Ank nsrYrgn! Yj Cl rggq

Polyhedron	Lamina	Surface	Space
Curve	PlaneCurve		
PeriodicTiling	NonperiodicTiling		
Graph	Knot	FiniteGroup	
MathematicalFunction	IntegerSequence	ContinuedFraction	
FunctionSpace	TopologicalSpaceType		
FamousMathProblem	FamousMathGame	ComputationalComplexityClass	
MathWorld	ContinuedFractionResult	ContinuedFractionSource	
	FunctionalAnalysisSource		

GeometricScene	- символические изображения известных конструкций и теорем планиметрии в вычислимой форме
-----------------------	---

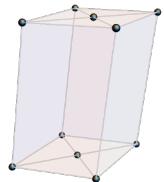
In[5]:= **RandomEntity["GeometricScene", 5]**

Out[5]= {Pasch's theorem, Euclid book 6 proposition 27, Euclid book 7 definition 5, Euclid book 4 proposition 2, Euclid book 6 proposition 3b}

Lattice	- известные математические решетки
LatticeSystem	- систем 3D-решеток: кубическая, гексагональная, моноклинная, орторомбическая, ромбоэдрическая, тетрагональная и триклинная
Solid	- именованные математические тела: замкнутые заполненные трехмерные области. Классы основаны на математических классификациях, основанных на геометрических и топологических характеристиках.

In[1]:= **base-centered monoclinic LATTICE** ["Graphics3D"]

Out[1]=



In[2]:= **hexagonal LATTICE SYSTEM** ["CrystalFamily"]

hexagonal CRYSTAL FAMILY

RandomEntity["Solid", 5]

Out[2]=

{ Steinmetz 4-solid, trirectangular tetrahedron, second gömböc, half-torus, Steinmetz 3-solid }

In[3]:= **ball SOLID** ["Volume"]

Out[3]=

$$\text{Function}[a, \frac{4 \pi a^3}{3}]$$

["ThetaSeriesFunction"][x]

KcbgYj Cl rggq

AnatomicalStructure	Neuron	Disease	CognitiveTask
AnimalAnatomicalStructure		ICDNine	MedicalTest
AnatomicalConceptIdentifier		ICDTen	
AnatomicalFunctionalConcept			
AnatomicalTemporalConcept			

MpeYl gk Rnncq

TaxonomicRank	Їрүипмлмк жүгпиж оўлбжбоснинц мөбўлжик ма
TaxonomicSpecies	Їрүипмлмк жүгпиж жт жоңғапиж вўллциг мөбўлжик маҳд жамрлишую ўрглжузубожумай ўугзү
Ўўиргожж жажосима	

RandomEntity["TaxonomicRank", 5] (*примеры сущностей*)

Out[4]=

{ section, tribe, subclass, subtribe, no rank }

In[$\#$]:= **polar bear** SPECIES SPECIFICATION [EntityProperty["TaxonomicSpecies", "MaximumRecordedLength"]]

Out[$\#$]=

3.0 m

In[$\#$]:= EntityValue[**giant panda** SPECIES SPECIFICATION, "TaxonomicSequence"]

(*таксономическая последовательность для животного*)

Out[$\#$]=

```
{superkingdom → eukaryotes, kingdom → animals, phylum → chordates,
subphylum → Craniata, superclass → Sarcopterygii, class → mammals,
superorder → Laurasiatheria, order → carnivores, suborder → Caniformia,
family → bears, genus → Ailuropoda, species → giant panda}
```

AnimalAnatomicalStructure	AnimalUse	CatBreed	Dinosaur	EggSize
HorseBreed				
AnimalCoatColorAndPattern		DogBreed		
AnimalEggColor	HorseBreedConservationStatus			
AnimalFeatherColor		GoatBreed		
	HorseCoatColorAndPattern			
AnimalSkinColor		PigBreed		
	HorseType		PigeonBreed	
				SheepBreed
HorseUse				

Plant	PlantDisease	PlantLifeCycle	PlantPhysiologicalResistance
PlantUse	CultivatedPlant		
PlantGrowthForm	PlantHeredity	PlantPollinator	PlantSoilMoisturePreference
PlantEarliness	CultivatedPlantType		
PlantStructure	PlantLeafRetention	PlantPropagationMethod	PlantSunRequirement
			USDAPlantShadeTolerance

Species
мобұлжын ма

Ірүипмемкі жүгіншілдегі жаңылардың мөлдірліктерінің жиынтығынан табылады.

In[$\#$]:= **cats** SPECIES SPECIFICATION ["SubEntities"]

Out[$\#$]=

```
{Acinonyx, Caracal, Catopuma, Felis, jaguarundi, Leopardus,
Leptailurus, Lynx, Neofelis, Oncifelis, andean mountain cat, Otocolobus,
Panthera, Pardofelis, Prionailurus, Profelis, Puma, Uncia}
```

SpeciesColor	SpeciesAntipredatorAdaptation	SpeciesSensoryChannel
SpeciesBehavioralFeature	SpeciesPhyleticGroup	SpeciesCaptivityStatus
SpeciesFeedingBehavior	SpeciesThermalAdaptation	SpeciesDiet
SpeciesDevelopmentalFeature	SpeciesTrophicGroup	SpeciesConservationStatus
SpeciesTemporalConcept	SpeciesOxygenTolerance	SpeciesSexualDimorphism
SpeciesReproductiveFeature		SpeciesHabitatStatus
SpeciesThermoregulation		

Nf nqayj Qaq/l acq

Chemical	Element	Isotope	FamousChemistryProblem
Mineral	CrystalFamily	CrystalSystem	CrystallographicSpaceGroup
Particle	Laser	PhysicalSystem PhysicalConstant FamousPhysicsProblem	Color ColorSet LightColor

MeasurementDevice

BioSequenceType Gene
 GeneticTranslationTable Element Isotope

Элементы периодической таблицы Element

Entity["Element", /Yk c] жжж name ELEMENT
 предоставляют сущность типа "Element" с наименованием /Yk c

Вгру́жённые памзра

Сущности из типа "Element" представляют мельчайшие строительные блоки материи, которые невозможно разбить на составные части химическим путем.

Встроенные объекты "Element" можно указывать на естественном языке с помощью `ctrl =:`

= gold →

In[]:= gold ELEMENT ... ✓
 Out[]:= gold

Список всех классов в типе “Element” в У-базе сущностей можно получить с помощью EntityclassList[“Element”].
Образцы классов сущностей в типе “Element”:

```
In[1]:= EntityValue["Element", "SampleEntityClasses"]
Out[1]= {{"alkaline earth metals"}, {"group 1 elements"}, {"group 8 elements"}, {"lanthanides"}, {"noble metals"}, {"non-metallic elements"}, {"platinum metals"}, {"rare earth metals"}, {"solid elements"}, {"stable elements"}}
```

Образцы сущностей в типе “Element”:

```
In[2]:= EntityValue["Element", "SampleEntities"]
Out[2]= {"beryllium", "oxygen", "carbon", "hydrogen", "gold", "xenon", "germanium", "bromine", "uranium", "darmstadtium"}
```

Число сущностей в типе “Element”:

```
In[3]:= EntityValue["Element", "EntityCount"]
Out[3]= 118
```

Количество свойств, доступных для сущностей в “Element”:

```
In[4]:= EntityValue["Element", "PropertyCount"]
Out[4]= 120
```

Доступные *и^кУпщ гамз пра* для сущностей в “Element”:

```
In[5]:= EntityValue["Element", "PropertyClasses"]
Out[5]= {{"YZs/bYl acq"}, {"pCYarggw"}, {"cjcarpk Yel crg nptmcrgc"}, {"k YqqYZs/bYl acq"}, {"k Yrcfj nptmcrgc"}, {"lsajcYprnptmcrgc"}, {"mrgYj nptmcrgc"}, {"rfcpk nbwYk g nptmcrgc"}}
```

Памз пра^Упсч лмпргз в типе “Element” (полученные с помощью команды Entity[“Element”][“Properties”]):

Лўжайлж^z

Name	название
AlternateNames	альтернативные наименования
AtomicSymbol	атомный символ
CASNumber	регистрационный номер CAS (численный идентификатор соединений в базе данных Химической реферативной службы (Chemical Abstracts Service))

⁷ ККМРОМНШ^z

AllotropeNames	название аллотропов
AllotropicMultiplicities	количество атомов в молекуле аллотропов

Оўпномпроўляллмгъ

CrustAbundance	распространённость в коре
CrustMolarAbundance	молярное содержание в коре
HumanAbundance	распространённость в человеческом организме
HumanMolarAbundance	молярное содержание в человеческом организме
MeteoriteAbundance	распространённость в метеоритах
MeteoriteMolarAbundance	молярное содержание в метеоритах
OceanAbundance	распространённость в океане
OceanMolarAbundance	молярное содержание в Мировом океане
SolarAbundance	распространённость на Солнце
SolarMolarAbundance	молярное содержание на Солнце
UniverseAbundance	распространённость во Вселенной
UniverseMolarAbundance	молярное содержание во Вселенной

Origins	происхождение
SpaceGroup	космическая группа
SpaceGroupNumber	номер космической группы IUCr (Международного союза
кристаллографов)	

Нмкмдглж а нгожмвжхглимз рўЎкжфц

AtomicNumber	атомный номер
Block	блок
Group	группа
Period	период
Series	группа элементов

УЎоЎиргожпражижЎрмкЎз

AtomicMass	атомная масса
AtomicRadius	атомный радиус
CovalentRadius	ковалентный радиус
VanDerWaalsRadius	радиус Ван-дер-Ваальса

ElectronCount	количество электронов
ElectronicConfiguration	электронная конфигурация
ElectronShellConfiguration	конфигурация электронной оболочки
OrbitalOccupationList	список занятых орбиталей
ShortElectronicConfiguration	краткая электронная конфигурация
TermSymbol	спектральный терм

Жўимрмнц

IsotopeAbundances	распространённость изотопов
IsotopeHalfLives	периоды полураспада изотопов

IsotopeLifetimes	время жизни изотопов
KnownIsotopes	известные изотопы
StableIsotopes	стабильные изотопы
UnstableIsotopes	нестабильные изотопы

Памз праўэвоЖ

NeutronCount	число нейтронов на изотоп
NeutronCrossSection	нейтронное сечение
NuclearDiameter	диаметр ядра
NuclearRadius	ядерный радиус
ProtonCount	количество протонов

DecayMode	вид распада
HalfLife	период полураспада
NeutronMassAbsorption	поглощение массы нейтрона

MolarRadioactivity	молярная радиоактивность
SpecificRadioactivity	удельная радиоактивность

БоЎт жалкувжбўйк к щиокмўйд глжо:

LewisDotStructureDiagram	структурная диаграмма с точечными структурами Льюиса
Image	изображение

Огўифимплўэ пінгмўлміръ

ElectronAffinity	сродство к электрону
Electronegativity	электроотрицательность
IonizationEnergies	энергия ионизации
SpecificIonizationEnergies	удельные энергии ионизации
Valence	валентность
ValenceElectronCount	число валентных электронов

KnownOxidationStates	известные степени окисления
MostCommonOxidationStates	наиболее распространенные степени окисления

Бкгиромжўблжрлшг памз праў

ElectricalConductivity	электропроводность
ElectricalType	электрический тип
MagneticType	магнитный тип
Resistivity	удельное сопротивление
ThresholdFrequency	пороговая частота
WorkFunction	рабочая функция

MassMagneticSusceptibility	массовая магнитная восприимчивость
----------------------------	------------------------------------

MolarMagneticSusceptibility	молярная магнитная восприимчивость
VolumeMagneticSusceptibility	объемная магнитная восприимчивость

CuriePoint	точка Кюри
NeelPoint	антиферромагнитная точка Кюри
SuperconductingPoint	сверхпроводящая точка

Памз праўагч гпраў

GasAtomicMultiplicities	количество атомов в молекуле газа
MolarMass	молярная масса

BulkModulus	модуль объемной упругости
DebyeCharacteristicTemperature	температура Дебая
FusionHeat	молярная теплота плавления $\Delta_{d,q}F$
LiquidDensity	плотность жидкости
MolarVolume	молярный объем
PoissonRatio	коэффициент Пуассона
SoundSpeed	скорость звука
SolidificationHeat	молярная теплота затвердевания
SpecificSolidificationHeat	удельная теплота затвердевания
TensileYieldStrength	сверхпроводящая точка
ThermalConductivity	теплопроводность
ThermalExpansion	тепловое расширение
UltimateTensileStrength	предел прочности на растяжение

ShearModulus	модуль сдвига
YoungModulus	модуль Юнга

BrinellHardness	твёрдость по Бринеллю
MohsHardness	твёрдость по шкале Мооса
VickersHardness	твёрдость по Виккерсу

Мнржглихг памз праў

Color	цвет
IonColor	цвет иона
RefractiveIndex	показатель преломления

Ргок мвжлўк жхглихг памз праў

AdiabaticIndex	адиабатический индекс
Phase	фазовое состояние
SpecificFusionHeat	удельная теплота плавления $\Delta_{d,q}F$
SpecificHeat	удельная теплоёмкость a_n

BoilingPoint	точка кипения
CriticalTemperature	критическая температура
MassDensity	плотность

MeltingPoint	точка плавления R_k
MolarHeatCapacity	молярная теплоемкость
VaporizationHeat	молярная теплота парообразования и конденсации $\Delta_{t\gamma n}F$

Памз праў и ножпрукма:

CrystalStructure	решетка Браве
CrystalStructureImage	изображение схемы решетки
LatticeAngles	углы решетки
LatticeConstants	константы решетки

Паэжълышг памз праў

Discoverers	первооткрыватели
DiscoveryCountries	страны открытия
DiscoveryDate	дата открытия
CommonCompounds	сущности "Chemical", содержащие элемент
EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей, к которому принадлежит объект
Memberships	членство в классах сущностей
Price	цена

Ножглощ

Применим различные типы записей электронных конфигураций элемента:

```
In[]:= phosphorus ELEMENT [{"ElectronicConfiguration", "ElectronConfigurationString",
  "ElectronShellConfiguration", "ShortElectronicConfiguration"}, "Association"]
Out[=]= {{"ElectronicConfiguration" → 1s^2 2s^2 2p^6 3s^2 3p^3, "ElectronConfigurationString" → [Ne] 3s^2 3p^3,
  "ElectronShellConfiguration" → {2, 8, 5}, "ShortElectronicConfiguration" → [Ne] 3s^2 3p^3}}
```

Найдём теплопроводность вольфрама (в единицах СИ):

```
In[]:= tungsten ELEMENT ["ThermalConductivity"]
Out[=]= 170. W/(m K)
```

Получим набор данных всех доступных свойств для сущности:

```
In[#]:= iron ELEMENT ["Dataset"]
```

```
Out[#]=
```



Найдём пять самых электроотрицательных элементов:

```
In[]:= EntityClass["Element", {"Electronegativity" → TakeLargest[5]}] // EntityList
```

```
Out[=] = {fluorine, oxygen, chlorine, nitrogen, krypton}
```

Найдём английское название элемента и список альтернативных названий:

```
In[]:= tungsten ELEMENT [ {"Name", "AlternateNames"} ]
```

```
Out[=] = {tungsten, {wolfram}}
```

Мпмъгллмпржножжглглжэ

Классы свойств

Получим набор данных класса свойств элемента:

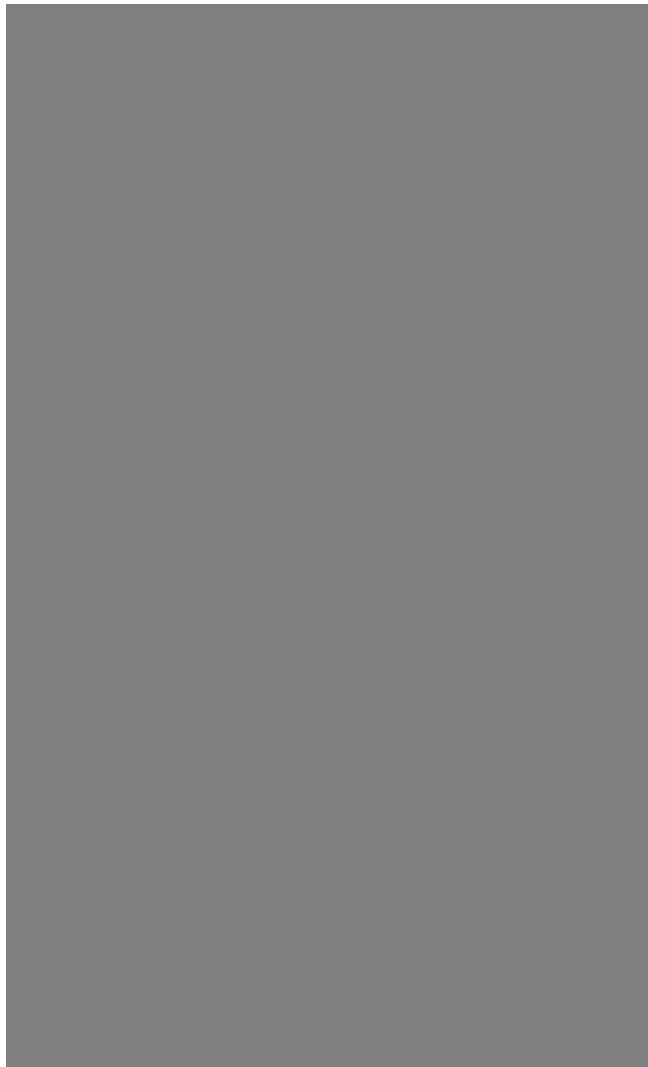
```
In[]:= EntityValue[darmstadtium ELEMENT, {"sajcYpnpmcrgq"}, "Dataset"]
```

```
Out[=]
```



```
In[=]:= EntityValue[cerium ELEMENT, "k Yrcpøj jnpmcpgq", "Dataset"]
```

Out[=]=



Неявные классы сущностей

Найдём наиболее распространенные элементы в метеоритах:

```
In[=]:= EntityClass["Element",
  EntityProperty["Element", "MeteoriteAbundance"] → TakeLargest[5]] // EntityList
```

Out[=]=

{oxygen, iron, silicon, magnesium, sulfur}

Перечислим все элементы, открытые в первой половине XX века:

```
In[]:= EntityClass["Element", {"DiscoveryYear" →
  (Between[#, {DateObject[{1900}], DateObject[{1950}]}] &)}] // EntityList
Out[]= {technetium, promethium, europium, lutetium,
  hafnium, rhenium, astatine, radon, francium, protactinium,
  neptunium, plutonium, americium, curium, berkelium, californium}
```

Entity Instances

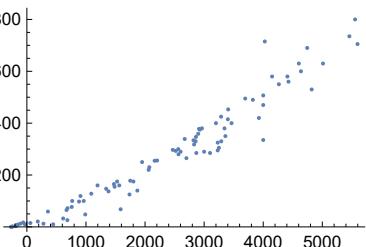
Найдём цену на золото в 2010 г.:

```
In[]:= EntityInstance[gold ELEMENT, "Date" → DateObject[{2010}]]["Price"]
Out[]= TimeSeries[]
```

Ножкындағы

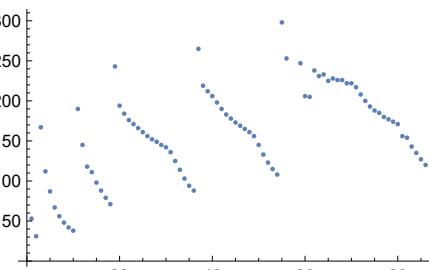
Найдём зависимость между энталпийей испарения и температурой кипения:

```
In[]:= ListPlot[EntityValue["Element", {"BoilingPoint", "VaporizationHeat"}]]
Out[]=
```



Построим график влияния на атомные радиусы элементов структуры электронной оболочки:

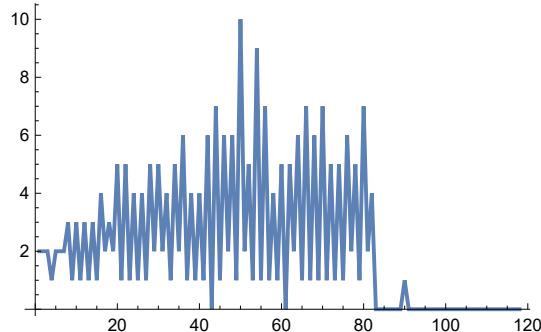
```
In[]:= ListPlot[EntityValue["Element", "AtomicRadius"]]
Out[]=
```



Построим график количества стабильных изотопов каждого элемента:

```
In[]:= ListLinePlot[Length /@ EntityValue["Element", "StableIsotopes"]]
```

```
Out[=]=
```



Анализ списка известных изотопов элемента (из сущностей "Isotope"):

```
In[]:= curium ELEMENT ["KnownIsotopes"]
```

```
Out[=]=
```

```
{curium-231, curium-232, curium-233, curium-234, curium-235,
 curium-236, curium-237, curium-238, curium-239, curium-240,
 curium-241, curium-242, curium-243, curium-244, curium-245, curium-246,
 curium-247, curium-248, curium-249, curium-250, curium-251, curium-252}
```

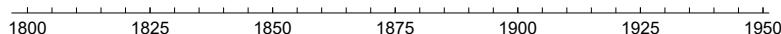
```
In[]:= EntityValue[%, "Spin", "EntityAssociation"]
```

```
Out[=]=
```

```
(| curium-231 → Missing[NotAvailable], curium-232 → Missing[NotAvailable],
 curium-233 → 3/2, curium-234 → 0, curium-235 → 5/2, curium-236 → 0, curium-237 → 5/2,
 curium-238 → 0, curium-239 → 7/2, curium-240 → 0, curium-241 → 1/2, curium-242 → 0,
 curium-243 → 5/2, curium-244 → 0, curium-245 → 7/2, curium-246 → 0, curium-247 → 9/2,
 curium-248 → 0, curium-249 → 1/2, curium-250 → 0, curium-251 → 1/2, curium-252 → 0 |)
```

Визуализация открытия лантаноидов:

```
In[]:= TimelinePlot[EntityValue[lanthanides ELEMENTS, "DiscoveryYear", "EntityAssociation"]]  
Out[]=
```



Найдём, сколько элементов было открыто в каждой стране:

```
In[]:= Tally[Flatten[EntityValue["Element", "DiscoveryCountries"]]] // SortBy[Last]  
Out[]=
```

{Denmark, 1}, {Finland, 1}, {India, 1}, {Italy, 1}, {Japan, 1},
{Mexico, 1}, {Peru, 1}, {Poland, 1}, {Romania, 1}, {Spain, 1},
{Missing[NotAvailable], 1}, {Missing[NotDiscovered], 1}, {Austria, 2},
{Switzerland, 3}, {Russia, 8}, {Missing[Prehistoric], 12}, {France, 15},
{Sweden, 19}, {Germany, 20}, {United States, 22}, {United Kingdom, 25}

Изотопы химического элемента Isotope

```
Entity["Isotope", /Yk c] жжк name ISOTOPЕ  
представляют сущность типа "Element" с наименованием /Yk c
```

В группах ячеек

Сущности из типа "Isotope" представляют все изотопы всех химических элементов.

Встроенные объекты "Isotope" можно указывать на естественном языке с помощью **ctrl =**:

carbon 14 →

```
In[1]:= carbon-14 ISOTOPe 
```

```
Out[1]= carbon-14
```

Список всех классов в типе “Isotope” в *UJ*-базе сущностей можно получить с помощью `EntityclassList["Isotope"]`. Классы сущностей “Isotope” включают группировки изотопов по различным свойствам, например, изотопы одного и того же химического элемента, изотопы с одинаковым первичным режимом распада и изотопы с одинаковым атомным номером.

Образцы классов сущностей в типе “Isotope”:

```
In[2]:= EntityValue["Isotope", "SampleEntityClasses"]
```

```
Out[2]= {double beta decay, isotopes with A = 40, alpha emission,
stable isotopes, positron emission, electron capture, fermion,
oxygen-18 cluster emission, spontaneous fission, beta delayed two neutron emission}
```

Образцы сущностей в типе “Isotope”:

```
In[3]:= EntityValue["Isotope", "SampleEntities"]
```

```
Out[3]= {uranium-235, carbon-14, iodine-131, sodium-22, dubnium-269,
strontium-90, radium-223, cesium-137, chlorine-35, thorium-220}
```

Число сущностей в типе “Isotope”:

```
In[4]:= EntityValue["Isotope", "EntityCount"]
```

```
Out[4]= 3558
```

Количество свойств, доступных для сущностей в “Isotope”:

```
In[5]:= EntityValue["Isotope", "PropertyCount"]
```

```
Out[5]= 56
```

Памът прајпсч лмпргз в типе “Isotope” (полученные с помощью команды `Entity["Isotope"]["Properties"]`):

ЛУЖАЛЖ ζ

Name	название
AtomicSymbol	атомный символ

УЛОУИРГОЖТРЖАЖУРМКҮЗ

AtomicNumber	атомный номер
AtomicMass	атомная масса
FullSymbol	полный символ
Symbol	символ

Уйоғыр жоктарынан маңыздылық түрдөр

IsotopeAbundances	распространённость изотопов
Stable	стабильность
NeutronNumber	количество нейтронов

Памзаттарынан маңыздылық түрдөр

BranchingRatios	коэффициенты ветвления
DecayMode	вид распада
DaughterNuclides	дочерние нуклиды
DecayConstant	константа распада
DecayEnergies	энергия распада
DecayModeSymbols	режимы распада
DecayProducts	продукты распада
EffectiveHalfLife	эффективный период полураспада
FinalDecayProducts	конечные продукты распада
HalfLife	период полураспада

Орталық радиоактивтүрдөр

Radioactivity	радиоактивность
MolarRadioactivity	молярная радиоактивность
SpecificRadioactivity	удельная радиоактивность

Памзаттарынан маңыздылық түрдөр

BindingEnergy	энергия связи на нуклон
CriticalDiameter	критический диаметр
CriticalMass	критическая масса
EffectiveLifetime	эффективный срок жизни
ExcitedStateEnergies	C_{ictc_j}
ExcitedStateHalfLives	half-life $\tau_{1/2}$
ExcitedStateLifetimes	время жизни возбужденного состояния
ExcitedStateParities	частицы возбужденного состояния
ExcitedStateSpins	возбужденное состояние спинов
ExcitedStateWidths	ширина возбуждённого состояния Δ
ExternalExposure	указание по внешнему воздействию
HalfValueLayer	слой половинного значения
Lifetime	средняя продолжительность жизни
MagneticMoment	магнитный момент
MassDefect	дефект массы
MassExcess	избыток массы
MassNumber	массовое число
MolarMass	молярная масса
Parity	паритетность
QuadrupoleMoment	электрический квадрупольный момент
QuantumStatistics	тип частицы
Spin	спин

SpinParity	часть спина $\frac{1}{2}$
TenthValueLayer	слой десятичного значения
Width	ширина

ЖМКМБЖХГ ПАМЗ ПРАЙ

BiologicalHalfLife	период биологического полураспада
BiologicalLifetime	продолжительность биологической жизни
CriticalOrgans	критические органы
DiagnosticApplications	диагностические приложения
DiagnosticApplicationsToDiseases	диагностические приложения к заболеваниям
MedicalApplications	диагностические приложения

ПАЭЖДЛЛЩГ ПАМЗ ПРАЙ

EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей, к которому принадлежит объект
Memberships	членство в классах сущностей

НОЖГОЩ

Найдём значение свойства для объекта:

```
In[1]:= thorium-220 ISOTOPE ["HalfLife"]
Out[1]= 10. \mu s
```

Получим набор данных всех доступных свойств для сущности:

```
In[8]:= cesium-137 ISOTOPE ["Dataset"]
```

Out[8]=



Найдём изотопы урана с самым большим периодом полураспада:

```
In[1]:= EntityClass["Isotope", {"AtomicNumber" → 92, "HalfLife" → TakeLargest[5]}] // EntityList
Out[1]= {uranium-238, uranium-235, uranium-236, uranium-234, uranium-233}
```

Ножкаглглжт

Распад урана 232

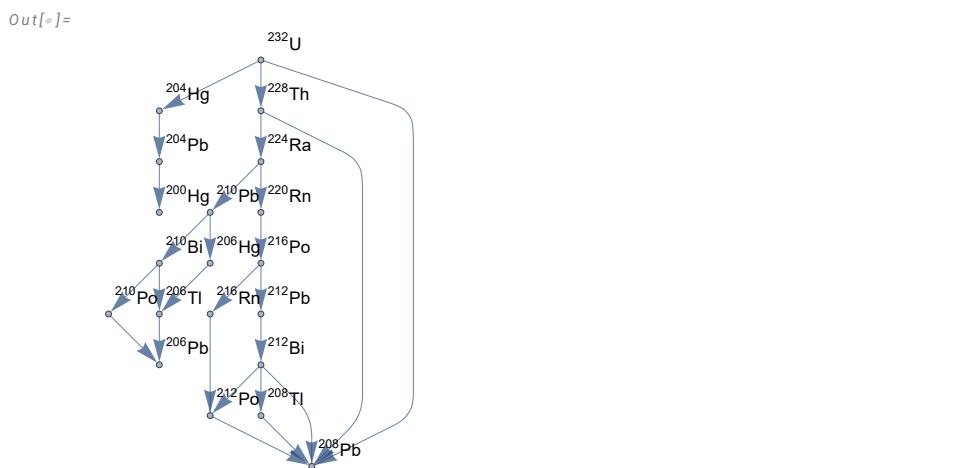
Найдём все нуклиды в заданной начальной точке:

```
In[2]:= DaughterNuclides[s_List] := DeleteMissing[Flatten[bysefrcplesajfpcq[s]]];
ReachableNuclides[s_List] := FixedPoint[Union[Join[#, DaughterNuclides[#]]] &, s];
DaughterNuclidesQ[s1_, s2_] := (s1 != s2 && MemberQ[DaughterNuclides[{s1}], s2]);
vertices = ReachableNuclides[{uranium-232 ISOTOPE}]

Out[2]= {bismuth-210, bismuth-212, lead-204, lead-206, lead-208,
lead-210, lead-212, mercury-200, mercury-204, mercury-206,
polonium-210, polonium-212, polonium-216, radium-224, radon-216,
radon-220, thallium-206, thallium-208, thorium-228, uranium-232}
```

Визуализируем цепочки распада урана 232:

```
In[3]:= labels = Normal@EntityValue[vertices, "Symbol", "EntityAssociation"];
RelationGraph[DaughterNuclidesQ, vertices, VertexLabels → labels]
```



Химические соединения Chemical

Entity["Chemical", /Yk c] жеке name CHEMICAL

представляют сущность типа "Chemical" с наименованием / Yk c

В группах по ламзра

Сущности из типа "Chemical" имеют постоянный состав и не могут быть разделены на более мелкие сущности без разрыва связей.

Классы сущностей "Chemical" включают вещества, имеющие общую характеристику (например, функциональную группу).

Встроенные объекты "Chemical" можно указывать на естественном языке с помощью `ctrl =`:

`= tri-methyl orthobutyrate` →

```
In[1]:= trimethyl orthobutyrate CHEMICAL ... ✓
```

```
Out[1]= trimethyl orthobutyrate
```

Список всех классов в типе "Chemical" в УБазе сущностей можно получить с помощью [EntityclassList](#) ["Chemical"].

Образцы классов сущностей в типе "Chemical":

```
In[2]:= EntityValue["Chemical", "SampleEntityClasses"]
```

```
Out[2]= { carboxylic acids, chlorofluorocarbons, diatomic molecules, fullerenes, hydrobromofluorocarbons, NOx compounds, polar solvents, pyridines, quinolines, thiophenes }
```

Образцы сущностей в типе "Chemical":

```
In[3]:= EntityValue["Chemical", "SampleEntities"]
```

```
Out[3]= { L-lysine, acetic acid, benzene, water, methane, ammonia, carbon dioxide, hydrogen sulfide, D-(+)-glucose, sodium hydroxide }
```

Число сущностей в типе "Chemical":

```
In[4]:= EntityValue["Chemical", "EntityCount"]
```

```
Out[4]= 44 096
```

Количество свойств, доступных для сущностей в "Chemical":

```
In[5]:= EntityValue["Chemical", "PropertyCount"]
```

```
Out[5]= 119
```

Доступные **икүйлшіламз пра** для сущностей в “Chemical”:

```
In[1]:= EntityValue["Chemical", "PropertyClasses"]
Out[1]= {{"ZYqfa нртсргзқ", {"ZnI b g dрк Yrgt"}, {"qjjsrgt нртсргзқ", {"mngdawнртсргзқ}}
```

Памз пра “**ЛМПРГЗ**” в типе “Chemical” (полученные с помощью команды `Entity["Chemical"]["Properties"]`):

Лұжадылж жт моксаның

AlternateNames	альтернативные наименования
FormattedName	отформатированное имя
Name	название
IUPACName	наименование по ИЮПАК

Formula	формула
FormulaString	строка формулы
HillFormula	формула Хилла
HillFormulaString	строка формулы Хилла

ЖӘГЛРДІК ЖИЛДАМАШЫҚ

InChI	идентификатор InChI
IsomericSMILES	изомерный идентификатор SMILES
SMILES	идентификатор SMILES

BeilsteinNumber данных Beilstein)	регистрационный номер Бейльштейна (идентификатор соединений в базе
CASRegistryNumber данных Химической реферативной службы (Chemical Abstracts Service))	регистрационный номер CAS (численный идентификатор соединений в базе
DOTHazardClass	класс опасности DOT
DOTNumbers	номера DOT
EDECNumber	номер ЕГЭК
EUNumber List of Notified Chemical Substances)	номер EC (European Inventory of Existing Commercial Substances или European
GmelinNumber	номер Гмелина (база данных по неорганической и металлоганической химии)
MDLNumber	идентификатор MDL (MDL Discovery Knowledge)

NFPAFireRating опасных веществ для реагирования на чрезвычайные ситуации)	рейтинг пожаробезопасности NFPA (Стандартная система идентификации
NFPAHazards	опасности NFPA
NFPHealthRating	числовая оценка здоровья NFPA
NFPALabel	изображение этикетки NFPA
NFPAReactivityRating	рейтинг реактивности NFPA
NSCNumber рака (NCI)	числовой идентификатор веществ, представленных в Национальном институте
RTECSClasses Toxic Effects of Chemical Substances)	классы RTECS (Реестр токсических эффектов химических веществ; Registry of
RTECSNumber	идентификатор RTECS

Гларуа к мөгискүйгү просирсолыңг памз праў.

AromaticAtomCount	количество ароматических атомов
AromaticQ	подтверждение, является ли вещество ароматическим
AtomCoordinates	позиции атомов
AtomCount	количество атомов
NonHydrogenCount	количество атомов, за исключением атомов водорода
AtomDiagramCoordinates	координаты атомной диаграммы
ElementTypes	сущности элементов, входящие в состав молекулы
ElementCounts	количество элементов
ElementMassFraction	массовая доля элемента
IonCounts	количество ионов
IonEquivalents	эквиваленты ионов
BondCounts	количество химических связей
BondEnergies	энергии связи
BondLengths	длины связей
ChemicalHybridization	орбитальная гибридизация
EdgeRules	правила связей атомов
EdgeTypes	типы связей атомов
RotatableBondCount	количество вращающихся связей
Isomers	изомеры
TautomerCount	количество таутомеров
ElectronAffinity	сродство к электрону
ElectronCount	количество электронов
FormalCharges	формальный заряд
NetCharge	суммарный заряд
ProtonAffinity	сродство к протону
ProtonCount	количество протонов
HBondAcceptorCount	количество акцепторов водородной связи
HBondDonorCount	количество доноров водородных связей
OxidationStates	степени окисления атомов в молекуле
NonStandardIsotopeCount	количество нестандартных изотопов
NonStandardIsotopeCounts	количество нестандартных изотопов
NonStandardIsotopeNumbers	числа нестандартных изотопов

Боўт жижжвжўбоўккүц

BlackStructureDiagram	чёрно-белая структурная диаграмма
CHBlackStructureDiagram	чёрно-белая структурная диаграмма с указанием атомов углерода и водорода
CHColorStructureDiagram	цветная структурная диаграмма с указанием атомов углерода и водорода
ColorStructureDiagram	цветная структурная диаграмма

LewisDotStructureDiagram	структурная диаграмма с точечными структурами Льюиса
MoleculePlot	пространственная модель молекулы
SpaceFillingMoleculePlot	структура молекулы по модели заполнения пространства
StickMoleculePlot	3D-структура скелета молекулы
StructureGraph	структурный граф

VertexTypes	атомы, находящиеся в вершинах графа
-------------	-------------------------------------

Ашхалткэгк ўжжглиж жт жижхглиж памз праū

MolarMass	молярная масса
MolarVolume	молярный объем
MolecularMass	молекулярная масса
RelativeMolecularMass	относительная молекулярная масса

AcidityConstants	константа диссоциации кислоты / γ
AutoignitionPoint	точка самовоспламенения
BoilingPoint	точка кипения
CombustionHeat	теплота сгорания
CriticalPressure	критическое давление
CriticalTemperature	критическая температура
DielectricConstant	диэлектрическая проницаемость
DipoleMoment	дипольный момент
FlashPoint	точка возгорания
FusionHeat	теплота плавления
HenryLawConstant	константа Генри
HildebrandSolubility	параметр растворимости Гильдебранда
IsoelectricPoint	изоэлектрическая точка
LightSpeed	скорость света
LogAcidityConstants	n/γ
LowerExplosiveLimit	нижний предел взрываемости
MassDensity	плотность
MeltingPoint	точка плавления
OdorThreshold	порог обнаружения запаха
OdorType	запах
PartitionCoefficient	коэффициент разделения (XLogP)
pH	pH
Phase	фаза
RefractiveIndex	показатель преломления
Resistivity	удельное сопротивление
SideChainLogAcidityConstant	n/γ боковой цепи
Solubility	растворимость
SurfaceTension	поверхностное натяжение
ThermalConductivity	молярная теплопроводность
UpperExplosiveLimit	верхний предел взрываемости
VanDerWaalsConstants	константы Ван-дер-Ваальса
VaporDensity	давление пара
VaporizationHeat	теплота парообразования и конденсации
VaporPressure	давление пара
Viscosity	динамическая вязкость

MeanFreePath	средняя длина свободного пробега молекулы
TopologicalPolarSurfaceArea	топологическая полярная площадь поверхности

Пәрәйлүштік памзапраў

Codons	кодоны
DrugInteractions	взаимодействие с лекарственными средствами

EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей, к которому принадлежит объект
Memberships	членство в классах сущностей
Molecule	представляет объект "Molecule"

Ножкгощ

Найдём составные части (элементы, ионы) химических веществ:

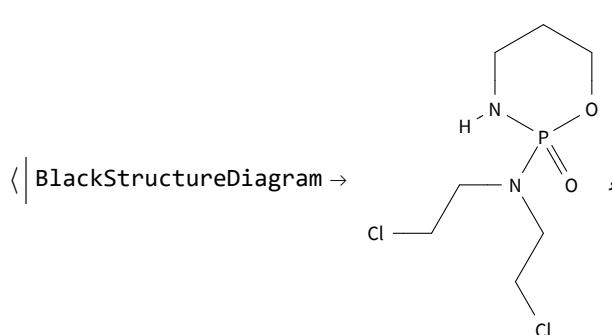
```
In[1]:= EntityValue[α-(tert-butyl)hydrocinnamic acid CHEMICAL, "ElementTypes"]
Out[1]= {carbon, hydrogen, oxygen}

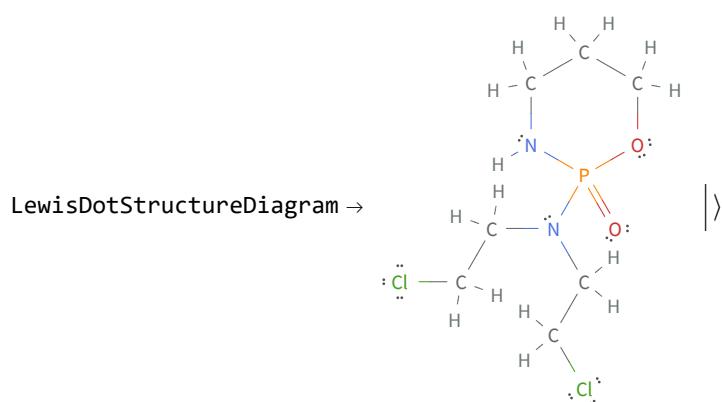
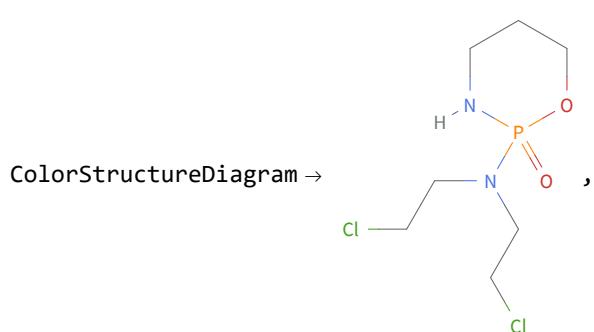
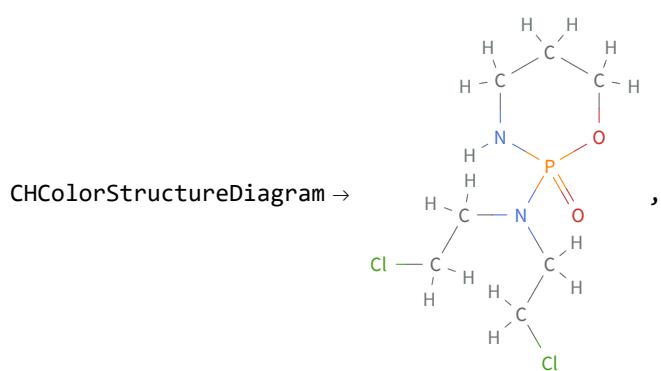
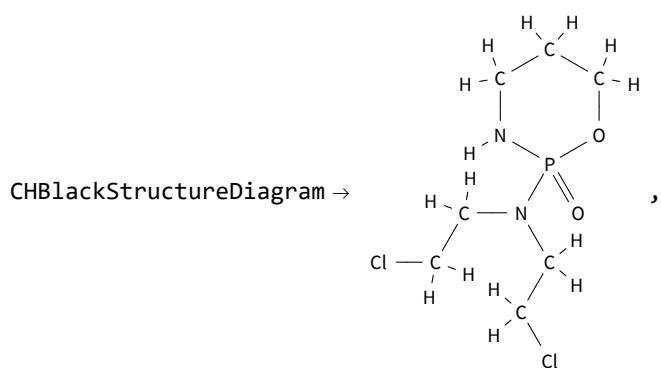
In[2]:= EntityValue[α-(tert-butyl)hydrocinnamic acid CHEMICAL,
 {"ElementCounts", "ElementMassFraction"}, "PropertyAssociation"]
Out[2]= {ElementCounts → {carbon → 13, hydrogen → 18, oxygen → 2}, ElementMassFraction → {carbon → 75.69%, hydrogen → 8.80%, oxygen → 15.512%} }

In[3]:= EntityValue[sulfuric acid CHEMICAL, {"IonCounts", "IonEquivalents"}]
Out[3]= {IonHydrogen → 2, IonSulfate → 1}
```

Построим различные структурные диаграммы для молекулы:

```
In[4]:= cyclophosphamide CHEMICAL [
 {"BlackStructureDiagram", "CHBlackStructureDiagram", "CHColorStructureDiagram",
 "ColorStructureDiagram", "LewisDotStructureDiagram"}, "Association"]
Out[4]= {BlackStructureDiagram →
```

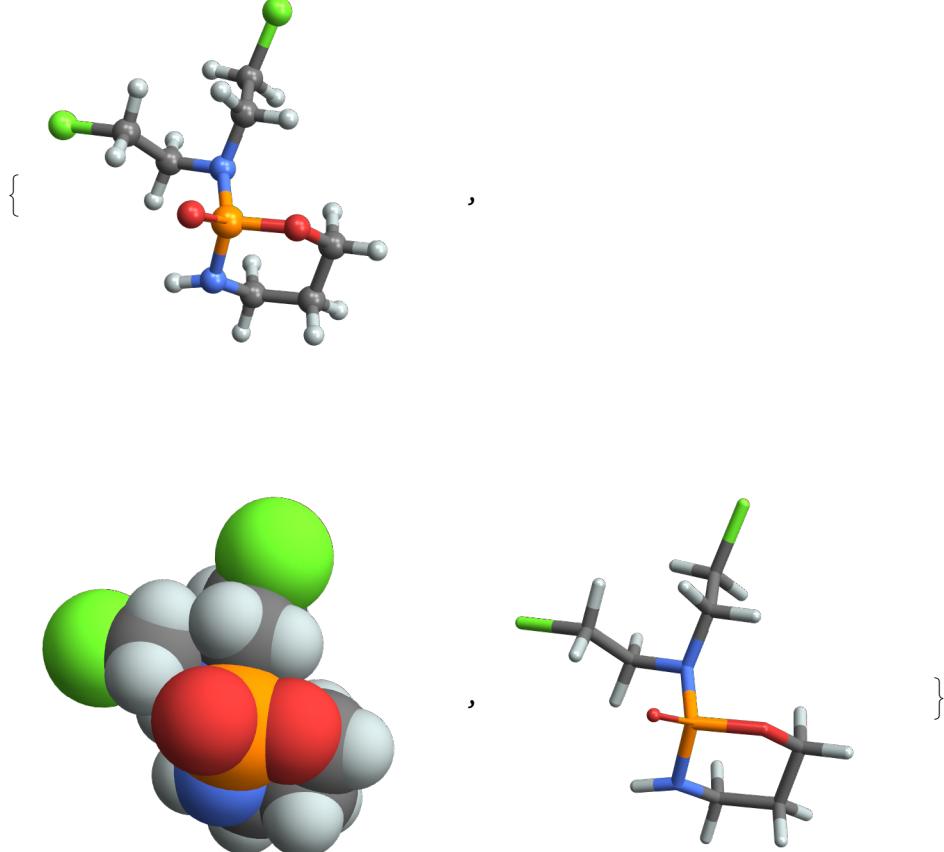




Построим различные пространственные модели молекулы:

```
In[8]:= cyclophosphamide CHEMICAL [
  {"MoleculePlot", "SpaceFillingMoleculePlot", "StickMoleculePlot"}]
```

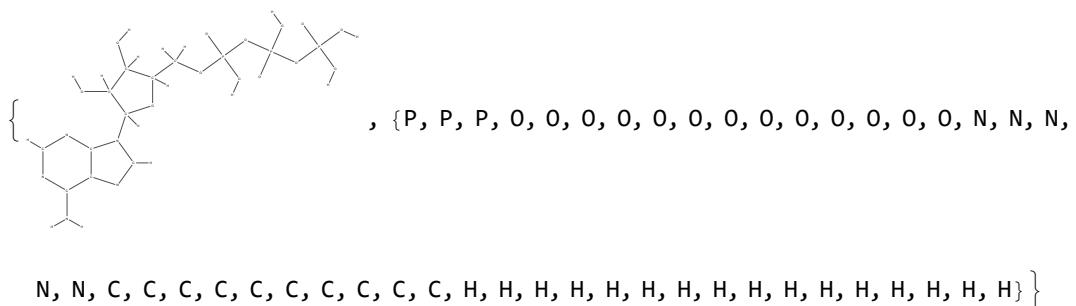
Out[8]=



Построим граф молекулы АТФ и найдём, какие атомы находятся в вершинах:

```
In[9]:= adenosine triphosphate CHEMICAL [{"StructureGraph", "VertexTypes"}]
```

Out[9]=



Представим результаты значений свойства для сущностей класса:

```
In[=]:= EntityValue[ "butyl hydroxide" CHEMICALS , { "k_njYpk_Yqq" , "k_njYptnjsk_c" , "k_njcasjYpk_Yqq" } , "PropertyEntityAssociation" ]
Out[=]= <| "k_njYpk_Yqq" ->
  <| "1-butanol" -> 74.12 g/mol , "sec-butanol" -> 74.12 g/mol , "isobutyl alcohol" -> 74.12 g/mol ,
    "(R)-2-butanol" -> 74.12 g/mol , "(S)-(+)-2-butanol" -> 74.12 g/mol , "t-butanol" -> 74.12 g/mol |> ,
  "k_njYptnjsk_c" -> <| "1-butanol" -> 91.5099 cm3/mol , "sec-butanol" -> 91.7364 cm3/mol ,
    "isobutyl alcohol" -> 92.3076 cm3/mol , "(R)-2-butanol" -> 91.8501 cm3/mol ,
    "(S)-(+)-2-butanol" -> 92.3076 cm3/mol , "t-butanol" -> 94.0 cm3/mol |> ,
  "k_njcasjYpk_Yqq" -> <| "1-butanol" -> 74.12 u , "sec-butanol" -> 74.12 u , "isobutyl alcohol" -> 74.12 u ,
    "(R)-2-butanol" -> 74.12 u , "(S)-(+)-2-butanol" -> 74.12 u , "t-butanol" -> 74.12 u |> |> |>
```

Получим набор данных всех доступных свойств для сущности:

```
In[=]:= Entity["Chemical", "DimethylSulfoxide"] ["Dataset"]
```

```
Out[=]=
```

Найдём химические вещества с наибольшей молярной массой:

```
In[]:= EntityClass["Chemical", {"MolarMass" → TakeLargest[5]}] // EntityList
Out[=]= {antihemophilic factor, botulinum toxin type b, botulinum toxin type a, eculizumab, muromonab}
```

Найдём единицу измерения свойства:

In[1]:= **EntityValue**["Unit"]

Out[1]=

Angstroms^2

Ножки глглжт

Получим набор данных всех основных свойств химического вещества:

In[2]:= **EntityValue**["ammonia" , "Dataset"]

Out[2]=

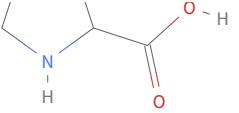
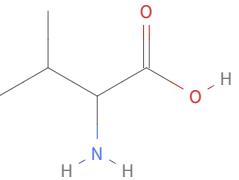
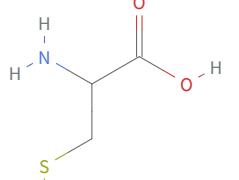
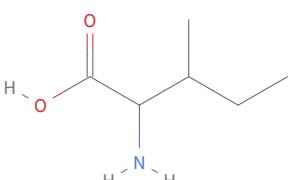
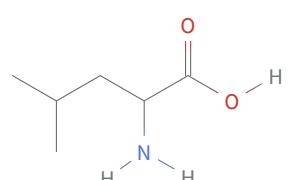
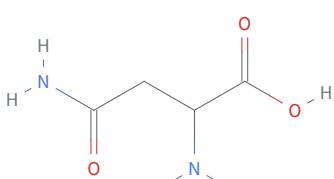
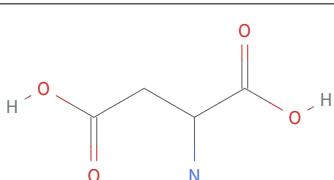
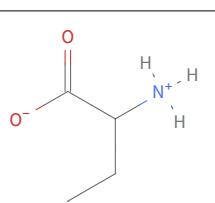


Исследуем структуру молекул аминокислот:

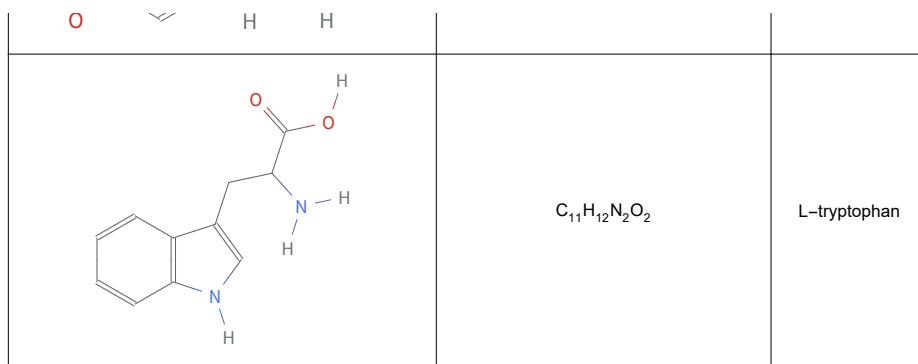
GraphicsGrid[**EntityValue**[**EntityClass**["Chemical" , "AminoAcids"] , { "ColorStructureDiagram" , "Formula" , "Name" }] , **Frame** → All]

Out[2]=

	<chem>NH2CH2COOH</chem>	glycine
	<chem>C3H7NO2</chem>	L-alanine
	<chem>C3H7NO3</chem>	L-serine
	-	

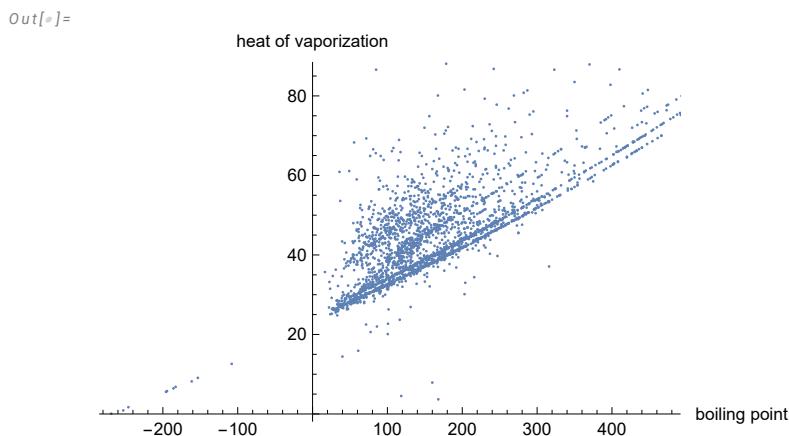
	$C_5H_9NO_2$	L-proline
	$C_5H_{11}NO_2$	L-valine
	$C_4H_9NO_3$	L-threonine
	$HSCH_2CH(NH_2)CO_2H$	L-cysteine
	$C_6H_{13}NO_2$	L-isoleucine
	$(CH_3)_2CHCH_2CH(NH_2)CO_2H$	L-leucine
	$C_4H_8N_2O_3$	L-asparagine
	$HO_2CCH_2CH(NH_2)CO_2H$	L-aspartic acid
	$C_5H_{10}N_2O_3$	L-glutamine

	$\text{H}_2\text{N}(\text{CH}_2)_4\text{CH}(\text{NH}_2)\text{CO}_2\text{H}$	L-lysine
	$\text{HO}_2\text{CCH}_2\text{CH}_2\text{CH}(\text{NH}_2)\text{CO}_2\text{H}$	L-glutamic acid
	$\text{C}_5\text{H}_{11}\text{NO}_2\text{S}$	L-methionine
	$\text{C}_6\text{H}_9\text{N}_3\text{O}_2$	L-histidine
	$\text{C}_6\text{H}_5\text{CH}_2\text{CH}(\text{NH}_2)\text{CO}_2\text{H}$	L-phenylalanine
	$\text{H}_2\text{NC}(=\text{NH})\text{NH}(\text{CH}_2)_3\text{CH}(\text{NH}_2)\text{CO}_2\text{H}$	L-arginine
	$(\text{HO})\text{C}_6\text{H}_4\text{CH}_2\text{CH}(\text{NH}_2)\text{CO}_2\text{H}$	L-tyrosine



Построим график для списка значений свойств:

```
In[1]:= ListPlot[EntityValue[liquid compounds CHEMICALS, {"BoilingPoint", "VaporizationHeat"}],  
AxesLabel -> {"boiling point", "heat of vaporization"}]
```



Химические законы FamousChemistryProblem

Entity["FamousChemistryProblem", / Yk c] жкж name FAMOUS CHEMISTRY PROBLEM
представляют сущность типа "FamousChemistryProblem" с наименованием / Yk c

В группах по темам

Сущности из типа "FamousChemistryProblem" включают в себя известные и поименованные результаты, законы, отношения и принципы в химии.

Классы сущностей "FamousChemistryProblem" группируют сущности:

- или по типу результата: например, [chemical effects](#), [chemical paradoxes](#)
- или по природе вещества, к которому они применяются: например, [gas laws](#),
- [Faraday's laws of electrolysis](#).

Встроенные объекты "FamousChemistryProblem" можно указывать на естественном языке с помощью [ctrl =](#):

ideal gas law →

In[1]:= **ideal gas law** CHEMISTRY PROBLEM

Out[1]=

ideal gas law

Список всех классов в типе “**FamousChemistryProblem**” в *W*-базе сущностей можно получить с помощью EntityClassList [“**FamousChemistryProblem**”].

Образцы классов сущностей в типе “**FamousChemistryProblem**”:

In[2]:= **EntityValue** [“**FamousChemistryProblem**”, “**SampleEntityClasses**”]

Out[2]=

{ **chemical effects** , **Faraday's laws of electrolysis** , **Fick's laws of diffusion** ,
gas laws , **laws of chemistry** , **chemical models** , **chemical paradoxes** ,
chemical phenomena , **chemical relations** , **chemical theorems** }

Образцы сущностей в типе “**FamousChemistryProblem**”:

In[3]:= **EntityValue** [“**FamousChemistryProblem**”, “**SampleEntities**”]

Out[3]=

{ **Amagat's law** , **Beer-Lambert law** , **Born-Oppenheimer approximation** ,
Boyle's law , **Fick's second law** , **Gibbs-Dalton law** , **Hess's law** ,
ideal gas law , **Le Châtelier's principle** , **Onsager reciprocal relations** }

Число сущностей в типе “**FamousChemistryProblem**”:

In[4]:= **EntityValue** [“**FamousChemistryProblem**”, “**EntityCount**”]

Out[4]=

22

Количество свойств, доступных для сущностей в “**FamousChemistryProblem**”:

In[5]:= **EntityValue** [“**FamousChemistryProblem**”, “**PropertyCount**”]

Out[5]=

22

Пам'ятні праці в типе “**FamousChemistryProblem**” (полученные с помощью команды Entity[“**FamousChemistryProblem**”][“Properties”]):

In[6]:= **Entity** [“**FamousChemistryProblem**”] [“**Properties**”]

Out[6]=

{ , , , ,
 , , , , , ,
 , , , , ,
 , , , , , , }

AdditionalPeople

дополнительно вовлеченные люди

AlternateNames

альтернативные наименования

AlternateStatements

альтернативные заявления

AssociatedEquations	связанные уравнения
AssociatedPeople	учёные, связанные с формулировкой и решением проблемы
Classes	классы
CurrentEvidence	текущие доказательства
Discoverers	первооткрыватели
DiscoveryDate	дата открытия
DiscoverySource	источник открытия
EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей, к которому принадлежит объект
FormulationDate	дата формулировки
FormulationSource	источник формулировки
Formulators	составители формул
Limitations	ограничения
Name	наименование
PrizeAwards	врученные награды
ProofDate	дата проверки
ProofSource	источник доказательства
Provers	испытатели
Statement	состояние

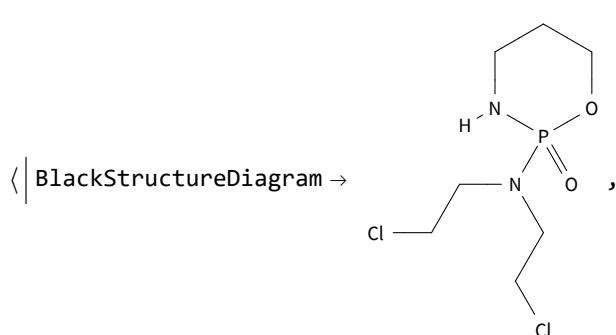
Нажмите

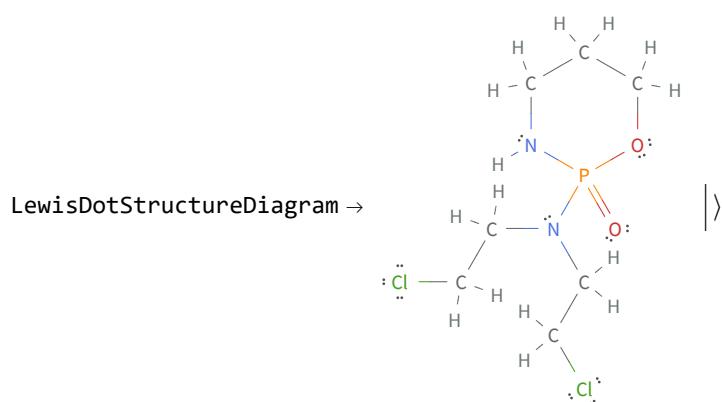
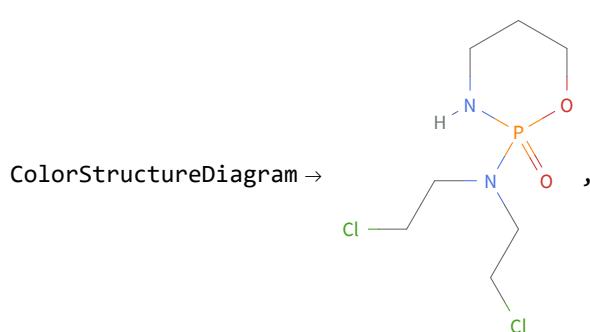
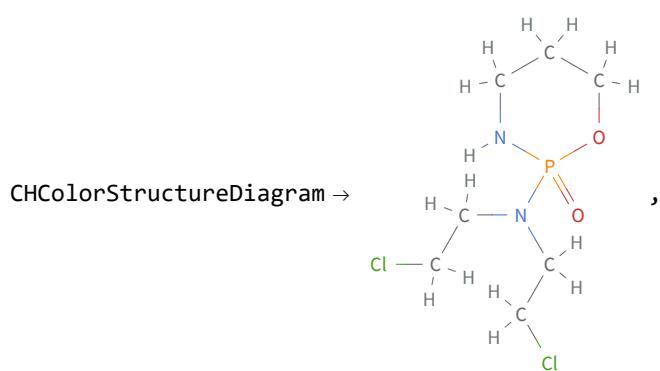
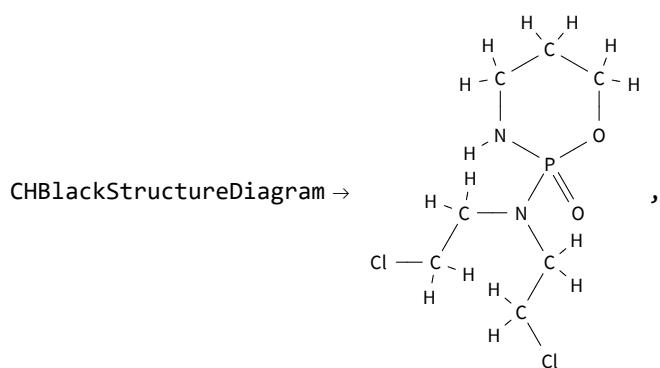
Найдём значение свойства для объекта:

```
In[1]:= Boyle's law CHEMISTRY PROBLEM ["AssociatedEquations"]
Out[1]= {p V == constant}
```

Построим различные структурные диаграммы для молекулы:

```
In[2]:= cyclophosphamide CHEMICAL [
  {"BlackStructureDiagram", "CHBlackStructureDiagram", "CHColorStructureDiagram",
   "ColorStructureDiagram", "LewisDotStructureDiagram"}, "Association"]
Out[2]=
```

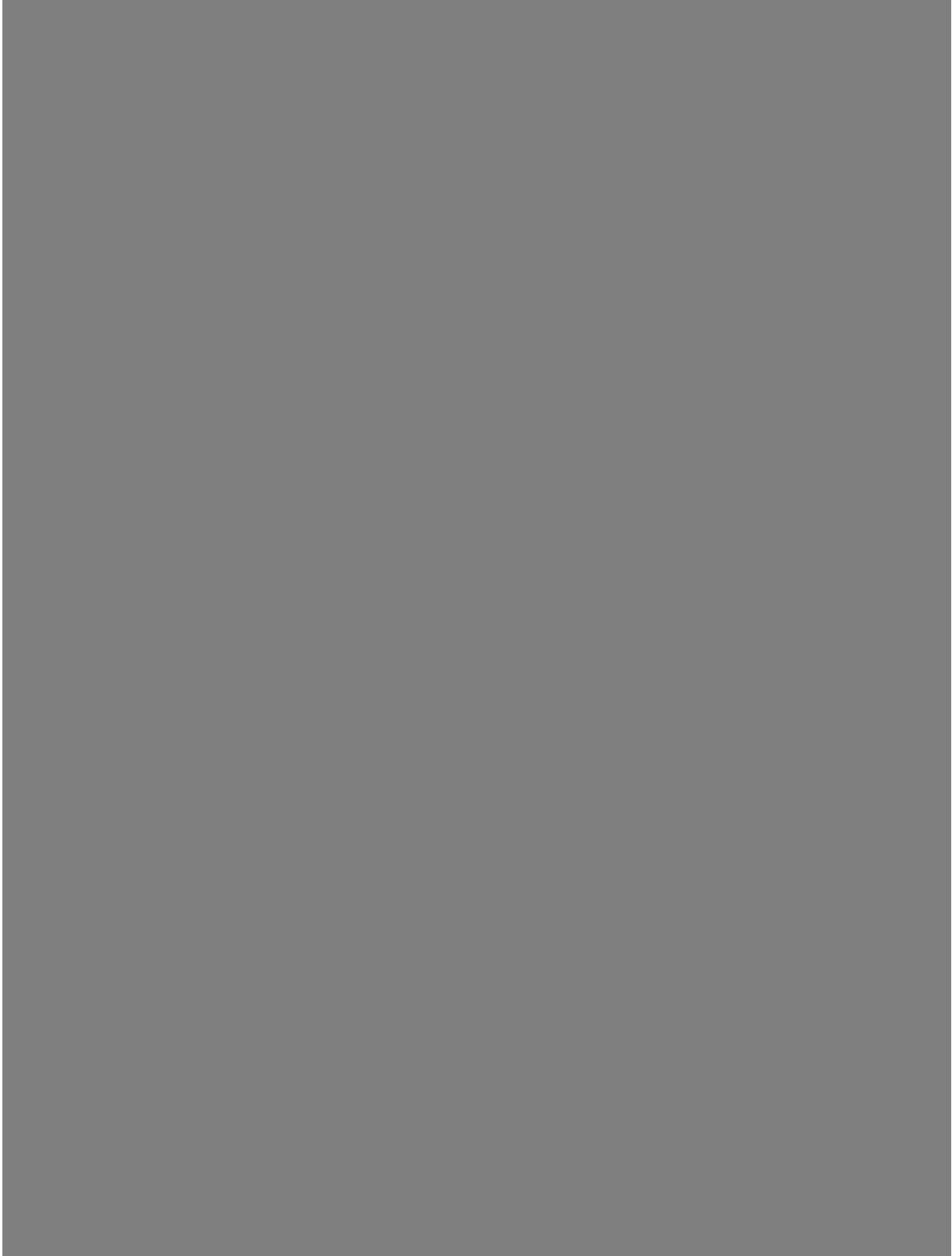




Получим набор данных всех доступных свойств для сущности:

In[8]:= Avogadro's law CHEMISTRY PROBLEM ["Dataset"]

Out[8]=



Получим газовые законы:

```
In[1]:= EntityClass["FamousChemistryProblem",
  "Classes" → ContainsAll[{ "GasLaw"}]] // EntityList
```

```
Out[1]= {Amagat's law, Avogadro's law, Boyle's law, Charles's law,
combined gas law, Dalton's law, Gibbs-Dalton law, Henry's law,
ideal gas law, law of combining volumes, pressure-temperature law}
```

Белок

Protein

Entity["Protein", / Yк с] жжк **name** PROTEIN
представляют сущность типа "Protein" с наименованием / Yк с

В группах по памяти

Каждая из сущностей "Protein" представляет собой определенный известный белок.

Классы сущностей "Protein" группируют сущности:

- или по биологическому процессу, в котором они участвуют: например, **actin filament binding** PROTEINS,

fever PROTEINS;

- или по более крупной структуре, к которой они принадлежат: например,

6 phosphofructokinase complex PROTEINS, **actin cytoskeleton** PROTEINS.

Встроенные объекты "Protein" можно указывать на естественном языке с помощью **ctrl =**:

acan →

```
In[2]:= aggregcan isoform 1 precursor PROTEIN
```

```
Out[2]=
```

aggregcan isoform 1 precursor

Список всех классов в типе "Protein" в UJ-базе сущностей можно получить с помощью **EntityclassList** ["Protein"].
Образцы классов сущностей в типе "Protein":

```
In[3]:= EntityValue["Protein", "SampleEntityClasses"]
```

```
Out[3]=
```

```
{af2 domain binding, arfgt pase activator activity,
atp synthesis coupled proton transport, autophagic vacuole membrane,
beta 14 mannosylglycoprotein 4 beta n acetylglucosaminyltransferase activity,
beta n acetylglucosaminylglycopeptide beta 14 galactosyltransferase activity, beta tubulin folding,
bleb, boron transport, brain specific angiogenesis inhibitor activity}
```

Образцы сущностей в типе “Protein”:

```
EntityValue["Protein", "SampleEntities"]
```

Out[*n*] =

```
{ Rho GDP dissociation inhibitor (GDI) alpha , breakpoint cluster region isoform 1 ,
breast cancer 1, early onset isoform 1 , CD1A antigen precursor , cyclin-dependent kinase 5 ,
fatty acid binding protein 1, liver , FK506-binding protein 4 , glycyl-tRNA synthetase ,
GM2 ganglioside activator precursor , integrin alpha M precursor }
```

Число сущностей в типе “Protein”:

```
In[n] := EntityValue["Protein", "EntityCount"]
```

Out[*n*] =

```
27479
```

Количество свойств, доступных для сущностей в “Protein”:

```
In[n] := EntityValue["Protein", "PropertyCount"]
```

Out[*n*] =

```
27
```

Пам'ятка по API в типе “Protein” (полученные с помощью команды Entity[“Protein”][“Properties”]):

AdditionalAtomTypes	дополнительные типы атомов
BallAndStickDiagram	шаростержневая модель молекулы
BiologicalProcesses	биологические процессы, в которых участвует белок
CellularComponents	клеточные компоненты, содержащие белок
ChainLabels	метки цепей белка
DomainIDs	доменные идентификаторы
DomainPositions	позиции домена
Domains	домены
EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей
Gene	ген белка
GenelD	идентификатор гена
GyrationRadius	радиус гирации (радиус вращения) белка
Memberships	членство
MolecularFunctions	молекулярные функции
MolecularWeight	молекулярный вес
Name	название
NCBIAccessions	NCBI идентификатор
PDBIDList	список PDBID
PrimaryPDBID	первичный PDBID

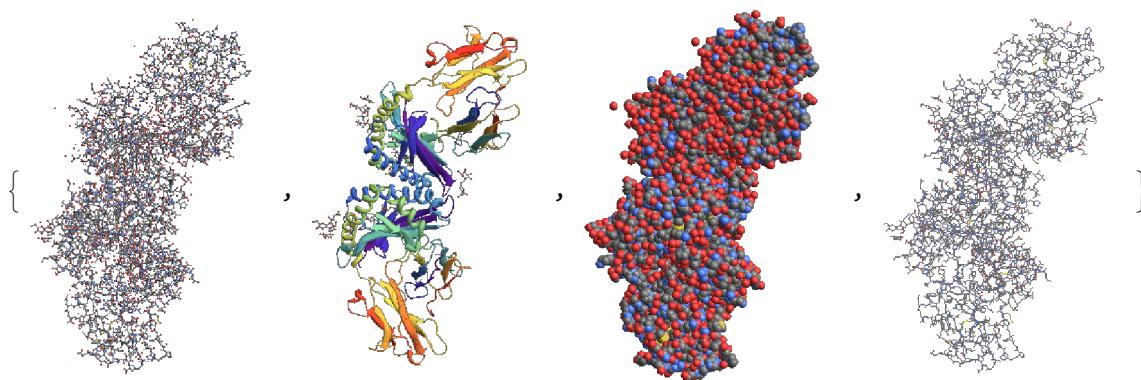
PrimaryPDBURL	основной URL-адрес PDB
SecondaryStructureRules	правила для вторичной структуры
Sequence	последовательность
SequenceLength	длина референсной последовательности гена
SpacefillingDiagram	диаграмма для модели заполнения пространства
StructureDiagram	ленточная диаграмма
WireframeDiagram	каркасная диаграмма

Нажмите

Построим различные пространственные модели белка:

```
In[1]:= Entity["Protein", "CD1A"] [{"BallAndStickDiagram",
  "StructureDiagram", "SpacefillingDiagram", "WireframeDiagram"}]
```

Out[1]=



Представим результаты значений свойства для сущностей класса:

```
EntityValue[ fever PROTEINS , eclc , "EntityAssociation"]
```

Out[1]=

```
{| interleukin 1, alpha proprotein → 3552 , interleukin 1, beta proprotein → 3553 |}
```

Получим набор данных всех доступных свойств для сущности:

```
In[2]:= Entity["Protein", "BRCA1Isoform1"] ["Dataset"]
```

Out[^o]=



Определим важные белки, участвующие в дифференцировке нейронов:

```
In[=]:= EntityClass["Protein",
  "BiologicalProcesses" → MemberQ["NeuronDifferentiation"] ] // EntityList
Out[=]= {plasma membrane calcium ATPase 2 isoform 1, cyclin-dependent kinase 5,
sphingosine-1-phosphate receptor 1, ephrin receptor EphA2, empty spiracles homeobox 2,
GATA binding protein 2, homeo box HB9, T-cell leukemia homeobox 1, homeobox C8,
neurogenin 1, nuclear receptor subfamily 4, group A, member 2, Brn3b POU domain transcription factor,
mitogen-activated protein kinase kinase 1, tubulin, beta 2, superiorcervical ganglia, neural specific 10,
BR serine/threonine kinase 1, CCAAT/enhancer binding protein beta, empty spiracles homolog 1,
inhibitor of DNA binding 3, LIM homeobox transcription factor 1, beta, reticulon 1 isoform A,
SWI/SNF-related matrix-associated actin-dependent regulator of chromatin a1 isoform a,
B-cell translocation gene 2, N-ethylmaleimide-sensitive factor attachment protein, alpha,
LIM domain binding 1 isoform 3, BR serine/threonine kinase 2,
tubulin, beta, 4, neugrin, neurite outgrowth associated isoform 1,
B-cell translocation gene 4, neurogenin 2, myotrophin,
proprotein convertase subtilisin/kexin type 9 preproprotein, BarH-like homeobox 2, tubulin, beta 2B}
```

Найдём источники информации о свойстве:

```
In[=]:= ZgjnegrYj prmasqqq ["Source"]
Out[=]= {NCBI Reference Sequence Database, RCSB Protein Data Bank}
```

Гены для ряда модельных организмов

Gene

Entity["Gene", / Yk c] жкк **name** GENE
представляют сущность типа "Gene" с наименованием / Yk c

В группах по теме

Каждая из сущностей “Gene” представляет собой ген из модельного организма (человека, плодовой мушки, дрожжей, крысы и др.).

Данные о генах предоставляются только для некоторых известных модельных организмов.

Классы сущностей “Gene” соответствуют семействам генов, которые имеют сходство в структуре, функции, в клеточном контексте и эволюционной истории.

В свойствах генов содержатся только те данные, которые можно надежно связать с референтными геномами, предоставленными Национальным центром биотехнологической информации (NCBI). Поэтому они могут не содержать последних результатов исследований.

Встроенные объекты “Gene” можно указывать на естественном языке с помощью `ctrl =`:

`In[1]:= Drosophila Tinman gene` →

`In[2]:= tinman (fruit fly) GENE` ✓

`Out[2]=`

`tinman (fruit fly)`

Список всех классов в типе “Gene” в *W*-базе сущностей можно получить с помощью `EntityclassList["Gene"]`.

Образцы классов сущностей в типе “Gene”:

`In[3]:= EntityValue["Gene", "SampleEntityClasses"]`

`Out[3]=`

{`bend`, `voltage-gated ion channels`, `igko`, `la ribonucleoprotein domain family`,
`peptidyl arginine deiminases`, `phactr`, `prame`, `SRY (sex determining region y)-box genes`,
`zinc fingers, CCHC domain containing`, `zinc fingers, MYND-type`}

Образцы сущностей в типе “Gene”:

`In[4]:= EntityValue["Gene", "SampleEntities"]`

`Out[4]=`

{`ADP-ribosylation-like factor 6 interacting protein 6 (rat)`, `Atg5p (yeast)`,
`BCL2-antagonist/killer 1 (human)`, `cytochrome c oxidase subunit I (human)`, `F-box protein 8 (chimpanzee)`,
`CPS-53 (KpLE1) prophage; bactoprenol-linked glucose translocase (flippase) (EscherichiaColi)`,
`integrin, beta 2 (rat)`, `mitogen-activated protein kinase kinase 3 (cow)`,
`OTU domain, ubiquitin aldehyde binding 2 (cow)`, `tinman (fruit fly)`}

Число сущностей в типе “Gene”:

`In[5]:= EntityValue["Gene", "EntityCount"]`

`Out[5]=`

294 947

Количество свойств, доступных для сущностей в “Gene”:

```
In[1]:= EntityValue["Gene", "PropertyCount"]
Out[1]= 18
```

Доступные **иерархии памят** для сущностей в “Gene”:

```
In[2]:= EntityValue["Gene", "PropertyClasses"]
Out[2]= {{"jnaYrgt"}}
```

Памят праўпсч лімпрэз в типе “Gene” (полученные с помощью Entity[“Gene”][“Properties”]):

AlternateNames	альтернативные названия
BiologicalProcesses	название биологических процессов
CellularComponents	названия клеточных компонентов
Context	контекст
EntityClasses	в какой класс сущностей вложен объект
EntityTypeList	список типов сущностей
FullSequencePosition	положение хромосомы
GeneSymbol	символ гена
Length	длина гена
Locus	локус
MolecularFunctions	названия молекулярных функций
Name	название
ProteinMolecularWeight	молекулярный вес белка
Proteins	названия белков
ReferenceSequence	секвенированная последовательность
Species	название вида модельного организма
Strand	нить
SubLabel	модельный организм

Нажмите

Найдём значение свойства для объекта:

Entity[“Gene”, {"FBXO8", {"Species" → "PanTroglodytes"}}] (*выберем сущность*)

```
Out[1]= F-box protein 8(chimpanzee)
```

```
In[2]:= F-box protein 8(chimpanzee) GENE ["Locus"]
```

```
Out[2]= {chromosome 4}
```

```
In[3]:= alcohol dehydrogenase, iron containing, 1(human) GENE ["EntityClasses"]
```

```
Out[3]= {"alcohol dehydrogenases", "human chromosome 8 genes"}
```

Представим результаты вычисления свойства для класса сущности в виде ассоциации:

```
In[]:= EntityValue[ "acyl-CoA thioesterases" GENES, "ecl c jcl erf", "Association"]

Out[=]= {{"acyl-CoA thioesterase 11 (human)" → 86 611 bp, "acyl-CoA thioesterase 2 (human)" → 6588 bp}, {"acyl-CoA thioesterase 7 (human)" → 129 495 bp, "acyl-CoA thioesterase 8 (human)" → 15 689 bp}, {"acyl-CoA thioesterase 9 (human)" → 39 631 bp, "acyl-CoA thioesterase 13 (human)" → 38 035 bp}, {"acyl-CoA thioesterase 12 (human)" → 64 042 bp, "acyl-CoA thioesterase 4 (human)" → 4061 bp}, {"acyl-CoA thioesterase 1 (human)" → 6571 bp, "acyl-CoA thioesterase 6 (human)" → 3045 bp}}
```

Получим набор данных всех доступных свойств сущности:

```
Entity["Gene", {"Arl6ip6", {"Species" → "RattusNorvegicus"} } ] (*выберем сущность*)
```

```
Out[=]= ADP-ribosylation-like factor 6 interacting protein 6 (rat)
```

```
In[6]:= ADP-ribosylation-like factor 6 interacting protein 6 (rat) GENE ["Dataset"]
```

```
Out[6]=
```



Получим набор данных для класса свойств “Локации” для сущности:

```
In[7]:= EntityValue[ cyclin-dependent kinase 9 (chimpanzee) GENE , {"jnaYrgt"}, "Dataset"]
```

```
Out[7]=
```



Найдём источники информации о свойстве:

```
In[]:= jmasq ["Source"]
Out[=] = {NCBI Reference Sequence Database}
```

Алфавиты и соотношения, позволяющие строкам представлять биологические последовательности **BioSequenceType**

Entity["BioSequenceType", / *Yk c*] **жкж** **name** BIO SEQUENCE TYPE
представляют сущность типа "BioSequenceType" с наименованием / *Yk c*

В гру́ппах памз пра

Сущности "**BioSequenceType**" включают стандартизованные алфавиты, наиболее часто используемые при обработке биологических последовательностей.

Тип "**BioSequenceType**" в U-J базе сущностей представлен одним классом сущностей:

biomolecular sequence types BIOMOLECULAR SEQUENCE TYPES

Образцы сущностей в типе "**BioSequenceType**":

```
In[]:= EntityValue["BioSequenceType", "SampleEntities"]
Out[=] = {DNA, RNA, peptide, circular DNA, circular RNA, circular peptide}
```

Число сущностей в типе "**BioSequenceType**":

```
In[]:= EntityValue["BioSequenceType", "EntityCount"]
Out[=] = 6
```

Встроенные объекты "**BioSequenceType**" можно указывать на естественном языке с помощью **ctrl =**:

protein biosequence type →

```
In[]:= peptide BIO MOLECULAR SEQUENCE TYPE ✓
Out[=] = peptide
```

Количество свойств, доступных для сущностей в "**BioSequenceType**":

```
In[]:= EntityValue["BioSequenceType", "PropertyCount"]
Out[=] = 15
```

Памз пра́йсч лмпргз в типе "**BioSequenceType**" (полученные с помощью **Entity**["BioSequenceType"]["Properties"]):

AdditionalBondTypes	дополнительные типы связей
AllowedBondingRules	правила для допустимых связей

Alphabet	алфавит
AlphabetRules	правила алфавита
AlternateEncodingRules	альтернативные правила кодирования
BondTypes	типы связей
Caption	заголовок
CircularQ	является ли кругом
ComplementLetterRules	правила дополнения
DegenerateAlphabet	вырожденный алфавит
DegenerateLetterRules	правила для вырожденных букв
EntityTypeList	список типов сущностей
FullAlphabet	полный алфавит
Name	название
PrimaryStructureBondType	тип связи первичной структуры

В свойствах содержатся только те данные, которые можно надежно связать с референтными геномами, предоставленными Национальным центром биотехнологической информации (NCBI). Поэтому они могут не содержать последних результатов исследований.

Ножкгощ

Найдём значение свойства для объекта:

```
In[1]:= RNA BIOMOLECULAR SEQUENCE TYPE ["AlphabetRules"]
Out[1]= {A → adenine, C → cytosine, G → guanine, U → uracil}
```

Получим набор данных всех доступных свойств сущности:

```
In[2]:= Entity["BioSequenceType", "DNA"] (*выберем сущность*)
Out[2]= DNA
```

In[$\#$]:= **DNA** BIOMOLECULAR SEQUENCE TYPE ["Dataset"]

Out[$\#$]=



Ножкмд глжг

Сущности "BioSequenceType" можно использовать для построения выражений **BioSequence**:

In[$\#$]:= **BioSequence** [**DNA** BIOMOLECULAR SEQUENCE TYPE , "GCCACG"]

Out[$\#$]=

BioSequence [ Type: DNA Sequence
Content: GCCACG (6 letters)]

Тип любого объекта `BioSequence` — это сущность “`BioSequenceType`”:

In[$\#$]:= `BioSequence` [ Type: DNA Sequence
Content: GCCACG (6 letters)] ["SequenceType"]

Out[$\#$]=

DNA

Ножки глыж

Создадим набор данных интерпретации для каждой буквы, разрешенной в последовательности РНК:

```
In[=]:= Dataset[Sort[
  Join[
    KeyValueMap[<|"Letter" → #1, "Interpretation" → StringRiffle[#2, " or "]|> &,
    RNA BIOMOLECULAR SEQUENCE TYPE ["DegenerateLetterRules"]], 
    KeyValueMap[<|"Letter" → #1, "Interpretation" → CommonName[#2]|> &,
    RNA BIOMOLECULAR SEQUENCE TYPE ["AlphabetRules"]]]]]
```

Out[=]=



различных биосистемах

Кодирование кодонов для перевода нуклеиновых кислот в белки в
GeneticTranslationTable

`Entity["GeneticTranslationTable", /Yk c] жкж name GENETIC TRANSLATION TABLE`

представляют сущность типа "GeneticTranslationTable" с наименованием /Yk c

В гру́кожу́фко памз пра

Сущности “**GeneticTranslationTable**” включают в себя соответствия кодонов и аминокислот, наиболее часто встречающихся в процессах генетической трансляции различных биологических систем.

Тип “**GeneticTranslationTable**” в *WJ*-базе сущностей представлен одним классом сущностей:

genetic translation tables GENETIC TRANSLATION TABLES

Образцы сущностей в типе “**GeneticTranslationTable**”:

```
In[]:= EntityValue["GeneticTranslationTable", "SampleEntities"]
```

Out[]:=

```
{standard, vertebrate mitochondrial, yeast mitochondrial, mold mitochondrial,
invertebrate mitochondrial, ciliate nuclear, echinoderm mitochondrial, euplotid nuclear,
bacterial, archaeal and plant plastid, alternative yeast nuclear, ascidian mitochondrial,
alternative flatworm mitochondrial, blepharisma macronuclear, chlorophycean mitochondrial,
trematode mitochondrial, scenedesmus obliquus mitochondrial, thraustochytrium mitochondrial,
pterobranchia mitochondrial, candidate division sr1 and gracilibacteria,
pachysolen tannophilus nuclear, karyorelict nuclear, condylostoma nuclear,
mesodinium nuclear, peritrich nuclear, blastocrithidia nuclear, balanophoraceae plastid}
```

Число сущностей в типе “**GeneticTranslationTable**”:

```
In[]:= EntityValue["GeneticTranslationTable", "EntityCount"]
```

Out[]:=

26

Встроенные объекты “**GeneticTranslationTable**” можно указывать на естественном языке с помощью :

→

```
In[]:= bacterial, archaeal and plant plastid GENETIC TRANSLATION TABLE
```

Out[]:=

bacterial, archaeal and plant plastid

Количество свойств, доступных для сущностей в “**GeneticTranslationTable**”:

```
In[]:= EntityValue["GeneticTranslationTable", "PropertyCount"]
```

Out[]:=

11

Памз пра́у псу́ч лмпргз в типе “**GeneticTranslationTable**” (полученные с помощью `Entity["GeneticTranslationTable"]["Properties"]`):

AlternateNames	альтернативные названия
AminoAcidTable	таблица аминокислот
CodonTranslations	трансляторы кодонов
EntityTypeList	список типов сущностей

Name	название
NCBIndex	индекс NCBI
NCBIVersion	версия NCBI
StartCodons	стартовые кодоны
StartingAminoAcidTable	начальная таблица аминокислот
StartingCodonTranslations	начальные кодоны трансляции
StopCodons	стоп-кодоны

В свойствах содержатся только те данные, которые можно надежно связать с референтными геномами, предоставленными Национальным центром биотехнологической информации (NCBI). Поэтому они могут не содержать последних результатов исследований.

Ножкиощ

Найдём значение свойства для объекта:

```
In[1]:= vertebrate mitochondrial GENETIC TRANSLATION TABLE ["CodonTranslations"]
Out[1]= <| TTT → F, TTC → F, TTA → L, TTG → L, TCT → S, TCC → S, TCA → S, TCG → S, TAT → Y, TAC → Y,
TAA → *, TAG → *, TGT → C, TGC → C, TGA → W, TGG → W, CTT → L, CTC → L, CTA → L,
CTG → L, CCT → P, CCC → P, CCA → P, CCG → P, CAT → H, CAC → H, CAA → Q, CAG → Q,
CGT → R, CGC → R, CGA → R, CGG → R, ATT → I, ATC → I, ATA → M, ATG → M, ACT → T,
ACC → T, ACA → T, ACG → T, AAT → N, AAC → N, AAA → K, AAG → K, AGT → S, AGC → S,
AGA → *, AGG → *, GTT → V, GTC → V, GTA → V, GTG → V, GCT → A, GCC → A, GCA → A,
GCG → A, GAT → D, GAC → D, GAA → E, GAG → E, GGT → G, GGC → G, GGA → G, GGG → G |>
```

Получим набор данных всех доступных свойств сущности:

```
In[2]:= Entity["GeneticTranslationTable", "YeastMitochondrial"] (*выберем сущность*)
```

yeast mitochondrial GENETIC TRANSLATION TABLE

In[\circ]:= yeast mitochondrial GENETIC TRANSLATION TABLE ["Dataset"]

Out[\circ]=



Ножкындағы жаңы

Сущности "GeneticTranslationTable" можно использовать в качестве аргумента для BioSequenceTranslate :

In[\circ]:= BioSequenceTranslate[

BioSequence["DNA", "CTGATATAC"], vertebrate mitochondrial GENETIC TRANSLATION TABLE]

Out[\circ]=

BioSequence[Type: Peptide Sequence
Content: LMY (3 letters)]

Сущности "GeneticTranslationTable" можно использовать в качестве спецификатора для некоторых модификаций, вносимых BioSequenceModify :

In[\circ]:= BioSequenceModify[

BioSequence["DNA", "ACTGATATAC"], {"DropToStartCodon", vertebrate mitochondrial GENETIC TRANSLATION TABLE }]

Out[\circ]=

BioSequence[Type: DNA Sequence
Content: ATATA (6 letters)]

Единичные вариации между референтным геномом и выборочными

популяциями

SNP

`Entity["SNP", / Yk c] жкж name SNP`

представляют сущность типа сущностей "SNP" с наименованием / Yk c

В группах памзра

Каждая из сущностей "SNP" представляет собой:

- или полиморфизм одного нуклеотида ;
- или различие в одной паре оснований между выборочной популяцией и эталонным геномом человека.

Сущности "SNP" представляются только для людей.

Сущности "SNP" будут включать только обновления по поддерживаемым референтным геномам. Поэтому они могут не содержать недавно опубликованные SNP.

Встроенные объекты "SNP" можно указывать на естественном языке с помощью `ctrl =`:`rs15291 →``█``Out[] =``rs15291`В типе "SNP" содержится один класс `SNPs`.Образцы сущностей в типе "SNP": `rs15291 SNP ... ✓``In[] := EntityValue["SNP", "SampleEntities"]``Out[] =``{ rs10132251 , rs11071174 , rs11666725 , rs1203140 ,
rs12716168 , rs12968956 , rs1323042 , rs13389833 , rs15291 }`Список всех классов в типе "SNP" в UJ-базе сущностей можно получить с помощью `EntityclassList["SNP"]`.

Число сущностей в типе "SNP":

`In[] := EntityValue["SNP", "EntityCount"]``Out[] =``8 439 598`

Количество свойств, доступных для сущностей в "SNP":

`In[] := EntityValue["SNP", "PropertyCount"]``Out[] =``9`

Доступные **икүйпіштіктер** для сущностей в “SNP”:

```
In[1]:= EntityValue["SNP", "PropertyClasses"]
Out[1]= {{"jnaYrgt"}}
```

Ламз праўпсч лампргз в типе “SNP” (полученные с помощью Entity[“SNP”][“Properties”]):

ChromosomePosition	расположение хромосомы
EntityClasses	в какой класс сущностей вложен объект
EntityTypeList	список типов сущностей
Gene	стандартное название гена
GeneLength	длина гена
GenePosition	расположение гена
Locus	локус
Name	название
WeightedAlleleFrequencies	взвешенные частоты аллелей

Нажмите

Найдём значение свойства для объекта:

```
In[2]:= Entity["SNP", "RS12716168"] ["Locus"]
Out[2]= {chromosome 5, p15}
```

Получим набор данных всех доступных свойств сущности:

```
In[3]:= Entity["SNP", "RS1323042"] (*выберем сущность*)
Out[3]= rs1323042
```

In[$\#$]:= **rs1323042 SNP** ["Dataset"]

Out[$\#$]=



Получим набор данных для класса свойств “Локации” для сущности:

In[$\#$]:= **EntityValue**[**rs12716168 SNP**, **jmaYrgt**, "Dataset"]

Out[$\#$]=



Найдём источники информации о свойстве:

In[$\#$]:= **jnasq** ["Source"]

Out[$\#$]=

{NCBI Reference Sequence Database}

классификации болезней (МКБ)

Заболевания и симптомы, указанные в Международной
Disease

Entity["Disease", / $Yk c$] **jjkj** **name** DISEASE

представляют сущность типа "Disease" с наименованием / $Yk c$

В группах пампра

Сущности из типа "Disease" включают заболевания, симптомы, различные медицинские состояния:

ICDNine153 == (рак толстой кишки);
 == (нарушение целостности раны).

Встроенные объекты "Disease" можно указывать на естественном языке с помощью `ctrl =`:

→

```
In[1]:= influenza DISEASE ... ✓
Out[1]= influenza
```

Список всех классов в типе "Disease" в УУбазе сущностей можно получить с помощью `EntityclassList["Disease"]`:

```
In[2]:= EntityclassList["Disease"]
Out[2]= {contagious diseases, coronavirus diseases, diseases}
```

Образцы сущностей в типе "Disease":

```
In[3]:= EntityValue["Disease", "SampleEntities"]
Out[3]= {chicken pox, latent schizophrenia, autistic disorder,
Parkinson's disease, narcolepsy, mononeuritis of lower limb,
rheumatic chorea, influenza, peripartum cardiomyopathy, common truncus}
```

Число сущностей в типе "Disease":

```
In[4]:= EntityValue["Disease", "EntityCount"]
Out[4]= 10924
```

Количество свойств, доступных для сущностей в "Disease":

```
In[5]:= EntityValue["Disease", "PropertyCount"]
Out[5]= 29
```

Пампра в типе "Disease" (полученные с помощью команды `Entity["Disease"]["Properties"]`):

Вълпшг нѣфкглрѣ

AgeMean	средний возраст пациента
AgeOfOnset	возраст начала заболевания

БҮЛЛҮҮГ МӨК МРОУЧ

BodyMassIndexMean	средний индекс массы тела
BodyTemperatureMean	средняя температура
DiastolicMean	среднее диастолическое артериальное давление
HeightMean	средний рост пациента
SystolicMean	среднее систолическое артериальное давление
WeightMean	средний вес пациента

МПМҮГЛЛМРЖЖҮМКГАҮЛЖӘЗ

Name	наименование
Definition	определение
Specialty	медицинская специальность

Causes	причины
CommonSymptoms	общие симптомы
Duration	продолжительность
ExposureToDiseaseOnset	причины начала заболевания
Prevalence	распространенность
PrevalenceRate	уровень распространенности
Prevention	профилактика
Risks	риски
Transmission	пути передачи инфекции
Treatment	лечение

AssociatedAnatomicalSite	связанные анатомические области
AssociatedGenes	ассоциированные гены
BasicReproductionNumber	основной номер воспроизведения

ICDNineCode	код МКБ - 9
ICDTenCode	код МКБ - 10
Image	изображение

ПАЭЖҮЛЛҮҮГ ПАМЗ ПРАЙ

EntityClasses	классы сущностей, в которые данная сущность вложена
EntityTypeList	список типов сущностей, к которому принадлежит объект

Для того, чтобы вычислить значения свойств заболевания только для пациентов, у которых оно является основным диагнозом, в канонических названиях заболеваний необходимо указать "Primary": Entity["Disease", {..., "Primary"}].

```
In[4]:= Entity["Disease", "ICDNine052"] [ {"AgeMean", "BodyTemperatureMean"}]
```

```
Out[4]=
```

```
{ 11. yr , 37. °C }
```

```
In[]:= Entity["Disease", {"ICDNine052", "Primary"}] [ {"AgeMean", "BodyTemperatureMean"}]  
Out[=]= {10. yr, 37. °C}
```

Ножкгощ

Найдём значение свойства для объекта:

```
In[=]= chicken pox DISEASE [ YtcdYec nYrgl r Yec ]  
Out[=]= 11. yr
```

Получим набор данных всех доступных свойств для сущности:

```
In[#]:= narcolepsy DISEASE ["Dataset"]
```

```
Out[#]=
```



Найдём пять болезней с самой высокой зарегистрированной средней температурой:

```
In[]:= EntityClass["Disease", {"BodyTemperatureMean" → TakeLargest[5]}] // EntityList
Out[=]= {intestinal infection due to rotavirus, viral enteritis,
unspecified type of osteomyelitis involving the ankle and foot,
acute vascular insufficiency of the intestine, postsurgical nonabsorption}
```

измерения	Распространенные медицинские лабораторные анализы и MedicalTest
-----------	--

```
Entity["MedicalTest", /Yk c] жжк name MEDICAL TEST
предоставляют сущность типа сущностей "MedicalTest" с наименованием /Yk c
```

В группах памяти

В типе "MedicalTest" группируются общие медицинские тесты и измерения, результаты которых можно использовать для различных медицинских целей, например, при диагностике заболеваний и скрининге. Сущности представляют:

- общие измерения: `waist circumference` MEDICAL TEST, `weight` MEDICAL TEST ... ;
- специализированные лабораторные тесты (например, различные анализы крови и мочи): `serum albumin`, `urine creatinine`.

Встроенные объекты "MedicalTest" можно указывать на естественном языке с помощью `ctrl =`:

Cholesterol →

```
In[=]= serum total cholesterol MEDICAL TEST ... 
Out[=]= cholesterol
```

Список всех классов в типе "MedicalTest" в базе сущностей можно получить с помощью `EntityclassList` ["MedicalTest"].

Образцы классов сущностей в типе "MedicalTest":

```
In[=]= EntityValue["MedicalTest", "SampleEntityClasses"]
Out[=]= {anemia screening, arterial blood gas, cardiac screens,
comprehensive metabolic panel, physical examination, hepatitis B screening,
hepatitis screening, lipid panel, NHANES Demographic, PSA screening}
```

Образцы сущностей в типе "MedicalTest":

```
In[]:= EntityValue["MedicalTest", "SampleEntities"]
Out[=] = {serum iron, serum rubella antibody test, serum LDL cholesterol,
          serum retinyl palmitate, serum retinyl stearate, serum selenium,
          serum total protein, systolic blood pressure, urine iodine, urine lead}
```

Число сущностей в типе "MedicalTest":

```
In[]:= EntityValue["MedicalTest", "EntityCount"]
Out[=] = 200
```

Количество свойств, доступных для сущностей в "MedicalTest":

```
In[]:= EntityValue["MedicalTest", "PropertyCount"]
Out[=] = 18
```

Памз праўпсч лмпргз в типе "MedicalTest" (полученные с помощью Entity["MedicalTest"]["Properties"]):

AlternateNames	альтернативные имена
BMDistributionPlot	график распределения ИМТ (индекс массы тела)
CPTCode	код СРТ
DistributionPlot	график распределения (гистограммы)
EntityClasses	в какой класс сущностей вложен объект
EntityTypeList	список типов сущностей
Mean	среднее значение
Median	медиана значений
Name	название
NHANESCode	код теста в NHANES
NHANESYears	цикли тестирования в NHANES
PlausibleRange	диапазон вероятностей
ReferenceRange90	90% референтный диапазон
ReferenceRange95	95% референтный диапазон
ReferenceRange99	99% референтный диапазон
SamplePopulationValue	выборочная совокупность
StandardDeviation	стандартное отклонение σ
StandardDeviationInterval	$\pm 1 \sigma$ диапазон

Иоўрий Э ўйойргожажел LF LCQ

Исследования в рамках Национальной программы социального исследования (National Health and Nutrition Examination Survey; NHANES) проводятся Национальным Центром статистики здравоохранения США с 1971 г. (с 1999 г. ежегодно) [www9]. Цель программы - среднестатистическая оценка состояния здоровья и питания населения всех возрастов.

Обследование включает в себя медицинские осмотры, лабораторные анализы, опросы, позволяющие получить сведения о демографии, социоэкономическом положении населения и особенностях питания.

Полученная информация систематизируется и используется при анализе связей особенностей питания, образа жизни и заболеваний. Данные NHANES являются основой некоторых национальных американских стандартов (рост, вес, кровяное давление).

Исследователи могут получить доступ к данными NHANES, в том числе, к данным программы биообразцов NHANES (ДНК, сыворотки,

плазмы, мочи), и объединить их с собственными результатами.

Это может послужить базисом для разработки тестов, программ и услуг, связанных со здоровьем, питанием и воздействием окружающей среды. Такие исследования могут позволить решить многие медицинские, экологические и здравоохранительные проблемы.

Ножкгош

Найдём значение свойства для объекта:

```
In[]:= Entity["MedicalTest", "IronPanelSerumIron"] ["ReferenceRange90"]
```

```
Out[=]
```

(41.7038 to 133.922) $\mu\text{g}/\text{dL}$

Получим набор данных всех доступных свойств сущности:

```
In[]:= Entity["MedicalTest", "UrineLead"] (*выберем сущность*)
```

```
Out[=]
```

urine lead

In[⁶]:= **urine lead MEDICAL TEST** ["Dataset"]

Out[⁶]=



Получим альтернативные названия тестов из класса “Cardiac Screens”:

In[=]:= EntityValue[cardiac screens MEDICAL TESTS, Yjrcp Yrc l Yk cq, "Dataset"]

Out[=]=



Ножкаглжт

Найдём источники информации о свойстве:

In[=]:= Yjrcp Yrc l Yk cq ["Source"]

Out[=]=

{National Health and Nutrition Examination Survey, Ambulatory Health Care Data}

Найдём значение свойств, связанных с доступом к NHANES

In[=]:= EntityValue[hepatitis B screening MEDICAL TESTS, {LF LCQrcqr anbc, LF LCQrcqr awajcq}, "EntityPropertyAssociation"]

Out[=]=

$\langle \boxed{\text{serum or plasma hepatitis B core antibody}} \rightarrow \langle \boxed{\text{LF LCQrcqr anbc}} \rightarrow \text{LBXHBC},$
 $\boxed{\text{LF LCQrcqr awajcq}} \rightarrow \{ \text{Year: 2000}, \text{Year: 2002}, \text{Year: 2004}, \text{Year: 2006} \} \rangle,$
 $\boxed{\text{serum or plasma hepatitis B surface antibody}} \rightarrow \langle \boxed{\text{LF LCQrcqr anbc}} \rightarrow \text{LBXHBS},$
 $\boxed{\text{LF LCQrcqr awajcq}} \rightarrow \{ \text{Year: 2000}, \text{Year: 2002}, \text{Year: 2004}, \text{Year: 2006} \} \rangle,$
 $\boxed{\text{serum or plasma hepatitis B surface antigen}} \rightarrow \langle \boxed{\text{LF LCQrcqr anbc}} \rightarrow \text{LBDHBG},$
 $\boxed{\text{LF LCQrcqr awajcq}} \rightarrow \{ \text{Year: 2000}, \text{Year: 2002}, \text{Year: 2004}, \text{Year: 2006} \} \rangle \rangle$

Анатомические структуры человека

AnatomicalStructure

`Entity["AnatomicalStructure", /Yк с] жжк name ANATOMICAL STRUCTURE`

представляют сущность типа "AnatomicalStructure" с наименованием /Yк с

В группах по памяти

Сущности из типа "AnatomicalStructure" включают:

- системы и органы организма,
- отдельные скелетные структуры, мышцы, нервы и кровеносные сосуды.

Классы сущностей "AnatomicalStructure" охватывают органы и ткани, связанные структурно или функционально.

Встроенные объекты "AnatomicalStructure" можно указывать на естественном языке с помощью `ctrl =`:

`lung` →

`In[=]:= lung ANATOMICAL STRUCTURE`

`Out[=]=`

`lung`

Список всех классов в типе "AnatomicalStructure" в УИ-базе сущностей можно получить с помощью `EntityclassList["AnatomicalStructure"]`.

Образцы классов сущностей в типе "AnatomicalStructure":

`In[=]:= EntityValue["AnatomicalStructure", "SampleEntityClasses"]`

`Out[=]=`

`{ muscles , bones , organ systems , vertebrae , spinal nerves , sensory organs , joints , teeth , ribs , fingers }`

Образцы сущностей в типе "AnatomicalStructure":

`In[=]:= EntityValue["AnatomicalStructure", "SampleEntities"]`

`Out[=]=`

`{ heart , lung , brain , stomach , biceps brachii , skeleton , pancreas , kidney , cerebellum , alimentary system , hand , esophagus , sciatic nerve , femur , ankle , hepatic lymphatic chain , diaphragm , vertebral vein , renal artery , orbit }`

Число сущностей в типе "AnatomicalStructure":

`In[=]:= EntityValue["AnatomicalStructure", "EntityCount"]`

`Out[=]=`

`99 253`

Количество свойств, доступных для сущностей в "AnatomicalStructure":

`In[=]:= EntityValue["AnatomicalStructure", "PropertyCount"]`

`Out[=]=`

`139`

Памз пра \bar{y} псч лимпрз в типе “AnatomicalStructure” (полученные с помощью команды Entity[“AnatomicalStructure”][“Properties”]):

Жөнгпржт жи \bar{y} рмошц

Name	AssociatedICDNine
LatinName	AssociatedICDTen
AlternateNames	UMLSConceptID
Abbreviation	

Кмот мкмбжэз

Definition	Shape	TypicalNumber
------------	-------	---------------

Имкожигпрагллшг памз пра \bar{y} з

Depth	Volume	Density	Mass
Diameter	Width		WeightPercentageOfTotalBody
Length			

Жо \bar{y} м \bar{y} о \bar{y} д глж \bar{z}

BodyLocationImage	Graphics3D	Region	Typ-
calMorphology		RegionalLocationHighlightedImage	
Image		RegionalLocationImage	

Бгмк гро \bar{y} хглиж памз пра \bar{y} з

EnclosingCuboid	Graphics3DPrimitives	MeshArea	MinimumCentroid-
Sphere	RegionalParts		
EnclosingSphere		MeshRegion	SurfaceArea
	RegionBounds		
EnclosingSphereDiameter		MeshVolume	
	RegionCentroid		
EnclosingSphereRadius			

Тслифж

Function	FunctionalCategories
----------	----------------------

Мпрмбглгж

DevelopmentalOrigin	GermOrigin
---------------------	------------

Просирсош

AnatomicalGroup	BilateralComponents	AdjacentStructures	Exter-
nalStructure	PrimarySegmentalSupply		

AnatomicalRegion larArchitecture AnatomicalSpace nalStructure StructuralGroup byStructures StructureType ContinuousStructure SystemicGroup	ConstitutionalParts SecondarySegmentalSupply IncludedOrganCount SegmentalSupply IncludedOrgans Members	AttachedStructure ContinuousStructures DerivedStructures DistallyContinuousStructure DividingStructure	Fascicu- Inter- Near- Proximally-
CellCountcell	CellSurfaceSpecialization		
AfferentVessel ArterialSupply EfferentVessel	IncludedArteries IncludedVeins Tributaries	VenousDrainage VenousSource	
AdjacentJointCount AdjacentJoints IncludedJointCount IncludedJoints	ArticulatingBones BoneArticulation BoneArticulationCount	IncludedBoneCount IncludedBones	IncludedCartilages IncludedLigaments IncludedTendons
AttachedMuscleCount AttachedMuscles	IncludedMuscleCount IncludedMuscles InsertedMuscleCount InsertedMuscles	MuscleAction MuscleAntagonist MuscleAttachmentSites MuscleInsertion MuscleOrigin	OriginatedMus- OriginatedMus-
Branches	ContinuationBranches		
Axons Dendrites	IncludedNerves	NerveSupply NeuronalInput	Innervations PrimaryInnervations
Neurons tions		NeuronalOutput	SecondaryInnerva- SpinalNerveInnerva- tions
AssociatedCerebralLobe	AssociatedCorticalArea		
ReceptorType	SensoryFiberType	Stimulus	
IncludedLymphaticVessels	LymphaticDrainage LymphaticSource		

Паэжъллшц памз праў

InputPathway OutputPathway	PathwayComponents PathwayDestination	Root
-------------------------------	---	------

	PathwayDiagram PathwayDirection PathwayOrigin PathwayProjectionSide	
SiteOfDecussion	Stem	Supplies
Contents	содержание	
ContextualizingEntity	контекстуализирующая сущность	
EntityClasses	классы сущностей, в которые данная сущность вложена	
EntityTypeList	список типов сущностей, к которому принадлежит объект	

Некоторые свойства изображений принимают квалификаторы “Qualifier”:

```
In[1]:= EntityValue[g Yec, {"Qualifiers", "QualifierValues"}]
Out[1]= {{Size}, {Size → {LargeThumbnail, MediumThumbnail, SmallThumbnail}}}

In[2]:= EntityValue[μεγι Yj jnəYrgn g Yec, {"Qualifiers", "QualifierValues"}]
Out[2]= {{View}, {View → {All, Back, Bottom, Front, Left, Right, Top}}}

In[3]:= EntityValue[rwngYjk npnf njnew, {"Qualifiers", "QualifierValues"}]
Out[3]= {{View}, {View → {All, Back, Bottom, Front, Left, Right, Top}}}
```

Для группировки объектов сущностей “AnatomicalStructure” можно использовать свойства других типов сущностей, в том числе “AnatomicalFunctionalConcept”, “Neuron”.

Ножки гощ

Найдём значение свойства для объекта:

```
In[1]:= Entity["AnatomicalStructure", "Heart"]
Out[1]= heart

In[2]:= heart ANATOMICAL STRUCTURE ["LatinName"]
Out[2]= {cor}

In[3]:= heart ANATOMICAL STRUCTURE ["Density"]
Out[3]= 1.1 g/cm3
```

Найдём единицу измерения свойства:

In[[#]]:= **bcnrf** ["Unit"]

Out[[#]]=

Inches

Получим набор данных всех доступных свойств для сущности:

```
In[#]:= brain ANATOMICAL STRUCTURE ["Dataset"]
```

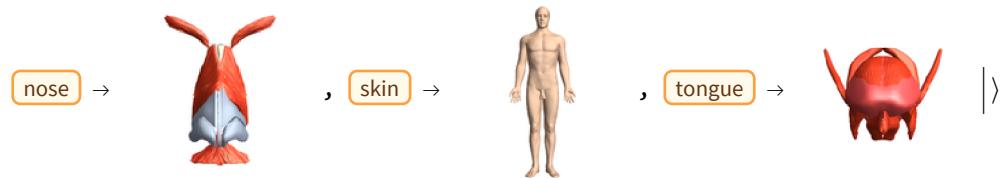
```
Out[#]=
```



Получим различные изображения для органов и систем:

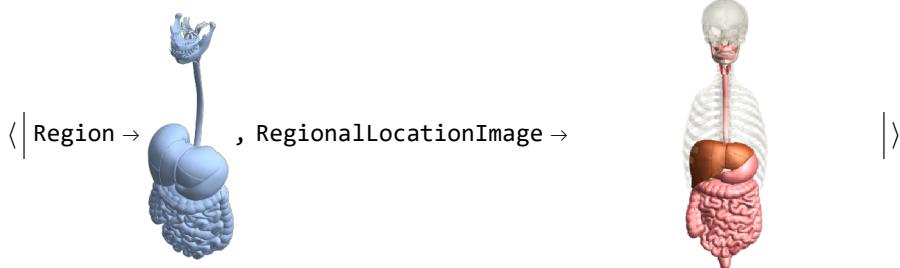
In[$\#$]:= sensory organs ANATOMICAL STRUCTURES ["Image", "EntityAssociation"]

Out[$\#$]=



In[$\#$]:= alimentary system ANATOMICAL STRUCTURE [{"Region", "RegionalLocationImage"}, "Association"]

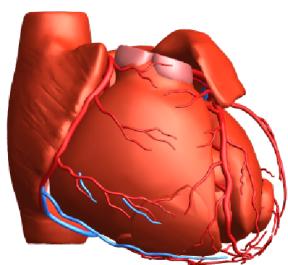
Out[$\#$]=



Визуализируем сущность:

In[$\#$]:= Entity["AnatomicalStructure", "Heart"]["TypicalMorphology"]

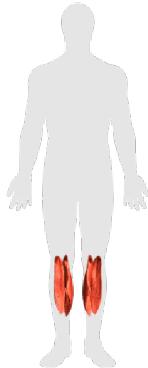
Out[$\#$]=



Найдём расположение сущности в теле:

In[$\#$]:= **gastrocnemius** ANATOMICAL STRUCTURE ["BodyLocationImage"]

Out[$\#$]=



Выделим объект в области тела:

In[$\#$]:= **left third metacarpal bone** ANATOMICAL STRUCTURE ["RegionalLocationHighlightedImage"]

Out[$\#$]=



Ножки глагла

Вычислим параметры Mesh-области:

In[$\#$]:= **left humerus** ANATOMICAL STRUCTURE ["MeshRegion"]

Out[$\#$]=



In[$\#$]:= **left humerus** ANATOMICAL STRUCTURE [{"MeshArea", "MeshVolume"}]

Out[$\#$]=

$$\{ 28181.9 \text{ mm}^2, 155676. \text{ mm}^3 \}$$

Найдём 5 самых крупных по весу анатомических структур:

```
In[]:= EntityClass[ "anatomical structures" ANATOMICAL STRUCTURES , {"WeightPercentageOfTotalBody" → TakeLargest[5]} ] // EntityList
Out[]= {cardiovascular system, muscle, bone, skeletal system, skin}
```

Найдём органы пищеварения:

```
In[]:= EntityValue[ "Entities" ]
Out[]= {dentition, esophagus, gallbladder, large intestine,
liver, pancreas, pharynx, small intestine, stomach, tongue}
```

Найдём предполагаемую общую длину пищеварительного тракта:

```
In[]:= EntityValue[ EntityGroup[ {pharynx ANATOMICAL STRUCTURE, esophagus ANATOMICAL STRUCTURE, stomach ANATOMICAL STRUCTURE,
small intestine ANATOMICAL STRUCTURE, large intestine ANATOMICAL STRUCTURE} ], "Length" ]
Out[=]
(8.6 to 9.8) m
```

Мпмъгллмпржножкглглж

Использование неявного класса сущностей:

Определим неявный класс сущностей и найдём анатомические структуры, которые функционируют при слуховом восприятии:

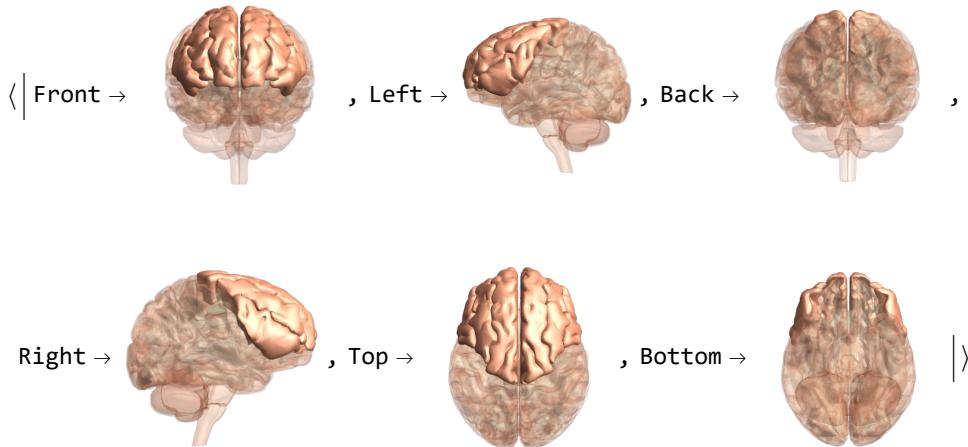
```
In[]:= EntityClass[ "anatomical structures" ANATOMICAL STRUCTURES , {"FunctionalCategories" →
{ "Sensory", "Motor" } } ] // EntityList
Out[=]
{auditory cortex, Brodmann area 32, cochlear nerve, cochlear nucleus,
ear, inferior colliculus, lateral lemniscus, left inferior colliculus,
left temporal lobe, medial geniculate body, right inferior colliculus,
right temporal lobe, superior olivary complex, temporal lobe, trapezoid body}
```

Применение квалификаторов “Qualifier”

Визуализируем сущность, рассматривая её с разных сторон:

```
In[]:= prefrontal cortex ANATOMICAL STRUCTURE ["RegionalLocationImage", {"View" → "All"}]
```

Out[]:=



Отношения с другими сущностями типа

Определим нейроны, обнаруженные в сетчатке:

```
In[]:= Entity["AnatomicalStructure", "Retina"] [  
  EntityProperty["AnatomicalStructure", "Neurons"]] // Shallow
```

Out[>//Shallow=

```
{ retina off-upilon ganglion cell , retina on-midget ganglion cell , retina amacrine cell , retina bipolar cell ,  
 retina ganglion cell , retina medium complex ganglion cell , retina medium simple ganglion cell ,  
 retina midget bipolar cell , retina photoreceptor L cone cell , retina photoreceptor M cone cell , <<31>> }
```

Отношения с сущностями другого типа

Найдём клетки вкусовых сосочков, используя свойство типа “Neuron”:

```
In[]:= taste bud type 1 cell NEURON [EntityProperty["Neuron", "SomaLocation"]]
```

Out[]:=

```
{ epiglottis , tongue , palate }
```

Найдём области мозга, связанные с принятием решений, используя свойство типа “AnatomicalFunctionalConcept”:

```
Entity["AnatomicalFunctionalConcept", "DecisionMaking"] [
  EntityProperty["AnatomicalFunctionalConcept", "AssociatedAnatomicalSites"]]
(*результатами являются сущности типа "AnatomicalStructure")
```

Out[=] =

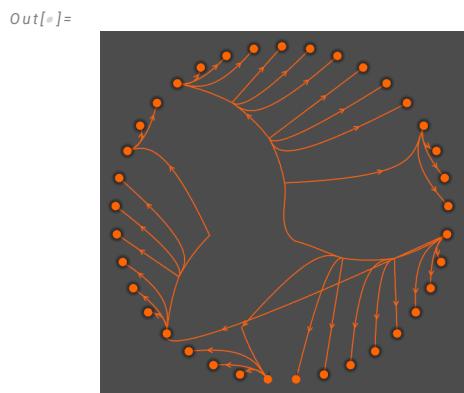
```
{frontal lobe, striatum, prefrontal cortex, anterior cingulate gyrus,
striatum of right cerebral hemisphere, Brodmann area 47, orbitofrontal cortex, Brodmann area 9}
```

Ножкамд глжг

Визуализируем ветви плечевой артерии:

In[=]:=

```
NestGraph[
 Cases[
 EntityValue[#, "Branches"], _Entity] &, brachial artery ANATOMICAL STRUCTURE,
 3, VertexLabels → Placed["Name", Tooltip], PlotTheme → "Marketing",
 GraphLayout → {"EdgeLayout" → "HierarchicalEdgeBundling"}, ImageSize → 300]
```



Визуализируем нервы, иннервирующие левую руку:

In[=]:=

```
nerve = Entity["AnatomicalStructure", "LeftHand"] [
 EntityProperty["AnatomicalStructure", "NerveSupply"]]
```

Out[=] =

```
{left median nerve, left radial nerve, left ulnar nerve}
```

```
AnatomyPlot3D[{nerve, Opacity[.3],  
skeleton of left free upper limb ANATOMICAL STRUCTURE , left hand ANATOMICAL STRUCTURE }]  
Out[=]
```



Анатомические функции человеческого тела AnatomicalFunctionalConcept

`Entity["AnatomicalFunctionalConcept", /Yk c] жәж name ANATOMICAL FUNCTIONAL CONCEPT`

представляют сущность типа "AnatomicalFunctionalConcept" с наименованием /Yk c

В группах пампра

Сущности из типа "AnatomicalFunctionalConcept" описывают:

- физиологические функции: действия мышц, движения суставов и т.п.;
- когнитивную деятельность.

Встроенные объекты "AnatomicalFunctionalConcept" можно указывать на естественном языке с помощью `ctrl =`:

`= visual perception →`

`In[=]:= visual perception ANATOMICAL FUNCTIONAL CONCEPT ... ✓`

Out[=]=

`visual perception`

Тип "AnatomicalFunctionalConcept" в UJ-базе сущностей включает один класс сущностей:

`In[=]:= EntityclassList["AnatomicalFunctionalConcept"]`

Out[=]=

`{ anatomical functional concepts }`

Образцы сущностей в типе “AnatomicalFunctionalConcept”:

```
In[]:= EntityValue["AnatomicalFunctionalConcept", "SampleEntities"]
Out[=] = {auditory tone detection, arithmetic processing, decision making,
feature comparison, flexion, language processing, speech perception,
visual face recognition, visual pattern recognition, working memory updating}
```

Число сущностей в типе “AnatomicalFunctionalConcept”:

```
In[]:= EntityValue["AnatomicalFunctionalConcept", "EntityCount"]
Out[=] = 919
```

Количество свойств, доступных для сущностей в “AnatomicalFunctionalConcept”:

```
In[]:= EntityValue["AnatomicalFunctionalConcept", "PropertyCount"]
Out[=] = 20
```

Пам'ятка в типе “AnatomicalFunctionalConcept” (полученные с помощью команды Entity[“AnatomicalFunctionalConcept”][“Properties”]):

Name	наименование
Definition	определение
AssociatedAnatomicalSites	связанные анатомические области
EntityTypeList	список типов сущностей, к которому принадлежит объект
StandardExperimentalTasks	стандартные экспериментальные задачи
BroaderConcepts	более широкие концепции
NarrowerConcepts	более узкие концепции
SubsetConcepts	подмножество концепций
SupersetConcepts	расширенные концепции

Жарг

ActivationAreas3DGraphic	3D-графика мозговой активности
BrainActivation3DGraphic	3D-изображение областей активации мозга
BrainGraphicBack	вид сзади
BrainGraphicBottom	вид снизу
BrainGraphicFront	фронтальный вид
BrainGraphicLeft	вид слева
BrainGraphicRight	вид справа
BrainGraphicTop	вид сверху
BrainImageCoronalSlices	коронарные срезы изображения мозга
BrainImageHorizontalSlices	горизонтальные срезы изображения мозга
BrainImageSagittalSlices	сагиттальные срезы изображения мозга

Нажмите

Найдём значение свойства для объекта:

```
In[1]:= Entity["AnatomicalFunctionalConcept", "Adduction"]
```

```
Out[1]=
```

adduction

```
In[2]:= adduction ANATOMICAL FUNCTIONAL CONCEPT ["Definition"]
```

```
Out[2]=
```

movement toward the midline of the body

Найдём описание свойства:

```
In[3]:= Yqymag'rcb Yl Yrnk gYj agcq ["Description"]
```

```
Out[3]=
```

associated anatomical sites

Получим набор данных всех доступных свойств для сущности:

```
In[4]:= visual face recognition ANATOMICAL FUNCTIONAL CONCEPT ["Dataset"]
```

```
Out[4]=
```





Найдём функции, связанные с областью Бродмана 4:

В качестве связанной анатомической области укажем сущность “Область Бродмана 4” из типа сущностей “*AnatomicalStructure*”:

```
In[1]:= EntityClass["AnatomicalFunctionalConcept", {"AssociatedAnatomicalSites" ->
    EntityValue[#, "AssociatedAnatomicalSites"] &}] // EntityList
```

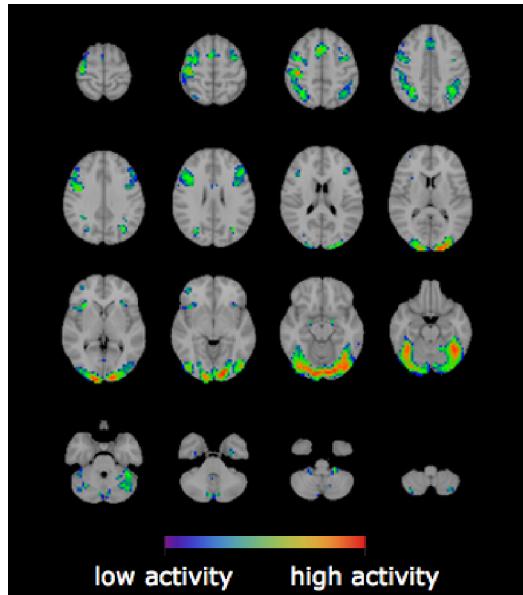
```
Out[1]= {attention, left-hand response execution, lexical encoding,
motor control, motor movement for speech, procedural memory}
```

Ножкаглглж

Получим изображения фрагментов мозговой активности, связанной с визуальным распознаванием лиц:

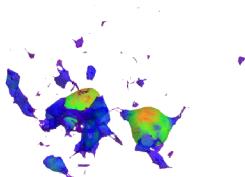
```
In[2]:= visual face recognition ANATOMICAL FUNCTIONAL CONCEPT [ ZpVg gk Yec f npgntrYj qjgacq ]
```

Out[2]=



Извлечение 3D-графики мозговой активности, связанной с обработкой эмоций:

```
In[=]:= emotion ANATOMICAL FUNCTIONAL CONCEPT [ YargYrgn! YpcYqzB epYnfg ] // Rasterize
Out[=]=
```



МплУгллмпржножжглглжэ

Использование неявного класса сущностей:

Определим неявный класс сущностей и найдём когнитивные функции, оцениваемые с помощью заданной стандартной экспериментальной задачи:

```
In[=]:= EntityClass["AnatomicalFunctionalConcept",
  {"StandardExperimentalTasks" → tone counting COGNITIVE TASK } ] // EntityList
Out[=]= { auditory tone discrimination , response selection ,
  visual number recognition , working memory maintenance , working memory updating }
```

Отношения с сущностями другого типа

Найдём области мозга, связанные с принятием решений, используя свойство типа "AnatomicalFunctionalConcept":

```
Entity["AnatomicalFunctionalConcept", "DecisionMaking"] [
```

```
EntityProperty["AnatomicalFunctionalConcept", "AssociatedAnatomicalSites"] ]
```

```
Out[=]= { frontal lobe , striatum , prefrontal cortex , anterior cingulate gyrus ,
  striatum of right cerebral hemisphere , Brodmann area 47 , orbitofrontal cortex , Brodmann area 9 }
```

Найдём анатомические структуры, связанные с обонянием:

olfaction ANATOMICAL FUNCTIONAL CONCEPT [*YqnaeYrcb Yl Ymk gYj qgcq*]

(*результатами являются сущности типа "AnatomicalStructure")

Out[*n*]=

```
{ diencephalon, mammillary body, nose, olfactory bulb,
  set of olfactory nerves, olfactory tract, Brodmann area 28, Brodmann area 44,
  Brodmann area 33, Brodmann area 45, Brodmann area 11, Brodmann area 32,
  Brodmann area 47, Brodmann area 24, olfactory cortex, primary olfactory cortex }
```

Найдём стандартные тесты, оценивающие способность извлекать информацию из памяти:

spatial working memory ANATOMICAL FUNCTIONAL CONCEPT [*qYl bYpb cvncpk cl rYj rYqj q*]

(*результатами являются сущности типа "CognitiveTask")

Out[*n*]=

```
{ spatial delayed response task, spatial n-back task,
  spatial span test, spatial working memory task, visual object learning test }
```

Нажмите для просмотра

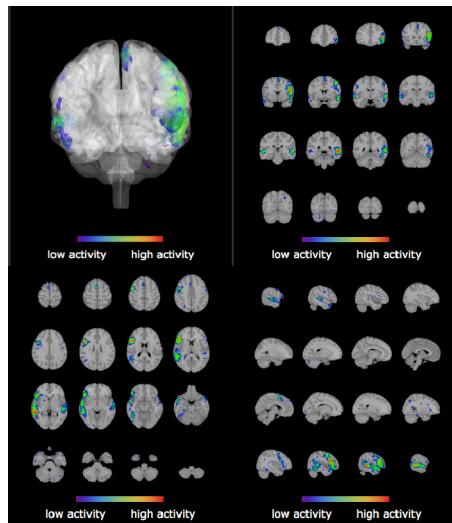
Визуализируем активность мозга, связанную с работой языка:

In[*n*]:= **ImageCollage**[

```
Map[ language ANATOMICAL FUNCTIONAL CONCEPT #[ &, {
  EntityProperty["AnatomicalFunctionalConcept", "BrainGraphicFront"],
  ZpYg gk Yec anpt Yj qgcq, ZpYg gk Yec fnpgrml rYj qgcq, ZpYg gk Yec qYegrYj qgcq }],
```

ImageSize → 350]

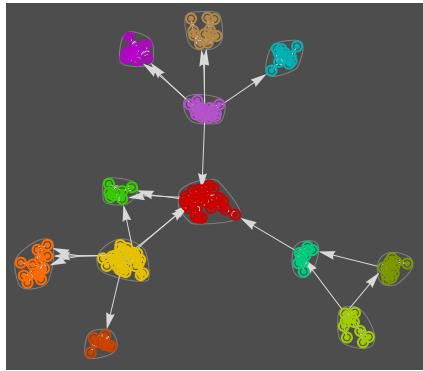
Out[*n*]=



Визуализируем категориальные связи исполнительных функций мозга:

```
With[{cog = {decision making ANATOMICAL FUNCTIONAL CONCEPT,
memory ANATOMICAL FUNCTIONAL CONCEPT, perception ANATOMICAL FUNCTIONAL CONCEPT}},
CommunityGraphPlot@NestGraph[
Cases[
EntityValue[#, EntityProperty["AnatomicalFunctionalConcept", "NarrowerConcepts"]], _Entity, Infinity] &, cog, 4, PlotTheme -> "Marketing", GraphHighlight -> cog,
GraphLayout -> "RadialEmbedding", VertexSize -> 1, EdgeStyle -> LightGray,
VertexStyle -> Orange, VertexLabels -> Placed["Name", Tooltip]]]
```

Out[*#*] =



Нервные клетки, обнаруженные в нервной системе человека Neuron

```
Entity["Neuron", /Yk c] жжк name NEURON
предоставляют сущность типа сущностей "Neuron" с наименованием /Yk c
```

В групкожъфю памз пра

К сущностям типа “Neuron” относятся сенсорные и двигательные нейроны, а также интернейроны нервной системы.

Встроенные объекты “Neuron” можно указывать на естественном языке с помощью `ctrl =`:

`neocortex basket neuron` →

In[*#*] = `neocortex basket cell` NEURON

Out[*#*] =

`neocortex basket cell`

Тип “Neuron” в UJ-базе сущностей представлен одним классом **neurons**.

```
In[1]:= EntityclassList["Neuron"]
```

```
Out[1]= {neurons}
```

Образцы сущностей в типе “Neuron”:

```
In[2]:= EntityValue["Neuron", "SampleEntities"]
```

```
Out[2]= {cerebellum basket cell, cerebellum granule cell, dentate gyrus basket cell, hippocampus CA1 pyramidal cell, hippocampus CA3 pyramidal cell, inferior olfactory nucleus principal neuron, neocortex basket cell, olfactory bulb (main) granule cell, retina bipolar cell, spinal cord ventral horn motor neuron}
```

Число сущностей в типе “Neuron”:

```
In[3]:= EntityValue["Neuron", "EntityCount"]
```

```
Out[3]= 344
```

Количество свойств, доступных для сущностей в “Neuron”:

```
In[4]:= EntityValue["Neuron", "PropertyCount"]
```

```
Out[4]= 35
```

Пам'ятка по типу “Neuron”:

```
In[5]:= Entity["Neuron"]["Properties"]
```

```
Out[5]= {YZZpxtgrgt, YtcpYec FNYk njgsbc, YtcpYec FNbsptrgt, YtcpYec aYnYagYiac, YtcpYec gnsrpxqgrYiac, YtcpYec k ck ZpYl crg c ant qYl r, YtcpYec pxqrg e k ck ZpYl c nmrl rgyj, YtcpYec pfcnZYqc, YtcpYec qng c Yk njgsbc, YtcpYec qng c fYjdil gfrf, YtcpYec qng c rfpxqf njb, YtcpYec qnt rYl cnsqdt e pYrc, Yvnt jcl erf, Yvnt k ucgj Yrgt, Yvnt npqeg, Yvnt npnkarqnt jYrcpljgw, ZpYl af g e rwnc, acjj qnk Y bgk crcp, acjj qnk Y qf Ync, acjj qnk Y qgc, acjjsjYpqnt Ynrga rYpecr, bcd#g e apgcpg, bcl bpgc jnqYrgt, bcqapqrgt, cl rgwrrwnc jgr, dgf e nYrrcpd q, gk Yec, jnqYrgt mdbgrYl r Yvnt YpZnpqgYrgt, jnqYrgt mdjnqYj Yvnt YpZnpqgYrgt, lYk c, lcspmplYl dk grcpqjcYqcb, lsk Zcpndnpk Ypbcl bpgcq, pjc, qnk Y jnqYrgt, qng c bcl qgwnt bcl bpgcq}
```

Нажмите

Найдём значение свойства для объекта:

```
In[]:= Entity["Neuron", "RetinaBipolarCell"]["AverageSpikeThreshold"]  
Out[=] = -63. mV
```

Получим набор данных всех доступных свойств сущности:

```
In[]:= Entity["Neuron", "HippocampusCA3PyramidalCell"] (*выберем сущность*)  
Out[=] = hippocampus CA3 pyramidal cell
```

```
In[8]:= hippocampus CA3 pyramidal cell NEURON ["Dataset"]
```

```
Out[8]=
```



Найдём нейроны пирамидальной формы и большого размера:

```
In[]:= EntityClass["Neuron",
  {"CellSomaSize" → "large", "CellSomaShape" → MemberQ["pyramidal"]}] // EntityList
Out[]= {amygdala basolateral nuclear complex pyramidal neuron, hippocampus CA1 pyramidal cell,
hippocampus CA3 pyramidal cell, neocortex primary motor area pyramidal layer 5 callosal cell,
neocortex primary motor area pyramidal layer 5 corticopontine-tectal cell,
neocortex primary motor area pyramidal layer 5 corticospinal cell,
neocortex primary motor area pyramidal layer 5 corticostriate cell,
neocortex pyramidal cell, neocortex pyramidal cell layer 5-6,
olfactory cortex deep pyramidal cell, olfactory cortex superficial pyramidal cell}
```

Найдём единицу измерения свойства:

```
In[]:= YtcpYec FNYk njgsbc ["Unit"]
Out[]= Millivolts
```

Asqmk iBcđ cb Cl rgwRuncq

НмкъямаЎргкъпиж ржнщпсч лмпргз

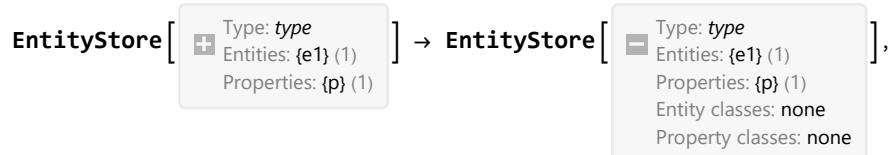
сущностей	EntityStore	Локальное или удаленное хранилище
	EntityRegister	Регистрация нового базового типа сущности
сущностей	EntityStores	Список всех зарегистрированных хранилищ
сущностей	EntityUnregister	Отмена регистрации сущностей в хранилище

EntityStore [" <i>rwns</i> "]	создаёт пустое хранилище сущностей для сущностей типа <i>rwns</i>
EntityStore [" <i>rwns</i> " → <i>bYrY</i>]	создаёт хранилище сущностей для сущностей типа <i>rwns</i> со свойствами и значениями, определяемыми данными <i>bYrY</i>
EntityStore [{ <i>rqns_a</i> , <i>rqns_b</i> , ...}]	создаёт хранилище сущностей для сущностей нескольких типов
EntityStore [<i>RelationalDatabase</i> [...]]	создаёт хранилище сущностей на основе схемы реляционной базы данных
EntityStore [{ <i>rqns_a</i> , <i>rqns_b</i> , ...}, <i>bZqnsa</i>]	создает хранилище сущностей путем сопоставления имен таблиц в базе данных, указанных в <i>bZqnsa</i> , с типами, указанными в <i>rqns_a</i>

Вгрўккюлфж памз пра

Если объект [EntityStore](#) включен в список [EntityStores](#)[], то типы сущностей, которые он содержит, доступны автоматически при использовании [Entity](#).

Объект [EntityStore](#) имеет следующий вид:



где указаны тип, наименования и свойства сущностей и классов сущностей, если они имеются.

Если разрешения не установлены иным образом, его содержимое может быть изменено назначениями формы $c/rvg [pmtcprv] = tYjsc$.

Доступ к данным в объекте `EntityStore` можно получить напрямую с помощью $qtrx [nzh, "pmtcprv"]$, где nzh – объект `Entity`, `EntityClass`, `EntityProperty`, `EntityPropertyClass` или `EntityType`.

В `EntityStore [{ rgncag, rgncav, ... }]` каждый из $rgnca_{\cdot}$ может быть либо “ $rgnsc_{\cdot}$ ”, либо “ $rws_{\cdot} \rightarrow bYrY_{\cdot}$ ”.

В `EntityStore [“rws” → bYrY]` данные – это ассоциация, которая определяет сущности, свойства и значения, связанные с указанным типом rws .

Элемент данных “`Entities`” может задавать связь со свойствами сущностей данного типа: “`Entities`” → < | “ $/ Yk c_{\cdot}$ ” → $pmtq_{\cdot}, \dots$ | >.

$pmtq_{\cdot}$ задают значения свойств для указанной сущности: < | “ $pmtq_1 \rightarrow tYj_{g1}$, “ $pmtq_2 \rightarrow tYj_{g2}, \dots$ | >.

Специальное свойство “`Label`” может быть задано для любой сущности, указывающее, как она должна отображаться.

Элемент данных “`Properties`” предоставляет информацию о свойствах, например, о том, как должны отображаться свойства и как должны вычисляться значения свойств, которые явно не указаны.

Возможные записи в ассоциации ***bYrY***:

“ <code>Entities</code> ”	ассоциация, дающая значения свойств для каждой сущности
“ <code>EntityClasses</code> ”	ассоциация, дающая сущности и свойства для классов сущностей
“ <code>Label</code> ”	метка для использования при отображении rws
“ <code>LabelPlural</code> ”	форма множественного числа метки, используемая при отображении rws
“ <code>Properties</code> ”	ассоциация, предоставляющая информацию о свойствах rws
“ <code>PropertyClasses</code> ”	ассоциация, предоставляющая информацию о классе, к которому принадлежит rws
“ <code>EntityTypes</code> ”	ассоциация, дающая свойства и значения для дочерних rws

Запись “`Properties`” в данных имеет вид < | “ pmt_y ” → < | $k_1 \rightarrow v_1, \dots$ | >, “ pmt_2 ” → ..., ..., | >.

Возможные записи в ассоциации, приведенной для каждого свойства:

“ <code>DefaultFunction</code> ”	функция по умолчанию для вычисления значения этого свойства
“ <code>Label</code> ”	метка для использования при отображении этого свойства

При настройке «`DefaultFunction`» → $\{$ для заданного свойства $\}$ применяется к сущности для вычисления значения этого свойства, если оно явно не указано в ассоциации «`Entities`».

Возможные записи для каждого класса сущностей в «`EntityClasses`» :

“ <code>Entities</code> ”	список имен или предикатов, определяющих сущности в классе
“ <code>Label</code> ”	метка для использования при отображении класса, к которому принадлежит сущность
сущность	

В ассоциации для определенного класса сущностей “`Entities`” → { “ $/ Yk c_{\cdot}$ ”, “ $/ Yk c_{\cdot}$ ”, ..., | > дает явный список сущностей в классе сущностей.

“Entities” → “*пред*” → “*пред*” указывает, что сущности в классе — это те, для которых *пред* дает True при применении к значению свойства *пред*.

Возможные записи для каждого класса свойств в «PropertyClasses»:

“Label”	метка, используемая при отображении этого класса свойств
“Properties”	названия свойств в этом классе свойств

Элемент данных “EntityTypes”:

- предоставляет информацию о дочерних типах данного типа сущности. Дочерний тип состоит из подмножества сущностей этого типа (родительского типа), к которому применяются дополнительные свойства.
- имеет форму <| “*afgb*” → *bYrY*, ... |>, где *bYrY*, соответствующие дочернему типу «*afgb*», имеют ту же форму, что и *bYrY*.

В *внхголж ржнц* наследуют сущности от своих родительских типов.

Свойства, классы сущностей и классы свойств дочернего типа “*afgb*” типа “*runc*” имеют форму:

EntityProperty [“*runc*” → “*afgb*”, ...], *EntityClass* [“*runc*” → “*afgb*”, ...] и *EntityPropertyClass* [“*runc*” → “*afgb*”, ...].

Более глубоко вложенные типы имеют форму “*runc*” → ... → “*epYl bafgb*”.

В *EntityStore* [{"*runc*” → *k crY*, “*runc*” → *k crY*, ...}, *bYrYZYqcqncs*], *k crY* может иметь форму:

{“ <i>anj</i> ”, “ <i>anj</i> ”, ...}	список наименований столбцов, которые будут отображены
{" <i>пред</i> ” → “ <i>anj</i> ”, ...}	“ <i>anj</i> ” переименован в “ <i>пред</i> ”
{..., “ <i>пред</i> ” → <i>EntityFunction</i> [...], ...}	“ <i>пред</i> ” вычисляется из <i>EntityFunction</i>
<i>ajYqf</i>	весь тип построен из вычисляемого класса сущностей

В *EntityStore* [{"..., “*runc*” → *ajYqf*, ...}], *ajYqf* может быть любой функцией из: *EntityClass*, *FilteredEntityClass*, *ExtendedEntityClass*, *SampledEntityClass*, *SortedEntityClass*,

AggregatedEntityClass, *CombinedEntityClass*.

Памз праў уоўлжжкі лсч лмпргз унмввгод жаўгк щу огкэфмплицк жўўжк жвўллщул

В хранилищах сущностей, поддерживаемых реляционными базами данных, структура первого аргумента *EntityStore* имеет вид:

<| “Types” -><|{“*runc*” → <|...|>, “*runc*” → <|...|>, ...|>|>,
где каждая ассоциация в правой части каждого “*runc*” может иметь следующие ключи:

“Properties”	ассоциация, определяющая свойства
“EntityTypeExtractor”	наименование таблицы или вычисляемого класса
“CanonicalNameProperties”	свойства, которые формируют <i>CanonicalName</i>

В реляционных базах данных *памз праў ўшадаў ррогу ажвма*: столбцы исходной базы данных, вычисляемые свойства или свойства отношения.

Памз праў прмкўф определяются с помощью ассоциации, содержащей:

“ColumnPrefix”	название таблицы (мЎцхлм)
“ColumnName”	название столбца

Ашкаптәгкүй памзпрау определяются с помощью ассоциации, содержащей:

“Function”	EntityFunction, определяющая свойство
------------	---------------------------------------

Памзпрау мрлмц глжэ — это свойства, которые принимают значения Entity или EntityClass, ссылающиеся на типы в EntityStore.

Они определяются ассоциацией, содержащей:

“DestinationEntityType”	тип сущности, на который ссылается отношение
“EntityTypeMapping”	сопоставление свойств входящего и целевого типа сущности

“EntityTypeMapping” имеет ту же спецификацию, что и CombinedEntityClass.

Возможные значения:

“ <i>пртп</i> ”	входящий и целевой типы сущности имеют одинаковые значения для свойства
“ <i>пртп</i> “	
{“ <i>пртп_в</i> ”, “ <i>пртп_ц</i> ”, …}	входящий и целевой типы сущности имеют одинаковые значения для всех
<i>фт</i>	
“ <i>jcd</i> ” → “ <i>прfr</i> ”	исходный тип имеет значения “ <i>jcd</i> ”, которые совпадают со значениями “ <i>прfr</i> ”
целевого типа	
{“ <i>jcd_в</i> ” → “ <i>прfr_в</i> ”, …}	исходный тип имеет значения “ <i>jcd_в</i> ”, которые совпадают со значениями “ <i>прfr_в</i> ”
целевого типа	
EntityFunction[[<i>jcd</i> , “ <i>прfr</i> ”, …]]	значение отношения для исходной сущности слева — это все сущности
	конечного типа справа, для которых результат EntityFunction True

“EntityTypeMapping” имеет ту же спецификацию, что и CombinedEntityClass.

Возможные значения:

“ <i>пртп</i> ”	входящий и целевой типы сущности имеют одинаковые значения для свойства
“ <i>пртп</i> “	
{“ <i>пртп_в</i> ”, “ <i>пртп_ц</i> ”, …}	входящий и целевой типы сущности имеют одинаковые значения для всех
<i>фт</i>	
“ <i>jcd</i> ” → “ <i>прfr</i> ”	исходный тип имеет значения “ <i>jcd</i> ”, которые совпадают со значениями “ <i>прfr</i> ”
целевого типа	
{“ <i>jcd_в</i> ” → “ <i>прfr_в</i> ”, …}	исходный тип имеет значения “ <i>jcd_в</i> ”, которые совпадают со значениями “ <i>прfr_в</i> ”
целевого типа	
EntityFunction[[<i>jcd</i> , “ <i>прfr</i> ”, …]]	значение отношения для исходной сущности слева — это все сущности
	конечного типа справа, для которых результат EntityFunction True

В хранилищах сущностей, поддерживаемых реляционными базами данных, форматирование с использованием свойств «Label» отключено, чтобы гарантировать, что все вызовы EntityValue и EntityList выполняют один вызов базы данных.

Возможные **множок** функции и значения по умолчанию:

Initialization	None	выражение для оценки при регистрации хранилища сущностей
----------------	------	--

Создадим хранилище сущностей, содержащее выражение инициализации:

```
In[4]:= store = EntityStore["type", Initialization :> Echo["hello"]];
```

Выражение инициализации вычисляется при регистрации:

```
In[1]:= EntityRegister[store]
```

» hello

```
Out[1]= {type}
```

```
In[2]:= EntityUnregister[store]
```

MetaInformation	< >	метаинформация, связанная с хранилищем сущностей
-----------------	------	--

Создадим хранилище сущностей, включающую метаинформацию:

```
In[3]:= store = EntityStore["endocrine" → <|"Entities" → <|
  "hypothalamus" → <|"anatomical regions" → {hypothalamus ANATOMICAL STRUCTURE}>|,
  "lymphatic" → <|"NearbyStructures" → {thymus ANATOMICAL STRUCTURE}>|>|>],
  MetaInformation → <|"CreationDate" → DateObject[{2025}]>]
```

```
Out[3]=
```

	Type: endocrine Entities: {hypothalamus, lymphatic} (2) Properties: {anatomical regions, NearbyStructures} (2)
--	---

Посмотрим метаинформацию:

```
In[4]:= Options[store, MetaInformation]
```

```
Out[4]=
```

	MetaInformation → < CreationDate → Year: 2025 > >
--	---

```
In[5]:= EntityUnregister[store]
```

	EntityRegister [<i>хрнц</i>] регистрирует сущности в хранилище сущностей <i>хрнц</i> , чтобы к ним можно было получить прямой доступ с помощью Entity
--	--

В гръцкото име на

Список хранилищ зарегистрированных сущностей предоставляется **EntityStores** [].

Entity [“*ннс*”, / *Yk c*] интерпретируется как сущность из первого хранилища сущностей в **EntityStores** [], которое содержит “*ннс*”.

Типы сущностей, которые появляются в **EntityStores** [], рассматриваются раньше встроенных типов сущностей.

Изменение хранилища зарегистрированных сущностей добавлением новой сущности: **Entity** [“*ннс*”, / *cu / Yk c*] [*пртсрв*] = *tYsc*.

Данные хранятся в первом **EntityStore** в **EntityStores** [], который содержит “*ннс*”.

Если ни один **EntityStore** в **EntityStores** [] не содержит “*ннс*”, регистрируется новый **EntityStore**.

EntityRegister […] возвращает список типов сущностей, которые были зарегистрированы.

	EntityStores […] список всех зарегистрированных хранилищ сущностей, доступ к которым осуществляется при использовании Entity
--	---

	EntityUnregister [“ <i>ннс</i> ”] отмена регистрации всех сущностей, определённых “ <i>ннс</i> ”, в первом хранилище сущностей, найденного в списке, возвращаемом EntityStores []
--	---

EntityUnregister [*сущ*]
соответствующих *сущ*

отмена регистрации всех сущностей в хранилище, буквально

В гру́ккожұфқа памзара

Для очистки хранилища, содержащего несколько сущностей можно использовать **EntityUnregister** /@ {"*нурс*", "*нурс*", ...}.

Ножкгош

Создадим хранилище сущностей для сущностей типа «нуклеотиды»:

```
In[1]:= store = EntityStore["нуклеотиды" → <|"Entities" → <|"Аденин" → <|"обозначение" → "A" |>, "Гуанин" → <|"обозначение" → "G" |>, "Цитозин" → <|"обозначение" → "C" |>, "Тимин" → <|"обозначение" → "T" |>, "Урацил" → <|"обозначение" → "U" |>|>|>]
```

Out[1]=

```
EntityStore[  
  + Type: нуклеотиды  
  Entities: {Аденин, Гуанин, Цитозин, Тимин, Урацил} (5)  
  Properties: {обозначение} (1)]
```

В глобальном списке хранилищ сущностей нет зарегистрированных объектов:

```
In[2]:= EntityStores[]
```

Out[2]= {}

Зарегистрируем созданное хранилище сущностей в глобальном списке хранилищ сущностей:

```
In[3]:= EntityRegister[storeNUCL]
```

Out[3]= {нуклеотиды}

В глобальном списке хранилищ сущностей появился объект:

```
In[4]:= EntityStores[]
```

Out[4]= {EntityStore[
 + Type: нуклеотиды
 Entities: {Аденин, Гуанин, Цитозин, Тимин, Урацил} (5)
 Properties: {обозначение} (1)]}

Теперь значения свойств для сущности автоматически ищутся в созданном хранилище сущностей:

```
In[5]:= Entity["нуклеотиды", "Цитозин"] ["обозначение"]
```

Out[5]= С

Получим список сущностей указанного типа:

```
In[6]:= EntityList["нуклеотиды"]
```

Out[6]= {Аденин, Гуанин, Цитозин, Тимин, Урацил}

Найдём сущности, свойство «обозначение» которых соответствуют «А»:

```
In[1]:= EntityClass["нуклеотиды", "обозначение" → EqualTo["A"]] // EntityList
Out[1]= {Аденин}
```

Получим связь всех сущностей и значений для указанного свойства:

```
In[2]:= EntityValue["нуклеотиды", "обозначение", "Association"]
Out[2]= {Аденин → A, Гуанин → G, Цитозин → C, Тимин → T, Урацил → U}
```

Отменим регистрацию хранилища сущностей:

```
In[3]:= EntityUnregister["нуклеотиды"]
```

Объект исчез из глобального списка хранилищ сущностей:

```
In[4]:= EntityStores[]
```

```
Out[4]= {}
```

Создадим хранилище сущностей с вычисляемым свойством «р3» и классами сущностей и свойств:

```
In[5]:= store = EntityStore["MyType" → <|"Label" → "my type", "LabelPlural" → "my type entities",
  "Entities" → <|"e1" → <|"p1" → 1, "p2" → 2, "Label" → "E1"|>,
  "e2" → <|"p1" → 3, "p2" → 4, "Label" → "E2"|>, "e3" → <|"Label" → "E3"|>|>,
  "Properties" → <|"p1" → <|"Label" → "P1"|>,
  "p3" → <|"DefaultFunction" → Function[e, e["p1"] + e["p2"]], "Label" → "P3"|>|>,
  "EntityClasses" → <|"ec1" → <|"Entities" → {"e1", "e2"}, "Label" → "EC1"|>,
  "ec2" → <|"Entities" → {"p3" → GreaterThan[3]}|>|>,
  "PropertyClasses" → <|"pc1" → <|"Properties" → {"p1", "p3"}, "Label" → "PC1"|>|>|>]
```

```
Out[5]= EntityStore[

|                                     |
|-------------------------------------|
| Type: MyType                        |
| Entities: {e1, e2, e3} (3)          |
| Properties: {p1, p3, p2, Label} (4) |

]
```

```
In[6]:= EntityRegister[store];
```

Рассмотрим созданный EntityStore подробнее:

EntityList["MyType"] (*список сущностей*)

```
Out[6]= {E1, E2, E3}
```

EntityClassList["MyType"] (*список классов сущностей*)

```
Out[7]= {EC1, ec2}
```

EntityList[EC1 MY TYPE ENTITIES] (*список сущностей в "ec1"-классе*)

```
Out[8]= {E1, E2}
```

```
EntityList[] (*список сущностей в "ec2"-классе*)
```

Out[*#*] =

{}

```
E1  ["Properties"] (*свойства сущностей*)
```

Out[*#*] =

{, , , }

Получим вычисленное значение:

```
In[#] := Entity["MyType", "e2"] ["p3"]
```

Out[*#*] =

7

Найдём второй член "ec2" в созданном "EntityClasses":

```
In[#] := EntityList[EntityClass["MyType", "ec2"]]
```

Out[*#*] =

{}

Получим ассоциацию значений всех свойств в «pc1» для одной сущности "e1":

```
In[#] := Entity["MyType", "e1"] [EntityPropertyClass["MyType", "pc1"], "PropertyAssociation"]
```

Out[*#*] =

{| → 1, → 3 |}

Получим для одной сущности "e1" прямой доступ к хранилищу сущностей:

```
In[#] := store[Entity["MyType", "e1"], "p1"]
```

Out[*#*] =

1

```
store[Entity["MyType", "e1"], "p3"] (*значения не вычисляются*)
```

Out[*#*] =

Missing[NotAvailable]

Доступ к свойствам класса сущности "ec1":

```
In[#] := store[EntityClass["MyType", "ec1"], "Label"]
```

Out[*#*] =

EC1

Извлечём необработанные данные, хранящиеся для "Entities":

```
In[#] := store[EntityClass["MyType", "ec1"], "Entities"]
```

Out[*#*] =

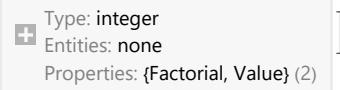
{e1, e2}

```
In[#] := EntityUnregister["MyType"]
```

Создадим хранилище сущностей всех целых чисел:

```
integerStore = EntityStore[
  "integer" → <|
    "EntityValidationFunction" → IntegerQ@*Interpreter["Integer"], "Properties" → <|
      "Factorial" → <| "DefaultFunction" → Function[entity, entity["Value"]!] |>,
      "Value" → <| "DefaultFunction" → Interpreter["Integer"]@*CanonicalName |> |> |>
    (*сущности задаются функцией*)
  EntityRegister[integerStore];
```

Out[*#*] =

```
EntityStore[]
```

Получим числовое значение сущности:

In[*#*] := Entity["integer", "4"] ["Value"]

Out[*#*] =

4

Вычислим факториал сущности «4»:

In[*#*] := Entity["integer", "4"] ["Factorial"]

Out[*#*] =

24

Недействительные сущности не вычисляются:

In[*#*] := Entity["integer", "abc"] ["Factorial"]

Out[*#*] =

Missing[UnknownEntity, {integer, abc}]

Изменим значение «Факториала» сущности «7»:

In[*#*] := Entity["integer", "7"] ["Factorial"] = 7

Out[*#*] =

7

Введём сущность “integer” вместе с новым значением в зарегистрированное хранилище сущностей:

In[*#*] := newIntegerStore = Entity["integer"] ["EntityStore"]

Out[*#*] =

```
EntityStore[]
```

In[*#*] := newIntegerStore[Entity["integer", "7"], "Factorial"]

Out[*#*] =

7

Другие сущности не имеют явного значения:

```
In[1]:= newIntegerStore[Entity["integer", "2"], "Factorial"]
Out[1]= Missing[UnknownEntity, {integer, 2}]

In[2]:= EntityUnregister["integer"]
```

Создадим хранилище сущностей, содержащее помеченные сущности и классы сущностей:

```
In[3]:= store = EntityStore[
  "type" → <|"Entities" → <|"e1" → <|"Label" → "E1"|>, "e2" → <|"Label" → "E2"||>|>,
  "EntityClasses" → <|"ec1" → <|"Entities" → {"e1", "e2"}, "Label" → "EC1"||>|>|>];
EntityRegister[store];
```

Обычно EntityClass ведет себя как список сущностей, которые он представляет.

Получим «Label» для каждой сущности в классе:

```
In[4]:= EntityClass["type", "ec1"] ["Label"]
Out[4]= {E1, E2}
```

Для получения «Label» класса обратимся напрямую к EntityStore:

```
In[5]:= store[EntityClass["type", "ec1"], "Label"]
Out[5]= EC1
```

```
In[6]:= EntityUnregister[store]
```

Поместим хранилище сущностей в облако:

```
In[7]:= co = CloudPut[EntityStore["t" → <|
  "Entities" → <|"e1" → <|"p1" → 99, "p2" → 88||>, "e2" → <|"p1" → 77, "p2" → 66||>|>|>]
Out[7]= CloudObject[https://www.wolframcloud.com/obj/60d6b8f2-73a5-45d0-aad4-7f15476d5eb9]
```

Сделаем созданное хранилище, помещённое в облако, доступным для того, кто захочет использовать содержащиеся в нем сущности:

```
In[8]:= EntityRegister[CloudGet[co]];
In[9]:= Entity["t", "e2"] ["p2"]
Out[9]= 66
```

```
In[10]:= EntityUnregister["t"]
```

Изменим хранилище зарегистрированных сущностей, добавив новую сущность, и удалим получившуюся сущность:

Создадим и зарегистрируем новое хранилище сущностей:

```
In[11]:= store = EntityStore["type" → <|"Entities" → <|"e1" → <|"p" → "prop1"||>|>|>];
EntityRegister[store];
```

In[*#*]:= EntityStores[]

```
{EntityStore[ Type: type
Entities: {e1} (1)
Properties: {p} (1)]}
```

Добавим в хранилище зарегистрированных сущностей новую сущность:

In[*#*]:= Entity["type", "e2"] ["p"] = "prop2";

EntityStores[] (*новая сущность заменила первую store*)

Out[*#*]=

```
{EntityStore[ Type: type
Entities: {e1, e2} (2)
Properties: {p} (1)]}
```

Отменим регистрацию измененного хранилища:

In[*#*]:= EntityUnregister[store];

Хранилище, содержащее тип «*type*», по-прежнему зарегистрировано, поскольку оно не соответствует исходному хранилищу **store**:

In[*#*]:= EntityStores[]

Out[*#*]=

```
{EntityStore[ Type: type
Entities: {e1, e2} (2)
Properties: {p} (1)]}
```

Отменим регистрацию хранилища, обратившись к нему по типу «*type*»:

In[*#*]:= EntityUnregister["type"]
EntityStores[]

Out[*#*]=

{ }

Ножки глыж

Добавление новых данных в одно и то же хранилище сущностей:

Для каждого нового типа сущности создаются отдельные хранилища:

In[*#*]:= store1 = EntityStore["type1" → <|"Entities" → <|"e" → <|"p" → "prop1"|>|>|>];
store2 = EntityStore["type2" → <|"Entities" → <|"f" → <|"q" → "prop2"|>|>|>];
EntityRegister[store1];
EntityRegister[store2];

In[*#*]:= EntityStores[]

Out[*#*]=

```
{EntityStore[ Type: type2
Entities: {f} (1)
Properties: {q} (1)], EntityStore[ Type: type1
Entities: {e} (1)
Properties: {p} (1)]}
```

EntityUnregister /@ {"type1", "type2"}; (*очистим хранилище*)

Чтобы обе сущности находились в одном и том же хранилище, зарегистрируем для них одно хранилище:

```
In[1]:= EntityRegister[EntityStore[{"type1", "type2"}]];
```

```
In[2]:= EntityStores[]
```

```
Out[2]=
```

EntityStore	Types: Entity count: Property count:	type1 0 0	type2 0 0]	}
-------------	--	-----------------	-----------------	---	---

Выполним новые назначения:

```
In[3]:= Entity["type1", "e"]["p"] = 915;
Entity["type2", "f"]["q"] = "abc";
```

```
In[4]:= EntityStores[]
```

```
Out[4]=
```

EntityStore	Types: Entity count: Property count:	type1 1 1	type2 1 1]	}
-------------	--	-----------------	-----------------	---	---

Для отмены регистрации всего хранилища сущностей достаточна ссылка на любой из типов сущностей:

```
In[5]:= EntityUnregister["type1"]
```

```
In[6]:= EntityStores[]
```

```
Out[6]=
```

```
{ }
```

Восстановление первоначального варианта сущности:

Зарегистрируем хранилище, содержащее два типа сущностей:

```
In[1]:= EntityRegister[
EntityStore[{ "t1" → <|"Entities" → <"e" → <"p" → Entity["t2", "f"]|>|>|>, "t2" → <"f" → <"q" → 19|>|>}]]
```

```
Out[1]=
```

```
{t1, t2}
```

```
In[2]:= EntityStores[]
```

```
Out[2]=
```

EntityStore	Types: Entity count: Property count:	t1 1 1	t2 1 1]	}
-------------	--	--------------	--------------	---	---

Проверим свойства сущностей:

```
In[1]:= Entity["t1", "e"] ["p"]
%["q"]
```

Out[1]=

f

Out[1]=

19

Зарегистрируем ещё одно хранилище, которое копирует «t2», но с другим значением свойства.

```
In[2]:= EntityRegister[
EntityStore[
"t2" → <|"Entities" → <|"f" → <|"q" → 7891|>|>]];
```

EntityRegister: Types {t2} in EntityStore[<|Types → <|t2 → <| Entities → <|f → <|q → 7891|>|>], Properties → <|q → <| |>|>|> already exist. The existing ones will be shadowed.

Список всех зарегистрированных хранилищ:

```
In[3]:= EntityStores[]
```

Out[3]=

EntityStore[+ Type: t2 Entities: {f} (1) Properties: {q} (1)	EntityStore[+ Types: Entity count: t1 t2 Property count: 1 1
---	---

Теперь новая сущность «t2» находится в недавно зарегистрированном хранилище и её свойства напрямую влияют на свойства «t1»:

```
In[4]:= Entity["t1", "e"] ["p"]
%["q"]
```

Out[4]=

f

Out[4]=

7891

Извлечём хранилище сущностей, содержащее «t2»:

```
In[5]:= Entity["t2"] ["EntityStore"]
```

Out[5]=

EntityStore[+ Type: t2 Entities: {f} (1) Properties: {q} (1)

Отмена регистрации типа «t2» восстановит предыдущий результат:

```
In[6]:= EntityUnregister["t2"]
```

```
In[7]:= Entity["t1", "e"] ["p"]
%["q"]
```

Out[7]=

f

Out[7]=

19

```
In[8]:= EntityUnregister["t1"] (*очистим хранилище*)
```

```
In[#]:= EntityStores[]

Out[#]= {}
```

Класс агрегированных сущностей

[AggregatedEntityClass](#)

EntityInstance

Сущность с заданными значениями свойств

EntityInstance [<i>cl rgw os Yj</i> → <i>tYj</i>] квалификатор которой <i>os Yj</i> имеет значение <i>tYj</i> EntityInstance [<i>cl rgw{os Yj_g</i> → <i>tYj₁</i> , <i>os Yj_v</i> → <i>tYj₂</i> , ...}] квалификаторы которой <i>os Yj_g</i> имеют значения <i>tYj_i</i> EntityInstance [<i>cl rgw os Yl rgw</i>] определяемую количеством <i>os Yl rgw</i>

предоставляет сущность,

предоставляет сущность,

предоставляет собой сущность,

В грекоязычной памяти

Квалификатор *os Yj* может быть квалификатором:

- свойства [EntityProperty](#) сущности ,
- [QuantityVariable](#) ,
- аргумента [Function](#).

Если *cl rgw* возвращает [Function](#)[*nYrk q Znbi*] с параметром *nYrk q* равным *os Yj*, то соответствующее значение подставляется в *Znbi*, а *os Yj* удаляется из *nYrk q*

Если параметров не осталось, то оболочка [Function](#) удаляется.

Выражения, содержащие неизвестные или неприменимые свойства *os Yj*, игнорируются. Результаты соответствуют результатам исходной сущности:

```
In[#]:= EntityValue[half-torus SOLID, "SurfaceArea"]

Out[#]= Function[{a, c},  $2 \pi (c + a \pi)$ ]

In[#]:= EntityValue[EntityInstance[half-torus SOLID, {"blah" → 2}], "SurfaceArea"]

Out[#]= Function[{a, c},  $2 \pi (c + a \pi)$ ]
```

При задании *{os Yj_g → tYj₁, os Yj_v → tYj₂, ...}* применяются только соответствующие *os Yj_g*:

```
In[#]:= EntityValue[EntityInstance[half-torus SOLID, {"blah" → 2, a → 5}], "SurfaceArea"]

Out[#]= Function[{c},  $2 \pi (c + 5 \pi)$ ]
```

Ножкиощ

Представим сущность, значение для свойства которого возвращается как чистая функция:

In[1]:= `half-torus SOLID` ["SurfaceArea"]

Out[1]=
`Function[{a, c}, 2 c π (c + a π)]`

Укажем значения для данного свойства:

In[2]:= `eIn = EntityInstance[half-torus SOLID, {a → 1, c → 2}]`

Out[2]=
`EntityInstance[half-torus, {a → 1, c → 2}]`

Найдём значения свойства для созданной сущности:

In[3]:= `EntityValue[eIn, "SurfaceArea"]`

Out[3]=
`4 π (2 + π)`

Найдём значения двух свойств для сущности, значение свойства которого возвращается как чистая функция:

In[4]:= `EntityValue[EntityInstance[ball SOLID, a → 1], {"SurfaceArea", "Volume"}]`

Out[4]=
`{4 π, 4 π / 3}`

Укажем значения для свойств, содержащих количественные переменные:

In[5]:= `eIn = EntityInstance[harmonic oscillator PHYSICAL SYSTEM,`

`QuantityVariable["ω", "AngularFrequency"] → Quantity[1, ("Radians") / ("Seconds")]]`

In[6]:= `EntityValue[eIn, {JYerpl eg'l, cos Yrgn qmk mgt}, "PropertyAssociation"]`

Out[6]=

$$\langle \left| \begin{aligned} JYerpl eg'l &\rightarrow \left(-\frac{1}{2} \text{ rad}^2/\text{s}^2 \right) m x[t]^2 + \frac{1}{2} m x'[t]^2, \\ \cos Yrgn qmk mgt &\rightarrow \left\{ \left(1 \text{ rad}^2/\text{s}^2 \right) x[t] + x''[t] = 0 \right\} \end{aligned} \right| \rangle$$

Ножкаглгж

Найдём объём для сущности молекулярного кислорода с заданной массой химического вещества:

In[7]:= `eIn = EntityInstance[oxygen CHEMICAL, Quantity[1, "Kilograms"]];`

In[8]:= `EntityValue[eIn, "AbsoluteVolume"]`

Out[8]=
`699.8 L`

Аргументы чистой функции

Получим переменные, входящие в свойства твердого тела:

In[1]:= **half-torus SOLID** [*tYpYZjcq*]

Out[1]=

{*a*, *c*}

Рассмотрим свойства, которые являются чистыми функциями переменных:

In[2]:= **half-torus SOLID** [{ *acl rpmθ*, *φpdac YpxY*, *bcdgdeglcosYjggq* }, "Association"]

Out[2]=

$\langle \left| \begin{array}{l} \text{acl rpm}\theta \rightarrow \text{Function}[\{\mathbf{a}, \mathbf{c}\}, \left\{ \frac{\mathbf{a}^2 + 4 \mathbf{c}^2}{2 \mathbf{c} \pi}, 0, 0 \right\}], \\ \text{φpdac YpxY} \rightarrow \text{Function}[\{\mathbf{a}, \mathbf{c}\}, 2 \mathbf{c} \pi (\mathbf{c} + \mathbf{a} \pi)], \text{bcdgdeglcosYjggq} \rightarrow \text{Function}[\{\mathbf{a}, \mathbf{c}\}, \text{Function}[\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}, \left(\mathbf{c} - \sqrt{\mathbf{x}^2 + \mathbf{y}^2} \right)^2 + \mathbf{z}^2 \leq \mathbf{a}^2 \& \mathbf{x} \geq 0]] \end{array} \right| \rangle$

Определив переменную *π* получим функции оставшейся переменной:

In[3]:= **EntityValue**[
EntityInstance[**half-torus SOLID**, {*c* → 10}],
{ *acl rpmθ*, *φpdac YpxY*, *bcdgdeglcosYjggq* }, "Association"]

Out[3]=

$\langle \left| \begin{array}{l} \text{acl rpm}\theta \rightarrow \text{Function}[\{\mathbf{a}\}, \left\{ \frac{\mathbf{a}^2 + 4 \times 10^2}{2 \times 10 \pi}, 0, 0 \right\}], \\ \text{φpdac YpxY} \rightarrow \text{Function}[\{\mathbf{a}\}, 2 \times 10 \pi (10 + \mathbf{a} \pi)], \text{bcdgdeglcosYjggq} \rightarrow \text{Function}[\{\mathbf{a}\}, \text{Function}[\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}, \left(10 - \sqrt{\mathbf{x}^2 + \mathbf{y}^2} \right)^2 + \mathbf{z}^2 \leq \mathbf{a}^2 \& \mathbf{x} \geq 0]] \end{array} \right| \rangle$

Определим все переменные:

In[4]:= **EntityValue**[
EntityInstance[**half-torus SOLID**, {*c* → 10, *a* → 2}],
{ *acl rpmθ*, *φpdac YpxY*, *bcdgdeglcosYjggq* }, "Association"]

Out[4]=

$\langle \left| \begin{array}{l} \text{acl rpm}\theta \rightarrow \left\{ \frac{101}{5 \pi}, 0, 0 \right\}, \text{φpdac YpxY} \rightarrow 20 \pi (10 + 2 \pi), \\ \text{bcdgdeglcosYjggq} \rightarrow \text{Function}[\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}, \left(10 - \sqrt{\mathbf{x}^2 + \mathbf{y}^2} \right)^2 + \mathbf{z}^2 \leq 2^2 \& \mathbf{x} \geq 0] \end{array} \right| \rangle$

Определение количественных переменных

Определим переменные, от которых зависит гармонический осциллятор :

In[5]:= **harmonic oscillator PHYSICAL SYSTEM** [*qmrck tYpYZjcq*]

Out[5]=

{*m*, *ω*}

Укажем значения количественных переменных и найдём уравнение движения:

```
In[]:= EntityValue[
  EntityInstance[harmonic oscillator PHYSICAL SYSTEM, {
    QuantityVariable["m", "Mass"] → 1,
    QuantityVariable["ω", "AngularFrequency"] → 1}], "EquationsOfMotionSolution"]

Out[=]
{x → Function[{t}, Cos[t 1] x[θ] + Sin[t 1] x'[θ]]}
```

Рассчёт массы заданного количества вещества

Создадим сущность: “1 моль воды”:

```
In[]:= eIn = EntityInstance[water CHEMICAL ..., Quantity[1, "Moles"]]

Out[=]
EntityInstance[water, 1 mol]
```

Определим её массу:

```
In[]:= EntityValue[eIn, "AbsoluteMass"]

Out[=]
18.015 g
```

Рассчитаем массу, используя свойство “молярная масса”:

```
In[]:= water CHEMICAL ... [k mJYpk Yqg] Quantity[1, "Moles"]

Out[=]
18.015 g
```

Получили одинаковые результаты:

```
In[]:= % == %

Out[=]
True
```

Рассчитаем содержание компонента в заданном количестве вещества

Создадим сущность: “50 грамм круассана”:

```
In[]:= eIn = EntityInstance[
  Entity["Food", {"FoodType" → croissant FOOD TYPE, "RelativeWaterContent" →
    Quantity[Interval[{0, 0.5}], "Grams" / "Grams"]}], Quantity[50, "Grams"]]

Out[=]
EntityInstance[✓ food +, 50 g]
```

Найдём абсолютное содержание витамина А:

```
In[1]:= eIn["AbsoluteVitaminAContent"]
Out[1]= 9. × 101 μg
```

Пользовательская база сущностей

Создадим базу сущностей «Формы» и зарегистрируем её:

```
In[2]:= EntityRegister[
  EntityStore["shape" → <|
    "Entities" → <|"rectangle" → <|"variables" → {a, b}, "area" → Function[{a, b}, a * b]|>,
    "disk" → <|"variables" → {r}, "area" → Function[r, π * r^2]|>|>
  |>]
]
Out[2]= {shape}
```

Создадим диск радиусом 2 метра и найдём его площадь:

```
In[3]:= EntityInstance[
  Entity["shape", "disk"], r → Quantity[2, "Meters"]
]
Out[3]= EntityInstance[disk, r → 2 m]

In[4]:= %[ "area"]
Out[4]= 4 π m2
```

Создадим прямоугольник с указанной одной стороной и определим его площадь, как функцию другой стороны:

```
In[5]:= EntityInstance[
  Entity["shape", "rectangle"], a → Quantity[9, "Centimeters"]
]
Out[5]= EntityInstance[rectangle, a → 9 cm]

In[6]:= %[ "area"]
Out[6]= Function[{b}, (9 cm) b]

In[7]:= EntityUnregister["shape"] (*отменим регистрацию базы сущностей*)
```

Задача двойного маятника

Получим уравнения движения двойного маятника, задав соответствующие значения:

```
In[6]:= eqn = EntityValue[EntityInstance[double pendulum PHYSICAL SYSTEM], 
  {QuantityVariable[Subscript["l", 1], "Length"] → 1, 
   QuantityVariable[Subscript["m", 1], "Mass"] → 1, 
   QuantityVariable[Subscript["l", 2], "Length"] → 1, 
   QuantityVariable[Subscript["m", 2], "Mass"] → 1, 
   QuantityVariable["g", "Acceleration"] → 9.81}], "EquationsOfMotion"]

Out[6]= {19.62 Sin[θ1[t]] + Sin[θ1[t] - θ2[t]] θ2'[t]^2 + 2 θ1''[t] + Cos[θ1[t] - θ2[t]] θ2''[t] == 0, 
  9.81 Sin[θ2[t]] + Cos[θ1[t] - θ2[t]] θ1'[t] + θ2''[t] == Sin[θ1[t] - θ2[t]] θ1'[t]^2}
```

Выберем начальные условия:

```
In[7]:= theta10 = 0.7;
theta20 = 0;
```

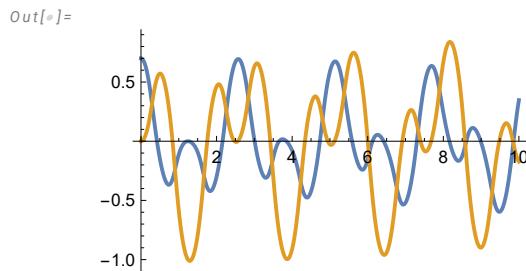
Решим уравнения:

```
In[8]:= {theta1sol, theta2sol} =
NDSolveValue[
  Join[
    eqn /. {QuantityVariable[Subscript["θ", 1], "Angle"] → theta1,
      QuantityVariable[Subscript["θ", 2], "Angle"] → theta2}, {
      theta1[0] == theta10,
      theta1'[0] == 0,
      theta2[0] == theta20,
      theta2'[0] == 0}], {
    theta1, theta2}, {QuantityVariable["t", "Time"], 0, 10}]
```

```
Out[8]= {InterpolatingFunction[ Domain: {{0, 10.}} Output: scalar], 
  InterpolatingFunction[ Domain: {{0, 10.}} Output: scalar]}
```

Построим график движения двойного маятника для каждого шарнира:

```
In[9]:= Plot[{theta1sol[t], theta2sol[t]}, {t, 0, 10}]
```



Анимация двойного маятника:

```
In[=]:= Animate[
Graphics[{Disk[{0, 0}, 0.1],
With[{p1 = RotationMatrix[theta1sol[t]].{0, -1}}, {
Line[{{0, 0}, p1}], Disk[p1, 0.1],
With[{p2 = p1 + RotationMatrix[theta2sol[t]].{0, -1}}, {
Line[{p1, p2}],
Disk[p2, 0.1]}]}]}, PlotRange -> {{-3, 3}, {-3, 1}}], {t, 0, 10}, SaveDefinitions -> True]
```

Out[=]=

*Виѣ Yk gøYjjwAmI qрs arcb Cl rgwAjYqçq**ВжпЎк жжг пижпмюв ўллшг ик ўпшц псч лмпргз*

Класс агрегированных сущностей

[AggregatedEntityClass](#)[AggregatedEntityClass](#) [*ajYqç* “*nptm*” → *d*
пртм которого является

класс сущностей, значение одного из свойств

результатом применения функции *f*

ко всему указанному классу сущностей

[AggregatedEntityClass](#) [*ajYqç* {“*nptm*₁” → *d*, “*nptm*₂” → *d*, ...}]конструирует несколько свойств *nptm*, путёмприменения *d* к соответствующим свойствам класса сущностей *ajYqç*[AggregatedEntityClass](#) [*ajYqç* *prtpcsa*, *eprtm*]формирует группы элементов из класса *class* всоответствии со значением свойства *eprtm*,

затем генерирует класс сущностей из

этих групп

[AggregatedEntityClass](#) [*ajYqç* *prtpcsa*, “*nl Yk c*” → *d*формирует группы элементов класса *ajYqç* в

соответствии со значениями, полученными

путем применения функции сущности

и *d*, с результирующим свойством *nl Yk c*

AggregatedEntityClass [*ажъднпртфса*, {*ефса*₁, *ефса*₂,...}] формирует группы, для которых набор значений, определяемый *ефса*_ρ различается

В группах по памзра

Функция **d** может быть:

- либо объектом **EntityFunction** [...];
- либо функцией *d*, записанная как **EntityFunction** [*v*, *d*/[*v*["prop"]]].

AggregatedEntityClass генерирует сущности, соответствующие определённым группам. Полученные сущности фактически соответствуют неявным сущностям.

```
In[1]:= agg = AggregatedEntityClass["Element", "AtomicNumber" → Max, "Period"];
```

```
In[2]:= EntityList[agg]
```

```
Out[2]= {1, 2, 3, 4, 5, 6, 7}
```

```
In[3]:= EntityProperties[agg]
```

```
Out[3]= {mk gLsk Zcp, ncpnib}
```

```
In[4]:= agg[{mk gLsk Zcp, ncpnib}]
```

```
Out[4]= {{1, 2}, {2, 10}, {3, 18}, {4, 36}, {5, 54}, {6, 86}, {7, 118}}
```

Объекты **EntityFunction** в *пртфса* фактически применяются к ассоциации свойств, значения которых представляют собой список значений для сущностей в *ажъд*

Выражения **EntityFunction** во втором аргументе **AggregatedEntityClass** применяются не к отдельным сущностям из первого аргумента, а:

- или ко всему первому аргументу, если третий аргумент отсутствует,
- или к классам сущностей, полученным путем группировки в соответствии с третьим

аргументом.

Выражения **EntityFunction** в третьем аргументе **AggregatedEntityClass** применяются отдельно к индивидуальным сущностям в первом аргументе.

В отличие от **GroupBy**, **AggregatedEntityClass** [*ажъд пртфса*, {*пртм*₁, *пртм*₂,...}] не выполняет иерархическую группировку, если её третий аргумент является списком.

Он создаст класс с рядом сущностей, заданных различными значениями **EntityValue** [*ажъд* {*пртм*₁, *пртм*₂,...}].

```
EntityProperties[AggregatedEntityClass[ажъд{"I Yk c1"→cl rgnd/a1,"I Yk c2"→cl rgnd/a2,...}, {"пртм1","пртм2",...}]]  
возвращает  
{EntityProperty[Yee,"I Yk c1"], EntityProperty[Yee,"I Yk c2"],..., EntityProperty[Yee,"пртм1"],  
EntityProperty[Yee,"пртм2"],...}
```

, где *Yee* — это все выражение **AggregatedEntityClass**.

При использовании сущностей из реляционных баз данных **AggregatedEntityClass** с двумя аргументами соответствует использованию агрегатных функций в операторе SELECT.

Ножгош

Создадим совокупность всех сущностей в указанном классе и вычислим общее значение указанного свойства:

Рассчитаем общее количество людей, прошедших кардиологический скрининг, результаты тестирования которых содержатся в базе :

```
EntityProperty["MedicalTest", "SamplePopulationValue"] ["Description"]
(*используемое свойство*)
```

```
Out[=]=
sample population
```

[] (*измеряемые показатели*)

```
Out[=]=
{serum total cholesterol, serum LDL cholesterol, serum HDL cholesterol,
serum triglycerides, systolic blood pressure, diastolic blood pressure,
serum C-reactive protein, serum insulin, plasma homocysteine, serum ferritin}
```

```
AggregatedEntityClass[  ,
"TotalNumber" → EntityFunction[e, Total[e["SamplePopulationValue"]]]] [
"TotalNumber"] (*число прошедших тестирование людей*)
```

```
Out[=]=
{ 175 170 people }
```

Эквивалентное вычисление, подходящее для небольших наборов данных :

```
In[=]:= Total[EntityValue[  , "SamplePopulationValue"]]
```

```
Out[=]=
175 170 people
```

Сформируем класс, состоящий из сущностей “Элементы ... периода”. Найдём среднюю атомную массу для элементов в каждом периоде:

```
In[=]:= AggregatedEntityClass["Element", "AtomicMass" → Mean, "Period"] ["AtomicMass"]
```

```
Out[=]=
{ 2.50530 u, 13.49 u, 30.10 u, 60.679 u, 106.2 u, 174.2 u, 261. u }
```

Мпмўгллмпржножжглглж

[EntityFunction](#) может быть третьим аргументом:

Создадим [AggregatedEntityClass](#) на основе заданной функции “ММ” и найдём среднее значение заданного свойства “VD” для всех сформированных групп класса :

```
In[=]:= agg = AggregatedEntityClass[EntityClass["Chemical", "Lipids"],
"VD" → EntityFunction[c, c[EntityProperty["Chemical", "MolarVolume"]] /
c[EntityProperty["Chemical", "MassDensity"]]],
"MM" → EntityFunction[c, Round[c["MolecularMass"], 150]]];

agg // EntityList(*сущности сгруппированы по значению их молекулярных масс*)
```

```
Out[=]=
{  ,  ,  ,  ,  }
```

```
In[]:= EntityProperties[agg]
Out[]= {TB, KK}

In[]:= agg[KK]
Out[=] {150 u, 300 u, 450 u, 750 u, 900 u}

In[]:= agg[TB]
Out[=] {{101. cm6/ (g mol), 211.94 cm6/ (g mol), 184.48 cm6/ (g mol),
175.349 cm6/ (g mol), 169.503 cm6/ (g mol), 1, 157.724 cm6/ (g mol),
1, 214.856 cm6/ (g mol), 194.876 cm6/ (g mol), 1, 1, 1, 1, 1},
{393.619 cm6/ (g mol), 1, 1, 1, 1, 1, 352.139 cm6/ (g mol), 352.139 cm6/ (g mol),
350.212 cm6/ (g mol), 350.249 cm6/ (g mol), 333.214 cm6/ (g mol),
355.459 cm6/ (g mol), 1, 1, 1, 316.025 cm6/ (g mol), 1, 1, 334.888 cm6/ (g mol), 1},
{477.249 cm6/ (g mol), 1, 1}, {1, 1}, {1058.84 cm6/ (g mol)}}
```

Применение `EntityFunction` к результату (обычно это функция агрегации):

```
hepatitis B screening MEDICAL TESTS // EntityList (*оцениваемый класс медицинских тестов*)
Out[=] {serum or plasma hepatitis B core antibody,
serum or plasma hepatitis B surface antibody, serum or plasma hepatitis B surface antigen}
```

Создадим функцию, которая работает на значениях свойства "Mean" всех сущностей класса:

```
AggregatedEntityClass[hepatitis B screening MEDICAL TESTS,
"MeanGroupMT" → EntityFunction[e, f[e[EntityProperty["MedicalTest", "Mean"]]]][
"MeanGroupMT"]
Out[=] {f[{1.95225, 1.73082, 1.99724}]}
```

Вычислим среднее значения свойства:

```
In[]:= AggregatedEntityClass[hepatitis B screening MEDICAL TESTS,
"Mean" → EntityFunction[e, Mean[e[EntityProperty["MedicalTest", "Mean"]]]][
"Mean"] // N
Out[=] {1.89344}
```

Создадим круговую диаграмму с результатом:

```
In[=]:= AggregatedEntityClass[ hepatitis B screening MEDICAL TESTS ,
  "ValueMTChart" → EntityFunction[e,
    PieChart[e[EntityProperty["MedicalTest", "Mean"]]]] [ "ValueMTChart" ]
Out[=]=
```

{ }

[AggregatedEntityClass](#) генерирует сущности, фактически соответствующие неявным сущностям.

Сформируем класс, состоящий из сущностей “Шоколадное мороженое” и найдём каллорийность продукта.

С помощью [Entity](#), указывая свойства:

```
In[=]:= Entity[ "Food",
  { dmb rnnc → ContainsExactly[ { ice cream FOOD TYPE } ],
    Ybbcb dmb rnncq → ContainsExactly[ {} ],
    gYtnp → chocolate FLAVOR }
  ] [ "RelativeTotalCaloriesContent" ]
Out[=]=
```

2.1 Cal/g

С помощью [AggregatedEntityClass](#):

```
In[=]:= AggregatedEntityClass[ EntityClass[ "Food", { EntityProperty[ "Food", "FoodType" ] →
  ContainsExactly[ { Entity[ "FoodType", "IceCream" ] } ],
  EntityProperty[ "Food", "AddedFoodTypes" ] → ContainsExactly[ {} ],
  EntityProperty[ "Food", "Flavor" ] → Entity[ "FoodFlavor", "Chocolate" ] }
  ], "RelativeTotalCaloriesContent" → Median@*DeleteMissing] [
  "RelativeTotalCaloriesContent"]
Out[=]=
```

{ 2.1 Cal/g }

Класс комбинированных сущностей

[CombinedEntityClass](#)

[CombinedEntityClass](#)[*entity1*, *entity2*, *prop*]

пар сущностей из *entity1* и *entity2*,

класс сущностей, полученных путем объединения

для которых значение свойства *prop* одинаково для двух сущностей в

паре

`CombinedEntityClass[ajYqq, ajYqq, пртп6 → пртп7]`

которых

объединяет пары сущностей из *ajYqq* и *ajYqq*, для

значение свойства *пртп*₆ сущности из *ajYqq* совпадает со значением

*пртп*₇ для сущности из *ajYqq*

`CombinedEntityClass[ajYqq, ajYqq, {попса1, попса2,...}]` объединяет пары сущностей, для которых все спецификации свойств *попса*₁ совпадают

`CombinedEntityClass[ajYqq, ajYqq, d]`

объединяет пары сущностей, для которых применение

`EntityFunction` *d*ает значение `True`

`CombinedEntityClass[ajYqq, ajYqq, попса, "渴са"]` использует "*渴са*" для определения того, когда следует разрешить включение сущностей с отсутствующими свойствами.

В групповой форме пампра

Функция используется для создания класса сущностей путём объединения пар сущностей из имеющихся баз новых свойств.

`EntityProperties[CombinedEntityClass[ajYqq, ajYqq,...]] == Join[EntityProperties[ajYqq], EntityProperties[ajYqq]]`

Функция *d* должна иметь вид `EntityFunction[{c6, c7,...}]`, где *c*₆ и *c*₇ обозначают соответственно сущности *ajYqq*₆ и *ajYqq*₇.

Свойства в результате `CombinedEntityClass[ajYqq, ajYqq,...]` имеют форму `EntityProperty [инс6 ...]`, где *инс*₆ — это типы сущностей *ajYqq*₆.

`CombinedEntityClass[θYjg'q6 → ajYqq, θYjg'q7 → ajYqq,...]` можно использовать для задания свойств в форме `EntityProperty [θYjg'q6 → инс6 ...]`.

В `CombinedEntityClass[ajYqq, ajYqq,...]` имеют форму `EntityProperty [инс6 ...]`, где *инс*₆ — это типы сущностей *ajYqq*₆.

В `CombinedEntityClass[ajYqq, ajYqq,...]` классы *ajYqq* должны различаться.

`CombinedEntityClass["渴са" Yjg'q → ajYqq]` можно использовать для идентичных классов.

В `CombinedEntityClass[a1, a2, попса, "渴са"]` спецификация *渴са* определяет, какие сущности *Y_ε* из класса *a*₁ и *Z_ε* из класса *a*₂ должны быть сохранены в зависимости от того, выполняется ли условие, определенное *am b*. Возможные спецификации соединения классов такие же, как и для `JoinAcross`:

"Inner" оставить только *Y_ε* и *Z_ε*, для которых условие *am b* выполняется (нискмкхУлжь)

"Left" разрешить *Y_ε*, для которого нет соответствия с *Z_ε*

"Right" разрешить *Z_ε*, для которого нет соответствия с *Y_ε*

"Outer" допускают как несовпадающие *Y_ε* так и *Z_ε*

Если значение свойства отсутствует, вместо него используется `Missing` ["Unmatched"].

В `CombinedEntityClass[ajYqq, ajYqq, EntityFunction[{v,w,True}]] эффективно реализует SQL CROSS JOIN .`

Это примерно эквивалентно `Outer [Join,...]`, применяемой к «PropertyAssociation», полученному из обоих классов.

При использовании сущностей, поддерживаемыми `RelationalDatabase`, `CombinedEntityClass` фактически эквивалентен оператору `Join`.

Ножгощ

Объединим элементы I периода с их изотопами и сравним их атомные массы:

```
In[1]:= CombinedEntityClass["Isotope", period 1 elements ELEMENTS, "Element" → "Entity"] [
  EntityProperty["Isotope", "Name"], EntityProperty["Element", "AtomicMass"],
  EntityProperty["Isotope", "AtomicMass"]]

Out[1]= {{protium, 1.0080 u, 1.0078250319 u}, {deuterium, 1.0080 u, 2.0141017778 u},
  {tritium, 1.0080 u, 3.016049281 u}, {hydrogen-4, 1.0080 u, 4.026 u},
  {hydrogen-5, 1.0080 u, 5.035 u}, {hydrogen-6, 1.0080 u, 6.045 u},
  {hydrogen-7, 1.0080 u, 7.05 u}, {helium-2, 4.002602 u, Missing[Unknown]},
  {helium-3, 4.002602 u, 3.016029322 u}, {helium-4, 4.002602 u, 4.002603254 u},
  {helium-5, 4.002602 u, 5.0121 u}, {helium-6, 4.002602 u, 6.018886 u},
  {helium-7, 4.002602 u, 7.0280 u}, {helium-8, 4.002602 u, 8.033934 u},
  {helium-9, 4.002602 u, 9.044 u}, {helium-10, 4.002602 u, 10.053 u}}
```

CombinedEntityClass обладает всеми свойствами обоих классов:

```
In[2]:= EntityProperties[
  CombinedEntityClass["Element", "Isotope", "AtomicMass" → "Entity"]] ===
  Union[EntityProperties["Element"], EntityProperties["Isotope"]]

Out[2]= True
```

Объединим элементы и изотопы по предикату:

```
CombinedEntityClass["Isotope", period 1 elements ELEMENTS,
  EntityFunction[{i, e}, i["AtomicMass"] < e["AtomicMass"]]] [
  EntityProperty["Isotope", "Name"], EntityProperty["Element", "Name"]]

Out[3]= {{protium, hydrogen}, {protium, helium}, {deuterium, helium},
  {tritium, helium}, {helium-3, helium}, {lithium-3, helium}}
```

Ножкаглаж

Различные спецификации соединения приведут к разному поведению в отношении несовпадающих сущностей:

Объединим два класса: элементы I периода и вновь созданный класс:

```
In[4]:= EntityList[period 1 elements ELEMENTS]

Out[4]= {hydrogen, helium}

In[5]:= isotope = FilteredEntityClass["Isotope",
  EntityFunction[i, i["Element"] == hydrogen ELEMENT || i["Element"] == lithium ELEMENT]];
```

```
In[]:= EntityList[isotope] (*отфильтрованный класс сущностей*)

Out[=]= {protium, deuterium, tritium, hydrogen-4, hydrogen-5, hydrogen-6,
hydrogen-7, lithium-3, lithium-4, lithium-5, lithium-6, lithium-7,
lithium-8, lithium-9, lithium-10, lithium-11, lithium-12, lithium-13}

CombinedEntityClass[isotope, period 1 elements ELEMENTS, "Element" → "Entity", "Inner"] [
{EntityProperty["Isotope", "Name"], EntityProperty["Element", "Name"]}]
(*объединяются только пары, для которых совпадают наименования элемента*)

Out[=]= {{protium, hydrogen}, {deuterium, hydrogen},
{tritium, hydrogen}, {hydrogen-4, hydrogen}, {hydrogen-5, hydrogen},
{hydrogen-6, hydrogen}, {hydrogen-7, hydrogen}]

CombinedEntityClass[isotope, period 1 elements ELEMENTS, "Element" → "Entity", "Left"] [
{EntityProperty["Isotope", "Name"], EntityProperty["Element", "Name"]}]
(*объединяются пары, для которых совпадают наименования элемента,
и изотопы, для которых нет соответствия с наименованием элемента*)

Out[=]= {{protium, hydrogen}, {deuterium, hydrogen}, {tritium, hydrogen},
{hydrogen-4, hydrogen}, {hydrogen-5, hydrogen}, {hydrogen-6, hydrogen},
{hydrogen-7, hydrogen}, {lithium-3, Missing[Unmatched]}, {lithium-4, Missing[Unmatched]}, {lithium-5, Missing[Unmatched]}, {lithium-6, Missing[Unmatched]}, {lithium-7, Missing[Unmatched]}, {lithium-8, Missing[Unmatched]}, {lithium-9, Missing[Unmatched]}, {lithium-10, Missing[Unmatched]}, {lithium-11, Missing[Unmatched]}, {lithium-12, Missing[Unmatched]}, {lithium-13, Missing[Unmatched]}}

CombinedEntityClass[isotope, period 1 elements ELEMENTS, "Element" → "Entity", "Right"] [
{EntityProperty["Isotope", "Name"], EntityProperty["Element", "Name"]}]
(*объединяются пары, для которых совпадают наименования элемента,
и элементы, для которых нет соответствия с элементами изотопа*)

Out[=]= {{protium, hydrogen}, {deuterium, hydrogen}, {tritium, hydrogen},
{hydrogen-4, hydrogen}, {hydrogen-5, hydrogen}, {hydrogen-6, hydrogen},
{hydrogen-7, hydrogen}, {Missing[Unmatched], helium}}
```

```
CombinedEntityClass[isotope, period 1 elements ELEMENTS, "Element" → "Entity", "Outer"] [
  EntityProperty["Isotope", "Name"], EntityProperty["Element", "Name"]}]
(*создаются пары из всех изотопов и элементов*)
Out[=]={{protium, hydrogen}, {deuterium, hydrogen},
{tritium, hydrogen}, {hydrogen-4, hydrogen}, {hydrogen-5, hydrogen},
{hydrogen-6, hydrogen}, {hydrogen-7, hydrogen},
{Missing[Unmatched], helium}, {lithium-3, Missing[Unmatched]},
{lithium-4, Missing[Unmatched]}, {lithium-5, Missing[Unmatched]},
{lithium-6, Missing[Unmatched]}, {lithium-7, Missing[Unmatched]},
{lithium-8, Missing[Unmatched]}, {lithium-9, Missing[Unmatched]},
{lithium-10, Missing[Unmatched]}, {lithium-11, Missing[Unmatched]},
{lithium-12, Missing[Unmatched]}, {lithium-13, Missing[Unmatched]}}
```

При объединении одного и того же типа с самим собой необходимо использовать псевдоним:

```
In[=]:= CombinedEntityClass["Element", "other" → "Element", "ElementType"] [
  EntityProperty["Element", "Name"],
  EntityProperty["other" → "Element", "Name"]]] // Short
Out[=]//Short={{hydrogen, hydrogen}, {helium, hydrogen}, {lithium, hydrogen},
<<13919>>, {tennessine, oganesson}, {oganesson, oganesson}}
```

Ножкидглж

Определим новый CombinedEntityClass и отфильтруем полученный класс по 17 (7) группе элементов:

```
In[=]:= combined = CombinedEntityClass["Isotope", "Element", "Element" → "Entity"];
filtered = FilteredEntityClass[combined, EntityFunction[e, e[ептн] == 17]];
Найдём сферы их диагностического применения, удалив те элементы, для которых это свойство не
поддерживается:
In[=]:= DeleteMissing[EntityValue[filtered, {IYkc, bgel nqg Ynnjg Yrgtq}], 1, 2]
Out[=]={{fluorine-18,
{chest cancer, hypoxia, liver disease, squamous cell carcinoma, vocal cord cancer}},
{iodine-124, {apoptosis, mediastinal micrometastates}}}
```

ComplementedEntityClass

Класс дополненных сущностей

ComplementedEntityClass [*ajYqqjj, ajYqq, ajYqq, ...*] класс сущностей, содержащий все сущности класса *ajYqqjj*, которые не входят ни в один из классов *ajYqq*.

В гр҃кжийфю памз пра

ComplementedEntityClass [*ajYqqjj, ajYqq, ajYqq, ...*] фактически эквивалентно неотсортированному дополнению сущностей из класса *ajYqqjj* к сущностям в *ajYqq*.

Форма этих сущностей проявляется при применении `EntityList`.

Результирующие сущности `ComplementedEntityClass` имеют тип, указанному в объекте после применения `EntityList`.

При выполнении дополнения сущностей из разных типов (псч лмпргз лгпмак гржк щуржна) результирующие сущности будут иметь новый тип.

В `ComplementedEntityClass` [`ajYqqi,j`, `ajYqqi`, `ajYqqj`, ... , `SameTestProperties` → { `prtqi,j`, `prtqi`, ... }] использует свойства `prtq` из `ajYqq` для проверки идентичности сущностей.

Если `SameTestProperties` не установлен в `Automatic`, результирующие сущности будут иметь новый тип.

Опция `SameTestProperties` - опция для операций над множествами классов сущностей. Её настройка задает свойства, используемые для определения того, являются ли две заданные сущности одинаковыми.

При `SameTestProperties` → `Automatic` (жлўхглж нм скмкхўлж) равенство между сущностями определяется как равенство между выражениями `Entity`, возвращаемыми `EntityList` с одним аргументом.

`EntityProperties` [`ComplementedEntityClass` [`ajYqqi,j`, `ajYqqi`, `ajYqqj`, ...]] возвращает :

- и все свойства каждого класса `ajYqq`,
- и `EntityProperties` [`ComplementedEntityClass` [...], `prtq`], где `prtq` — это любое из свойств, которое появляется по крайней мере в двух `ajYqq`, когда эти классы имеют разные типы сущностей.

`ComplementedEntityClass` обычно эквивалентен `Complement` :

```
In[]:= EntityList[ComplementedEntityClass[  
    {basic metabolic panel MEDICAL TESTS, liver function test MEDICAL TESTS}]] ===  
Complement[EntityList[{basic metabolic panel MEDICAL TESTS}],  
EntityList[{liver function test MEDICAL TESTS}]]
```

Out[]=

True

При работе с сущностями, поддерживаемыми реляционной базой данных, `ComplementedEntityClass` соответствует оператору EXCEPT.

Ножкгош

Вычислим дополнение двух классов сущностей:

```
mitochondrion PROTEINS ["EntityCount"] (*белки митохондрий*)
```

Out[]=

861

```
axon PROTEINS ["EntityCount"] (*белки, локализованные в аксоне*)
```

Out[]=

33

```
In[1]:= EntityList[
  cEC = ComplementedEntityClass[{"axon PROTEINS", "mitochondrion PROTEINS"}]
(*сформируем класс из белков аксона, не локализованных в митохондриях*)

Out[1]= {adrenergic, beta-2-, receptor, surface, calcitonin isoform CALCA preproprotein,
cholecystokinin preproprotein, cyclin-dependent kinase 5, cyclin-dependent kinase 5, regulatory subunit 1,
cholinergic receptor, nicotinic, alpha 7 precursor, contactin 2 precursor,
contactin 4 isoform a precursor, copine 6, deleted in colorectal carcinoma,
dedicator of cytokinesis 7, dystrobrevin binding protein 1 isoform a, frequenin homolog isoform 1,
ghrelin precursor, glutamate receptor, metabotropic 2 isoform a precursor,
hyperpolarization activated cyclic nucleotide-gated potassium channel 1,
immunoglobulin mu binding protein 2, leucine rich repeat transmembrane neuronal 1,
microtubule-associated protein tau isoform 1, myotrophin,
neurofilament, heavy polypeptide 200kDa, neurofilament, light polypeptide 68kDa,
neurofilament, medium polypeptide 150kDa isoform 1, neurofibromin isoform 1,
palmitoyl-protein thioesterase 1 (ceroid-lipofuscinosis, neuronal 1, infantile), peripherin,
parvalbumin, sema domain, transmembrane domain (TM), and cytoplasmic domain, (semaphorin) 6A,
solute carrier family 6 (neurotransmitter transporter, GABA), member 1,
transforming growth factor, beta 1, transforming growth factor, beta 2}
```

In[2]:= cEC["EntityCount"]

Out[2]= 31

Свойства нового класса соответствуют свойствам типа исходных сущностей:

In[3]:= cEC["Properties"] === Entity["Protein"]["Properties"]

Out[3]= True

Найдём, к какому типу принадлежат результирующие сущности:

In[4]:= EntityTypeName[adrenergic, beta-2-, receptor, surface PROTEIN]

Out[4]= Protein

МпмЎгллмпржнојкглглжӘ

Дополнение сущностей несовместимых типов

Сущности из разных типов по умолчанию считаются разными:

```
In[1]:= homonuclear diatomic molecules CHEMICALS // EntityList
Out[1]= {hydrogen, nitrogen, oxygen, fluorine, chlorine, bromine, iodine}

In[2]:= gaseous elements ELEMENTS // EntityList
Out[2]= {hydrogen, helium, nitrogen, oxygen, fluorine,
neon, chlorine, argon, krypton, xenon, radon}

In[3]:= EntityList[
  cEC1 = ComplementedEntityClass[
    {homonuclear diatomic molecules CHEMICALS, gaseous elements ELEMENTS}]]]
Out[3]= {bromine, chlorine, fluorine, hydrogen, iodine, nitrogen, oxygen}
(*результатирующие сущности имеют новую форму*)

Свойства сущностей нового класса не равны свойствам исходных сущностей:
```

```
In[4]:= hydrogen CHEMICAL ["Properties"] === hydrogen ["Properties"]
Out[4]= False
```

```
In[5]:= hydrogen ELEMENT ... [checked] ["Properties"] === hydrogen ["Properties"]
Out[5]= False
```

SameTestProperties используют для определения равенства между сущностями:

Дополнение сущностей из различных типов фактически приведет к первому классу сущностей:

```
In[6]:= EntityClass["Element", "NobleGas"] // EntityList
Out[6]= {helium, neon, argon, krypton, xenon, radon, oganesson}

In[7]:= EntityClass["Chemical", "Refrigerants"] // EntityList // Shallow
Out[7]//Shallow=
{hydrogen, helium, methane, ammonia, water,
neon, nitrogen, ethylene, ethane, methylamine, <>83>}

In[8]:= ComplementedEntityClass [{noble gases ELEMENTS, refrigerants CHEMICALS}] ["Name"]
Out[8]= {argon, helium, krypton, neon, oganesson, radon, xenon}
```

Причина неправильного дополнения состоит в том, что ряд объектов из разных типов имеют одинаковое названия, но их сущности различны.

Так, сущность Entity["Chemical", "Neon"] не совпадает с сущностью Entity["Chemical", "Neon"].

```
In[1]:= neon ELEMENT === neon CHEMICAL
```

```
Out[1]=
```

False

Переопределение равенства по наименованию дает правильный результат

```
In[2]:= ComplementedEntityClass [ noble gases ELEMENTS ,  
 refrigerants CHEMICALS , SameTestProperties → "Name" ] ["Name"]
```

```
Out[2]=
```

{argon, oganesson, radon, xenon}

Дополнение неявных выражений классов сущностей

При выполнении дополнения неявных выражений **EntityClass** или **FilteredEntityClass** часто более эффективно использовать **And** и **Not** в условии:

```
In[3]:= EntityList [  
 ComplementedEntityClass [  
 FilteredEntityClass ["Element", EntityFunction[x, x["Group"] == 4]],  
 FilteredEntityClass ["Element", EntityFunction[x, x["Period"] == 3]]]]
```

```
Out[3]=
```

{hafnium, rutherfordium, titanium, zirconium}

Полученные классы эквивалентны (за исключением порядка сущностей в списке):

```
In[4]:= EntityList [  
 FilteredEntityClass ["Element", EntityFunction[x, x["Group"] == 4 && x["Period"] ≠ 3]]]
```

```
Out[4]=
```

{titanium, zirconium, hafnium, rutherfordium}

Класс расширенных сущностей

[ExtendedEntityClass](#)

[ExtendedEntityClass](#) [*сущность* / *свойство* → *функция*]
добавления нового свойства “/ *свойство*”,

класс сущностей, полученных из *сущности* путем
значение которого для каждой сущности

получается путем применения [EntityFunction](#) *d*

[ExtendedEntityClass](#) [*сущность* {"/ *свойство* → *функция*, / *свойство* → *функция*, ...}] добавляет свойство / *свойство* определенное
функциями *q*

В гру́жёному фи́ю памзгра

Функция используется для создания класса сущностей путём присвоения сущностям из имеющихся баз новых свойств.

Функция *d* должна быть объектом [EntityFunction](#).

[ExtendedEntityClass](#) работает с любым объектом [EntityStore](#), независимо от того, определен ли он явно или основан на объекте [RelationalDatabase](#).

При работе с сущностями, поддерживаемыми **SQL**, расширить класс сущности нескалярными значениями невозможно.

ExtendedEntityClass [*ажық* / *cu*] → фактически представляет собой виртуальную таблицу базы данных, полученную из *ажық*, с добавленным новым столбцом *all cu fti* значениями *all cl/gv4* для каждой сущности *cl/gv4*. Полное имя нового добавленного свойства: **EntityProperty[ExtendedEntityClass[*ажық*, *all cu fti*], *all cu fti*]**. Новое вычисляемое свойство добавляется даже если у класса уже есть свойство с именем *all cu fti*. Однако это новое свойство должно быть указано с использованием полного имени.

Если имя нового свойства не конфликтует с именами существующих свойств, то **EntityValue[*ажық* / *cu*]** можно использовать для получения значений нового свойства.

Ножкіншт

Добавим вычисляемое свойство (число электронов в молярном объёме) к классу сущностей “OxygenAllotropes”:

```
In[1]:= ExtendedEntityClass[
  EntityClass["Chemical", "OxygenAllotropes"],
  "ElectronMolarVolume" → EntityFunction[c, c[carpmans[r]] / c[knjYptnjsk[c]]][
  {"Name", "ElectronMolarVolume"}]

Out[1]= {{"oxygen", 0.0007145 mole/cm³}, {"ozone", 0.000981 mole/cm³}}
```

Добавим псевдоним для свойства (“AtomicMass”==“AtomicWeight”):

```
In[2]:= ExtendedEntityClass[
  EntityClass["Element", "Metal"],
  "AtomicWeight" → EntityProperty["Element", "AtomicMass"]]["AtomicWeight"]

Out[2]= {6.94 u, 9.0121831 u, 22.98976928 u, 24.305 u, 26.9815384 u, 39.0983 u,
40.078 u, 44.955907 u, 47.867 u, 50.9415 u, 51.9961 u, 54.938043 u, 55.845 u,
58.933194 u, 58.6934 u, 63.546 u, 65.38 u, 69.723 u, 74.921595 u, 85.4678 u,
87.62 u, 88.905838 u, 91.224 u, 92.90637 u, 95.95 u, 97. u, 101.07 u,
102.90549 u, 106.42 u, 107.8682 u, 112.414 u, 114.818 u, 118.710 u, 121.760 u,
132.90545196 u, 137.327 u, 138.90547 u, 140.116 u, 140.90766 u, 144.242 u,
145. u, 150.36 u, 151.964 u, 157.25 u, 158.925354 u, 162.500 u, 164.930329 u,
167.259 u, 168.934219 u, 173.045 u, 174.9668 u, 178.486 u, 180.94788 u,
183.84 u, 186.207 u, 190.23 u, 192.217 u, 195.084 u, 196.966570 u, 200.592 u,
204.38 u, 207.2 u, 208.98040 u, 209. u, 223. u, 226. u, 227. u, 232.0377 u,
231.03588 u, 238.02891 u, 237. u, 244. u, 243. u, 247. u, 247. u, 251. u,
252. u, 257. u, 258. u, 259. u, 262. u, 267. u, 268. u, 269. u, 270. u,
269. u, 277. u, 281. u, 282. u, 285. u, 286. u, 290. u, 290. u, 293. u}
```

Класс фильтрованных сущностей

FilteredEntityClass

FilteredEntityClass [*ажық*] → класс сущностей, в котором хранятся только сущности, для которых **EntityFunction** возвращает значение **True**

FilteredEntityClass [этическим параметром]

класс сущностей, в котором хранятся только сущности, для которых

свойство *прим* имеет значение **True**

В группировке памзера

Функция используется для создания класса сущностей путём выбора сущностей из существующих баз с помощью фильтрации по определённым свойствам.

FilteredEntityClass работает с любым объектом **EntityStore**:

- и определенным явно,
- и основанным на объекте **RelationalDatabase**.

EntityClass [*этическим, прим* → *многорядка*[*этическим*]] фактически эквивалентно **FilteredEntityClass**[*этическим*, **EntityFunction**[*в*, *многорядка*[*этическим*][*и[прим]*]],

за исключением случаев, когда оператором является **TakeLargest**, **TakeSmallest**, **TakeLargestBy**, **TakeSmallestBy** или **NearestTo**.

Ножкощ

Фильтр по логическому свойству сущности:

Найдём, сколько полимеров в классе сущностей содержат ароматические группы:

In[1]:= **FilteredEntityClass**[**polymers** **CHEMICALS**, **gjyptk Yrg**] ["EntityCount"]

Out[1]=

53

Фильтрация сущностей по **EntityFunction**:

Применяя **Boolean** функцию, создадим класс изотопов водорода и лития:

In[2]:= **EntityList**[**FilteredEntityClass**["Isotope",
 EntityFunction[*i*, *i*["Element"] == **hydrogen ELEMENT** || *i*["Element"] == **lithium ELEMENT**]]]

Out[2]=

{**protium**, **deuterium**, **tritium**, **hydrogen-4**, **hydrogen-5**, **hydrogen-6**,
hydrogen-7, **lithium-3**, **lithium-4**, **lithium-5**, **lithium-6**, **lithium-7**,
lithium-8, **lithium-9**, **lithium-10**, **lithium-11**, **lithium-12**, **lithium-13**}

Найдём полимеры, площадь полярной поверхности которых больше 140 \AA^2 (молекулы с полярной площадью поверхности более 140 \AA^2 считаются плохо проникающими через клеточные мембранны):

```
In[=]:= EntityList[FilteredEntityClass[{"polymers", "CHEMICALS"}, 
  EntityFunction[c, c["mmnjnegrYj nnjYpqs pdac YpcY"] > Quantity[140, ("Angstroms")^2]]][
  {"Name", "mmnjnegrYj nnjYpqs pdac YpcY"}]

Out[=]= {{chitosan oligosaccharide lactate, 201 Å²}, {nylon 6/66, 156 Å²}, 
  {bis(3-sulfopropyl)itaconate dipotassium salt, 167 Å²}, 
  {dipentaerythritol penta-/hexa-acrylate, 161 Å²}, 
  {lignosulfonic acid calcium salt, 162 Å²}, {galactan, 247 Å²}, 
  {pentaerythritol tetrakis(3,5-di-tert-butyl-4-hydroxyhydrocinnamate), 186 Å²}, 
  {chitosan, 808 Å²}, {cellulose acetate propionate, 666 Å²}, 
  {lithium polysilicate, 168 Å²}}
```

Получим список элементов, в ядре которых число протонов кратно девяти:

```
In[=]:= FilteredEntityClass["Element", EntityFunction[m, Divisible[m["AtomicNumber"], 9]]][
  {"Name", "AtomicNumber"}]

Out[=]= {{fluorine, 9}, {argon, 18}, {cobalt, 27}, {krypton, 36}, 
  {rhodium, 45}, {xenon, 54}, {europium, 63}, {hafnium, 72}, {thallium, 81}, 
  {thorium, 90}, {einsteinium, 99}, {hassium, 108}, {tennessine, 117}}
```

Класс пересекающихся сущностей IntersectedEntityClass

IntersectedEntityClass [*ajYqq*, *ajYqq*, ...] класс сущностей, содержащий все сущности, общие для всех классов *ajYqq*

В групповой форме памз пра

IntersectedEntityClass [*ajYqq*, *ajYqq*, ...] фактически эквивалентно неотсортированному пересечению сущностей в каждом классе *ajYqq*.

Форма этих сущностей проявляется при применении **EntityList**.

Результирующие сущности **IntersectedEntityClass** принадлежат к типу, указанному в объекте после применения **EntityList**.

При выполнении пересечения сущностей из разных типов (*лсч лмпргз лглмак гпржк щу ржнма*) результирующие сущности будут иметь новый тип.

В **IntersectedEntityClass [*ajYqq*, *ajYqq*, ...]** используется свойства *пртп* из *ajYqq* для проверки идентичности сущностей.

Если **SameTestProperties** не установлен в **Automatic**, результирующие сущности будут иметь новый тип (*лкъ Вгр҃кжк ѹфю памз пра Ank njck cl rcbCl rgwAjYqq*).

EntityProperties [IntersectedEntityClass [*ajYqq*, *ajYqq*, ...]] возвращает :

- все свойства каждого класса *ajYqq*;
- **EntityProperties [IntersectedEntityClass [...], *пртп*]**,
где *пртп* — это любое из свойств, которое появляется по

крайней мере в двух *ажық*, когда эти классы имеют разные типы сущностей.

`IntersectedEntityClass` обычно эквивалентен `Intersection`:

```
In[]:= EntityList[IntersectedEntityClass[
  {basic metabolic panel MEDICAL TESTS}, {liver function test MEDICAL TESTS}]] ===
Intersection[EntityList[{basic metabolic panel MEDICAL TESTS}],
 EntityList[{liver function test MEDICAL TESTS}]]
```

Out[]= True

При работе с сущностями, поддерживаемыми реляционной базой данных, `IntersectedEntityClass` соответствует оператору `INTERSECT`.

Ножкіншт

Вычислим пересечение двух классов сущностей:

```
{mitochondrion PROTEINS} ["EntityCount"] (*белки митохондрий*)
```

Out[]= 861

```
{axon PROTEINS} ["EntityCount"] (*белки, локализованные в аксоне*)
```

Out[]= 33

Сформируем класс из белков аксона, локализованных в митохондриях:

```
In[]:= EntityList[
 iEC = IntersectedEntityClass[{axon PROTEINS}, {mitochondrion PROTEINS}]]
```

Out[]= {cytochrome P450, family 17, dihydropyrimidinase-like 2}

Свойства нового класса соответствуют свойствам типа исходных сущностей:

```
In[]:= iEC["Properties"] === Entity["Protein"]["Properties"]
Out[]= True
```

Найдём, к какому типу принадлежат результирующие сущности:

```
In[]:= EntityTypeName[iEC[[1]]]
Out[]= Protein
```

Мемлекеттегі ножкіншт

Пересечение сущностей несовместимых типов

Пересечение сущностей из различных типов фактически приведет к созданию пустого класса сущностей:

```
In[1]:= homonuclear diatomic molecules CHEMICALS // EntityList
Out[1]= {hydrogen, nitrogen, oxygen, fluorine, chlorine, bromine, iodine}

In[2]:= gaseous elements ELEMENTS // EntityList
Out[2]= {hydrogen, helium, nitrogen, oxygen, fluorine, neon, chlorine, argon, krypton, xenon, radon}

In[3]:= EntityList[
  IntersectedEntityClass[
    homonuclear diatomic molecules CHEMICALS, gaseous elements ELEMENTS]]
Out[3]= {}
```

SameTestProperties используют для определения равенства между сущностями:

```
In[1]:= EntityClass["Element", "NobleGas"] // EntityList
Out[1]= {helium, neon, argon, krypton, xenon, radon, oganesson}

In[2]:= EntityClass["Chemical", "Refrigerants"] // EntityList // Shallow
Out[2]//Shallow= {hydrogen, helium, methane, ammonia, water, neon, nitrogen, ethylene, ethane, methylamine, <<83>>}

In[3]:= IntersectedEntityClass[noble gases ELEMENTS, refrigerants CHEMICALS] // EntityList
Out[3]= {}
```

Причина неправильного пересечения состоит в том, что ряд объектов из разных типов имеют одинаковое название, но их сущности различны.

Так, сущность Entity["Chemical", "Neon"] не совпадает с сущностью Entity["Chemical", "Neon"].

```
In[1]:= neon ELEMENT == neon CHEMICAL
Out[1]= False
```

Переопределение равенства по наименованию даст правильный результат:

```
In[1]:= IntersectedEntityClass[noble gases ELEMENTS,
  refrigerants CHEMICALS, SameTestProperties → "Name"] // EntityList
Out[1]= {helium, krypton, neon} (*результатирующие сущности имеют новый тип*)
```

Дополнение неявных выражений классов сущностей

При выполнении пересечения неявных выражений `EntityClass` или `FilteredEntityClass` часто более эффективно в условии использовать конъюнкцию:

```
In[]:= EntityList[
  IntersectedEntityClass[
    EntityClass["Element", "Period" → 2],
    EntityClass["Element", "Group" → 2]]]

Out[]= {beryllium}
```

Полученные классы эквивалентны (за исключением порядка в списке):

```
In[]:= EntityList[EntityClass["Element", {"Period" → 2, "Group" → 2}]]

Out[]= {beryllium}
```

сущностей *UJ*-базы знаний

`SampledEntityClass`

Класс сущностей, выбранных из типа

`SampledEntityClass[ajYqq /]`

представляет собой класс сущностей, содержащий /

сущностей из класса *ajYqq*

представляет собой класс сущностей, содержащий

`SampledEntityClass[ajYqq /k]`

первые с *k*-го по *l*-ный члены класса *ajYqq*

В группах баз знаний

`SampledEntityClass[ajYqq /]` предоставляет первые / сущностей из типа или класса сущностей *WL*-базы знаний. Это представление может измениться при выполнении операций над классом.

`SampledEntityClass[SortedEntityClass [ajYqq M, /]]` возвращает первые / сущностей из класса `SortedEntityClass`.

Если класс содержит менее / сущностей, `SampledEntityClass[ajYqq /]` эквивалентно классу *ajYqq*.

При использовании сущностей, поддерживаемых реляционной базой данных, `SampledEntityClass` компилируется в операторы LIMIT и OFFSET.

Нажмите

Просмотрим сущности, составляющие тип сущностей:

Просмотрим первые три сущности из типа "Dinosaur":

```
In[]:= EntityList[SampledEntityClass["Dinosaur", 3]]

Out[]= {Sauropsida, Abelisauria, Abelisauridae}
```

Следующие сущности можно просмотреть, указав смещение:

```
In[=]:= EntityList[SampledEntityClass["Dinosaur", {4, 7}]]  
Out[=]= {Abelisauroidea, Allosauridae, Anchisauridae, Ankylopellexia}
```

Просмотрим сущности, составляющие тип “Isotope” и один из классов сущностей этого типа:

```
In[=]:= SampledEntityClass["Isotope", 3] // EntityList  
Out[=]= {protium, deuterium, tritium}  
  
SampledEntityClass[alpha emission ISOTOPES, 3] // EntityList  
Out[=]= {beryllium-8, antimony-104, tellurium-105}
```

МпмУгллмпржножглгжэ

Если сущностей больше, чем запрошено, возвращаются все сущности:

```
In[=]:= EntityList[vitamins CHEMICALS]  
Out[=]= {niacin, pyridoxine, ascorbic acid, pantothenic acid, vitamin A, vitamin B1, riboflavin, vitamin D3, vitamin D2, α-tocopherol, D-alpha-tocopherol}  
  
In[=]:= SampledEntityClass[vitamins CHEMICALS, 11] // EntityList  
Out[=]= {niacin, pyridoxine, ascorbic acid, pantothenic acid, vitamin A, vitamin B1, riboflavin, vitamin D3, vitamin D2, α-tocopherol, D-alpha-tocopherol}
```

Мрлмц глкж

Применение [SampledEntityClass](#) к [SortedEntityClass](#)

```
In[=]:= EntityList[SampledEntityClass[SortedEntityClass[  
    "Dinosaur", EntityProperty["Dinosaur", "Length"] → "Descending"], 2]]  
Out[=]= {Amphicoelias, Argentinosaurus}
```

Тот же класс можно выразить проще, используя [SortedEntityClass](#) с тремя аргументами:

```
In[=]:= EntityList[  
    SortedEntityClass["Dinosaur", EntityProperty["Dinosaur", "Length"] → "Descending", 2]]  
Out[=]= {Amphicoelias, Argentinosaurus}
```

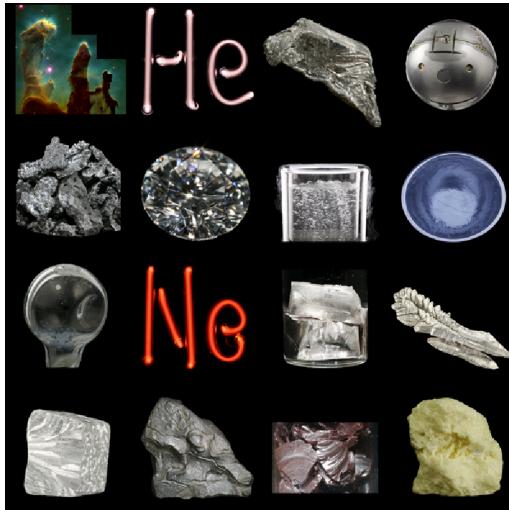
Некоторые типы сущностей поддерживают специальное свойство «**SampleEntities**»:

```
In[]:= EntityValue["Chemical", "SampleEntities"]
Out[=] = {L-lysine, acetic acid, benzene, water, methane, ammonia,
carbon dioxide, hydrogen sulfide, D-(+)-glucose, sodium hydroxide}
```

Ножкаглглжт

Создадим коллаж из 16 элементов:

```
In[]:= ImageCollage[SampledEntityClass["Element", 16] ["Image"]]
Out[=]
```



Класс отсортированных сущностей

SortedEntityClass

SortedEntityClass [*ajYqq пртт*]

класс сущностей, полученный из класса *ajYqq* путем

сортировки в соответствии со значениями свойства *пртт*

класс сущностей, полученный из класса *ajYqq* путем

сортировки по свойству *пртт* в указанном порядке

SortedEntityClass [*ajYqq {пртт₁, пртт₂, ...}*]

класс сущностей, полученный из класса *ajYqq* путем

разрыва связей, последовательно используя значения спецификаций *пртт_и*

SortedEntityClass [*ajYqq фтргса, l*]

представляет первые *l* сущностей класса при

сортировке по *фтргса*

SortedEntityClass [*ajYqq фтргса, {k ul}*]

представляет класс сущностей, содержащий первые с

k-го по *l*-ный члены класса, полученного при сортировке *ajYqq* по *фтргса*

В грүккәйфю памз пра

Когда класс *ajYqq* содержит менее *l* сущностей, в **SortedEntityClass** [*ajYqq фтргса, l*] используются все сущности.

пртт_и может быть задано:

- либо строкой;

- либо EntityProperty;
- либо функцией EntityFunction.

SortedEntityClass [әjYqq npfpcı, /] фактически эквивалентно SortedEntityClass [әjYqq npm → “Descending”, /].

При использовании сущностей, поддерживаемых реляционной базой данных, SortedEntityClass эквивалентен оператору ORDER BY.

Ножкгош

Сортировка класса сущностей:

```
In[1]:= EntityList[Entity["CatBreed"]] (*сущности класса "Порода кошек"*)
Out[1]= {Abyssinian, American Bobtail, American Curl, American Shorthair, American Wirehair,
Australian Mist, Balinese, Bengal, Birman, Bombay, British Longhair,
British Shorthair, Burmese, Burmilla, California Spangled, Chantilly, Chartreux,
Chausie, Chinese Li Hua Mao, Colorpoint Shorthair, Cornish Rex, Cymric, Devon Rex,
Donskoy, Egyptian Mau, European Burmese, European Shorthair, Exotic Shorthair,
German Rex, Havana Brown, Highlander, Himalayan, Japanese Bobtail,
Javanese, Korats, Kurilian Bobtail, LaPerm, Maine Coon, Manx, Minskin,
Munchkin, Nebelung, Norwegian Forest, Ocicat, Ojos Azules, Oriental Longhair,
Oriental Shorthair, Persian, Peterbald, Pixie-Bob, Ragamuffin, Ragdoll,
Russian Blue, Savannah, Scottish Fold, Selkirk Rex, Serengeti, Siamese Modern,
Siamese Traditional, Siberian, Singapura, Snowshoe, Sokoke, Somali, Sphynx,
Tiffanie, Tonkinese, Toyger, Turkish Angora, Turkish Van, York Chocolate}
```

```
In[=]:= EntityList[SortedEntityClass["CatBreed", "MeanWeight"]]
(*сущности класс "Порода кошек", отсортированного по среднему весу*)

Out[=]= {Singapura, Colorpoint Shorthair, Devon Rex, Japanese Bobtail, Munchkin, Minskin,
Cornish Rex, Egyptian Mau, Nebelung, Havana Brown, Korats, Balinese,
Turkish Angora, Javanese, Russian Blue, Peterbald, American Curl, Bombay,
Chantilly, Selkirk Rex, Siamese Modern, Snowshoe, Tonkinese, Donskoy,
Serengeti, Siamese Traditional, Ojos Azules, Toyger, LaPerm, Somali,
Scottish Fold, Cymric, Manx, Kurilian Bobtail, Ocicat, Oriental Shorthair,
European Burmese, Burmese, Chinese Li Hua Mao, Exotic Shorthair, German Rex,
American Bobtail, Tiffanie, Sokoke, Sphynx, American Wirehair, American Shorthair,
European Shorthair, Himalayan, Oriental Longhair, Persian, Abyssinian, Australian Mist,
Burmilla, Chartreux, Birman, Turkish Van, British Shorthair, California Spangled,
Pixie-Bob, Norwegian Forest, Ragamuffin, Siberian, Highlander, Ragdoll,
York Chocolate, Bengal, Maine Coon, British Longhair, Chausie, Savannah}
```

EntityValue[{Singapura CAT BREED, Savannah CAT BREED}, "MeanWeight"]

(*средний вес первой и последней породы*)

```
Out[=]= {2.9 kg, 10. kg}
```

Изменим направление сортировки:

```
EntityList[SortedEntityClass["CatBreed", "MeanWeight" → "Descending"]]
```

```
Out[=]= {Savannah, Chausie, British Longhair, Maine Coon, Bengal, Highlander,
Ragdoll, York Chocolate, Siberian, Ragamuffin, Norwegian Forest, Pixie-Bob,
California Spangled, British Shorthair, Turkish Van, Birman, Australian Mist,
Burmilla, Chartreux, Abyssinian, American Shorthair, European Shorthair,
Himalayan, Oriental Longhair, Persian, American Wirehair, Sphynx, Sokoke,
American Bobtail, Tiffanie, German Rex, Exotic Shorthair, Chinese Li Hua Mao,
Burmese, European Burmese, Oriental Shorthair, Ocicat, Kurilian Bobtail, Cymric,
Manx, Scottish Fold, LaPerm, Somali, Ojos Azules, Toyger, Donskoy,
Serengeti, Siamese Traditional, Chantilly, Selkirk Rex, Siamese Modern, Snowshoe,
Tonkinese, Bombay, American Curl, Peterbald, Russian Blue, Javanese, Balinese,
Turkish Angora, Havana Brown, Korats, Nebelung, Egyptian Mau, Cornish Rex,
Minskin, Munchkin, Devon Rex, Japanese Bobtail, Colorpoint Shorthair, Singapura}
```

Список из первых пяти сущностей:

```
In[=]:= EntityList[  
  SortedEntityClass[ rat mitochondrion genes GENES , npmcg k njcasjYpu cgef r → "Descending", 5 ] ]  
  
Out[=]= { ATPase subunit 6 (rat) , ATPase subunit 8 (rat) ,  
 cytochrome c oxidase subunit 1 (rat) , COXII (rat) , cytochrome c oxidase subunit 3 (rat) }  
  
In[=]:= EntityList[ SortedEntityClass[ rat mitochondrion genes GENES ,  
 npmcg k njcasjYpu cgef r → "Descending", {3, 7} ] ]  
  
Out[=]= { cytochrome c oxidase subunit 1 (rat) , COXII (rat) ,  
 cytochrome c oxidase subunit 3 (rat) , cytochrome b (rat) , NADH dehydrogenase subunit 1 (rat) }
```

Сортировка по нескольким критериям:

```
In[=]:= EntityList[ SortedEntityClass[ "Element", epn ] ]  
  
Out[=]= { hydrogen , lithium , sodium , potassium , rubidium , cesium , francium , beryllium ,  
 magnesium , calcium , strontium , barium , radium , scandium , yttrium , lutetium ,  
 lawrencium , titanium , zirconium , hafnium , rutherfordium , vanadium , niobium ,  
 tantalum , dubnium , chromium , molybdenum , tungsten , seaborgium , manganese ,  
 technetium , rhenium , bohrium , iron , ruthenium , osmium , hassium , cobalt ,  
 rhodium , iridium , meitnerium , nickel , palladium , platinum , darmstadtium ,  
 copper , silver , gold , roentgenium , zinc , cadmium , mercury , copernicium ,  
 boron , aluminum , gallium , indium , thallium , nihonium , carbon , silicon ,  
 germanium , tin , lead , flerovium , nitrogen , phosphorus , arsenic , antimony ,  
 bismuth , moscovium , oxygen , sulfur , selenium , tellurium , polonium ,  
 livermorium , fluorine , chlorine , bromine , iodine , astatine , tennessine ,  
 helium , neon , argon , krypton , xenon , radon , oganesson , lanthanum ,  
 cerium , praseodymium , neodymium , promethium , samarium , europium ,  
 gadolinium , terbium , dysprosium , holmium , erbium , thulium , ytterbium ,  
 actinium , thorium , protactinium , uranium , neptunium , plutonium , americium ,  
 curium , berkelium , californium , einsteinium , fermium , mendelevium , nobelium }
```

In[1]:= EntityList[SortedEntityClass["Element", "g1 gYrgt cl cpegcq"]]

Out[1]=

```
{lawrencium, rutherfordium, dubnium, seaborgium, bohrium, hassium, meitnerium,
darmstadtium, roentgenium, copernicium, nihonium, flerovium, moscovium,
livermorium, tennessine, oganesson, hydrogen, polonium, astatine, radon,
francium, protactinium, neptunium, plutonium, americium, curium, berkelium,
californium, einsteinium, fermium, mendelevium, nobelium, helium, tantalum,
tungsten, osmium, iridium, platinum, gold, radium, actinium, uranium,
lithium, technetium, ruthenium, rhodium, palladium, silver, cadmium,
iodine, xenon, cesium, barium, mercury, thallium, beryllium, gallium,
indium, neodymium, promethium, samarium, europium, gadolinium, terbium,
dysprosium, holmium, erbium, thulium, ytterbium, hafnium, rhenium,
thorium, boron, germanium, tin, lanthanum, praseodymium, lutetium, lead,
carbon, arsenic, zirconium, antimony, cerium, bismuth, nitrogen, selenium,
niobium, tellurium, oxygen, bromine, fluorine, neon, sodium, magnesium,
aluminum, silicon, phosphorus, sulfur, chlorine, argon, potassium, rubidium,
strontium, yttrium, calcium, scandium, titanium, vanadium, chromium,
manganese, iron, cobalt, nickel, copper, zinc, krypton, molybdenum}
```

```
In[1]:= EntityList[SortedEntityClass["Element", {"epn", "g1gYrgt cl cpeq"}]]
```

```
Out[1]= {hydrogen, francium, lithium, cesium, sodium, potassium, rubidium, radium, barium, beryllium, magnesium, strontium, calcium, lawrencium, lutetium, yttrium, scandium, rutherfordium, hafnium, zirconium, titanium, dubnium, tantalum, niobium, vanadium, seaborgium, tungsten, chromium, molybdenum, bohrium, technetium, rhenium, manganese, hassium, osmium, ruthenium, iron, meitnerium, iridium, rhodium, cobalt, darmstadtium, platinum, palladium, nickel, roentgenium, gold, silver, copper, copernicium, cadmium, mercury, zinc, nihonium, thallium, gallium, indium, boron, aluminum, flerovium, germanium, tin, lead, carbon, silicon, moscovium, arsenic, antimony, bismuth, nitrogen, phosphorus, livermorium, polonium, selenium, tellurium, oxygen, sulfur, tennessine, astatine, iodine, bromine, fluorine, chlorine, oganesson, radon, helium, xenon, neon, argon, krypton, protactinium, neptunium, plutonium, americium, curium, berkelium, californium, einsteinium, fermium, mendelevium, nobelium, actinium, uranium, neodymium, promethium, samarium, europium, gadolinium, terbium, dysprosium, holmium, erbium, thulium, ytterbium, thorium, lanthanum, praseodymium, cerium}
```

МПМҔГЛМПРЖНОЖГЛГЛЖӘ

[EntityFunction](#) может быть вторым аргументом:

```
In[2]:= EntityList[SortedEntityClass[{"vitamins", "CHEMICALS"}, EntityFunction[c, c[EntityProperty["Chemical", "Viscosity"]] / c[EntityProperty["Chemical", "MassDensity"]]] → "Descending", 3]]
```

```
Out[2]= {D-alpha-tocopherol, α-tocopherol, pyridoxine}
```

Мрлмц глжә

Применение [SampledEntityClass](#) к [SortedEntityClass](#)

```
In[3]:= EntityList[SampledEntityClass[SortedEntityClass["Dinosaur", EntityProperty["Dinosaur", "Length"] → "Descending"], 2]]
```

```
Out[3]= {Amphicoelias, Argentinosaurus}
```

Тот же класс можно выразить проще, используя [SortedEntityClass](#) с тремя аргументами:

```
In[]:= EntityList[
  SortedEntityClass["Dinosaur", EntityProperty["Dinosaur", "Length"] → "Descending", 2]]

Out[]= {Amphicoelias, Argentinosaurus}
```