Въведение в *Mathematica*

Какво представлява *Mathematica*?

Mathematica е софтуерен пакет, преимуществено (но не единствено) предназначен за нуждите на тъй нареченото техническо (още инженерно) програмиране – разполагаме с информация от научни изследвания (която често може да бъде във вид на картина и/или звук) и по нея искаме да построим модел. Тук обаче не бива да се счита, че говорим само за статистическия подход на "фитване" на информация с прост линеен модел, а моделиране в най-общ смисъл (т.е. търсене на зависимост на базата на експерименти).

Ако искаме да сме по-точни, трябва да кажем, че по същество *Mathematica* представлява компютърна система за символно смятане (или още по-точно, както е на английски, система за компютърна алгебра: Computer Algebra System). Вж. [*1.1i*] и [*1.2i*]. За сравнение с други програми от вида вж. [*1.3i*]

Техническото програмиране само по себе си изисква множество специализирани инструменти за:

- документация и планиране
- анализ и обработка на експерименталните данни
- конструиране на математическия модел (ако желаем да изпробваме / комбинираме с модели от различен тип, което вече говори за качество на работния процес, ще се наложи да разполагаме с немалко софтуер, с който често се работи по различен начин – различните програми изискват изходната информация да е в специфичен тип, междинните изчисления се правят с различна точност и т.н.)
- визуализация на алгоритмите и получените данни от модела
- често програмите се пишат в среда, различна от тази, в която се разработва математическия модел
- тестване
- пакетиране

Видно е, че става дума за употребата на *голямо* количество *специализиран* софтуер. Това значително забавя процеса и създава известни неудобства. Тъкмо с цел създаването на *единна среда* за улеснено и *качествено* извършване на горните дейности в средата на 80-те започва разработката на специализирани софтуерни пакети, най-известните от които са „трите М“: *Mathematica*, *Matlab* и *Maple*.

Ето и накратко някои конкретни отговора на въпроса "Що е то *Mathematica*?":

- текстов редактор с възможност за директно въвеждане на математически формули от произволен тип (които от своя страна могат лесно да се конвертират в TeX например), удобен за изготвяне на учебници, техническа документация и т.н.
- калкулатор (смятащ с числа и символи, т.е. числено и аналитично) с голяма математическа функционалност:
  - познава както основните математически функции, така и ред специални
  - разбира от анализ: може да интегрира, диференцира, развива в редове, намира трансформациите на Фурие и Лаплас и др.
  - разбира от линейна алгебра: намира обратна матрица, детерминанта, собствени стойности и вектори; пресмята матрични експоненти, ранк; решава линейни системи и др.
  - разбира от диференциални уравнения
  - (естествено) разбира от числени методи
  - решава оптимизационни проблеми
  - представлява мощен статистически инструмент: "фитване" на данни, "изглаждане" на данни, анализ на статистически модели, ANOVA и др.
- изчерпателна база данни с научна информация, познаваща: структурата на гените; структурата на множество химически съединения; текущото метеорологическо състояние в дадена точка от света и др.
- инструмент за математическо моделиране и анализ на данни
- интерпретиран език от високо ниво, включващ множество парадигми на програмиране (макар и не винаги в обичайната им форма): процедурно, функционално, логическо и обектно-ориентирано, и работещ *директно* с информация от най-различен тип (картина, звук, анимация, 3D обекти и др). В такъв смисъл, *Mathematica* позволяв относително лесното (спрямо стандартните езици) писане на програми с научна насоченост

Как *Mathematica* намира такова широко приложение? Първо, структурата на *Mathematica* е разделена на няколко части. Второ, езикът на *Mathematica* е с изключително прост синтаксис, върху който са изградени и другите парадигми. Трето, *Mathematica* е снабдена с богато количество инструменти, предназначени именно за решаване на проблеми от техническото програмиране и в частност на математическото моделиране (бивайки част от процеса на първото).

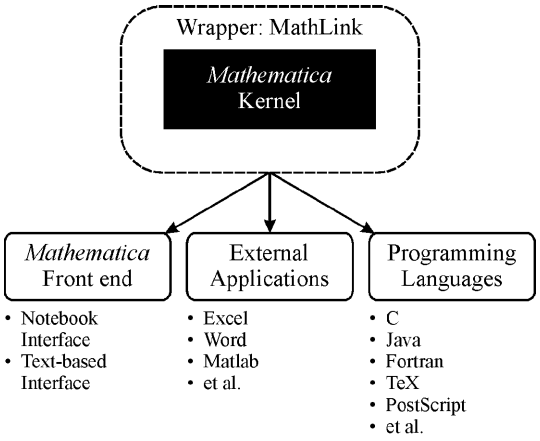
⌘ За повече информация по типичните приложения на *Mathematica*: [*2.1i*], [*2.2i*]
За информация по историята на създаването и развитието на *Mathematica*, както и *Version History*: [*2.3i*], [*2.4i*]
За информация относно съвместимостта между отделните версии: [*2.5m*]

■ *Mathematica*: софтуерна архитектура

Mathematica физически е разделена на два модула: *Kernel* (ядро – извършва необходимите пресмятания) и *Front end* (среда за управление), които комуникират чрез протокола *MathLink*. Тази програмна архитекура има следните предимства:

- Изчисленията могат да се извършват на специално предназначен за това компютър, докато се оперира от друга машина.
- Един *Front end* може да комуникира с няколко *Kernel*-а, ефективно разпределяйки товара между тях
  За повече информация по възможностите за паралелни изчисления (т.е. с няколко ядра / процесора) вж. [*2.6v*]
- Функционалността на *Mathematica* може да бъде използвана и от други програми, респективно от други езици за програмиране

На чертежа отдолу са показани основните модули на *Mathematica*.



За целите на курса работата с *Mathematica* посредством *Notebook Interface* ще е достатъчна. Затова и начините на работа чрез другите интерфейси няма да бъдат повече коментирани, а тук са дадени за пълнота. вж. [*3.1m*]

■ Езикът *Mathematica*.

Въпреки че под името *Mathematica* обикновено подразбираме ядрото и Notebook Interface-ът взети заедно, оттук нататък за удобство условно още ще наричаме *Mathematica* и "езикът на *Mathematica*".

Mathematica е интерпретиран език от високо ниво. За разлика от (изцяло) компилираните езици интерпретираните се изпълняват не на ниво програма, а команда по команда (или "ред по ред"). Затова и повечето езици за техническо програмиране са именно интерпретирани – за да въведем нова функция например, е необходимо единствено да изпълним кодът, касаещ се само до функцията, а не цялата програма. Също така, тъй като езикът е от високо ниво, се наблюдава съществено абстрахиране от хардуера и софтуера (*Mathematica* ще върви (почти) без промени на различен хардуер и операционни системи), т.е. можем да се съсредоточим върху математическия алгоритъм без излишни съображения.

Mathematica е силно абстрактен език. Главна причина за това е унифицираният подход за работа (*принципът за обобщеност*), постигнат чрез употребата на изрази (и коренящ се в идеята за компютрна алгебра).

Изрази

Една програма на *Mathematica* по същество представлява поредица от изрази. Израз наричаме нещо от вида:

```
h[e1, e2, e3, ...]
```

Всеки израз притежава точно една глава (h) и нула или няколко елемента (e_1, e_2, ...). Както елементите, така и главата могат да бъдат *изрази*. Т.е. изразите могат да се вграждат един в друг.

- **Атоми**

Видно е, че горната дефиниция за израз има рекурсивен характер и изисква някаква база (или начални, основни структури). Структурите от най-ниско ниво в *Mathematica* се наричат *атоми*. Има шест типа атоми: символи (*Symbol*), низове (*String*), цели числа (*Integer*), рационални дроби (*Rational*), реални числа (*Real*) и комплесни числа (*Complex*). Тогава можем да обобщим, че основните единици са: променливи (символи), текст и числа. От гледна точка на езика атомите имат само глава (думите в италик в миналото изречение) и нямат елементи, т.е. информацията, която съдържат, не се намира под формата на изрази (и в такъв смисъл не е спазен принципът за обобщеност). Символите представляват поредица от знаци (например: x, Sin, Exp, *t23*, α, β и т.н.), които чрез изразите влизат в някакви взаимовръзки, т.е. чрез тях ще изграждаме зависимостите между елементите в програмите си. *String*-овете са "мъртъв" текст, заграден в кавички (например: "Hey, hey!"), т.е. текст, който не води до зависимости.

Изразите са начин да изразим зависимост, при това йерархична. Представянето на информацията в този вид е не само ефективно от изчислителна гледна точка (очевидно изразите могат добре да се представят чрез дърво), но и удобно, тъй като ще видим, че те са напълно достатъчни за изграждането на всички по-сложни парадигми, заложени в *Mathematica*.

Нека онагледим казаното дотук с малък пример. Имаме:

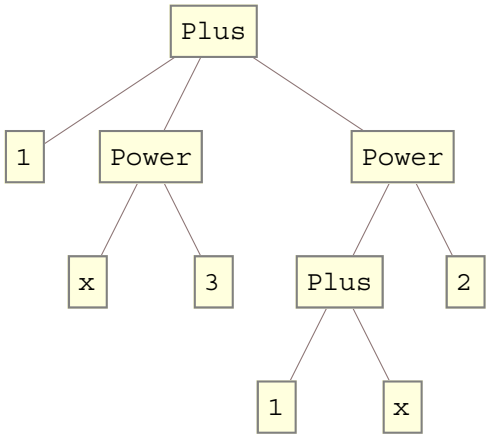
```
x^3 + (1 + x)^2 + 1
```

Във вид на *Mathematica*-изрази горното изглежда така (подробности по въпроса по-късно):

```
Plus[Power[x, 3], Power[Plus[1, x], 1]
```

Тук *Plus*, *Power*, *x*, *1*, *2*, *3* са атоми. Имаме изразите:
$h_1[e_{11}, e_{12}, e_{13}]$, където
h_1 = Plus,
$e_{11} = h_2[e_{21}, e_{22}]$, където
h_2 = Power,
$e_{21} = x$,
$e_{22} = 3$;
$e_{12} = h_3[e_{31}, e_{32}]$, където
h_3 = Power,
$e_{31} = h_4[e_{41}, e_{42}]$, където
h_4 = Plus,
$e_{41} = 1$,
$e_{42} = x$;
$e_{32} = 2$;
$e_{13} = 1$;

Представянето чрез дърво е по-нагледно.



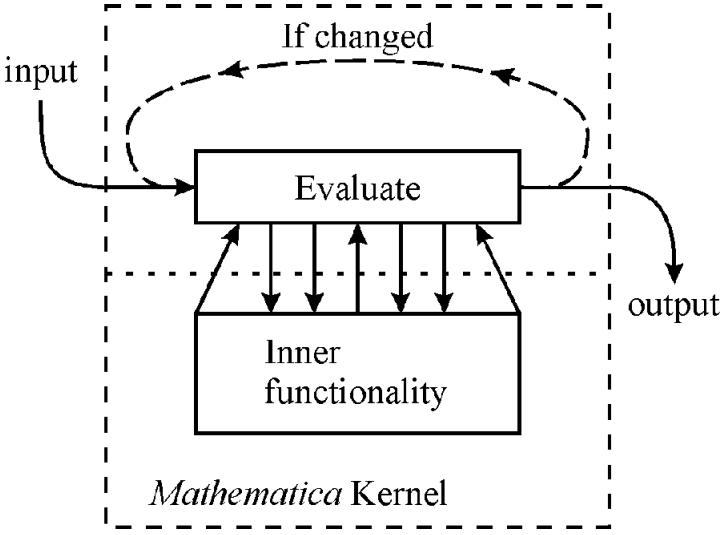
Забележка: тук елементите са подредени в каноничен ред (нещо, което *Mathematica* прави автоматично, стига редът да не е от значение. Повече информация по-долу.

■ Още няколко думи върху ядрото

От казаното дотук би трябвало да е станало ясно, че основната функция на ядрото е интерпретирането на изрази. Естествено, съществуват редица вътрешни (за ядрото) функции, които правят това възможно, но за потребителят е напълно достатъчно да мисли за ядрото само като за "изчислител на изрази". Процесът по обработката на изрази най-общо протича така:

1. Въвежда се входния израз
2. По определен ред се прилагат правила върху него, които най-общо представляват заместване на изрази с някакви други.
3. Ако изразът получен в 2. се различава от 1. повтаряме действието отначало. В противен случай връщаме получения израз.

Точка 2. е разгледана най-общо. По-задълбочено ще я разгледаме по-късно.
На следващата графика е изобразен процисът по обработката.



Основен синтаксис и правила на езика

Имена на символи

- *Mathematica* различава главни и малки букви: **sin** е различно от **Sin**.
- Имената не започват с цифра: *x*, *α*, *f*, *myfun*, *t$23424* са валидни имена
- Всички имена на вградените символи, започват с главна буква, например: **Integrate**, **Plot**, **Exp**, **Pi**, ...
- Често с цел яснота инената са цели думи, изключение правят обичайни съкращения от вида: **Abs**, **Cos**, **Det**, ...
- Ако името се състои от няколко думи, всяка такава е с главна буква и няма разстояния между тях, например: **ListPlot**, **FullForm**, **SetDelayed**, ...

Запетайки

Запетайките се използват за разделяне на елементите на израза. В по-конкретен план: разделят елементите на функция, списък и т.н.

Умножение

Тъй като използването на звезда (*) за умножение е неудобно (а и неестетично), *Mathematica* позволява то да бъде заменено с интервал (␣) или ×. По принцип интервалите се пренебрегват освен в случаите, когато могат да бъдат интерпретирани като умножение. В случаите, когато се подразбира, че става дума за умножение, интервали не се нужни, например: **3 (4 + *a*)**.

Скоби: "(", "[", "{"

- Кръгли скоби **( )**: Използват се за промяна на реда на действията (както в математиката).
- Квадратни скоби **[ ]**: Свързват главата с елементите на израза. В по-конкретен аспект, отделят елементите на функцията, например: Sin[x]. Тъй като вече знаем, че всичко в *Mathematica* е изрази и символи, използването на квадратни скоби (вместо кръгли) за функциите не бива да ни притеснява – по този начин прилагаме правилото за обобщеност, а и избягваме двусмислие (например **sin(*x*)** ще се интерпретира като **sin * *x***
- "Кадрави" скоби **{ }** се използват за конструирането на списъци, например: **{a, b, c, {d, e}}**

В допълнение на казаното горе, да отбележим, че, въвеждайки долните изрази, ще получим еднакви резултати.

```
{3 * 4, 3 × 4, 3 × 4, 3  4}
```

```
{12, 12, 12, 12}
```

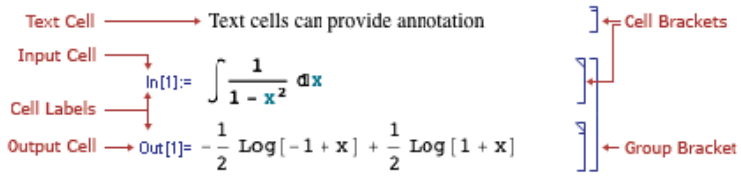
Още да напомним, че докато **2 *a*** е умножение на "2" с "*a*", то **a2** е име на символ.

Notebook interface.

Целта на Notebook интерфейсът е да улесни и същевременно ускори работата с *Mathematica*. При това той е и приемлив текстов редактор – идеален за формули и графики и нелош за книги. Не използвам твърде помпозни думи, тъй като, честно казано, най-слабата страна на *Mathematica* е именно Notebook интерфейсът. При случай, че пишем програми, чертаем графики, правим модели и т.н. надали ще се натъкнем на проблеми. Оформянето на текст, като статии и книги (като например текста на текущото упражнение), обаче е често съпроводено с известно количество "бъгове", които силно се забелязват, но не са документирани.

При стартирането на *Mathematica* забелязваме два основни елемента:
1. прозореца "Untitled", който представлява "тетрадката", в която ще пишем и чрез, която ще общуваме с ядрото; 2. менюто, с което управляваме тетрадката (и нейните елементи), както и някои аспекти на ядрото (например, команди за прекъсване/преустановяване на изчислени; избор на ядро и т.н.)

Всяка тетрадка се състои от текстови (*text*), входни (*input* – за въвеждане на код за изпълнение) и изходни (*output* – дават резултат от изчисление) клетки (*cells*). Между клетките има йерархия с цел по-добра визуализация и структуриране, а и по-мощна текстообработка (възможно е унаследяване на някакви опции от по-висша клетка и т.н.).



Всяка клетка съдържа "една порция" информация, например едно заглавие, един параграф, една бележка, една графика, една дефиниция на функция, една "сметка" и т.н. Естествено, като казвам "една", не искам да ограничa никого, а по-скоро говоря как обикновено (и вероятно най-логично) се процедира.

Нека напишем *Hello world!*. Виждаме, че отдясно се появи знак, индикиращ, че създадохме клетка. За да напишем втора клетка, необходимо е да застанем с курсора между две клетки (в случая над предишната или под нея) и да забележим, че курсорът придобива хоризонтален вид. Нека преместим курсора под предишната клетка. Поява се обявеният хоризонтален курсор. Кликваме веднъж с левия бутон на мишката и се появава дълга хоризонтална сива линия, индикираща, че сме "на път" да въведем нова клетка. Сега единствено се изисква да почнем да въвеждаме информацията. Нека напишем *Hello, indeed!*

По подразбиране новите клекти са от тип "input" затова и тъй като реално пишем непознати символи, текстът се оцветява в синьо. Нека изберем горната клетка чрез иконката в дясно. Сега от менюто Format → Style → Section. Да изберем и втората клетка по същия начин. После: Format → Style → Text. Сега вече всичко е в черно. Предвид целите на курса няма да се занимаваме повече с текстови клетки (поне не официално, на упражнения може да стане дума между другото).
За видео демострация на работа с Notebook Interface: [*4.1v*]

Input клетките съдържат изпълним код. Работата с тях става по следния начин: въвеждаме кода в клетката; натискаме SHIFT + ENTER; евентуално получаваме резултат в *output* клетка. Не получаваме резултат, когато: няма какво да се покаже (например въвеждаме дефиниция на функция); резултатът е бил нарочно скрит (чрез ";"); изчислението не е привършило, поради грешка.

Много от символите могат да се въвеждат от палитрите от менюто *Palettes*. Въпреки това, официалният начин за въвеждане (удобен, когато не можем да ползваме палитри) е **\[Name]**, където *Name* е името на символа. Например **\[Beta]** веднага се трансформира в β и т.н. Има и друг начин за вкарване на доста от символите: ESC*Name*ESC. Например чрез ESC b ESC пак ще получим β. Така се съкращават дори по-дълги символи, наример ESC sum t ESC дава $\sum_{\square=\square}^{\square}\square$ и т.н. Виж [*4.2m*] за по-пълна информация.

⌘ За повече информация върху *Notebook Interface*: [*4.3m*], [*4.4m*]

■ Още малко върху изразите. Основни операции с изрази

■ Форми на въвеждане и извеждане.

Нека отбележим, че в *Mathematica* често съществуват няколко начина за постигане на една и съща цел. Например за извличане на n-ия елемент от списъка **list** бихме могли да ползваме **Part[list, *n*]**, **Take[list, {*n*}]** или пък **list[[*n*]]** или **list⟦n⟧**. Има и още. Това се дължи на няколко основни причини:

- Функциите са от различно ниво на абстрактност. Някои са предназначени за по-общи структури (изрази например), докато други за по-конкретни (списъци например) и като такива имат известни убоства свързани с работата с последните или пък просто работят по-ефективно.
- Информацията, която се "предава" чрез изразите, често съществува в най-различен сходен вид. Пример за това е Part: "Записването във вида **Part[lst, *n*]** е по-неудобно, сравнено с **lst[[n]]**. От друга страна **lst⟦*n*⟧** (забележете разликата между двата знака "]]" и единия "⟧") е по-подходящо за книжен тескт например.

Последният факт не бива да ни притеснява – всеки е свободен да ползва това, което намира за най-удобно. Важно е обаче да разбираме, че:

- Има подобни функции, които дават еднакъв краен резултат, но са предназначени за различни цели. Ще се стараем да ползваме най-подходящата за дадена цел и при дадени условия.

Всичко това е свързано и с необходимостта за естественост (от класическа математическа гледна точка, т.е. от гледна точка на това как сме свикнали да пишем в тетрадките) на въведената и получена информация. Пример за това е естественото желание (и в случая възможност) вместо

```
Integrate[f[x], List[x, a, b]]
```

да запишем

$$\int_a^b f[x]\, dx.$$

Много от изразите ще ги въвеждаме тъкмо в последния вид чрез удобна палитра, предназначена именно за тази цел (или който желае, чрез shortcut).

В *Mathematica* има няколко форми за въвеждане и извеждане на изрази. За да е по-ясно ще разгледаме как изглежда един израз в дадената форма:

$$\frac{x^2}{a} + \sqrt{\text{Sin[y]}}$$

- **FullForm** е "вътрешната" форма на израза

```
Plus[Times[Power[a, -1], Power[x, 2]], Power[Sin[y], Rational[1, 2]]]
```

- **InputForm** е форма за въвеждане на изрази чрез знаци от клавиатурата

```
x^2/a + Sqrt[Sin[y]]
```

- **OutputForm** показва израза чрез знаци от клавиатурата, като ако трябва ги "принтира" на няколко реда

```
 2
x
-- + Sqrt[Sin[y]]
a
```

- **StandardForm** е най-удобната форма за въвеждане и извеждане на изрази, тъй като използва графически символи за някои операции и е недвусмислена (т.е. директно може да се използва за въвеждане)

$$\frac{x^2}{a} + \sqrt{\text{Sin[y]}}$$

- **TraditionalForm** извежда израза в класическа математическа нотация. Удобна за представяне на формули, но не и за въвеждане, тъй като е двусмислена (използва кръгли скоби вместо квадратни за функциите и т.н.).

$$\frac{x^2}{a} + \sqrt{\sin(y)}$$

Формите, за които говорим, представляват видът на записване на изразите в дадена клетка. Ясно е, че коя да е от горните конструкции (без последната) е аналогична на останалите. Ние, като най-удобно, ще пишем в **StandardForm**. Също така има едноименни функции за конвертиране на изрази, записани в един тип, в друг. Например чрез **FullForm[expr]** можем да видим вътрешното представяне на всеки израз. Например за горния израз, записан в **StandardForm**:

```
FullForm[x^2/a + Sqrt[Sin[y]]]
```

```
Plus[Times[Power[a, -1], Power[x, 2]], Power[Sin[y], Rational[1, 2]]]
```

Съществуват още функции за конвертиране на изрази във външни формати (например C, Fortran, TEX, т.н.). Вж. [*5.1m*] за пълна информация върху темата.

■ Основни операции с изрази

Сега мисля, че е време да въведем основните операции с изрази (т.е. операции на най-ниско ниво, които следва да са приложими и за по-конкретни структури). Всъщност тук не си струва да даваме пълен списък и обяснение на всички операции с изрази, тъй като списъкът е твърде дълъг (вж. [*5.2m*]). Ще се запознаем само с някои, а останалите ще се появят паралелно с работния процес.

Смислово се разграничават три вида оператори в *Mathematica*:

- prefix оператори, например **-x**
- infix оператори, например **x + y**
- postfix оператори, например **FullForm[...]**

С цел улесняване на въвеждането (а не писане по традиционния начин **f[g[x], y, ...]**, за всеки вид оператор може да се използва специална нотация.

Чрез f@x се записват префикс операторите. @ има голям приоритет и взима за аргумент на f стоящия директно след @ израз. Например:

```
f@x + y
```

```
y + f[x]
```

Или пък

```
f@x × y
```

```
y f[x]
```

Чрез **x ~ f ~ y** се записват инфиксните оператори.

```
{a, b, c} ~ Join ~ {c, d, e}
```

```
{a, b, c, c, d, e}
```

Последно, най-често използваната нотация е тази за постфикс операторите: **expr // f**

```
Sqrt[x + 1/2] // FullForm
```

```
Power[Plus[Rational[1, 2], x], Rational[1, 2]]
```

А сега операции. За да вземем главата на един израз, използваме **Head[expr]**

```
Head[x + y]
```

```
Plus
```

За да вземем дължината на израз (брой елементи) използваме **Length[expr]**

```
Length[x + y + z + w]
```

```
4
```

```
x + y + z + w // FullForm
```

```
Plus[w, x, y, z]
```

```
Length[{a, b, c, d}]
```

```
4
```

```
FullForm[{a, b, c, d}]
```

```
List[a, b, c, d]
```

```
Length[Sqrt[1/2 + x]]
```

```
2
```

```
Sqrt[1/2 + x] // FullForm
```

```
Power[Plus[Rational[1, 2], x], Rational[1, 2]]
```

Както виждаме, **Length** дава броят на елементите на израза (но не и на подизразите). Също така може да се ползва върху списъци.

За да вземем елемент(и) на израз използваме **Part**

```
Part[x + y + z + w, 3]
```

```
y
```

```
Part[x + y + z + w, {2, 4}]
```

```
x + z
```

```
Part[x + y + z + w, 2 ;; 4]
```

```
x + y + z
```

```
Part[Sqrt[1/2 + x], 1]
```

```
1/2 + x
```

```
Part[{a, b, c, d}, 3]
```

```
c
```

Синтаксисът е виден чрез примерите. Вторият пример показва как да вземем "избрани елементи", докато третият – серия елементи. Операторът ***a ;; b*** се тълкува "от *a* до *b*" и има подобен смисъл на оператора ":" в Matlab.

Удобен запис е **expr[[n]]** и има същия ефект като **Part[expr, n]**

```
(x + y + z + w)[[3]]
```

```
y
```

```
(x + y + z + w)[[{2, 4}]]
```

```
x + z
```

```
(x + y + z + w)[[2 ;; 4]]
```

```
x + y + z
```

Нулевият елемент връща главата на израза (аналогично на Head):

```
(x + y + z + w)[[0]]
```

```
Plus
```

```
{a, b, c, d}[[0]]
```

```
List
```

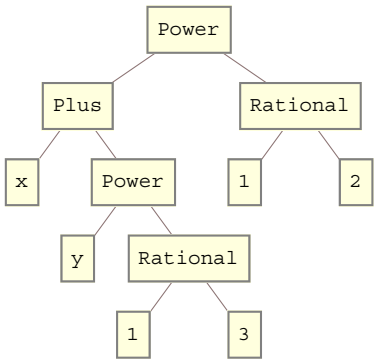
Често изразите включват под изрази. За да вземем техни елементи ползваме няколкоизмерен Part[]:

$$\sqrt{\sqrt[3]{y}+x}$$ **[[1, 2]]**

$$y^{1/3}$$

Горният пример връща първия вместо очаквания втори елемент. Това е поради факта, че изразът първо бива пресметнат (в случая няма какво да се пресметне, но въпреки това *Mathematica* сортира елементите по каноничен ред). Този ефект може да се забележи в дървото на израза:

$$\sqrt{\sqrt[3]{y}+x}$$ **// TreeForm**



Да разгледаме и по-ниските нива.

$$\sqrt{\sqrt[3]{y}+x}$$ **[[1, 2, 2]]**

$$\frac{1}{3}$$

Същото можем за постигнем и по следния начин

```
Sqrt[Surd[y, 3] + x] [[1]][[2]][[2]]
```

$$\frac{1}{3}$$

Ако искаме нещо да не се пресмята, ползваме HoldForm

■ Символите по-отблизо. Процесът на пресмятане в детайл.

Вж. [*11.1i*] и [*11.2i*]

■ Допълнителни ресурси

Ресурсите са дадени във вид: [#b] – препратки към книги, [#i] – към интернет страница, [#v] към онлайн видео, [#m] към страници от "help"-а на *Mathematica*. Последните се отварят, като пълният адрес се постави в полето за търсене на *help browser*-а.

[1.1i] http://bg.wikipedia.org/wiki/Компютърни_системи_за_символно_смятане
[1.2i] http://en.wikipedia.org/wiki/Computer_algebra_system
[1.3i] http://en.wikipedia.org/wiki/List_of_computer_algebra_systems

[2.1i] http://www.wolfram.com/technology/guide/
[2.2i] http://www.wolfram.com/products/mathematica/analysis/
[2.3i] http://www.wolfram.com/company/scrapbook/
[2.4i] http://www.wolfram.com/products/mathematica/quickrevisionhistory.html
[2.5m] tutorial/IncompatibleChanges
[2.6v] http://www.wolfram.com/broadcast/screencasts/parallelcomputing/

[3.1m] tutorial/TheInternalsOfMathematicaOverview

[4.1v] http://www.wolfram.com/broadcast/screencasts/handsonstart/
[4.2m] guide/SpecialCharacters
[4.3m] guide/NotebooksAndDocumentsOverview
[4.4m] tutorial/WorkingWithTheNotebookInterfaceOverview

[5.1m] tutorial/FormsOfInputAndOutput
[5.2m] tutorial/OperatorInputForms

[6.1i] http://functions.wolfram.com/

[11.1i] http://www.verbeia.com/mathematica/tips/HTMLLinks/Tricks_Misc_4.html
[11.2i] http://www.verbeia.com/mathematica/tips/HTMLLinks/Tricks_Misc_3.html