## Вариативная самостоятельная работа №2

## Задание 2.2.2

## Справочник по математическим объектам и их представлению в Scilab

В среду Scilab встроены типы данных, которые в большинстве случаев являются основой для проведения математических расчетов. Мы будем называть их **объектами**. Внутренняя структура объектов заранее предопределена внутри среды и скрыта от глаз пользователей. Работу с памятью при обработке объектов среда также берет на себя. В дальнейшем эти предопределенные объекты могут послужить для создания пользовательских объектов.

В основе своей объекты представляют собой массивы, в которых сохранены определенные данные. В этом разделе мы научимся создавать и манипулировать основными предопределенными объектами, без которых не обходится ни один из сеансов работы с программой.

**Переменная в Scilab** (англ. *Variable*) — это именованный массив всего с одним полем, которое хранит данные некоторого типа.

Создать переменную в среде не составляет труда. Для этого, как вы уже знаете, достаточно ввести ее имя и присвоить ей какое-либо начальное значение. Для переменной будет автоматически выделено место в памяти, а ее область видимости по умолчанию становится локальной.

Некоторые правила, которым должны удовлетворять имена переменных и вообще любых объектов среды:

* Имя переменной может состоять из букв латинского алфавита (верхнего и нижнего регистра) и цифр;
* Имя переменной не может начинаться с цифры, но может начинаться с символов '%', '\_', '#', '!', '$', '?';
* Регистр в имени играет роль, то есть переменные с именами var,VAR ,Var и т. п. разные;
* Запрещено совпадение имени переменной с зарезервированными словами, такими как имена объявленных функций, констант и др.;

Для начала попробуем создать несколько переменных



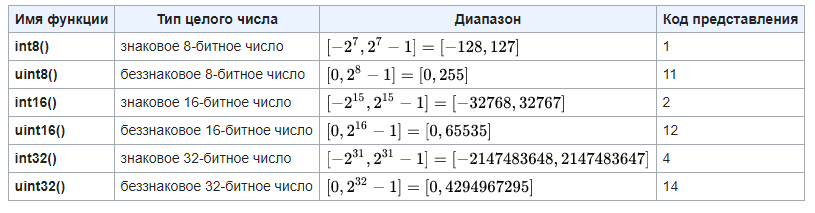
В результате мы получили 4 переменных, 3 из которых хранят вещественные числа и одна — комплексное число. Обратите внимание, как вводится мнимая единица. В среде Scilab мнимая единица является предопределенной константой с именем ' i '. Знак процента указывает на то, что вы обращаетесь к константе.

Целые числа.

В Scilab целые числа возможно создавать только через специальные функции. Во всех остальных случаях числовому значению всегда будет присваиваться вещественный тип данных.

Для хранения целого числа в памяти может быть использовано разное число битов, а именно 8, 16 и 32. От количества используемых битов зависит диапазон целых чисел. Кроме того, имеется возможность включения и отключения знакового бита, что бывает полезно, когда отрицательные целые числа не требуются.

В таблице ниже приведены функции для создания целых чисел и диапазоны возможных значений. Во всех случаях функция возвращает целое число, указанное в качестве аргумента, с определенным способом его хранения в памяти.



Какой тип целочисленной переменной следует применить, пользователь решает сам. Целые числа могут быть использованы как в чисто математических задачах, так и программировании, в качестве счетчиков.

Для работы с целочисленными переменными существует две функции:

* iconvert(X, itype) — меняет представление в памяти в общем случае матрицы из вещественных или целых чисел. Функции при этом следует передавать имя этой матрицы X и код внутреннего представления целого числа itype из последней колонки таблицы выше;
* inttype(X) — возвращает код внутреннего представления в общем случае матрицы целочисленных данных.

Матрицы и векторы

**Вектор в Scilab** — это упорядоченная совокупность элементов (одномерный массив) одного типа данных. Упорядоченность для пользователя в этом смысле проявляется в том, что к каждому элементу вектора можно обратиться по его уникальному порядковому номеру или *индексу*. В среде Scilab все индексы начинаются с единицы, что немного не привычно, так как например в программировании на языке Си те же индексы массивов начинаются с нуля.

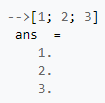
Ранее мы уже не раз создавали вектора и вы уже знаете, что для этого нужно элементы заключить в квадратные скобки, то есть



При объявлении вектора совершенно не обязательно отделять их друг от друга запятыми — достаточно простого пробела. Например, следующее объявление не приведет к синтаксической ошибке.

Вектор может быть записан столбцом или строкой. В первом случае он называется вектор-столбцом, а во втором — вектор-строкой. От того, как представлен вектор (столбцом или строкой), зависит применимость некоторых операторов. Например, из линейной алгебры вы знаете что скалярное произведение двух векторов возможно только, если первый множитель представлен строкой, а второй — столбцом.

Выше мы создавали векторы-строки. Векторы-столбцы создаются путем отделения каждого элемента точкой с запятой, то есть



**Матрица в Scilab** — это двухмерный массив однотипных элементов. Можно понимать матрицу как несколько векторов-строк, записанных столбцом.

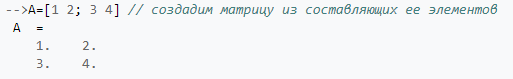
Создать матрицу в Scilab можно одним из нескольких способов:

* Матрицу можно создать из составляющих ее элементов;
* Из имеющихся векторов, упорядочив их строками или столбцами;
* Одной из специальных функций.

В общем случае синтаксическая конструкция имеет вид

[x11, x12, …, x1n; x21, x22, …, x2n; …; xm1, xm2, …, xmn]

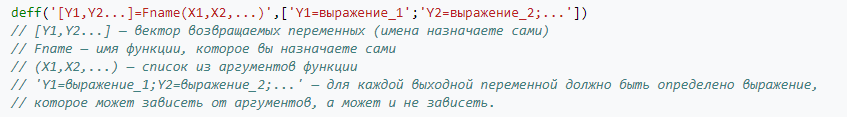
Таким образом, вы создаете векторы-строки, которые отделяете точкой с запятой. Запятая в этом случае, как и с вектором, не обязательна.



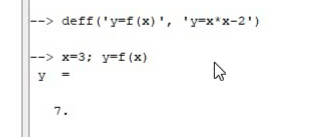
**Функции**

В Scilab есть предопределенные математические функции, такие как тригонометрические, экспонента и другие. Но что, если вам необходимо определить собственную функцию? Далее мы будем отличать математические функции и программируемые функции. Разница между ними заключается в том, что программируемые функции призваны реализовывать некоторый алгоритм, в то время как математическая функция отражает связь между множеством аргументов и множеством значений функции.

Отметим, что математическую функцию можно объявить через программирование, но так как обычно ее тело состоит из одной строчки, то рациональнее всего объявлять ее через специальную функцию **deff()**. Общий синтаксис имеет вид



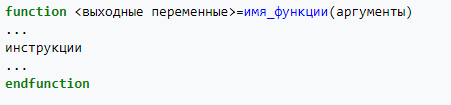
Пример использования



Также следует обратить внимание на функцию **funcprot()** (от англ. *function protection*). С помощью **funcprot()** вы можете защитить ваши функции от случайного их переопределения. Функция принимает один целочисленный аргумент:

* 0 — отключение механизма защиты;
* 1 — формирование предупреждения (используется по умолчанию);
* 2 — формирует ошибку при попытке переопределения существующей функции.

Вообще говоря, при определении функции через **deff()** она может иметь сколь угодно много инструкций, которые необходимо записывать в вектор. Однако, запись становится неудобочитаемой и в этом случае целесообразнее использовать уже вторую конструкцию function…endfunction. Общий синтаксис выглядит так



Данная конструкция, как правило, используется для написания пользовательских программируемых функций, однако, в зависимости от предпочтений пользователя, она может использовать и вместо функции **deff()**. Использовать эту конструкцию можно в интерпретаторе, который воспринимает ее по особому. Сначала необходимо ввести первую строку с ключевым словом *function*, именем функции, аргументами и выходными переменными. Далее вы можете нажать <Enter> и интерпретатор будет воспринимать все, что вы вводите, как инструкции для данной функции до тех пор, пока вы не введете ключевое слово *endfunction*. Если вы не допустили синтаксических ошибок в теле функции, то она будет готова к использованию, в противном случае интерпретатор выведет вам ошибку и объявлять придется заново.