Computing Fundamentals
Derived types

Salvatore Filippone

salvatore.filippone@uniroma2.it

2013–2014

Derived data types

- Collection of other data types, primitive or derived;
- Expand the set of representable objects;
- May have specific operators

Arrays

The first *derived* data type is the

Array: An ordered collection of items, all of the same type, identified by numeric indices.

- May have one or more indices;
- Has size;
- May be created dynamically

Most common cases: vectors and matrices.

Vectors

Entries in a vector are identified by a *position*.

```
a = [1, 4, 9, 16];
a(1)
a(3)
a(0)
a(5)
```

Vectors

Rules:

- The first entry is located at index 1;
- Attempting to read out of bounds causes an error;
- Attempting to write below 1 causes an error;
- Attempting to write above upper bound causes the vector to be automatically extended;
- Vectors can be empty;
- Vectors can be indexed with integer vectors;
- Vectors can be indexed with boolean vectors;

Vectors

Vectors can be empty

```
v = [];
```

and can be shortened

```
v = [1 ,2 ,3 ,4];
v(3) = [];
```

They can be concatenated

```
v1 = [1 ,2 ,3 ,4];
v2 = [5 ,6 ,7 ,8];
v  =  [v1 ,  v2];
```

and they can be indexed with other (integer) vectors

```
v = [1 ,4 ,9 ,16 ,25 ,36 ,49 ,64 ,81 ,100];
ix = [2 ,4 ,6 ,8 ,10];
v(ix)
```

Vectors

Array-valued functions:

- `zeros(1,n);`
- `ones(1,n);`
- `rand(1,n);`
- `linspace(a,b,n);`

Vectors

Auxiliary operators on vectors:

- `find();`
- `length();`
- `size();`
- `end;`
- Transpose `'`

Vectors

Operators on vectors:

- Unary minus - (change sign);
- Addition/subtraction by a scalar `v+1`;
- Multiplication/division by a scalar `alpha*v`;
- Element-by-element operators `.* ./ .^`
- Comparison and logical operators `< > <= >= any find | &`

Vectors

Vectors can be indexed with boolean vectors; for example, `v>0` can be used to index all positive entries, and if we need to double them we can use:

```
idxp = (v>0)
v(idxp)  = 2.*v(idxp)
v(~idxp) = - v(~idxp)
```

Vectors

Library function (and reductions)

- `sum();`
- `mean();`
- `max;`
- `min;`
- `round, ceil, floor, fix;`

Vectors

The triplet notation:

```
n1:str:n2
```

It is an array-valued function:

```
n1, n1+str, n1+2*str, n1+3*str, ..., n1+k*str
```

where `k` is the largest value such that

```
n1+k*str <= n2
```

It can be used to index other arrays.

```
v(1:2:end)
v(2:2:end)
v(end:-1:1)
```

When `str<0` the last constraint is `n1+k*str >= n2`

Matrices

Arrays can be multidimensional:

```
A=[ 1, 2;
    3, 4];
A(1,1)
A(3,4)
```

It is possible to select slices of a matrix:

```
A=[ 1, 2;
    3, 4];
A(1,:)
A(:,2)
```

Matrices have dimensions:

```
vsz=size(A)

[nr, nc] = size(A);
```

Matrices

Predefined matrices:

```
A=zeros(m,n)
```

```
B=ones(m,n);
R=rand(m,n);
```

Matrices can grow and shrink dynamically like vectors

```
A(1,5)=6;
```

```
A(3,:) = [];
```

Matrices

The `reshape` operator:

reshape(A, rows, cols)

rearranges the contents of the matrix to fit a different number of rows/columns.
Book examples

Matrices

Arithmetic operators are to be intended as in linear algebra:

C=A*B

is a valid matrix expression if `A` is `(m,k)`, `B` is `(k,n)` and `C` is `(m,n)`.