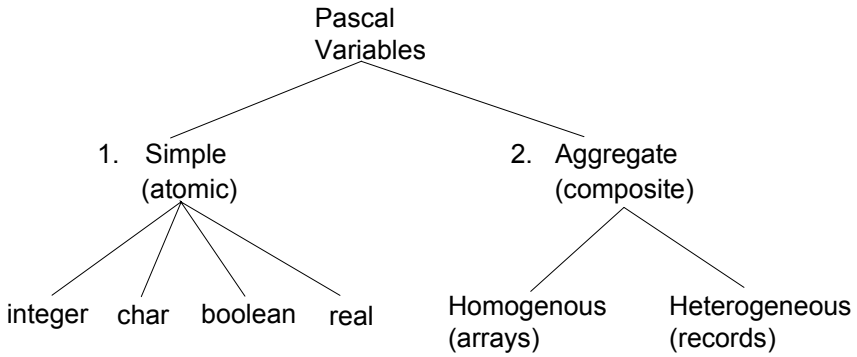
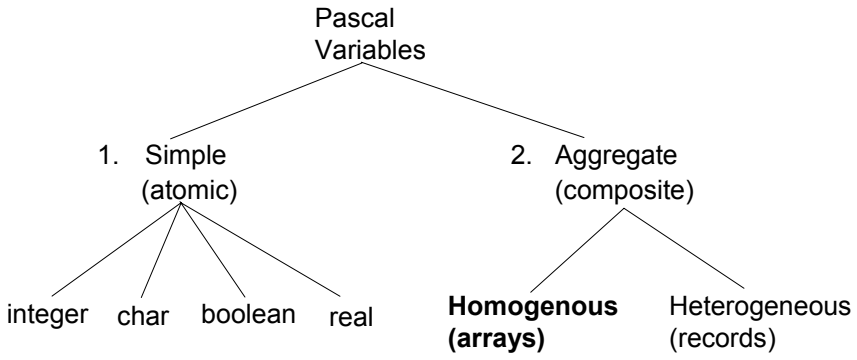
Arrays

In this section of notes you will be introduced to a homogeneous composite type arrays

Types Of Variables



Types Of Variables



Why Bother With Composite Types?

For a compilable example look in Unix under:
/home/231/examples/arrays/classList1.p

```
const
  CLASS_SIZE =  5;
begin
  var stu1     : real;
  var stu2     : real;
  var stu3     : real;
  var stu4     : real;
  var stu5     : real;
  var total    : real;
  var average : real;
```

Why Bother With Composite Types? (2)

```
  write('Enter grade for student number 1: ');
  readln(stu1);
  write('Enter grade for student number 2: ');
  readln(stu2);
  write('Enter grade for student number 3: ');
  readln(stu3);
  write('Enter grade for student number 4: ');
  readln(stu4);
  write('Enter grade for student number 5: ');
  readln(stu5);
  total := stu1 + stu2 + stu3 + stu4 + stu5;
  average := total / CLASS_SIZE;
  writeln('The average grade is ', average:6:2, '%');
```


With Bother With Composite Types? (3)

```
  (* Printing the grades for the class. *)
  writeln('Student1: ', stu1:6:2);
  writeln('Student2: ', stu2:6:2);
  writeln('Student3: ', stu3:6:2);
  writeln('Student4: ', stu4:6:2);
  writeln('Student5: ', stu5:6:2);
end.
```

With Bother With Composite Types? (3)

```
(* Printing the grades for the class. *)
  writeln('Student1: ', stu1:6:2);
  writeln('Student2: ', stu2:6:2);
  writeln('Student3: ', stu3:6:2);
  writeln('Student4: ', stu4:6:2);
  writeln('Student5: ', stu5:6:2);
end.
```

NO!

What's Needed

- A composite variable that is a collection of another type.
- The composite variable can be manipulated and passed throughout the program as a single entity.
- At the same time each element can be accessed individually.
- What's needed…an array!

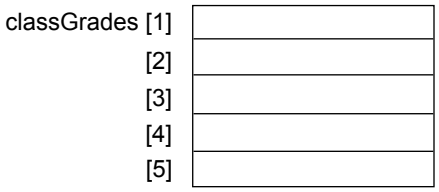
Declaring Arrays

Format:

name: array [*low index..high index*] of *element type*;

Example:

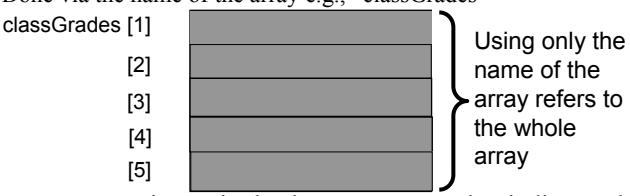
classGrades : array [1..CLASS_SIZE] of real;



Accessing Data In The Array

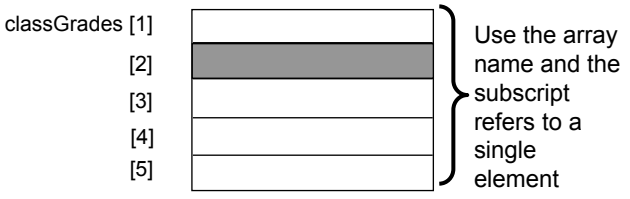
First you need to indicate which array is being accessed

- Done via the name of the array e.g., "classGrades"



If you are accessing a single element, you need to indicate which element that you wish to access.

- Done via the array index e.g., "classGrades[2]"



Assigning Data To The Array

Format:

(Whole array)	(One element)
name of array	name of array [index]

Examples (assignment via the assignment operator):

(Whole array)	(One element)
firstArray := secondArray;	classGrades [1] := 100;

Assigning Data To The Array (2)

Examples (assigning values via read or readln):

(Single element)

```
readln(classGrades[1]);
```

(Whole array – all elements)

```
for i: = 1 to CLASS_SIZE do
begin
    write('Input grade for student No. ', i, ': ');
    readln(classGrades[i]);
end;
```

Assigning Data To The Array (3)

(Whole array – all elements: Character arrays only)

```
var charArray : array [1..5] of char;
readln(charArray);
```


Accessing Data In The Array

Examples (displaying information):

(Single element)

```
writeln(classGrades[1]);
```

(Whole array – all elements)

```
for i := 1 to CLASS_SIZE do
    writeln('Grade for student No. ', i:2, ' ', classGrades[i]:6:2);
```

Accessing Data In The Array (2)

(Whole array – all elements: Character arrays only)

```
var charArray : array [1..5] of char;
write(charArray);
```


Revised Version Using An Array

For a compilable example look in Unix under:
/home/231/examples/arrays/classList2.p

```
const
  CLASS_SIZE =  5;
begin
  var classGrades : array [1..CLASS_SIZE] of real;
  var i           :  integer;
  var total       : real;
  var average     : real;
  total := 0;
```

Class Example Using An Array (2)

```
for i := 1 to CLASS_SIZE do
begin
    write('Enter grade for student no. ', i, ': ');
    readln (classGrades[i]);
    total := total + classGrades[i];
end;
average := total / CLASS_SIZE;
writeln;
writeln('The average grade is ', average:6:2, '%');

for i := 1 to CLASS_SIZE do
   writeln('Grade for student no. ', i, ' is ', classGrades[i]:6:2, '%');
```

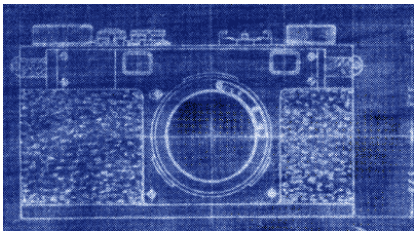

Passing Arrays As Parameters

1. Declare a type for the array.

   e.g.

   ```
   type
      Grades = array [1..CLASS_SIZE] of real;
   ```

   - Declaring A type does not create an instance
     - A type only describes the attributes of a new kind of variable that can be created and used.
     - No memory is allocated.



Passing Arrays As Parameters (2)

2. Declare an instance of this type.

   e.g., var lecture01 : Grades;
   - Memory is allocated!

3. Pass the instance to functions/procedures as you would any other parameter.

```
(Function/procedure call)
 displayGrades (lecture01, average);

(Function/procedure definition)
procedure displayGrades (lecture01 : Grades;
                         average   : real);
```


Passing Arrays As Parameters: An Example

The full example can be found in Unix under /home/231/examples/classList3.p):

```
program classList (input, output);
const
  CLASS_SIZE =  5;
type
  Grades = array [1..CLASS_SIZE] of real;

procedure tabulateGrades (var lecture01 : Grades;
                          var average   : real);
var
  i      : integer;
  total : real;
```

Passing Arrays As Parameters: An Example (2)

```
begin       (* tabulateGrades *)
  total := 0;
  for i := 1 to CLASS_SIZE do
  begin
    write('Enter grade for student no. ', i, ': ');
    readln(lecture01[i]);
    total := total + lecture01[i];
  end;
  average := total / CLASS_SIZE;
  writeln;
end;        (* tabulateGrades *)
```


Passing Arrays As Parameters: An Example (3)

```
procedure displayGrades (lecture01: Grades;
                         average  : real);
var
   i : integer;
begin
  writeln('Grades for the class...');
  for i := 1 to CLASS_SIZE do
    writeln('Grade for student no. ', i, ' is ', lecture01[i]:6:2, '%');
  writeln('The average grade is ', average:6:2, '%');
  writeln;
end;
```

Passing Arrays As Parameters: An Example (4)

```
begin
  var lecture01 : Grades;
  var average   : real;
  tabulateGrades (lecture01, average);
  displayGrades (lecture01, average);
end.
```


Returning Arrays From Functions

1. Declare a type for the array.

   e.g.

   ```
   type
      Grades = array [1..CLASS_SIZE] of real;
   ```

2. Declare an instance of this type.

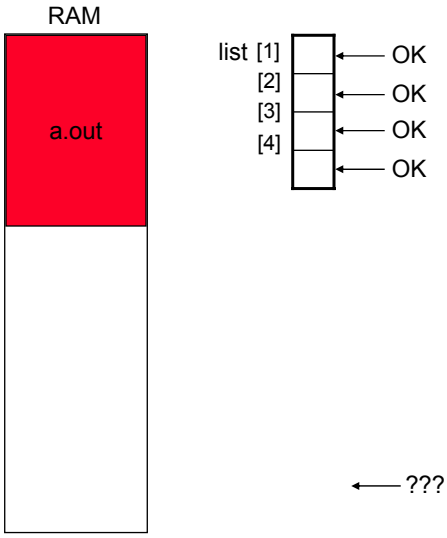
   e.g.,

   ```
   var lecture01 : Grades;
   ```

3. Return the instance of the array as you would any other return value.

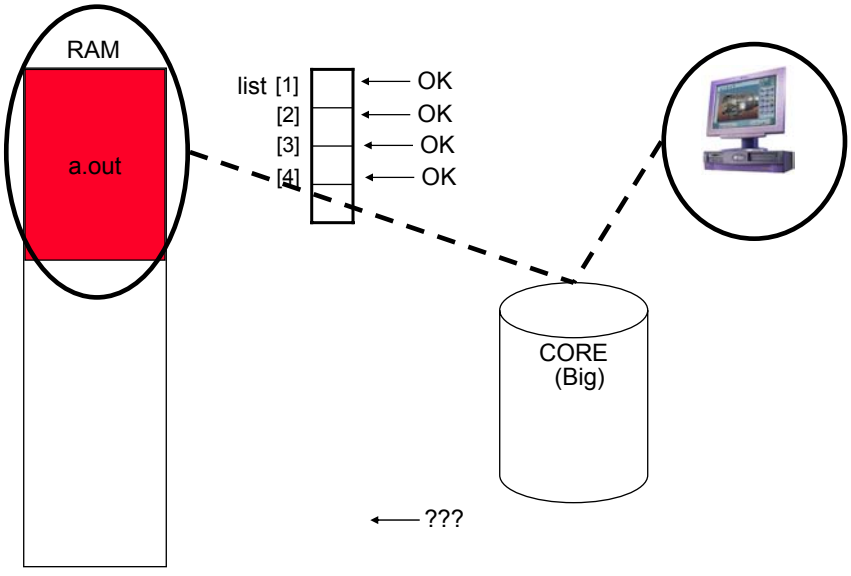
   ```
   (Function/procedure call)
   lecture01 := fun (lecture01);

   (Function/procedure definition)
   function fun (lecture01 : Grades ): Grades;
   ```

Segmentation Faults And Arrays



James Tam

Segmentation Faults And Arrays



Wav file from “The SImpsons”

James Tam

The String Type

It is a special type of character array.

Format for declaration:

var *name* : string [*SIZE*];

Example declaration:

var list2 : string[MAX];

Benefits Of The String Type

1. The end of array is marked.
2. There are a number of built in functions.

Marking The End Of The Array

The full example can be found in Unix under the path:
/home/231/examples/arrays/stringExample.p

```
program stringExample (output);
const
  MAX = 8;
begin
  var list1 : array [1..MAX] of char;
  var list2 : string[MAX];
  list1 := 'abcdefg';
  list2 := 'abcdefg';
  writeln('-', list1, '-');
  writeln('-', list2, '-');
end.
```


The Contents Of A String

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	NULL

Strings Are A Built-In Type[1]

This means that they can be passed as parameter in the same fashion as other built in types:

Format:

procedure *procedureName* (*stringName* : string);

OR

procedure *procedureName* (var *stringName* : string);

Examples:

procedure proc1 (list : string);

OR

procedure proc2 (var list : string);

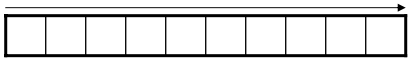
1 For many programming languages and some versions of Pascal

When To Use Arrays Of Different Dimensions

- Determined by the data – the number of categories of information determines the number of dimensions to use.

  Examples:
- (1D array)
  - Tracking grades for a class
  - Each cell contains the grade for a student i.e., grades[i]
  - There is one dimension that specifies the student

  One dimension (which student)



- (2D array)
  - Personal finances program
  - One dimension of information specifies the financial category (cash in or cash out).
  - The other dimension is used to specify the time

When To Use Arrays Of Different Dimensions (2)

•(2D array continued)

Time

Financial category

	January	February	March	…
Income				
-Rent				
-Food				
-Fun				
-Car				
-Misc				
Net income				

When To Use Arrays Of Different Dimensions (3)

•(2D array continued)

•Notice that each row is merely a 1D array

•(A 2D array is an array containing rows of 1D arrays)

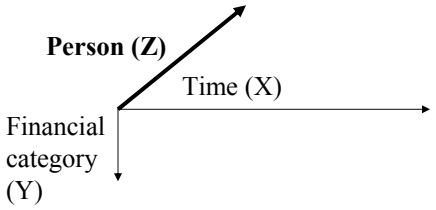
Columns

		[1]	[2]	[3]	[4]
[1]	Income				
[2]	-Rent				
[3]	-Food				
[4]	-Fun				
[5]	-Car				
[6]	-Misc				
[7]	Net income				

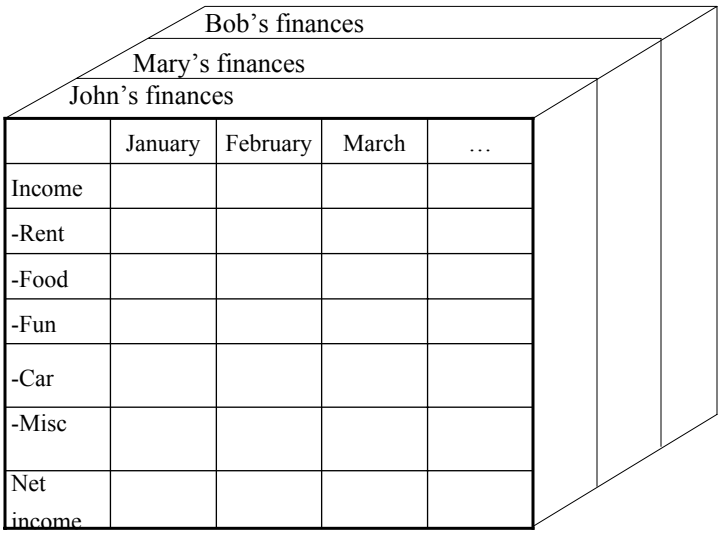
Rows

When To Use Arrays Of Different Dimensions (4)

- (3D array – take the 2D array but allow for multiple people)
- The third dimension specifies whose finances are being tracked.



When To Use Arrays Of Different Dimensions (5)



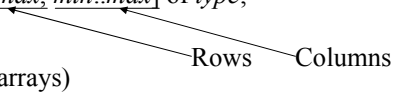
	January	February	March	…
Income				
-Rent				
-Food				
-Fun				
-Car				
-Misc				
Net income				

Declaring Multi-Dimensional Arrays

Format:

(Two dimensional arrays)

Name : array [*min..max*, *min..max*] of *type*;



(Three dimensional arrays)

Name : array [*min..max*, *min..max, min..max*] of *type*;

Example:

```
var johnFinances : array [1..7, 1..7] of real;
var cube         : array[1..3, 1..4, 1..6] of char;
```


Declaring Multi-Dimensional Arrays As A Type

Format:

Type declaration

Type name = array [*min..max*, *min..max*] of *element type*;

Type name = array [*min..max*, *min..max, min..max*] of *element type*;

Variable declaration

Array name : *Type name*;

Declaring Multi-Dimensional Arrays As A Type (2)

Example

Type declaration

```
Finances = array [1..7, 1..7] of real;
Cube = array[1..3, 1..4, 1..6] of char;
```

Variable declaration

```
var johnFinances : Finances;
var aCube        : Cube;
```


Accessing / Assigning Values To Elements

Format:

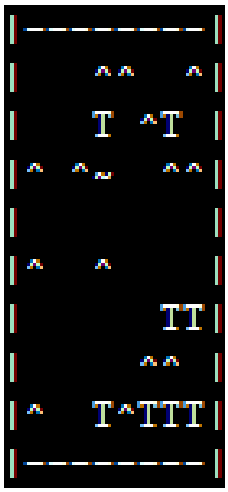
```
name [row][column] :=  name [row][column];
```

Example:

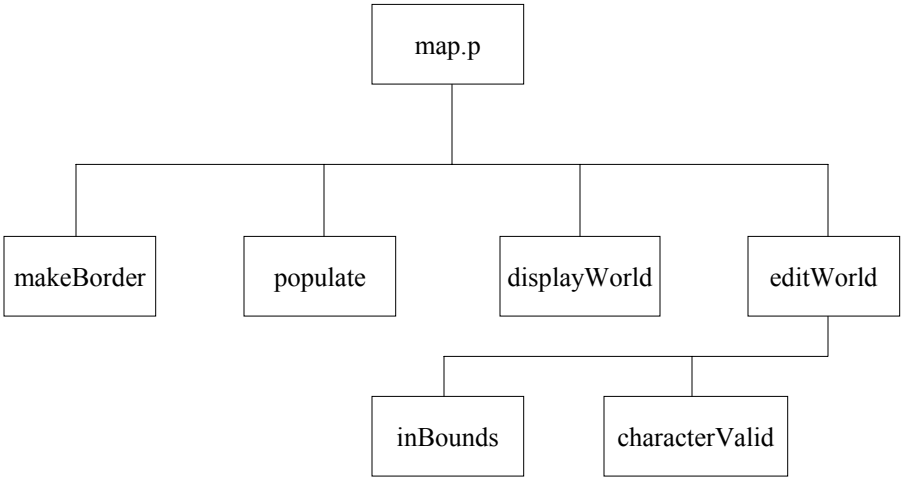
```
finances [1][1] := 4500;
writeln (finances[1][1]);
```

Example Program: Map Generator And Editor

You can find the full program in Unix under:
/home/231/examples/arrays/map.p



Example Program: Map Generator And Editor: Breaking The Problem Down



Example Program: Map Generator And Editor

```
program map (input, output);

const
  MAX_ROWS    = 10;
  MAX_COLUMNS =  10;

type
  Level =  array[1..MAX_ROWS, 1..MAX_COLUMNS] of char;
```


Example Program: Map Generator And Editor (2)

```
procedure makeBorder (var aLevel: Level);
var
  r : integer;
  c : integer;
begin
  for c := 1 to MAX_COLUMNS do
    aLevel[1][c] := '-';

  for c := 1 to MAX_COLUMNS do
    aLevel[MAX_ROWS][c] := '-';

  for r := 1 to MAX_ROWS do
    aLevel[r][1] := '|';

  for r := 1 to MAX_ROWS do
    aLevel[r][MAX_COLUMNS] := '|';

end; (* makeBorder *)
```

Example Program: Map Generator And Editor (3)

```
procedure populate (var aLevel :  Level);
var
  r           : integer;
  c           : integer;
  randomValue : real;
```


Example Program: Map Generator And Editor (4)

```
begin
  for r := 2 to (MAX_ROWS-1) do
  begin
    for c:= 2 to (MAX_COLUMNS-1) do
    begin
      randomValue := random;
      if (randomValue <= 0.05) then
        aLevel [r][c] := '~'
      else if (randomValue <= 0.25) then
        aLevel [r][c] := '^'
      else if (randomValue <= 0.40) then
        aLevel [r][c] := 'T'
      else
        aLevel [r][c] := ' ';
    end; (* inner for: traverse columns *)
  end; (* outer for: traverse rows *)
end; (* populate *)
```

Example Program: Map Generator And Editor (5)

```
procedure displayWorld (aLevel :  Level);
var
  r : integer;
  c :  integer;
begin
  for r := 1 to MAX_ROWS do
  begin
    for c := 1 to MAX_COLUMNS do
    begin
      write(aLevel[r][c]);
    end;
    writeln;
  end; (* for loop - displays world *)
end; (* displayWorld *)
```


Example Program: Map Generator And Editor (6)

```
function inBounds (row      : integer;
                   column : integer):boolean;
begin
  if (row < 2) OR
    (row > (MAX_ROWS-1)) OR
    (column < 2) OR
    (column > MAX_COLUMNS-1) then
     inBounds := false
  else
     inBounds := true;
end; (* inBounds *)
```

Example Program: Map Generator And Editor (7)

```
function characterValid (newCharacter :  char) : boolean;
begin
  if (newCharacter = '~') OR
    (newCharacter = '^') OR
    (newCharacter = 'T') OR
    (newCharacter = ' ') then
    characterValid := true
  else
    characterValid := false;
end; (* characterValid *)
```


Example Program: Map Generator And Editor (8)

```
procedure editWorld (var world : Level);
var
  editChoice       : char;
  charToChange  : char;
  rowToEdit        : integer;
  columnToEdit  : integer;
begin
  writeln;
  write('Enter "Y" or "y" if you wish to edit the world or the return ');
  write('key otherwise: ');
  readln(editChoice);
```

Example Program: Map Generator And Editor (9)

```
if (editChoice = 'Y') OR (editChoice = 'y') then
  begin
    writeln;
    write('Enter row (2 - 9) to edit: ');
    readln(rowToEdit);
    write('Enter column (2 - 9) to edit: ');
    readln(columnToEdit);
    if (inBounds(rowToEdit,columnToEdit) = false) then
    begin
      writeln('Value for row and column must be in the range of 2 - 9');
    end
```


Example Program: Map Generator And Editor (10)

```
    else
    begin
      writeln('What do wish to change this square to? Choices include:');
      writeln('"~" for water');
      writeln('"^" for trees');
      writeln('"T" for a town');
      writeln('" " (A space) for an open field');
      write('Enter choice and hit return: ');
      readln(charToChange);
      if (characterValid(charToChange) = true) then
      begin
        writeln('Changed: row ', rowToEdit,
              ', column ', columnToEdit, ' to ', charToChange);
        world[rowToEdit][columnToEdit] := charToChange;
      end
      else
         writeln('You can only populate the world with water, a forest,'
              ' a town or an empty space.');
    end; (* else *)
  end; (* if edit mode chosen. *)
end; (* editWorld *)
```

Example Program: Map Generator And Editor (11)

```
begin
  var outside      : Level;
  var quitChoice   : char;

  makeBorder(outside);
  populate(outside);
  repeat
  begin
    displayWorld(outside);
    editWorld(outside);
    write('Type "Q" or "q" to quit, or return to continue: ');
    readln(quitChoice);
  end; (* repeat loop *)
  until (quitChoice = 'Q') OR (quitChoice = 'q');
end. (* End of main program *)
```


You Should Now Know

- What is the difference between simple types (atomic) and composite types (aggregate)
- What is the benefit of using homogeneous composite types (arrays)
- How to declare arrays
- How to access or assign values to array elements
- How to work with an entire array
- How to pass instances of arrays into methods and how to return an array from a function.
- What is a segmentation fault and core dump file.
- How to declare and to use instances of a string type.
- The number of dimensions that should be set for an array
- How to declare arrays of multiple dimensions
- How to access and assign values to different parts (elements, rows etc.) of multi-dimensional arrays
- How to scan selected parts of the array using loops