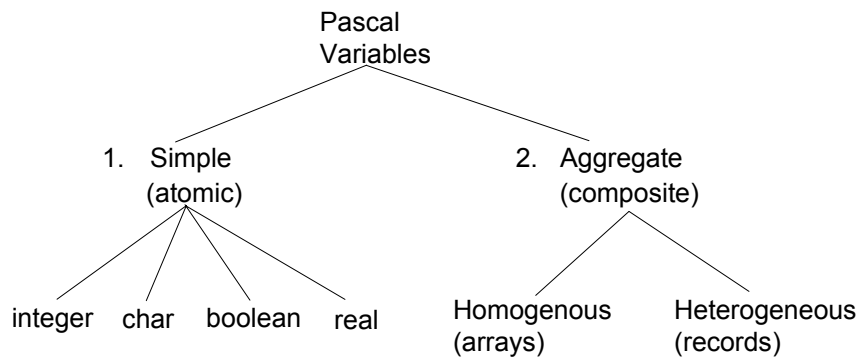


Arrays

In this section of notes you will be introduced to a composite type where all elements must be of the same type (homogeneous): arrays

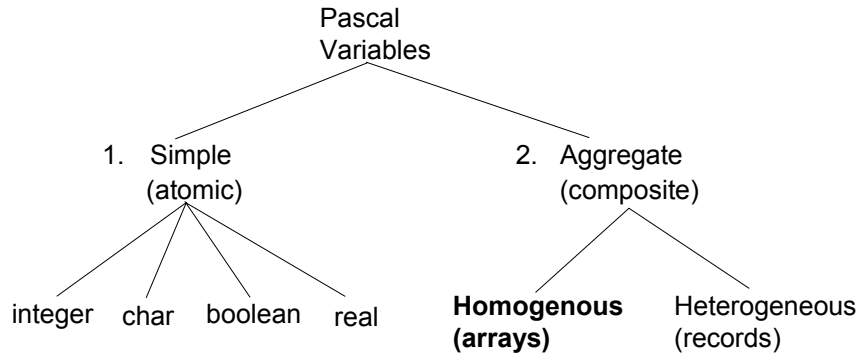
James Tam

Types Of Variables



James Tam

Types Of Variables



James Tam

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;
```

James Tam

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, '%');
```

James Tam

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.
```

James Tam

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:

const

CLASS_SIZE = 5;

: :

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

classGrades [1]

| |
|--|
| |
| |
| |
| |
| |

[2]

[3]

[4]

[5]

James Tam

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”

classGrades [1]

| |
|--|
| |
| |
| |
| |
| |

[2]

[3]

[4]

[5]

} Using only the **name of the array** refers to the whole array

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]”

classGrades [1]

| |
|--|
| |
| |
| |
| |
| |

[2]

[3]

[4]

[5]

} Use the **array name and a subscript** refers to a single element

James Tam

Assigning Data To The Array

Format:

(Whole array)

name of array := value;

(One element)

name of array [index] := value;

Examples (assignment via the assignment operator):

(Whole array)

firstArray := secondArray;

(One element)

classGrades [1] := 100;

James Tam

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;
```

James Tam

Assigning Data To The Array (3)

(Whole array – all elements: Character arrays only)

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

James Tam

Accessing The 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);
```

James Tam

Accessing The Data In The Array (2)

(Whole array – all elements: Character arrays only)

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

James Tam

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;
```

James Tam

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, '%');
```

James Tam

Passing Arrays As Parameters

1. Declare a type for the array.

e.g.

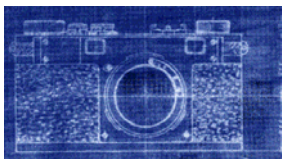
const

```
    CLASS_SIZE = 5;
```

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.



James Tam

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 *)
```

James Tam

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;
```

James Tam

Passing Arrays As Parameters: An Example (4)

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

James Tam

Returning Arrays From Functions

1. Declare a type for the array.

e.g.

```
const
```

```
    CLASS_SIZE = 5;
```

```
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 call)

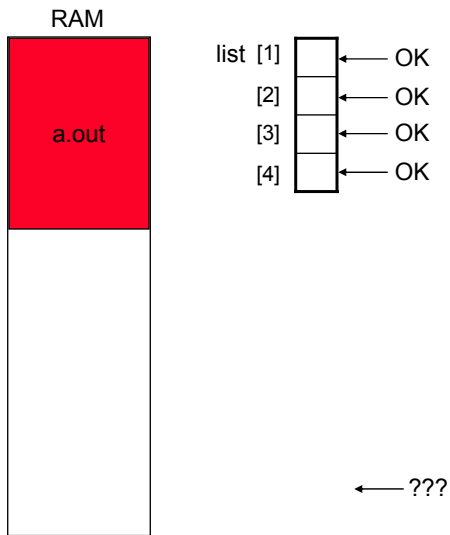
```
lecture01 := fun (lecture01);
```

(Function definition)

```
function fun (lecture01 : Grades) : Grades;
```

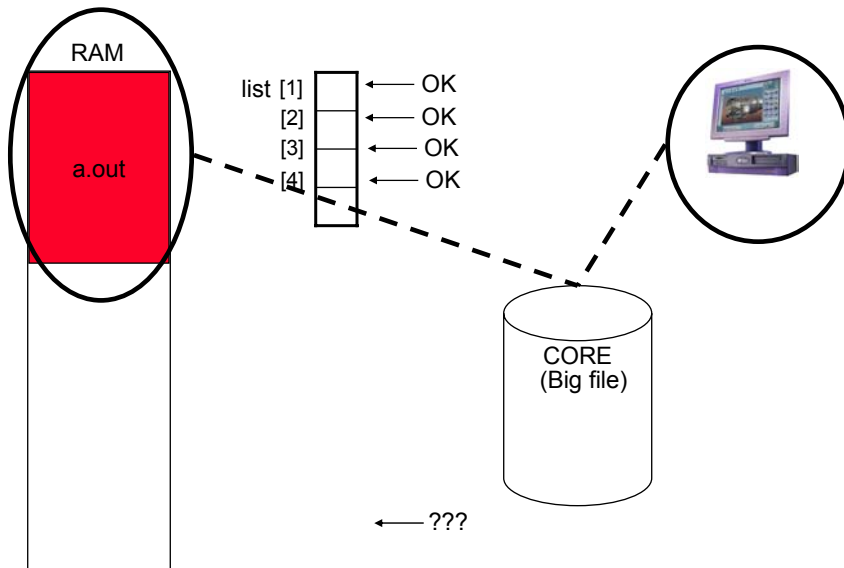
James Tam

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 firstName : string [MAX];
```

James Tam

Benefits Of The String Type

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

James Tam

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.
```

James Tam

The Contents Of A String

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|------|
| [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] |
| 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | NULL |

James Tam

Strings Are A Built-In Type¹

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);
```

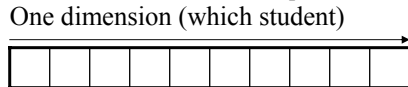
¹ For many programming languages and some versions of Pascal

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 which student's grades are being accessed



- (2D array)
 - Expanded grades program
 - Again there is one dimension that specifies which student's grades are being accessed
 - The other dimension can be used to specify the lecture section

When To Use Arrays Of Different Dimensions (2)

- (2D array continued)

Student

Lecture section

| | First student | Second student | Third student | ... |
|-----|---------------|----------------|---------------|-----|
| L01 | | | | |
| L02 | | | | |
| L03 | | | | |
| L04 | | | | |
| L05 | | | | |
| : | | | | |
| L0N | | | | |

James Tam

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

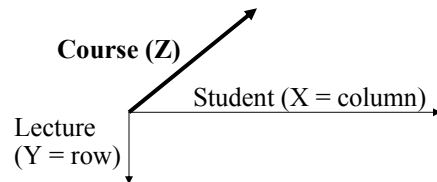
| | [1] | [2] | [3] | [4] |
|-----|-----|-----|-----|-----|
| [1] | L01 | | | |
| [2] | L02 | | | |
| [3] | L03 | | | |
| [4] | L04 | | | |
| [5] | L05 | | | |
| [6] | L06 | | | |
| [7] | L07 | | | |

Rows

James Tam

When To Use Arrays Of Different Dimensions (4)

- (3D array – take the 2D array but allow for multiple courses).
- The third dimension specifies which course grades are being tracked.



Note:

1. The standard approach for specifying the dimensions is to specify the row coordinate (Y) *and then* the column coordinate (X).
2. The size of a dimension must be the same for all elements along that dimension e.g., all rows must be of the same size

James Tam

When To Use Arrays Of Different Dimensions (5)

A 3D array structure representing course grades. The top face of the array is labeled with course numbers: CPSC 235, CPSC 233, and CPSC 231. The front face is a table with rows labeled L01 through L07 and columns labeled Student 1, Student 2, Student 3, and ...

| | Student 1 | Student 2 | Student 3 | ... |
|-----|-----------|-----------|-----------|-----|
| L01 | | | | |
| L02 | | | | |
| L03 | | | | |
| L04 | | | | |
| L05 | | | | |
| L06 | | | | |
| L07 | | | | |

James Tam

Declaring Multi-Dimensional Arrays

Format:

(Two dimensional arrays)

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

Rows Columns

(Three dimensional arrays)

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

Example:

var johnFinances : array [1..3, 1..7] of real;

var cube : array[1..6, 1..6, 1..6] of char;

James Tam

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*;

James Tam

Declaring Multi-Dimensional Arrays As A Type (2)

Example

Type declaration

```
Finances = array [1..3, 1..7] of real;
```

```
Cube = array [1..6, 1..6, 1..6] of char;
```

Variable declaration

```
var johnFinances : Finances;
```

```
var aCube       : Cube;
```

James Tam

Accessing / Assigning Values To Elements

Format:

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

Example:

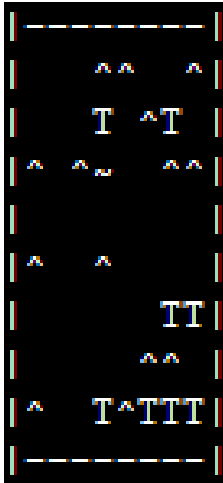
```
finances [1][1] := 4500;
```

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

James Tam

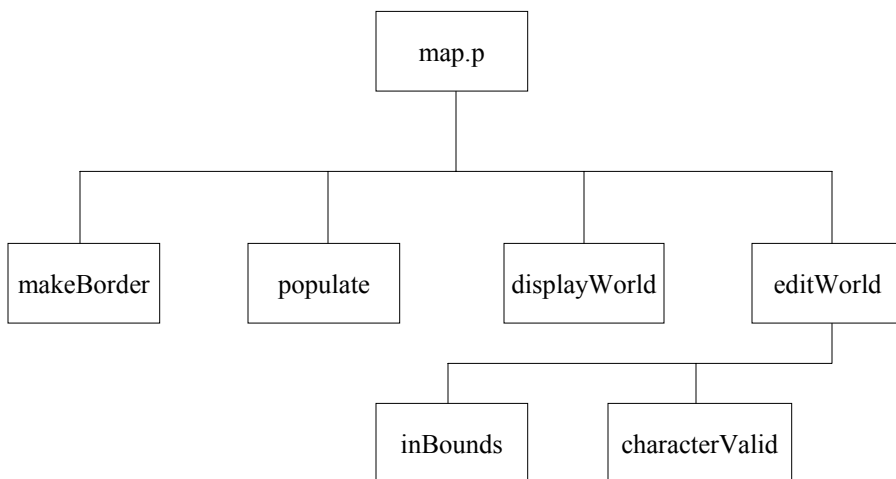
Example Program: Map Generator And Editor

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



James Tam

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



James Tam

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;
```

James Tam

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 *)
```

James Tam

Example Program: Map Generator And Editor (3)

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

James Tam

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 *)
```

James Tam

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 *)
```

James Tam

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 *)
```

James Tam

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 *)
```

James Tam

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);
```

James Tam

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
end
```

James Tam

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 *)
```

James Tam

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 *)
```

James Tam

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 functions and procedures and how to return an array from a function.
- What is a segmentation fault and what is a 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.

James Tam