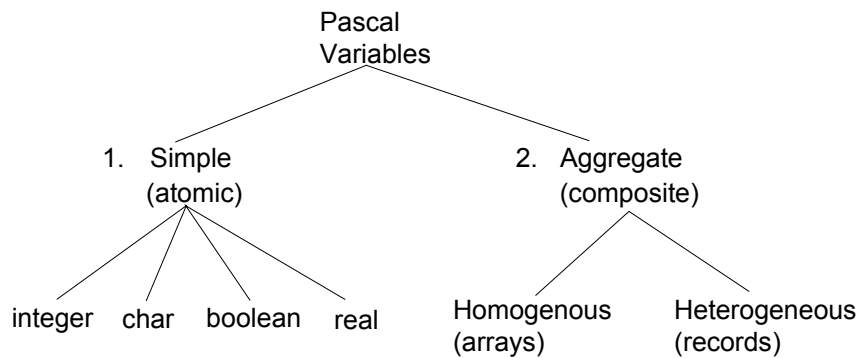


# Arrays

In this section of notes you will be introduced to a homogeneous composite type, one-dimensional arrays

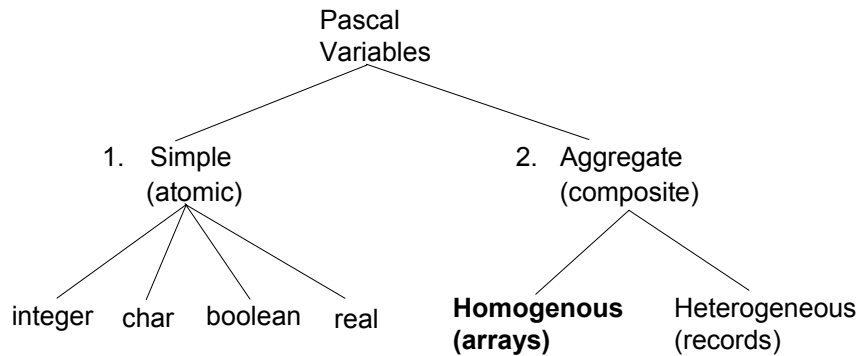
James Tam

## 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;
```

```
    write('Enter grade for student number 1: ');
```

```
    readln(stu1);
```

James Tam

## Why Bother With Composite Types? (2)

```
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:

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]		} Using only the name of the array refers to the whole array
[2]		
[3]		
[4]		
[5]		

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]		} Use the array name and the subscript refers to a single element
[2]		
[3]		
[4]		
[5]		

James Tam

## Assigning Data To The Array

Format:

(Whole array)  
name of array

(One element)  
name of array [index]

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..5] of char;  
readln(charArray);
```

James Tam

## 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);
```

James Tam

## Accessing Data In The Array (2)

(Whole array – all elements: Character arrays only)

```
var charArray : array [1..5] 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

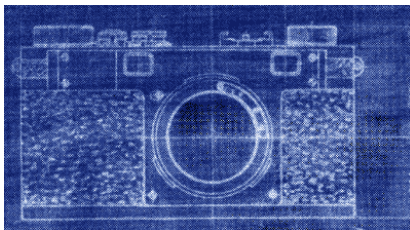
1. Declare a type for the array.

e.g.

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 (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 (L01, average);

(Function/procedure definition)  
procedure displayGrades (L01 : Grades;  
                          average : real);

James Tam

## 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;
```

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

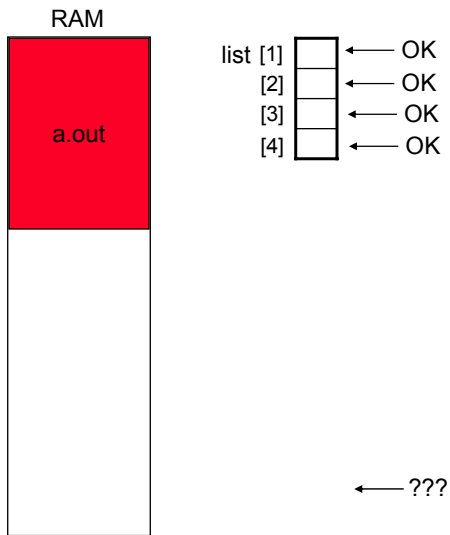
1. Declare a type for the array.  
e.g.  
type  
    Grades = array [1..CLASS\_SIZE] of real;
2. Declare an instance of this type.  
e.g.,  
var lecture01 : Grades;
3. Return the instance of the array as you would any other return value.

(Function/procedure call)  
lecture01 := fun (L01);

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

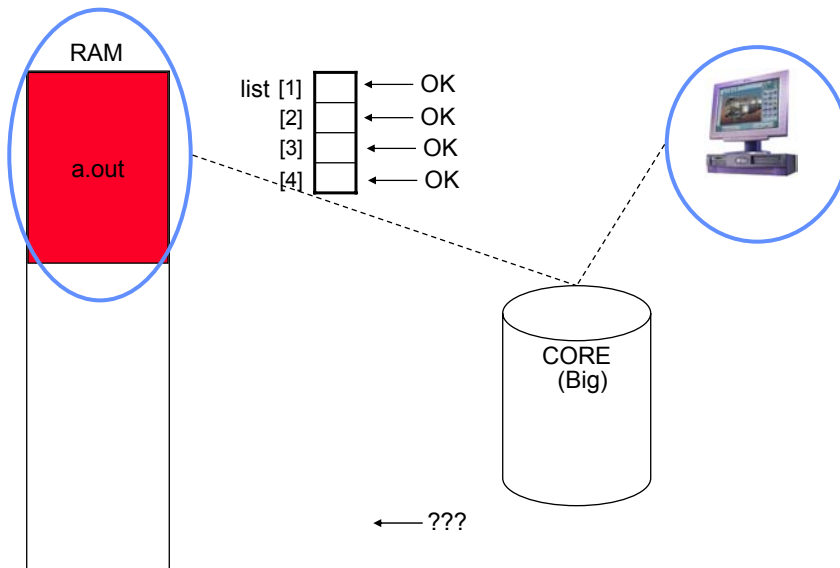
James Tam

## Segmentation Faults And Arrays



James Tam

## Segmentation Faults And Arrays



Wav file from "The Simpsons"

James Tam

## **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.