Arrays

CSCE 110

From James Tam's material
Typical (although simplified) Problem

Write a program that will track student grades in a class. The grades are saved as percentages. The program should allow the user to enter the grade for each student. Then it will display the grades for the whole class along with the average.
Why Bother With Composite Types?

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. *)
write('Student1: ', stu1:6:2);
write('Student2: ', stu2:6:2);
write('Student3: ', stu3:6:2);
write('Student4: ', stu4:6:2);
write('Student5: ', stu5:6:2);
end.

NO!
What’s Needed

• A ___________ variable that is a collection of another type.
  • The ___________ variable can be manipulated and passed throughout the
    program as a ___________ entity.
  • At the same time each ___________ can be accessed ________________.

• An array!
Data Types

simple
- ordinal
- real

structured
- array
- record
- set
- file

pointer

predefined
- integer
- char
- boolean

programmer-defined
- enumerated
- subrange

predefined
- text

programmer-defined
Declaring Arrays

As with any other variable, you must first create an array in memory by declaring an instance.

Format:

\[
\text{name: array [low index..high index] of element type;}
\]

Example:

```
const

CLASS_SIZE = 5;

::

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

classGrades [1]  
|  [2]  |  
|  [3]  |  
|  [4]  |  
|  [5]  |  
```
Accessing Data In The Array

To manipulate an array you need to first indicate which array is being accessed

• Done via the name of the array e.g., “classGrades”

If you are accessing ____________________, you need to indicate ______________ that you wish to access.

• Done via the array index e.g., “classGrades[2]”

J. Michael Moore
Assigning Data To The Array

Format:

(nameOfArray) := value; nameOfArray[index] := value;

Examples (assignment via the assignment operator):

(firstArray := secondArray; classGrades[1] := 100;
Assigning Data To The Array (2)

Examples (assigning values via read or readln):

(_________________________)

write('Input grade for student 1: ');
readln(classGrades[1]);

(_________________________)

for i: = 1 to CLASS_SIZE do
begin
  write('Input grade for student # ', i, ': ');
  readln(classGrades[i]);
Assigning Data To The Array (3)

Example: (Whole array – all elements: ____________________ arrays only)

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

Important note: arrays cannot be ____________________ to read or readln
(exception: one-dimensional character arrays can be passed)
Accessing The Data In The Array

Examples (displaying information):

(__________________________)

writeln('Grade for student 1: ', classGrades[1]:6:2);

(__________________________)

for i := 1 to CLASS_SIZE do

    writeln('Grade for student # ', i:2, ': ', classGrades[i]:6:2);
Accessing The Data In The Array (2)

Example: (Whole array – all elements: Character arrays only)

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

Important note: **arrays cannot be _______________________ to write or writeln**
(exception: one-dimensional character arrays can be passed)
Revised Version Using An Array

const
    CLASS_SIZE = 5;

begin
    var classGrades : array [1..CLASS_SIZE] of real;
    var i       : integer;
    var total   : real;
    var average : real;
Class Example Using An Array (2)

total := 0;
for i := 1 to CLASS_SIZE do
    begin
        write('Enter grade for student # ', 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 # ', i, ' is ',
        classGrades[i]:6:2, ' %');
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 (e.g. a _________________).
     
     - ________________________________.
Passing Arrays As Parameters (2)

2. Declare an instance of this type.
   e.g., \texttt{var lecture01 : Grades;}
   \texttt{\cdot \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots}

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

   (Function/procedure call)
   \texttt{displayGrades (lecture01, average);} 

   (Function/procedure definition)
   \texttt{procedure displayGrades (lecture01 : Grades;
                            average : real);}
Passing Arrays As Parameters: An Example

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 # ', i, ': ');
            readln(lecture01[i]);
            total := total + lecture01[i];
        end;
    average := total / CLASS_SIZE;
    writeln;
end;  (* tabulateGrades *)
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 # ', 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 __________________________________________________________.
   e.g.,
   
   \textbf{const}
   \begin{verbatim}
   CLASS\_SIZE = 5;
   \end{verbatim}
   
   \textbf{type}
   \begin{verbatim}
   Grades = array [1..CLASS\_SIZE] of real;
   \end{verbatim}

2. Declare __________________________________________________________.
   e.g.,
   \begin{verbatim}
   var lecture01 : Grades;
   \end{verbatim}

3. Return the __________________________________________________________.
   (Function call)
   \begin{verbatim}
   lecture01 := fun (lecture01);
   \end{verbatim}
   
   (Function definition)
   \begin{verbatim}
   function fun (lecture01: Grades ): Grades;
   \end{verbatim}
Segmentation Faults And The Array Bounds

```
 RAM
   a.out
   list[1] OK
   list[2] OK
   list[3] OK
   list[4] OK
   list[5]
      ???
   list[0]
```
Segmentation Faults And The Array Bounds

- RAM
- a.out
- CORE (Big file)
- list [1] OK
- [2] OK
- [3] OK
- [4] OK
- ???

J. Michael Moore
Segmentation Faults And The Array Bounds

• When using an array take care not to _________ the bounds.

• Ways of reducing the likelihood of exceeding the bounds of the array:
  1. Use a _____________ in conjunction with arrays e.g.,
     
     ```
     const
     MAX = 5;
     ```

  2. Refer to the _____________ when declaring an array:
     
     ```
     var aList : array [1..MAX] of integer;
     ```

  3. Refer to the _____________ when declaring the type for the array:
     
     ```
     type
     List = array [1..MAX] of integer;
     ```

  4. Refer to the _____________ when iterating/traversing through the array:
     
     ```
     for i := 1 to MAX do
     writeln('Grade for student # ', i, ' is ',
     lecture01[i]:6:2, '%');
     ```
Segmentation Faults And The Array Bounds

5. Make sure that array ___________________________________________.
   • You may need to verify this assumption with debugging statements.

What is the current value of index ‘i’?

Always _____________ your variables before using them: in this case the
index ‘i’ is set to a value __________
_________ of the array before it’s used.

program array1 (output);
begin
    var i    : integer;
    var list :
        array [1..2] of integer;
    list [i] := i;
    writeln (list[i]);
end.

program array2 (output);
begin
    var i    : integer;
    var list :
        array [1..2] of integer;
    i := 2;
    list [i] := i;
    writeln (list[i]);
end.
The String Type

It is a special type of ______________ array.

**Format for declaration:**
```
var name : string [NumberOfElements];
```

**Example declaration:**
```
var firstName : string[MAX];
```
Benefits Of The String Type

1. The _______ of the array is marked.
2. Many ___________ have already been implemented.
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.

Output:

-abcdfg-
-abcdfg-
### The Contents Of The Lists

#### List2 (String)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>‘a’</td>
<td>‘b’</td>
<td>‘c’</td>
<td>‘d’</td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### List1 (Array of characters)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>‘a’</td>
<td>‘b’</td>
<td>‘c’</td>
<td>‘d’</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Strings Are A Built-In Type\textsuperscript{1}

• This means that they can be passed as parameter in the same fashion as other built in types, no type needs to be defined beforehand.

Format:

\begin{verbatim}
procedure procedureName (stringName : string);
  OR
procedure procedureName (var stringName : string);
\end{verbatim}

Examples:

\begin{verbatim}
procedure proc1 (list : string);
  OR
procedure proc2 (var list : string);
\end{verbatim}

\textsuperscript{1} For many programming languages and some versions of Pascal