Arrays

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

Simple Types (Atomic)

1) Integer
2) Real
3) Char
4) Boolean

Composite Types (Aggregate)

1) Homogeneous
   - arrays

2) Heterogeneous
   - records

Why Bother With Composite Types?

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

```
const

   CLASSSIZE =  5;

var

   stu1, stu2, stu3, stu4, stu5: real;

   total, average              : real;

begin

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

   readln(stu1);
```

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 / CLASSSIZE;
writeln('The average grade is ', average:6:2, '%');
```

With Bother With Composite Types (3)

```
(* Printing the grades for the class. *)

Writeln('Student: ', 1, stu1);

Writeln('Student: ', 2, stu2);

Writeln('Student: ', 3, stu3);

Writeln('Student: ', 4, stu4);

Writeln('Student: ', 5, stu5);
```

With Bother With Composite Types (3)

```
(* Printing the grades for the class. *)

Writeln('Student: ', 1, stu1);

Writeln('Student: ', 2, stu2);

Writeln('Student: ', 3, stu3);

Writeln('Student: ', 4, stu4);

Writeln('Student: ', 5, stu5);
```

NO!

Revised Version Using An Array

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

```
const

   CLASSSIZE =  5;

var

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

   i                      :  integer;

   total, average : real;

begin

   total := 0;
```

Class Example Using An Array (2)

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

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

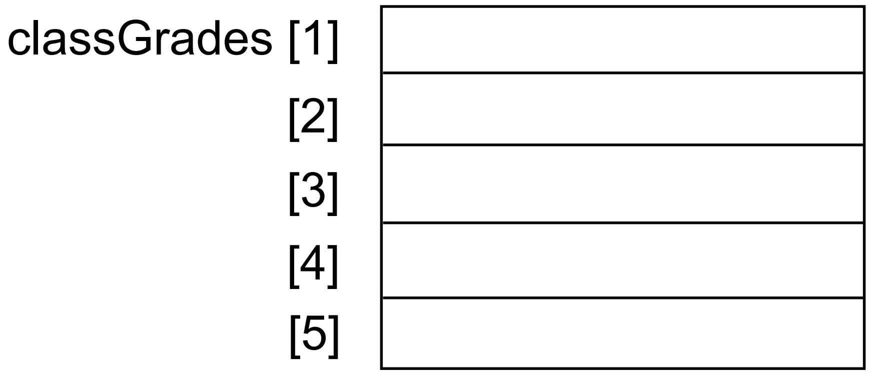
Declaring Arrays

Syntax:

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

Example:

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



Accessing Data In The Array

First need to indicate which array is being accessed

- Done via the name of the array

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

- Done via the array index

Syntax:

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

Examples (assignment via the assignment operator):

(Whole array)

```
firstArray := secondArray;
```

(One element)

```
classGrades [1] := 100;
```

Accessing Data In The Array (2)

Examples (assignment via read or readln):

(Single element)

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

(Whole array – all elements)

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

Accessing Data In The Array (3)

Examples (displaying information):

(Single element)

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

(Whole array – all elements)

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

Common[1] Array Operations

Declaration

- Done previously in this set of notes (slide No. 8, line of code No. 4)

Initialization / Assignment of all elements

- Done previously in this set of notes (slide No. 9, lines of code No. 1 – 4).

Extracting Elements

- Single element – done previously in this set of notes (slides No. 11 & 13)
- All elements – done previously in this set of notes (slide No. 9, line of code No. 11, slides No. 11 & 13)

In order copy between two arrays

- Using the assignment operator – done previously in this set of notes (slide No. 11)
- *Manual copy – coming up*

Reverse order copy between two arrays

- *Manual copy – coming up*

1) Common but by no means complete

In-Order Copy Between Arrays

Method 1: Using the assignment operator

- e.g., array1 := array2;

Method 2: Manual copy

- Use loops and copy from one array to another element-by-element

Example of manual copy (full example can be found in Unix under /home/231/examples/inorderArrayCopy.p)

```
const
   SIZE          =  5;
   MAXVALUE  = 11;
var
   array1 : array [1..SIZE] of integer;
   array2 : array [1..SIZE] of integer;
   i      : integer;
```

In-Order Copy Between Arrays (2)

```
begin
     for i:= 1 to SIZE do
          array1[i] := trunc (RANDOM * MAXVALUE);

    for i:= 1 to SIZE do
         array2[i] := array1[i];

   writeln;
   for i:= 1 to SIZE do
         writeln('array1: ', array1[i]:2, '     array2: ', array2[i]:2);
```

Reverse Order Copy Between Arrays

```
const
   SIZE                    = 5;
   MAXVALUE     = 11;
var
   array1 : array [1..SIZE] of integer;
   array2 : array [1..SIZE] of integer;
   i      : integer;
begin
   for i:= 1 to SIZE do
      array1[i] := trunc (RANDOM * MAXVALUE);
```

Reverse Order Copy Between Arrays (2)

```
for i:= 1 to SIZE do
  begin
    array2[SIZE-i+1] := array1[i];
  end;


writeln;
  for i:= 1 to SIZE do
    writeln('array1: ', array1[i]:2, '     array2: ', array2[i]:2);
  writeln;
```

Summary

What is the difference between simple types (atomic) and composite types (aggregate)?

Why is the benefit of using homogeneous composite types (arrays)?

How are some common operations performed with arrays in Pascal?

- Declaration
- Initialization and assignment
- Extracting elements
- In order copy of elements
- Reverse order copy of elements