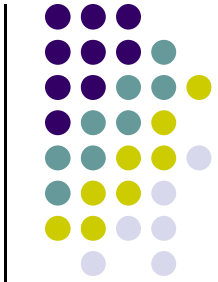
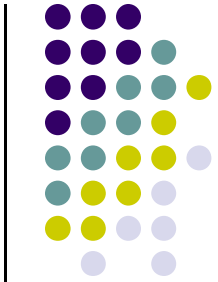
Arrays



Topics

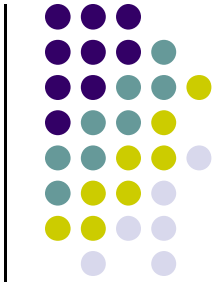
- Definition of a Data Structure
- Definition of an Array
- Array Declaration, Initialization, and Access
- Program Example Using Arrays

Data Types



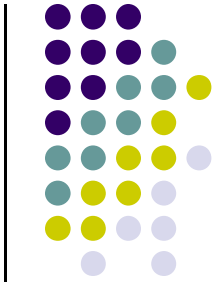
- So far, we have seen only simple variables.
- Simple variables can hold only one value at any time during program execution, although that value may change.
- A **data structure** is a data type that can hold multiple values at the same time.
- The **array** is one kind of data structure.

Arrays



- An array is a group of related data items that all have the same name.
- Arrays can be of any data type we choose.
- Each of the data items is known as an **element** of the array. Each element can be accessed individually.

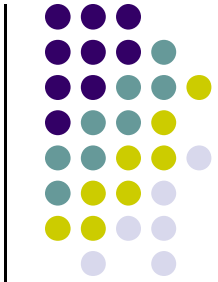
Array Declaration



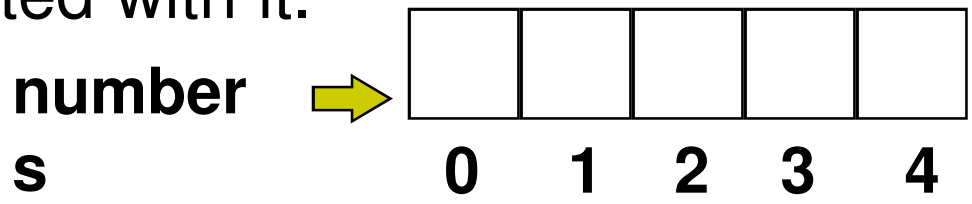
```
var numbers = new Array(5) ;
```

- The name of this array is "numbers".
- It does not initialize the array to 0 or any other value. They contain garbage.

Initializing and Modifying Elements

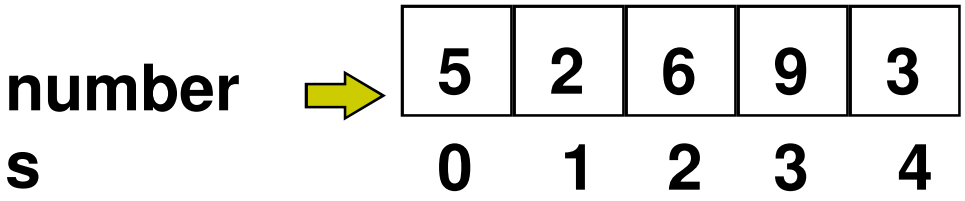


- Each element in an array has a **subscript** (**index**) associated with it.

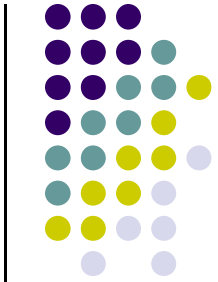


- We can put values into the array using indexing.

```
numbers[0] = 5 ;
numbers[1] = 2 ;
numbers[2] = 6 ;
numbers[3] = 9 ;
numbers[4] = 3 ;
```



Accessing Array Elements



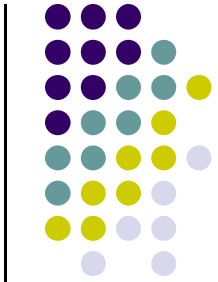
- For this class, subscripts are integers and always begin at zero.
- Values of individual elements can be accessed by **indexing** into the array. For example,

```
alert("The third element = " + numbers[2]);
```

would give the output

The third element = 6.

Accessing Array Elements

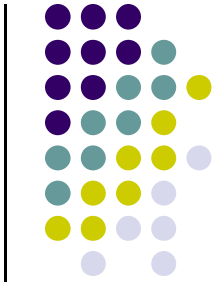


- A subscript can also be an expression that evaluates to an integer.

```
numbers[(a + b) * 2] ;
```

- Caution! It is a logical error when a subscript evaluates to a value that is out of range for the particular array. Some language will handle an **out-of-range error** gracefully and some will not.

Filling Large Arrays

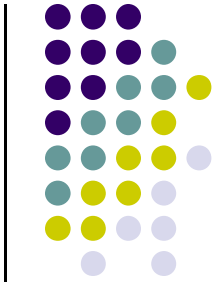


- Since many arrays are quite large, initializing each element individually can be impractical.
- Large arrays are often filled using a for loop.

```
for ( i = 0; i < 100; i++ )
{
    values [ i ] = 0;
}
```

would set every element of the 100 element array "values" to 0.

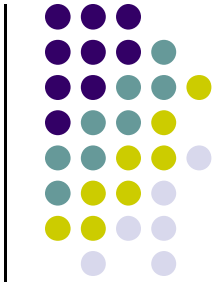
More Declarations



```
var scores = new Array(39);
var gradeCount  = new Array(5);
```

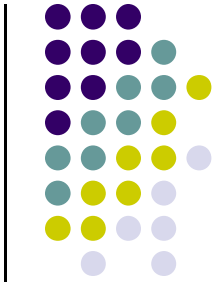
- Declares two arrays: `scores` and `gradeCount`.
- Neither array has been initialized.
- `scores` contains 39 elements (one for each student in a class).
- `gradeCount` contains 5 elements (one for each possible grade, A - F).

Example Using Arrays



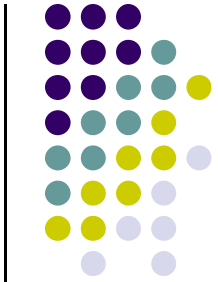
Problem: Find the average test score and the number of A's, B's, C's, D's, and F's for a particular class.

Example Using Arrays



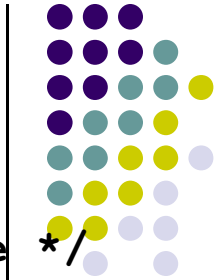
```
                .
                .
                .
<body>
 <script type="text/javascript">
   <!--
    var i;
    var scoreTotal = 0;
    var scores = new Array(39);
    var gradeCount = new Array(5);
    var averageScore;

    PrintInstructions();
```

Example Using Arrays



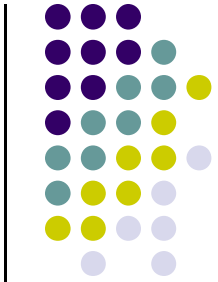
```
/* Initialize grade counts to zero */
for (i = 0; i < 5; i++)
{
  gradeCount[i] = 0;
}

/* Fill score array with scores */
for (i = 0; i < 39; i++)
{
  scores[i] = parseInt(prompt("Enter score:"));
}
```

Example Using Arrays



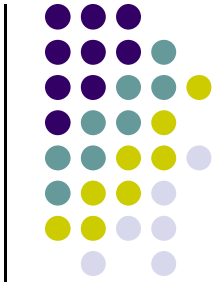
```
/* Calculate score total and count number of each grade */
for (i = 0; i < 39; i++)
{
  scoreTotal += scores[i];
  switch (Math.floor(scores[i]/10))
  {
     case 10:
     case  9: gradeCount[4]++;
              break;
     case  8: gradeCount[3]++;
              break;
     case  7: gradeCount[2]++;
              break;
     case  6: gradeCount[1]++;
              break;
     default: gradeCount[0]++;
   }
}
```

Example Using Arrays



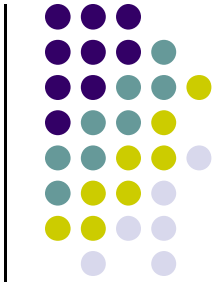
```
    average = FindAverage (scoreTotal, 39);

    /* Display the results to the user */
    string = "The class average is: ";
    string += average.toFixed(2) + "%";
    string += "\nThe grade distribution is:\n";
    string += gradeCount[4] + " A's\n";
    string += gradeCount[3] + " B's\n";
    string += gradeCount[2] + " C's\n";
    string += gradeCount[1] + " D's\n";
    string += gradeCount[0] + " F's";
    alert(string);

   //-->
  </script>
</body>
```

Example Using Arrays



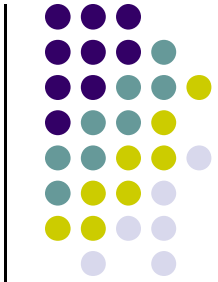
```
/*************************************************
** PrintInstructions - prints the user instructions
** Inputs:  None
** Outputs:  None
*************************************************/
function PrintInstructions()
{
  var string;

  string = "This program calculates the average score\n";
  string += "for a class of 10 students.  It also reports the\n";
  string += "number of A's, B's, C's, D's, and F's.  You will\n";
  string += "be asked to enter the individual scores.\n";
  alert(string);
}
```

Example Using Arrays



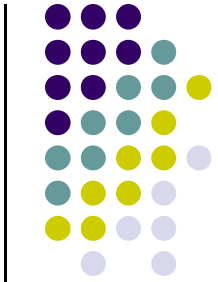
```
/*************************************************
 ** FindAverage - calculates an average
 ** Inputs:   sum - the sum of all values
 **           num - the number of values
 ** Outputs:  the computed average
 *************************************************/
function FindAverage(sum, num)
{
  var average;

  /* Make sure we don't do division by 0 */
  if (num != 0)
  {
    average = sum / num;
  }
  else
  {
    average = 0;
  }
  return average;
}
```

Improvements ?



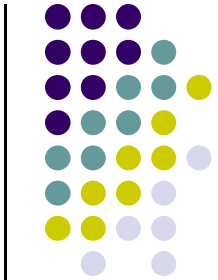
- We're trusting the user to enter valid grades. Let's add input error checking. For this program, the highest possible score is 110.
- If we aren't handling our array correctly, it's possible that we may be evaluating garbage rather than valid scores. We'll handle this by adding all the cases for F's (0 - 59) to our switch structure and using the default case for reporting errors.

Improved Input with Error Checking



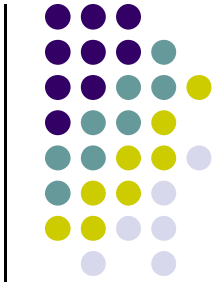
```
/* Fill score array with scores */
 for (i = 0; i < 39; i++)
 {
    scores[i] = parseInt(prompt("Enter score:"));

    /* Make sure score is within correct range */
    while (scores[i] < 0 || scores[i] > 110)
    {
       alert("Your number must be between 0 and 110.");
       scores[i] = parseInt(prompt("Enter score:"));
    }
 }
```

Improved switch() statment



```
switch (Math.floor(scores[i]/10))
{
   case 10:
   case  9: gradeCount[4]++;
            break;
   case  8: gradeCount[3]++;
            break;
   case  7: gradeCount[2]++;
            break;
   case  6: gradeCount[1]++;
            break;
  case 5:  case 4: case 3: case 2: case 1: case 0:
            gradeCount[0]++;
            break;
   default: alert("Error in score!");
            break;
}
```

Working Version of Grades Program



- A working version of the improved program can be found at:

http://userpages.umbc.edu/~dblock/arrays.html

- Note that it will ask for only 10 scores rather than 39.