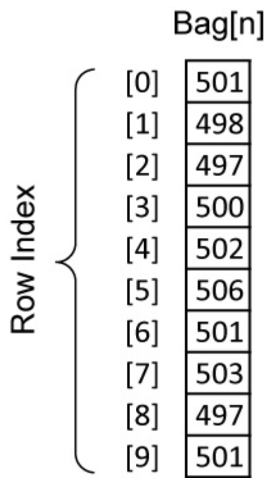
Electronics and Robotics I Week 27
Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings

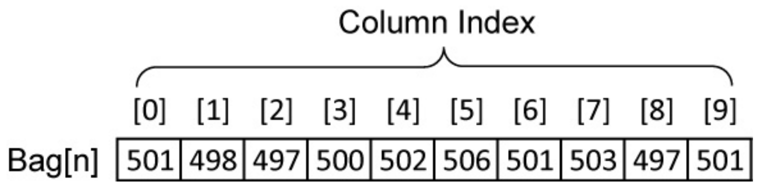
- **Administration:**
  - Prayer
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
- **Arrays:**
  - Definition: An array is a variable that holds multiple values of the same type that can be indexed. The terms in an array can be arranged in some geometric pattern, as in a matrix. A matrix is a rectangular arrangement of numbers in rows and columns.
  - Appearance:
    - One Dimensional Array:



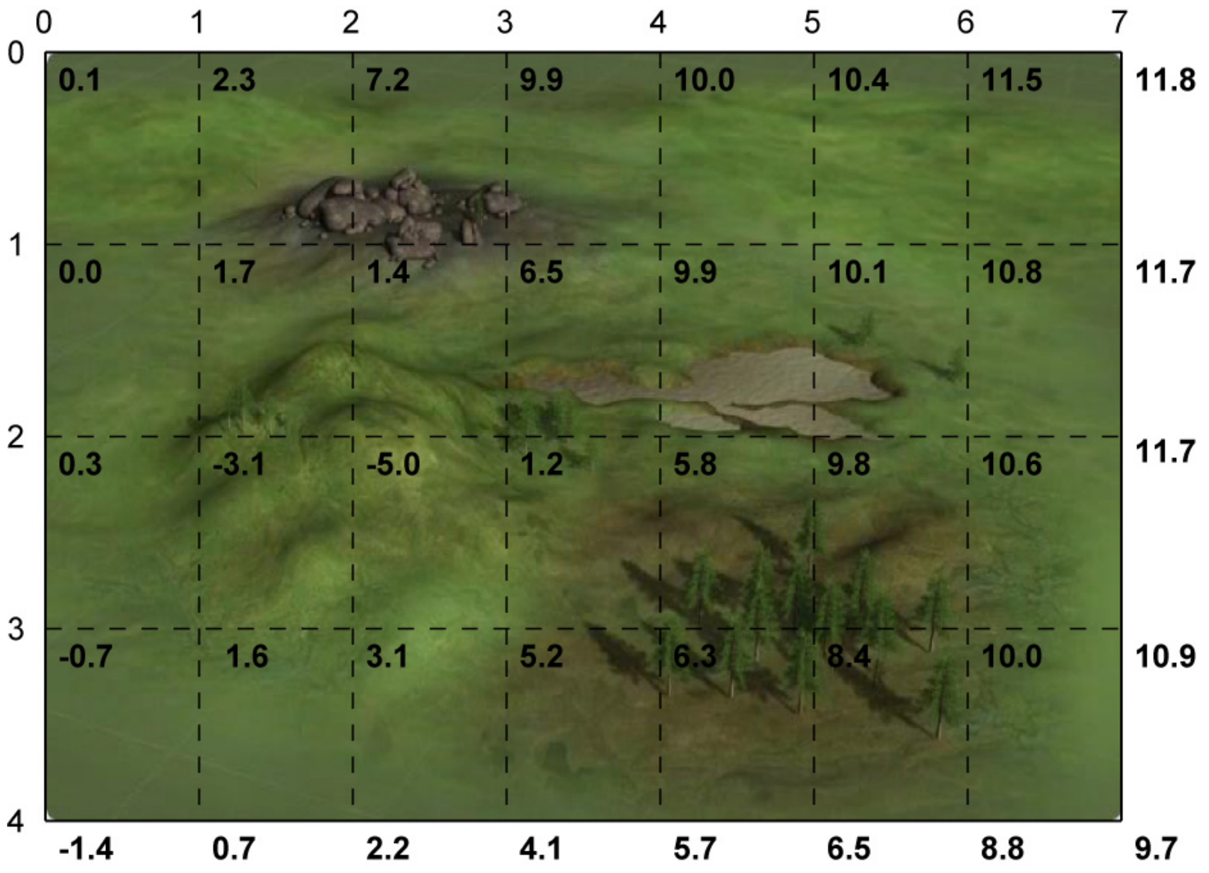
An array that holds the weight of 10 bags of rice. If written as a variable, it would look like this:

Bag[0] = 501
Bag[1] = 498
Bag[2] = 497
Bag[3] = 500
.
.
Bag[8] = 497
Bag[9] = 501

This array could also be written in column form:

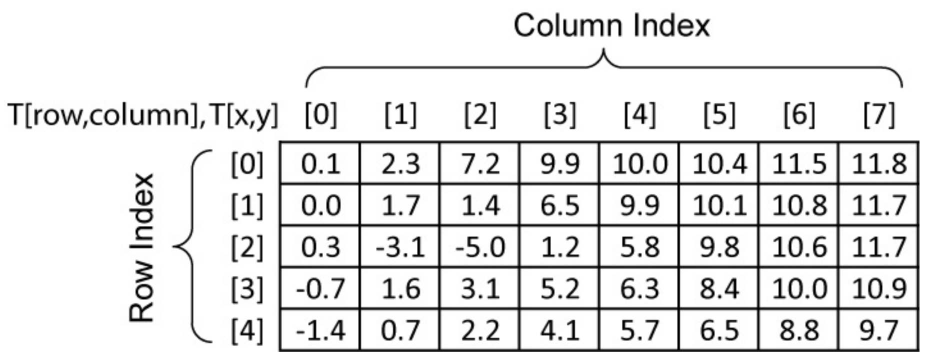


- Two Dimensional Array: Below is a map of a fictitious county called Wexford. It shows the temperatures in Celsius at each point on a 1 mile grid.



Temperatures in Wexford County (Degrees Celsius)

The temperatures can be organized into a 2 dimensional array, Y[x,y]: The first element refers to the row number; and the second element, to the column number.



T[row,column],T[x,y]	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[0]	0.1	2.3	7.2	9.9	10.0	10.4	11.5	11.8
[1]	0.0	1.7	1.4	6.5	9.9	10.1	10.8	11.7
[2]	0.3	-3.1	-5.0	1.2	5.8	9.8	10.6	11.7
[3]	-0.7	1.6	3.1	5.2	6.3	8.4	10.0	10.9
[4]	-1.4	0.7	2.2	4.1	5.7	6.5	8.8	9.7

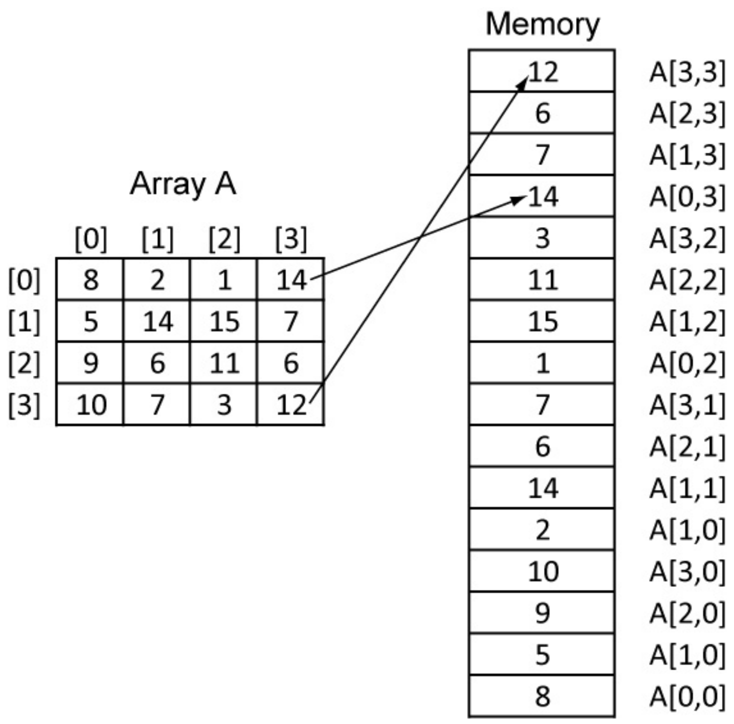
Column Index
Row Index

Two Dimension Array of Wexford County Temperature Data, T[x,y]

Sample values in this array are:

T[0,0] = 0.1^0 Celsius
T[1,2] = 1.7^0 Celsius
T[3,7] = 10.9^0 Celsius

- Memory Storage: Each element in an array is assigned its own data memory location. The mapping is demonstrated in the following figure:



Array Elements Ordered in Memory

- Arrays in PicBasic Pro:
  - Variable arrays are declared in a similar manner to variables.

    Label **VAR** Size[Number of elements]

    Label is any identifier, excluding keywords, as described above. Size is **BIT**, **BYTE** or **WORD**. Number of elements is how many array locations are desired. Some examples of creating arrays are:

```
  X    VAR   BYTE[5]     ‘ BYTE for each of 6 elements of array x[ ]
Bag    VAR   WORD[10]    ‘ WORD for each of 10 elements of array Bag[ ]
Temp   VAR   WORD[28]    ‘ WORD for each of 28 elements of array Temp[ ]
```

  - Regarding keywords, see section 7.4 and Appendix C in the PicBasic Pro Compiler Manual: http://www.microengineeringlabs.com/resources/index.htm#Manuals
- Perform Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings Lab 1 – Simple Arrays in PicBasic Pro.

- **Review PicBasic Pro Commands:**
  - **PULSIN:**
    Format:

    ```
    PULSIN Pin,State,Var
    ```

    EXPLANATION:
    Measures pulse width on Pin. If State is zero, the width of a low pulse is measured. If State is one, the width of a high pulse is measured. The measured width is placed in Var. If the pulse edge never happens or the width of the pulse is too great to measure, Var is set to zero. Pin is automatically made an input. Pin may be a constant, 0 - 15, or a variable that contains a number 0 - 15 (e.g. B0) or a pin name (e.g. PORTA.0).
    The resolution of PULSIN is dependent upon the oscillator frequency. If
    a 4MHz oscillator is used, the pulse width is returned in 10us increments.
    If a 20MHz oscillator is used, the pulse width will have a 2us resolution.
    Defining an OSC value has no effect on PULSIN. The resolution always
    changes with the actual oscillator speed.
    PULSIN normally waits a maximum of 65535 counts before it determines
    there is no pulse. If it is desired to wait fewer or more counts before it stops looking for a pulse or the end of a pulse, a DEFINE can be added:
    DEFINE PULSIN_MAX 1000
    This DEFINE also affects RCTIME in the same manner.
    ‘ Measure high pulse on Pin4 stored in W3
    PULSIN PORTB.4,1,W3

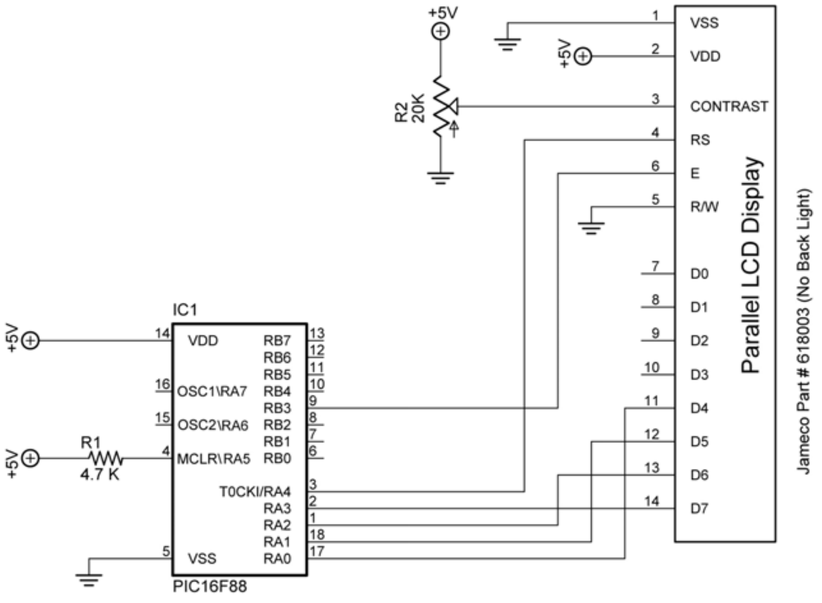
- Perform Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings Lab 2 – Programming the Sonar Car with sonar_car_b.pbp:

Electronics and Robotics I Week 27
Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings
LAB 1 – Simple Arrays in PicBasic Pro

- **Purpose:** The purpose of this lab is to acquaint the student with the use of arrays in a PicBasic Pro program.

- **Apparatus and Materials:**
    - 1 – Analog/Digital Trainer or Breadboard w/ +5V Power Supply
    - PIC16F88 Microcontroller
    - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
    - 20 K Potentiometer
    - 4.7 K Resistor

- **Procedure:**
    - Wire the circuit "array1 and array2" shown below:



array1 and array2

    - Import and run **array1.pbp**. See: http://cornerstonerobotics.org/code/array1.pbp
    - Discuss the operation of the program.
    - Import and run **array2.pbp**. See: http://cornerstonerobotics.org/code/array2.pbp
    - Discuss operation of the program.
- **Challenge:**
    - Using **array2.pbp** and **sonar1.pbp**, write a program that takes 4 ultra-sonic readings one second apart, then display each reading on an LCD for one second. For **sonar1.pbp**, see: http://cornerstonerobotics.org/code/sonar1.pbp

Electronics and Robotics I Week 27
Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings
LAB 2 – Programming the Sonar Car with sonar_car_b.pbp

- **Purpose:** The purpose of this lab is to review the second in a series of four programs that takes the class through the development of the final program sonar_car1.pbp. This lab reviews the program that adds taking ultra-sonic readings at each servo position then records data in dx_in & position arrays.

- **Apparatus and Materials:**
    - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics at: http://cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf and http://cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf

- **Procedure:**
    - Open the program as **sonar_car_b.pbp.** See: http://cornerstonerobotics.org/code/sonar_car_b.pbp
    - Discuss operation of the program.