Lab 15
Review of Arrays, Array of Objects and Vector
Dr. John Abraham, Professor

I have noticed over the years that students have great deal of difficulty dealing with composite and abstract data types. Therefore we are going spend an extra lab review material we have already learned. Vector data type, a variation of array, will be introduced as well. Vector data type (Vector Class) can be used when you need an array that grows dynamically. However, C++ does not allow us to declare size of an **array** dynamically like some other languages. It is important for you to declare the maximum size you will need. Suppose you are writing a program to keep grades. You have to predict the maximum class size you will have. At our university, UTPA, in computer science department we will not have more than 150 students. Most of our classrooms can only hold less than 100 students. So it would appropriate to declare an array size of 150, if you are writing a program to keep grades.

When you declare an array, declare it for the maximum size you will ever need. But, keep track of how many of array elements that were used. The remaining elements are assumed to have random values (garbage). Students often forget to keep the count of actual elements used in an array. So, when you pass an array, make it a practice to pass the array along with the number of elements used. A sentinel value is used to mark the end of valid elements. For example, if you are keeping grades: 88 78 77 99 100 68 87 67 89 -1; the -1 is the sentinel value. Even though 10 values are entered including -1, only 9 valid numbers are entered.

You can populate an array various ways. The simplest is to assign a value to each of the locations such as:

```
grades[0]=88;
grades[1]=78;
grades[2]=77;
```

Alternatively, you may read these grades in:

```
cin >> grades[0];
cin >> grades[1];
cin >> grades[2];
```

It would be easier to read the values in a loop:

```
For (i=1; i<=8; i++) cin >> grades[i];
```

For loop can only used if you know how many values we are dealing with. If you want to read an unknown number of values and stop reading when a sentinel value is entered, the while loop or do while loop should be used.

In the following program 15-1, we have grades[] and n to keep track of the array and the number of actual grades. Pay particular attention to passing of an array. Declaring a function with array parameters is similar to function with simple data types except to include the [], to indicate the array. Let us look at the function getGrades. It returns the number of actual grades entered (does not count the sentinel value entered). The function receives a parameter, int grades[]. When an array is passed, a pointer to the memory where the first element of the array can be found, is passed. Do not pass arrays as reference parameter, like int& grades[]. When this function is called in the main, n=getGrades(grades), the number of grades is received into n; note that the argument is grades, not grades[]. Once an array is declared, use its name and the compiler is aware that it is an array. When we display the grades both the array and the number of grades are passed.

Program 15-1

```
/************************************************
Read into an Array
By Dr. John Abraham
Created for 1370 students
************************************************/

#include <iostream>
using namespace std;

int getGrades (int grades[]);
void displayGrades(int grades[], int);
//--------------main-------------------------------------
int main ()
{
      int grades[150];
      int  n; //  number of scores read excluding negative number.
      n = getGrades(grades); //go get the scores and how many
      cout <<"\nHere are the scores entered:\n";
      displayGrades(grades,n);
      return(0);

}

//--------------function getscores----------------------

int getGrades(int grades[])
{
    int n=1;
      cout << "ENTER A SCORE AND PRESS ENTER. YOU QUIT ANY TIME BY
ENTERING A NEG NUMBER!\n";
```

```
	cout << "Enter score# " << n << " ";
	cin >> grades[n];
	while (grades[n] >= 0)
	{
		n++;
		cout << "Enter score# " << n <<" ";
		cin >> grades[n];
	}
return n-1; //n-1 actual scores read (ignore negative score).
}

//-------------function display scores------------------
void displayGrades(int grades[], int n)

{
	int i;
	for (i=1; i<=n; i++)
		cout << grades[i] <<endl;
}
```

Now that I have discussed reading into an array, we will turn to processing elements in an array. For this I refer you to the standard deviation program where we work with three arrays, one to store the data, the second one to find the difference of each score from the mean and the third one to find store the square of deviation of each score. The mean is calculated by adding all valid scores stored in the array and dividing by the number of scores. The deviation of each score is obtained by subtracting the score from the mean. Since this program drills you in array processing, I would ask all students to know this program thoroughly. Standard deviation is calculated using the following formula:

$$\sigma = \sqrt{\frac{\Sigma\ (x - \bar{x})^2}{n - 1}}$$

σ = **lower case sigma**
Σ = **capital sigma**
$\bar{x}$ = **x bar**

Lower case sigma means 'standard deviation'.
Capital sigma means 'the sum of'.
x bar means 'the mean'

Program 15-2

```
/*********************************************
Read scores from a file into an array.
Keep track of number of scores entered.
```

```
Find difference of each score from the Mean.
Square the differences and add the squares.
find stadard deviation
create a table
Teaching objective  - Pass Array as a parameter.
By Dr. John Abraham
Created for 1370 students
*************************************************/

#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath> //for square root function.

using namespace std;

int getscores (int scores[]);
float calcMean (int scores[], int);
float sumSquares (int, int scores[], float diff[], float
diffSq[],float);
float stdDeviation(int, float);
void Table(int, int scores[], float diff[], float diffSq[], float,
float);
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
int main ()
{
      int scores[90];float diff[90], diffSq[90];
      int n; // n for number of scores.
      float mean, sumSq, stDev;
      n = getscores(scores); //go get the scores and how many
      mean = calcMean(scores, n);
      sumSq = sumSquares (n,scores,diff,diffSq,mean);
      stDev = stdDeviation(n, sumSq);
      Table(n, scores, diff,diffSq, mean, stDev);
      cin >>n;
      return(0);

}

//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

int getscores(int scores[])
{
      ifstream infile;
      infile.open("scores1.dta");
      if (!infile) cout << "Data file does not exist!";
    int n=0;
      while(!infile.eof())

      {
            n++;
            infile >> scores[n];

      }
return n; //n-1 actual scores read.
}
```

```
//ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
float calcMean(int scores[], int n)
      {
        int sum=0;
        int i;    //use for counter
        float m;  //local variable for mean
        for (i=1; i <=n; i++) sum += scores[i]; //add all scores
        m = float(float(sum)/n);
        return m;
      }


//fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff

float sumSquares (int n, int scores[], float diff[],
                          float diffSq[],float mean)

      {
            int i; float ss=0.0;
            for (i=1; i <=n; i++)
            {
                  diff[i] = scores[i]-mean;
                  diffSq[i] = diff[i] * diff[i];
                  ss += diffSq[i];
            }
      return ss;
      }

//fffffffffffffffffffffffffffffffffffffffffffffffffffff
float stdDeviation(int n, float sumSq)

{
      float variance;
      variance = sumSq/(n-1);
      return (float(sqrt(variance)));
}

//fffffffffffffffffffffffffffffffffffffffffffffffffffff

void Table(int n, int scores[], float diff[],
               float diffSq[], float mean, float stDev)
{
      int i; float ss=0;
cout << "\nStadard Deviation for " << n<<" Scores.  Mean: "<<mean
<<endl;
cout <<"-----------------------------------------------------------\n";
cout <<setw(10) <<"Score"<<setw(10) <<"Dev" <<setw(10) << "Dev Sq"
        <<"  Sum of dev2)\n";
cout <<"-----------------------------------------------------------\n";
cout << setiosflags(ios::fixed);

for (i = 1; i<=n; i++)
{
            ss += diffSq[i];
            cout << setw(10) << setprecision(2) << scores[i]
                  << setw(10) << setprecision(2) << diff[i]
                  << setw(10) << setprecision(2) << diffSq[i]
```

```
                    << setw(10) << setprecision(2) << ss << endl;

        }
cout <<"-------------------------------------------------------------\n";
cout <<"Standard Deviation "<< setprecision(2) <<stDev <<endl;
}
```

The last program I would like to discuss with you dealing with array is the card shuffling program. Here we have an array with names of a deck of card. We do not actually shuffle this array, instead we create an array of integers (shuffled[]) and place random numbers from 1 to 52 in it. Once we have the random numbers, we just display the card names in that order.

```
string cards[52]={
          "CA","C2","C3","C4","C5","C6","C7","C8","C9","C10","CJ","CQ","CK",
     "DA","D2","D3","D4","D5","D6","D7","D8","D9","D10","DJ","DQ","DK",
     "HA","H2","H3","H4","H5","H6","H7","H8","H9","H10","HJ","HQ","HK",
     "SA","S2","S3","S4","S5","S6","S7","S8","S9","S10","SJ","SQ","SK"};


cout<<"\n\nHere is the suffled Deck: \n";
for (i=0;i <52;i++)
          {
          if (i%13==0) cout<<endl;
          cout <<cards[shuffled[i]] << " ";

         }
```

Vector Class

The vector is more flexible than arrays as it can grow in size dynamically. The array can be accessed using the subscript operator just like in array. However, to add an element into a vector use the method push_back as shown in the program below. You can remove the last element by using pop_back.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;



int getNames (vector<string> &nameVector );
void displayNames(int, vector<string> &nameVector);


int main ()
{
```

```
	vector<string> nameVector;
	int  n;
	n = getNames(nameVector);
	cout << "\n*****************************\n";
	displayNames(n, nameVector);
	return 0;
}

int getNames (vector<string> &nameVector)
{
	string name;
	int n=1;
	cout << "Enter a name or 'quit' to stop entry-> ";
	cin >> name;
	while (name != "quit")
	{
		nameVector.push_back(name);
		cout << "--------------------\n";
		n++;
		cout << "Enter a name or 'quit' to stop entry-> ";
		cin >> name;
	}

    return n-1;
}
void displayNames(int n, vector<string> &nameVector)
{

	int i;
	for (i=0; i<n; i++)
	{
		cout << nameVector[i] << endl;

	}
}
```

Assignment:
Complete the cards program without referring to the program you have done in the past.
Write the standard deviation program without the aid of the notes.
Create an object (use the previous notes), Make an array of these objects and read into these objects.

Caveat: It is important that you are able to do these programs without any help. I will be placing great deal of emphasis on these programs on the final exam.