CPT 111 – Principles of Programming
Week 11 Programming Lab
ONE DIMENSIONAL ARRAY

Learning Outcomes:
- Describe one dimensional array with multiple values and range.
- Apply one dimensional array elements and parallel arrays in data processing.
- Demonstrate arrays as function arguments.

---

1. Define the following arrays:

   a. `empNums`, a 100-element array of `ints`
   b. `payRates`, a 25-element array of `floats`
   c. `miles`, a 14-element array of `longs`
   d. `cityName`, a 26-element array of `string` objects
   e. `lightYears`, a 1,000-element array of `doubles`

2. Identify the error(s) in following array definitions.

```
int readings[-1];
float measurements[4.5];
int size;
string names[size];
```

3. State the output of the following code.

```
int values[5], count;
for (count = 0; count < 5; count++)
  values[count] = count + 1;
for (count = 0; count < 5; count++)
  cout << values[count] << endl;
```

4. The following program skeleton contains a 20-element array of `ints` called `fish`. When completed, the program should ask how many fish were caught by fishermen 1 through 20 and store this data in the array. Complete the program.

```
#include <iostream>
using namespace std;
int main()
{
  const int NUM_FISH = 20;
  int fish[NUM_FISH];

  // You must finish this program. It should ask how
  // many fish were caught by fishermen 1-20, and
  // store this data in the array fish.

  return 0;
}
```

5. Define the following arrays:

   a. `ages`, a 10-element array of `int`s initialized with the values 5, 7, 9, 14, 15, 17, 18, 19, 21, and 23.
   b. `temps`, a 7-element array of `float`s initialized with the values 14.7, 16.3, 18.43, 21.09, 17.9, 18.76, and 26.7.
   c. `alpha`, an 8-element array of `char`s initialized with the values 'J', 'B', 'L', 'A', '*', '$', 'H', and 'M'.

6. Determine each of the following a valid or invalid array definition. (If a definition is invalid, explain why.)

   a. `int numbers[10] = {0, 0, 1, 0, 0, 1, 0, 0, 1, 1};`
   b. `int matrix[5] = {1, 2, 3, 4, 5, 6, 7};`
      `double radii[10] = {3.2, 4.7};`
      `int table[7] = {2, , , 27, , 45, 39};`
   c. `char codes[] = {'A', 'X', '1', '2', 's'};`
   d. `int blanks[];`

7. State the output of the following program segment.

a.
```
int temp[5];
for (int i = 0; i < 5; i++)
    temp[i] = 2 * i - 3;
for (int i = 0; i < 5; i++)
     cout << temp[i] << " ";
cout << endl;

temp[0] = temp[4];
temp[4] = temp[1];
temp[2] = temp[3] + temp[0];
for (int i = 0; i < 5; i++)
  cout << temp[i] << " ";
cout << endl;
```

b.
```
double temp[5];
for (int i = 0; i < 5; i++)
  temp[i] = pow(i, 2.0) + 2;
for (int i = 0; i < 5; i++)
  cout << temp[i] << " ";
cout << endl;

temp[0] = pow(temp[1], 3);
temp[1] = temp[4] - temp[2];
temp[2] = temp[0] - 5;
for (int i = 0; i < 5; i++)
  cout << temp[i] << " ";
cout << endl;
```

8. State the content stored in `list` after the following C++ code executes.

```
int list[10];
for (int i = 0; i < 5; i++)
{
  list[i] = i * i - 5;
  if (i % 3 == 0)
    list[i] = list[i] + i;
  else
    list[i] = list[i] - i;
}
```

9. Given the following array definition:

```
int values[] = {2, 6, 10, 14};
```

State for each of the following display.

a. `cout << values[2];`
b. `cout << ++values[0];`
c. `cout << values[1]++;`
d.
```
x = 2;
cout << values[++x];
```

10. Given the following array definition:

```
int nums[5] = {1, 2, 3};
```

State the output for the following statement.

```
cout << nums[3];
```

11. State the output of the following C++ code. (You may need to use a calculator.)

```
double balance[5] = {100.0, 250.0, 325.0, 500.0, 1100.0};
const double INTRATE = 0.1;

cout << fixed << showpoint << setprecision(2);
for (int count = 0; count < 5; count++)
  cout << (balance[count] * INTRATE) << endl;
```

12. State the output of the following C++ code. (You may need to use a calculator.)

```
const int SIZE = 5;
int time[SIZE] = {1, 2, 3, 4, 5},
speed[SIZE] = {18, 4, 27, 52, 100}, dist[SIZE];

for (int count = 0; count < SIZE; count++)
  dist[count] = time[count] * speed[count];
for (int count = 0; count < SIZE; count++)
{
  cout << time[count] << " ";
  cout << speed[count] << " ";
  cout << dist[count] << endl;
}
```

13. Given the following array definitions

```
double array1[4] = {1.2, 3.2, 4.2, 5.2};
double array2[4];
```

will the following statement work? If not, why?

```
array2 = array1;
```

14. When an array name is passed to a function, what is actually being passed?
15. When used as function arguments, are arrays passed by value?
16. State the output of the following program. (You may need to consult the ASCII table.)

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

// Function prototypes
void fillArray(char [], int);
void showArray(const char [], int);

int main ()
{
   const int SIZE = 8;
   char prodCode[SIZE] = {'0', '0', '0', '0', '0', '0', '0', '0'};
   fillArray(prodCode, SIZE);
   showArray(prodCode, SIZE);
}

// Definition of function fillArray.
// (Hint: 65 is the ASCII code for 'A')
void fillArray(char arr[], int size)
{
   char code = 65;
   for (int k = 0; k < size; code++, k++)
     arr[k] = code;
}

// Definition of function showArray.
void showArray(const char codes[], int size)
{
   for (int k = 0; k < size; k++) cout << codes[k];
   cout << endl;
}
```

17. The following C++ program skeleton, when completed, will ask the user to enter 10 integers, which are stored in an array. The function `avgArray`, which you must write, is to calculate and return the average of the numbers entered.

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

// Write your function prototype here

int main()
{
  const int SIZE = 10;
  int userNums[SIZE];

  cout << "Enter 10 numbers: ";
  for (int count = 0; count < SIZE; count++)
  {
    cout << "#" << (count + 1) << " ";
    cin >> userNums[count];
  }
  cout << "The average of those numbers is ";
  cout << avgArray(userNums, SIZE) << endl; //function call
return 0;
}
```

// Write the function avgArray here.

18. Define a C++ program that lets the user enter 10 values into an array. The program should then display the **largest** and **smallest** values stored in the array.

19. Define a C++ program to find **Largest** and **Second Largest Even Numbers** in an array.

20. Define a C++ program that lets the user enter the total rainfall for each of 12 months into an array of `doubles`. The program should calculate and display the total rainfall for the year, the average monthly rainfall, and the months with the highest and lowest amounts.

    Input Validation: Do not accept negative numbers for monthly rainfall figures.

21. Define C++ program that lets a maker of chips and salsa keep track of sales for five different types of salsa: mild, medium, sweet, hot, and zesty. The program should use two **parallel 5-element arrays**: an array of strings that holds the five salsa names and an array of integers that holds the number of jars sold during the past month for each salsa type. The salsa names should be stored using an initialization list at the time the name array is created. The program should prompt the user to enter the number of jars sold for each type. Once this sales data has been entered, the program should produce a report that displays sales for each salsa type, total sales, and the names of the highest selling and lowest selling products.

    Input Validation: Do not accept negative values for number of jars sold.

22. In a C++ program, define a function definition that accepts three arguments: an array, the size of the array, and a number *n*. Assume that the array contains integers. The function should display all of the numbers in the array that are greater than the number *n*.

23. In a C++ program, given an array `arr` of integers of size `N`, the task is to find the count of positive numbers and negative numbers in the array. Define two function definitions `countPositiveNum` and `countNegativeNumb`, each accept two arguments: an array and the size of the array and each return the number of positive and negative values respectively.
    1. Traverse the elements in the array one by one.
    2. For each element, check if the element is less than 0. If it is, then increment the count of negative elements.
    3. For each element, check if the element is greater than 0. If it is, then increment the count of positive elements.
    4. Print the count of negative and positive elements.

~ooOOoo~