One dimensional array

array : a collection of variables having the same data type and referred to by the same name
Why array?

1. Array declaration: **data-type array-name[array-size];**

positive integer constant

```
float   scores[5];                    int     intValues[1000];
char    name[15];                     bool    vacation[365];
```

- specify the size of the array by using constant int declaration

```
const int ARRAY_SIZE = 10;
int intArray[ARRAY_SIZE];
```

2. Array subscript: access individual elements of an array

- array subscript starts with 0, ends with (specified array size – 1)

```
Assign the values to the10 elements of the array IntArray
intArray[0]=3;
intArray[1]=10;
…
intArray[8]=25;
intArray[9]=-14;
```

Example: Operation with array elements

```
int  sum= intArray[3]+intArray[5];
cin >> intArray[0];
cout << intArray[0];
if (sqrt(intArray[4]) >= 3)
```

- subscript of array can be constant, expression, or variable. It has to evaluate to an integer

```
int x=1;
intArray[2*3]= 53;
intArray [x] = 24;
intArray [x+4] = 94;
```

- Array reference error occurs when reference to or access array element with array subscription out of the specified array boundary : 0 - (array size – 1)
  C++ does not perform array boundary check. Program contains array reference error may not get compilation error, but can have run time error (memory violation)

```
intArray[10] = intArray[30]+intArray[3];
```

3. Assign values to array elements during declaration

Example

```
const int ARRAY_SIZE = 10;
int    intArray[ARRAY_SIZE]={2, 3, 4, 5, 10, -9, -2, 0, 1, 3};
```

- Number of items in initialization list > array size specified → compilation error or warning
- Number of items in initialization list < array size specified → values of the rest of the elements are not determined, no compilation error

Example

```
const int ARRAY_SIZE = 10;
int    intArray[ARRAY_SIZE]={0};  ← this initializes all elements of the array to 0
```

Example

```
int intArray[] = {3, 5, 7, 9};
```

4. Array iteration

(a)

```
const int ARRAY_SIZE = 15;
int      intValues[ARRAY_SIZE];
int i;

for (i=0; i<ARRAY_SIZE; i++)
   intValues[i] = i*i + 1;
```

```
for (i=0; i<ARRAY_SIZE; i++)
{
        cout << "Please enter an integer:  ";
         cin >> intValues[i];
}
```

(b)

```
for (i=0; i<ARRAY_SIZE; i++)
{
        cout << "value " << i+1 << ":" << intValues[i] << endl;
}
```

```
(C) sum=0;
for (i=0; i<ARRAY_SIZE; i++)
{
        sum = sum + intValues[i];
}
mean = (float)sum/ARRAY_SIZE;
sum=0;
for (i=0; i<ARRAY_SIZE; i++)
{
        sqDifference = pow((intValues[i] – mean), 2.0);
        sum = sum + sqDifference;
}
stdDeviation = sqrt( sum/(ARRAY_SIZE – 1));
cout << "mean = " << mean ", standard deviation = " << stdDeviation << endl;
```

5. Passing array to function as parameter

- array is always passed to function by reference
- if the content of the array is not to be modified in the function, pass the array as a constant parameter

Examples:

1. Write a function “CountFreezingDays” that counts the number of days below freezing in a year, assuming an array with 365 values is passed into this function. (how to protect the data in array “temperature” such that no data will be accidentally changed in the function)
2. Write a function that finds the coldest day temperature of the year
3. Write a function that computes the average temperature of the year.
4. Write a function “PickFortune” that randomly selects a fortune reading from a number of pre-stored readings. Assume an array of 100 readings (each of string type) is passed into the function as a parameter.
5. Write a function “CompareGrades” that compares the grades of two students to see if they make the same grades for each of the ten tests.
6. Write a C++ program that simulates the game “Deal or No Deal”.