C++ Arrays – Chapter 8

Simple data type – each variable can only store one value at a time (int, char, float, double, bool, etc)

Structured data type – each data item is a collection of other data items

Array – a collection of a **fixed number of components** all with the **same data type**

- allows multiple values stored in one variable to be separately accessed using a loop
- one-dimensional array – collection arranged in a list-like form

Declaration of an array -

syntax:

```
datatype arrayname[intexp];
```

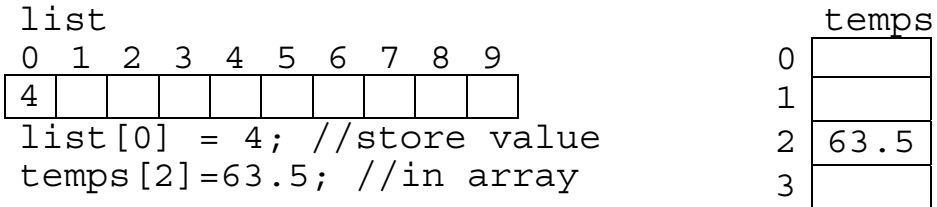
- datatype defines the type for the elements in the array
- arrayname must be a valid identifier name
- intexp is any constant expression that evaluates to a positive integer
- if intexp < 0, g++ will not allow compilation

semantics

- allocate `intexp` memory locations of the specified `datatype`
- the group of memory locations are assigned the `arrayname`

The contents of the elements of an array are **NOT automatically initialized** when the variable is declared.

```
const int MAX = 5;
int main()
{
  int list[10]; //creates array of 10 ints called list
  double temps[MAX]; //creates array of 5 doubles called temp
  . . .
```



- The value of the subscript must be between 0 and size-1.
- C++ does not check array bounds.
- There is no library function to return the size of an array.

Processing One-Dimensional Arrays –

- most processing of arrays is done one element at a time using a loop (usually for)
- typical tasks include – initialization, input, output, sum, largest, smallest, sort

```
//Program to sum and display in reverse order numbers using
//an array.
#include <iostream>
using namespace std;
const int MAX=4;
int main()
{
   int num[MAX], sum=0;
   cout << "please enter "<<MAX<<" integers: ";
   for (int i=0; i<MAX; i++)
   {
      cin >> num[i];
      sum = sum + num[i];
   }
   cout << "sum = " << sum << endl;
   cout << "values in reverse order are: ";
   for (int i=MAX-1; i>=0; i--)
      cout << num[i] << "  ";
   cout << endl;
   return 0;
}
```

Passing arrays as parameters

- arrays are passed by reference only
- & is <u>not</u> used in the formal parameter list, instead [] follows the parameter name
- size of array need not be specified inside the [ ]
  example: `void sample(double list[], int size)`
- when an array is passed its base address (address of first element) is sent to the function
- constant arrays can be created to avoid accidentally changing an array
  example: `void example (const int list[], int size)`

```
#include <iostream> //array.cpp and arraydata in cs135sampledir
using namespace std;
const int SIZE=4;
void read_array(double[],int); //prototypes – [] needed after data type
void sum_array(const double[],int);  //to indicate parameter is an array
void print_reverse(double[],int);
int main()
{
   double list[SIZE];
   read_array(list,SIZE); //calls – only name of array is needed if
   sum_array(list,SIZE);  //entire array is being passed
   print_reverse(list,SIZE);
   return 0;
}
void read_array(double data[], int n)//heading – [] needed to indicate
{                                    //that parameter is an array
   for (int i=0; i<n; i++)
      cin >> data[i];
}
void sum_array(const double list[], int n) //const prevents changes to
{                                          //array – not mandatory
   double sum=0.0;
   for (int i=0; i<n; i++)
      sum = sum + list[i];
   cout << "sum = " << sum << endl;
}
void print_reverse(double data[],int n)
{
   cout << "Array Elements in Reverse Order" << endl;
   for (int i=n-1; i>=0; i--)
      cout << data[i] << endl;
}
```

Passing individual elements of an array vs the entire array

```
//Assume a data file contains n+1 integers. The first value is n (number of
//values that follow. Read n and then store the next n values in an array.
//Compute and display the sum of the digits in each number. Find and display
//the smallest number.
//Assume maximum value of n will be 50.
//array2.cpp and array2data in cs135sampledir

#include <iostream>
#include <cmath>
using namespace std;
const int N=50;
int sum_digits(int);
int smallest(int[],int);
int main()
{
  int n;  int small;
  int numbers[N];
  cin >> n;
  for (int i=0; i<n; i++)
    cin >> numbers[i];
  small = smallest(numbers,n);      //pass whole array to find smallest #
  cout << "Smallest # is " << small << endl << endl;
  for (int i=0; i<n; i++)
    cout << "sum of the digits in " << numbers[i]     //pass one element
         << " is " << sum_digits(numbers[i]) << endl; //at a time
  return 0;
}
int smallest(int nums[],int n)
{
  int small = nums[0]; //assume 1st value is smallest
  for (int i=1; i<n; i++)
    if(small > nums[i])
      small = nums[i];
   return small;
}
int sum_digits(int num)
{
  int digit;
  int sum=0;
  num = int(fabs(num));
  while (num != 0)
    {
      digit = num%10;
      sum = sum+digit;
      num = num/10;
    }
  return sum;
}
```

Since summing of digits applies to an individual number, sending one value to the function at a time simplifies the code.

Using an array to store data when exact number of values unknown –

- have a collection of like type values
- know the maximum number of values that will have to be processed
- want to store values in an array to make processing easier
- use an eof-controlled loop to read and count data values, storing each value into an array

```
void get_data(datatype array[],int& count)
//Given a file of data values to be stored in an array, read and count
//the values. When loop terminates (at eof) the data values will be
//stored in the array (locations 0 through count-1).
{
   count = 0;
   cin >> array[count];
   while (cin)
   {
      count++;
      cin >> array[count];
   }
}
```

Design a program to read in a series of numbers from a file. Compute and display their average and display the list of values, 1 per line. Maximum number of values in file will be 50.

```
//array3.cpp and arra3data in cs135sampledir

#include <iostream>
#include <iomanip>
using namespace std;
const int MAX=50;
void get_data(double[],int&);
double average(double[],int);
void print_results(double,double[],int);
int main()
{
   double list[MAX]; //array to store values from file
   int n;            //# values read from the file
   double ave;       //average value
   get_data(list,n);
   ave = average(list,n);
   print_results(ave,list,n);
   return 0;
}
void get_data(double list[],int& n)
{
   n=0;
   cin >> list[n];
   while (cin)
   {
      n++;
      cin >> list[n];
   }
}
double average(double data[],int n)
{
   double sum=0.0;
   for (int j=0; j<n; j++)
      sum = sum + data[j];
   return sum/n;
}
void print_results(double ave, double list[],int n)
{
   cout << fixed << setprecision(2);
   cout << "Average = " << ave << endl << endl;
   cout << "Values in list:" << endl;
   for (int j=0; j<n; j++)
      cout << list[j] << endl;
}
```