

## Arrays

- **Problem:**

Assume that an instructor has given an exam in class and wants to find the average mark and the highest mark. Write a complete C++ program to do the following: Read in and print the data, find the average mark and the highest mark, find and print how many marks are above, how many marks are below, and how many marks are equal to the class average. Assume, there no more than 50 students in the class. The data format is as follows: a number 'num', called the parameter, saying how many marks there are for the class, followed by that many marks.

- **Sample Data:**

7 91 94 65 84 88 76 99

## ● First Draft of Program:

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

int main()
{
    int num,sum = 0;
    int mark;
    double avgmark;

    cout << "Enter the number of marks: ";
    cin >> num;

    if (num > 0)
        cout << "There are " << num << " marks" << endl;
    else
    {
        cout << "Invalid number of marks" << endl;
        exit(1);
    }

    // enter marks and find average mark
    for (int count = 1; count <= num; count + + )
    {
        cout << "Enter a mark: ";
        cin >> mark;
        sum += mark;
    }

    avgmark = sum/num;          // THIS IS WRONG!!
    cout << endl << "The average is " << avgmark << endl;
    return 0;
}
```

## Type Casting

- Casting is a method of explicitly requesting type conversion.

- Example:

```
avgmark = (double)sum / num;
```

- Example of Improper Casting:

```
avgmark = (double)(sum / num); //casting is too late
```

- **Complete Program (First Version):**

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

int main()
{
    int num,sum=0;
    int mark;
    double avgmark;

    cout << "Enter the number of marks: ";
    cin >> num;

    if (num > 0)
        cout << "There are " << num << " marks" << endl;
    else
    {
        cout << "Invalid number of marks" << endl;
        exit(1);
    }

    // enter marks and find average mark
    for (int count = 1; count <= num; count + +)
    {
        cout << "Enter a mark: ";
        cin >> mark;
        sum += mark;
    }

    avgmark = (double)sum/num;
    cout << endl << "The average is " << avgmark << endl;
    return 0;
}
```

● **Questions:**

Now that we calculated the average mark:

- What was the first mark?
- The second?
- The last?
- Which mark is closest to the average?
- How many marks are above, below, or equal to the average?

## Arrays

- An **array** is a collection of elements of the same type referenced by a single identifier. The individual elements are referenced through an index value into the array. The index is referred to as the **subscript** of the array.

- **Array Declaration:**

*data\_type identifier[num\_elements];*

```
int num[5];
```

	Array num
num[0]	
num[1]	
num[2]	
num[3]	
num[4]	

- **Using the Array:**

```
num[0] = 3;
```

```
num[1] = 15;
```

```
num[2] = -2;
```

```
num[3] = num[0] + num[1];
```

```
num[4] = num[0] * num[1] + num[2];
```

	Array num
num[0]	3
num[1]	15
num[2]	-2
num[3]	18
num[4]	43

- **Example: Subscript of the Array as a Variable:**

```
for (int i = 0; i < 5; i++)  
    num[i] = i * 3;
```

Array num

num[0]	0
num[1]	3
num[2]	6
num[3]	9
num[4]	12

- **Initializing Array Elements:**

```
int num[5] = {5,11,-1,17,-32};
```

Array num

num[0]	5
num[1]	11
num[2]	-1
num[3]	17
num[4]	-32

- **Partial Initialization of Array Elements:**

```
double sales[3] = {123.45,23456.78};
```

Array sales

sales[0]	123.45
sales[1]	23456.78
sales[2]	

- **Interpretation of Subscripts:**

```
/* Collect sales data for years 2004, 2005, 2006 */
```

```
double sales[3] = {0,0,0};
```

```
double amount;
```

```
int year,numtrans;
```

```
cout << "Enter the number of transactions: ";
```

```
cin >> numtrans;
```

```
for (int count = 0; count < numtrans; count + +)
```

```
{
```

```
    cout << "Enter the year: ";
```

```
    cin >> year;
```

```
    cout << "Enter the amount of sales for the year: ";
```

```
    cin >> amount;
```

```
    sales[year - 2004] + = amount;
```

```
}
```

```
for (year = 2004; year <= 2006; year + +);
```

```
    cout << "sales for " << year << " are " <<
```

```
        sales[year-2004] << endl;
```

- **Example of Array of char:**

```
char message[20] = {'w','e','i','r','d'};
int i = 0;
```

```
message[5] = ' ';           // this assigns a blank to message[5]
message[6] = 's';
message[7] = 't';
message[8] = 'u';
message[9] = 'f';
message[10] = 'f';
message[11] = '\0';       // this assigns a null character
```

```
while (message[i] != '\0') {
    cout << message[i];
    i++;
}                                     //output: weird stuff
cout << endl;
```

```
for (i = 0; message[i] != '\0'; i++)
    cout << message[i];           //output: weird stuff
cout << endl;
```

```
cout << message << endl;         //output: weird stuff
```

- **Note:** The last example will print the characters in the array message until it hits the null character. The string within message is called a **null terminated string**.

## ● Second Version of the Program:

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

const int SIZE = 50;

int main()
{
    int num,sum = 0;
    int mark[SIZE];
    double avgmark;

    cout << "Enter the number of marks: ";
    cin >> num;

    if (num > 0 && num <= SIZE)
        cout << "There are " << num << " marks" << endl;
    else
    {
        cout << "Invalid number of marks" << endl;
        exit(1);
    }

    // enter marks and find average mark
    for (int count = 0; count < num; count + + ) {
        cout << "Enter a mark: ";
        cin >> mark[count];
        cout << mark[count] << endl;
        sum + = mark[count];
    }

    avgmark = (double)sum/num;
    cout << endl << "The average is " << avgmark << endl;
    return 0;
}
```

- **Modular Version of the Program:**

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

const int SIZE = 50;

int main()
{
    int num,sum = 0;
    int mark[SIZE];
    double avgmark;

    cout << "Enter the number of marks: ";
    cin >> num;

    if (num > 0 && num <= SIZE)
        cout << "There are " << num << " marks" << endl;
    else
    {
        cout << "Invalid number of marks" << endl;
        exit(1);
    }

    // enter marks and print them
    for (int count = 0; count < num; count + +) {
        cout << "Enter a mark: ";
        cin >> mark[count];
        cout << "mark[" << count << "] = " << mark[count]
            << endl;
    }

    // find the average mark
    for (int count = 0; count < num; count + +)
        sum + = mark[count];
    avgmark = (double)sum/num;
    cout << endl << "The average is " << avgmark << endl;
    return 0;
}
```

## Arrays as Parameters of Functions

- **Problem:**

Write a function called **sumarray** that finds the sum of the elements of an array of 'n' integers.

- **Function Prototype:**

```
int sumarray(int [], int);
```

- **Function Header:**

```
int sumarray(int numbers[], int n)
```

- **The Function sumarray():**

```
/* Function sumarray()
 * Input:
 *   numbers - an array of integers
 *   n - the number of elements in the array
 * Process:
 *   finds the sum of the first n elements in the numbers
 *   array.
 * Output:
 *   returns the sum to the calling function.
 */
int sumarray(int numbers[], int n)
{
    int sum = 0;

    for (int count = 0; count < n; count + +)
        sum + = numbers[count];
    return(sum);
}
```

- **Function Usage:**

```
int sum;

sum = sumarray(mark,num);
```

- **The Function avgarray():**

```
/* Function avgarray()
 * Input:
 *   numbers - an array of integers
 *   n - the number of elements in the array
 * Process:
 *   calls sumarray to find the sum of the first n elements
 *   and then divides by n to find the average.
 * Output:
 *   returns the average to the calling function.
 */
double avgarray(int numbers[], int n)
{
    return ((double)sumarray(numbers,n)/n);
}
```

- **Function Prototype:**

```
double avgarray(int [], int);
```

- **Function Usage:**

```
double avgmark;
```

```
avgmark = avgarray(mark,num);
```

## Changing an Array Within a Function

Sending an array as a parameter to a function, sends the **actual location** of the array in memory. Consequently, changes made to the array within the function are retained upon return.

### ● The Function `readdata()`:

```
/* Function readdata()
 * Input:
 *   numbers - an array to be filled
 *   n - a reference to the number of elements in the array
 *   the parameters are uninitialized upon entry
 * Process:
 *   reads n and reads n values into the array
 * Output:
 *   the filled numbers array
 *   n - the number of elements in the array
 */
void readdata(int numbers[], int &n)
{
    cout << "Enter the number of marks: ";
    cin >> n;

    if (n > 0 && n <= SIZE)
        cout << "There are " << n << " marks" << endl;
    else
    {
        cout << "Invalid number of marks" << endl;
        exit(1);
    }

    // enter the marks
    for (int count = 0; count < n; count++) {
        cout << "Enter a mark: ";
        cin >> numbers[count];
    }
    return;
}
```

- **Revised Version of the Main Program:**

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

const int SIZE = 50;

/* Function Prototypes */
void readdata(int [], int &);
int sumarray(int [], int);
double avgarray(int [], int);

int main()
{
    int num;
    int mark[SIZE];
    double avgmark;

    // call function to read the marks
    readdata(mark,num);

    // print the mark array
    for (int count = 0; count < num; count + +)
        cout << "mark[" << count << "] = " << mark[count]
            << endl;

    // find and print the average mark
    avgmark = avgarray(mark,num);
    cout << endl << "The average is " << avgmark << endl;
    return 0;
}
```

## Finding the Largest Element of an Array

- Algorithm:
  - Initially, set the first element of the array as the largest.
  - Compare the current largest element to the next element within the array. If the next is larger, designate that element as the largest so far.
  - Repeat this process until all elements of the array have been compared to the largest so far.
  - Whichever element is the largest after the last comparison, is the largest element of the entire array.
- The Function `findmax()`:

```
/* Function findmax()
 * Input:
 *   numbers - an array of integers
 *   n - the number of elements in the array
 * Process:
 *   finds the largest value in the array
 * Output:
 *   returns the maximum value within the array
 */
int findmax(int numbers[], int n)
{
    int largest_so_far;

    largest_so_far = numbers[0];
    for (int count = 1; count < n; count++)
        if (largest_so_far < numbers[count])
            largest_so_far = numbers[count];
    return(largest_so_far);
}
```

## Finding Elements Above, Below, or Equal to a Threshold

### ● The Function `countmarks()`:

```
/* Function countmarks()
 * Input:
 *   numbers - an array of integers
 *   n - the number of elements in the array
 *   avgnum - the average value of the array elements
 *   num_below - a reference to the number of elements of
 *   the array that are below the average
 *   num_above - a reference to the number of elements of
 *   the array that are above the average
 *   num_equal - a reference to the number of elements of
 *   the array that equal to the average
 * Process:
 *   counts the number of elements that are above, below,
 *   and equal to avgnum.
 * Output:
 *   num_below - the number of elements below average
 *   num_above - the number of elements above average
 *   num_equal - the number of elements equal to the average
 */
void countmarks(int numbers[], int n, double avgnum,
int &num_below, int &num_above, int &num_equal)
{
    num_below = num_above = num_equal = 0;    //initialize

    for (int count = 0; count < n; count++)
        if ((double)numbers[count] < avgnum)
            num_below++;
        else if ((double)numbers[count] > avgnum)
            num_above++;
        else
            num_equal++;
    return;
}
```

● **Another Revision of the Main Program:**

```
/* Program to process exam grades */
#include <iostream>
using namespace std;

const int SIZE = 50;

/* Function Prototypes */
void readdata(int [], int &);
int sumarray(int [], int);
double avgarray(int [], int);
int findmax(int [], int);
void countmarks(int [], int, double, int &, int &, int &);

int main()
{
    int num;
    int mark[SIZE];
    double avgmark;
    int numabove,numbelow,numequal,hi_mark;

    // call function to read the marks and then print the array
    readdata(mark,num);

    // print the mark array
    for (count = 0; count < num; count + +)
        cout << "mark[" << count << "] = " << mark[count] << endl;

    // find and print the average mark
    avgmark = avgarray(mark,num);
    cout << endl << "The average is " << avgmark << endl;

    // find and print the highest mark
    hi_mark = findmax(mark,num);
    cout << "The highest mark is " << hi_mark << endl;

    // classify marks w.r.t. average mark
    countmarks(mark,num,avgmark,numbelow,numabove,numequal);
    cout << "Number of marks below the average is " <<
        numbelow << endl;
    cout << "Number of marks above the average is " <<
        numabove << endl;
    cout << "Number of marks equal to the average is " <<
        numequal << endl;
    return 0;
}
```

## Using cin.eof() to Check for the End of a Data Set

- **Note:**

When cin successfully reads in a data value, the method cin.eof() returns 0 (false). When cin reaches the end of an input stream (e.g., end of a file), then cin.eof() returns 1 (true).

### The Function readdata():

```
/* ... */
void readdata(int numbers[], int &n)
{
    // read and count the number of marks
    n = 0;
    cout << "Enter the marks - end with EOF: " << endl;

    cin >> numbers[n];
    while (!cin.eof())
    {
        n++;
        cin >> numbers[n];
    }
    return;
}
```

- **Note:**

In Windows or DOS, EOF is signaled by CTRL-Z, in Unix, EOF is signaled by entering CTRL-D.

## Alternate way to Check for the End of a Data Set

- **Note:**

When cin successfully reads in a data value, it returns 1 (true).  
When cin reaches the end of an input stream (e.g., end of a file), then it returns 0 (false).

### The Function readdata():

```
/* ... */  
void readdata(int numbers[], int &n)  
{  
    // read and count the number of marks  
    n = 0;  
    cout << "Enter the marks - end with EOF: " << endl;  
  
    while (cin >> numbers[n])  
        n+ +;  
  
    return;  
}
```

## Checking for EOF using an Input File

### The Function readdata():

```
/* ... */
void readdata(int numbers[], int &n)
{
    // declare and open input file
    ifstream infile("c:\\mypgms\\myinput.txt");
//    ifstream infile("con");           //un-comment for debugging

    // read and count the number of marks
    n = 0;

    while (infile >> numbers[n])
        n++;

    infile.close();           //close input file
    return;
}
```

- **Note:**

When infile successfully reads in a data value, it returns 1 (true). When infile reaches the end of the file, it returns 0 (false).

## Two-Dimensional Arrays

- **Problem:**

Assume that each student in a class has four grades, rather than one, representing marks on four exams. The instructor wishes to find various statistics:

- The average mark on each exam.
- The highest and lowest mark on each exam.
- Each student's average over all four exams.

- **Solution:**

To solve this problem efficiently, we need a two-dimensional array.

- **Two-Dimensional Array Declaration Syntax:**

*data\_type identifier[num\_rows][num\_columns];*

- **Example of a 3 x 6 array:**

```
int number[3][6];           //This is the declaration
                             columns
```

		0	1	2	3	4	5
r	0	95					-27
o	1				17		
w	2		68				
s							

- **Example Usage:**

```
number[0][0] = 95;
number[0][5] = -27;
number[1][3] = 17;
number[2][1] = 68;
```

## Processing a Two-Dimensional Array in a Main Program

```
/* program to read data into a two-dimensional array */
#include <iostream>
using namespace std;

const int MAXSIZE = 50;
const int NUMEXAMS = 4;

int main()
{
    int grade[MAXSIZE][NUMEXAMS];
    int class_size;

    cout << "How many students in the class? ";
    cin >> class_size;

    for (int stnum = 0; stnum < class_size; stnum++)
    {
        cout << "Type in four grades for student " << stnum
             << endl;
        for (int exam = 0, exam < NUMEXAMS; exam++)
            cin >> grade[stnum][exam];
        cout << "The grades for student " << stnum << " were:";
        for (int exam = 0, exam < NUMEXAMS; exam++)
            cout << " " << grade[stnum][exam];
        cout << endl;
    }
    return 0;
}
```

## Passing a Two-Dimensional Array as a Parameter

- **The Function findstudentavg():**

```
/* Function findstudentavg()
 * Input:
 *   grade - a 2-dimensional array of grades
 *   NUMEXAMS - numbers of exams for each student
 *   class_size - number of students in the class
 * Process:
 *   finds each student's average
 * Output:
 *   prints each student's average
 */
void findstudentavg(int grade[][NUMEXAMS], int class_size)
{
    int sum;
    double avg;

    for (int stnum = 0; stnum < class_size; stnum + +)
    {
        sum = 0;
        for (int exam = 0; exam < NUMEXAMS; exam + +)
            sum + = grade[stnum][exam];
        avg = (double)sum/NUMEXAMS;
        cout << "Student " << stnum << " had an average of "
            << avg << endl;
    }
    return;
}
```

- **Function Prototype:**

```
void findstudentavg(int [][][NUMEXAMS], int);
```

- **Function Usage:**

```
findstudentavg(grade,class_size);
```

## Processing Down a Column of a Two-Dimensional Array

- **The Function findexamavg():**

```
/* Function findexamavg()
 * Input:
 *   grade - a 2-dimensional array of grades
 *   NUMEXAMS - numbers of exams for each student
 *   class_size - number of students in the class
 * Process:
 *   finds the class average on each exam
 * Output:
 *   prints the class average on each exam
 */
void findexamavg(int grade[][NUMEXAMS], int class_size)
{
    int sum;
    double avg;

    for (int exam = 0; exam < NUMEXAMS; exam++) {
        sum = 0;
        for (int stnum = 0; stnum < class_size; stnum++)
            sum += grade[stnum][exam];
        avg = (double)sum/class_size;
        cout << "Exam " << exam <<
            " had a class average of " << avg << endl;
    }
    return;
}
```

- **Function Prototype:**

```
void findexamavg(int [][][NUMEXAMS], int);
```

- **Function Usage:**

```
findexamavg(grade,class_size);
```

- **Multi-Dimensional Arrays:**

```
data_type identifier[num_dim_1][num_dim_2]...[num_dim_n];
```