

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 40 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:**

3 91 94 65

- **First Draft of Program:**

```
/* Program to process exam grades */
#include <stdio.h>
void main()
{
    int count,num,sum= 0;
    int mark;
    double avgmark;

    printf("Enter the number of marks: ");
    scanf("%d",&num);
    if (num > 0)
        printf("There are %d marks\n",num);
    else {
        printf("Invalid number of marks\n",num);
        exit(1);
    }
    // enter marks and find average mark
    for (count = 1; count <= num; count+ + ) {
        printf("Enter a mark: ");
        scanf("%d",&mark);
        sum + = mark;
    }
    avgmark = sum/num;           // THIS IS WRONG!!
    printf("The average is %6.2f\n",avgmark);
}
```

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 <stdio.h>
void main()
{
    int count,num,sum= 0;
    int mark;
    double avgmark;

    printf("Enter the number of marks: ");
    scanf("%d",&num);
    if (num > 0)
        printf("There are %d marks\n",num);
    else {
        printf("Invalid number of marks\n",num);
        exit(1);
    }
    // enter marks and find average mark
    for (count = 1; count <= num; count+ + ) {
        printf("Enter a mark: ");
        scanf("%d",&mark);
        sum + = mark;
    }
    avgmark = (double)sum/num;
    printf("The average is %6.2f\n",avgmark);
}
```

- **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 z[5];
```

Array z

z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

- **Using the Array:**

```
z[0] = 3;
```

```
z[1] = 15;
```

```
z[2] = -2;
```

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

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

Array z

z[0]	3
z[1]	15
z[2]	-2
z[3]	18
z[4]	43

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

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

Array z

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

- **Initializing Array Elements:**

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

Array z

z[0]	5
z[1]	11
z[2]	-1
z[3]	17
z[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 1994, 1995, 1996 */

double sales[3] = {0,0,0};
double amount;
int year,count,numtrans;

printf("Enter the number of transactions: ");
scanf("%d",&numtrans);
for (count = 0; count < numtrans; count+ + ) {
    printf("Enter the year: ");
    scanf("%d",&year);
    printf("Enter the amount of sales for the year: ");
    scanf("%d",&amount);
    sales[year - 1994] + = amount;
}
for (year = 1994; year < = 1996; year+ + );
    printf("sales for %d are %8.2f\n",year,sales[year-1994]);
```

- **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') {
    printf("%c",message[i]);
    i+ + ;
}                                     //output: weird stuff
printf("\n");
```

```
for (i = 0; message[i] != '\0'; i+ + )
    printf("%c",message[i]);         //output: weird stuff
printf("\n");
```

```
printf("%s\n",message);             //output: weird stuff
```

- **Note:** The conversion specification **%s** means the value of a **null terminated string**.

- **Second Version of the Program:**

```
/* Program to process exam grades */
#include <stdio.h>
#define SIZE 40
void main()
{
    int count,num,sum= 0;
    int mark[SIZE];
    double avgmark;

    printf("Enter the number of marks: ");
    scanf("%d",&num);
    if (num > 0 && num <= SIZE)
        printf("There are %d marks\n",num);
    else {
        printf("Invalid number of marks\n");
        exit(1);
    }
    // enter marks and find average mark
    for (count = 0; count < num; count+ + ) {
        printf("Enter a mark: ");
        scanf("%d",&mark[count]);
        sum + = mark[count];
    }
    avgmark = (double)sum/num;
    printf("The average is %6.2f\n",avgmark);
}
```

- **Modular Version of the Program:**

```
/* Program to process exam grades */
#include <stdio.h>
#define SIZE 40
void main()
{
    int count,num,sum;
    int mark[SIZE];
    double avgmark;

    printf("Enter the number of marks: ");
    scanf("%d",&num);
    if (num > 0 && num <= SIZE)
        printf("There are %d marks\n",num);
    else {
        printf("Invalid number of marks\n");
        exit(1);
    }

    // enter marks and print them
    for (count = 0; count < num; count+ + ) {
        printf("Enter a mark: ");
        scanf("%d",&mark[count]);
        printf("mark[%d] = %d\n",count,mark[count]);
    }

    // find the average mark
    sum = 0;
    for (count = 0; count < num; count+ + )
        sum + = mark[count];
    avgmark = (double)sum/num;
    printf("The average is %6.2f\n",avgmark);
}
```

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 count,sum= 0;

    for (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
 *   parameter is uninitialized upon entry
 * Process:
 *   reads n and reads n values into the array
 * Output:
 *   fills the array
 *   returns n
 */
int readdata(int numbers[])
{
    int count,n;

    printf("Enter the number of marks: ");
    scanf("%d",&n);
    if (n > 0 && n <= SIZE)
        printf("There are %d marks\n",n);
    else {
        printf("Invalid number of marks\n",n);
        exit(1);
    }
    // enter marks
    for (count = 0; count < n; count+ + ) {
        printf("Enter a mark: ");
        scanf("%d",&numbers[count]);
    }
    return (n);
}
```

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

```
/* Program to process exam grades */
#include <stdio.h>
#define SIZE 40

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

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

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

    // print the mark array
    for (count = 0; count < num; count+ + )
        printf("mark[%d] = %d\n",count,mark[count]);

    // find and print the average mark
    avgmark = avgarray(mark,num);
    printf("The average is %6.2f\n",avgmark);
}
```

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 count, largest_so_far;

    largest_so_far = numbers[0];
    for (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
 * Process:
 *   counts the number of elements that are above, below,
 *   and equal to avgnum.
 * Output:
 *   prints these results
 */
void countmarks(int numbers[], int n, double avgnum)
{
    int count;
    int num_below = 0, num_above = 0, num_equal = 0;

    for (count = 0; count < n; count++ )
        if ((double)numbers[count] < avgnum)
            num_below++ ;
        else if ((double)numbers[count] > avgnum)
            num_above++ ;
        else
            num_equal++ ;
    printf("Number of marks below the average is %d\n",
        num_below);
    printf("Number of marks above the average is %d\n",
        num_above);
    printf("Number of marks equal to the average is %d\n",
        num_equal);
    return;
}
```

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

```
/* Program to process exam grades */
#include < stdio.h>
#define SIZE 40

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

void main()
{
    int count,num,hi_mark;
    int mark[SIZE];
    double avgmark;

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

    // print the mark array
    for (count = 0; count < num; count+ + )
        printf("mark[%d] = %d\n",count,mark[count]);

    // find and print the average mark
    avgmark = avgarray(mark,num);
    printf("The average is %6.2f\n",avgmark);

    // find and print the highest mark
    hi_mark = findmax(mark,num);
    printf("The highest mark is %d\n",hi_mark);

    // classify marks w.r.t. average mark
    countmarks(mark,num,avgmark);
}
```

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 <stdio.h>
#define MAXSIZE 50
#define NUMEXAMS 4

void main()
{
    int grade[MAXSIZE][NUMEXAMS];
    int class_size;
    int stnum;           // student number
    int exam;           // exam number

    printf("How many students in the class? ");
    scanf("%d",&class_size);
    for (stnum = 0; stnum < class_size; stnum+ + ) {
        printf("Type in four grades for student %d: ",stnum);
        for (exam = 0, exam < NUMEXAMS; exam+ + )
            scanf("%d",&grade[stnum][exam]);
        printf("The grades for student %d were",stnum);
        for (exam = 0, exam < NUMEXAMS; exam+ + )
            printf(" %d",grade[stnum][exam]);
        printf("\n");
    }
}
```

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 stnum,exam,sum;
    double avg;

    for (stnum = 0; stnum < class_size; stnum+ + ) {
        sum = 0;
        for (exam = 0; exam < NUMEXAMS; exam+ + )
            sum + = grade[stnum][exam];
        avg = (double)sum/NUMEXAMS;
        printf("Student %d had an average of %6.2f\n",
            stnum,avg);
    }
    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 stnum,exam,sum;
    double avg;
    for (exam = 0; exam < NUMEXAMS; exam+ + ) {
        sum = 0;
        for (stnum = 0; stnum < class_size; stnum+ + ) {
            sum + = grade[stnum][exam];
        }
        avg = (double)sum/class_size;
        printf("Exam %d had a class average of %6.2f\n",
            exam,avg);
    }
    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];
```

Using EOF to Check for the End of a Data Set

- **Reminder:**

`scanf()` returns the number of successful conversions, (this can be 0). If an end of the input stream is reached, the return value is the constant `EOF`.

- **The Function `readdata`:**

```
/* ... */
int readdata(int numbers[])
{
    int count= 0;

    // read and count number of marks
    while (scanf("%d",&numbers[count]) != EOF)
        count+ + ;
    return (count);
}
```