Arrays

Consider a program that will keep track of quiz scores. (Perhaps for EE 201.) There are 10 quizzes, with a possible 10 points each. I'd like to use a program to compute the total. Using what we have learned up to now, we probably proceed by declaring 10 integer variables, and use 10 scanf()s to enter the scores. Once the scores are entered, we can compute the total.

```c
//  pre-array
//
//  Created by Gary Tuttle on 9/15/16.
//  Copyright © 2016 Gary Tuttle. All rights reserved.

#include <stdio.h>

int main( void ){

    int quiz1, quiz2, quiz3, quiz4, quiz5;
    int quiz6, quiz7, quiz8, quiz9, quiz10;

    int quizTotal = 0;

    printf( "\nEnter score for quiz 1: ");
    scanf( "%d", &quiz1);

    printf( "\nEnter score for quiz 2: ");
    scanf( "%d", &quiz2);

    printf( "\nEnter score for quiz 3: ");
    scanf( "%d", &quiz3);

    printf( "\nEnter score for quiz 4: ");
    scanf( "%d", &quiz4);

    printf( "\nEnter score for quiz 5: ");
    scanf( "%d", &quiz5);

    printf( "\nEnter score for quiz 6: ");
    scanf( "%d", &quiz6);

    printf( "\nEnter score for quiz 7: ");
    scanf( "%d", &quiz7);

    printf( "\nEnter score for quiz 8: ");
    scanf( "%d", &quiz8);

    printf( "\nEnter score for quiz 9: ");
    scanf( "%d", &quiz9);

    printf( "\nEnter score for quiz 10: ");
    scanf( "%d", &quiz10);

    quizTotal = quiz1 + quiz2 + quiz3 + quiz4 + quiz5;
    quizTotal = quizTotal + quiz6 + quiz7 + quiz8 + quiz9 + quiz10;

    printf( "\n\nThe raw quiz score is %d out of 100 possible.\n\n", quizTotal);

    return 0;
}
```

```
Enter score for quiz 1: 8

Enter score for quiz 2: 9

Enter score for quiz 3: 10

Enter score for quiz 4: 9

Enter score for quiz 5: 3

Enter score for quiz 6: 9

Enter score for quiz 7: 10

Enter score for quiz 8: 9

Enter score for quiz 9: 8

Enter score for quiz 10: 10


The raw quiz score is 85 out of 100 possible.

Program ended with exit code: 0
```

Now, we would like to throw out the lowest score, and keep only the best nine. We can add some code to do that. Define one more integer variable, minQuiz, initialized to 10. Then the code below can be inserted at the end of the previous.

```
    if( quiz1 < minQuiz )
        minQuiz = quiz1;

    if( quiz2 < minQuiz )
        minQuiz = quiz2;

    if( quiz3 < minQuiz )
        minQuiz = quiz3;

    if( quiz4 < minQuiz )
        minQuiz = quiz4;

    if( quiz5 < minQuiz )
        minQuiz = quiz5;

    if( quiz6 < minQuiz )
        minQuiz = quiz6;

    if( quiz7 < minQuiz )
        minQuiz = quiz7;

    if( quiz8 < minQuiz )
        minQuiz = quiz8;

    if( quiz9 < minQuiz )
        minQuiz = quiz9;

    if( quiz10 < minQuiz )
        minQuiz = quiz10;

    printf( "The minimum quiz score was %d \n\n", minQuiz);

    printf( "The net quiz score is %d out of 90 possible.\n\n", quizTotal - minQuiz);


    return 0;
}
```

```
Enter score for quiz 1: 8

Enter score for quiz 2: 9

Enter score for quiz 3: 10

Enter score for quiz 4: 9

Enter score for quiz 5: 3

Enter score for quiz 6: 9

Enter score for quiz 7: 10

Enter score for quiz 8: 9

Enter score for quiz 9: 8

Enter score for quiz 10: 10


The raw quiz score total is 85 out of 100 possible.

The minimum quiz score was 3.

The net quiz score is 82 out of 90 possible.

Program ended with exit code: 0
```

However, this is terribly clunky and does not scale. A scanf() line is needed for each score. Each variable has to be separately listed when the scores are summed up. Each variable has a separate if statement when finding the minimum. Tedious. The variable storage can be scattered. This should not be a problem, because the compiler keeps track of memory alloction, but it looks untidy.

address	value	variable
00000		
00001	9	quiz8
00010		
00011		
00100	8	quiz1
00101		
00110		
00111		
01000	3	quiz5
01001		
01010		
01011		
01100	10	quiz10
01101		
01111		

address	value	variable
10000	9	quiz4
10001	9	quiz6
10010		
10011	10	quiz3
10100		
10101	9	quiz2
10110		
10111	10	quiz7
11000		
11001		
11010		
11011		
11100		
11101		
11111	8	quiz9

We can streamline matters significantly by using *arrays*. Arrays are blocks of variables that can be handled in a unified manner. The block is given a single variable name, and the individual items are referenced using an *index*.

When the array variable is declared, the total number of individual items in the array is also specified. For example:

```
int quizScore[10];
```

creates a block of 10 contiguous memory spaces. The block is known as quizScore and the individual items are accessed using the index.

quizScore[1] —> second item in the array

quizScore[6] —> seventh item in the array

quizScore[9] —> ninth item in the array

etc.

Niggling detail: Arrays in C are enumerated starting at index 0. So when an array of 10 items is created, they are referenced by

quizScore[0], quizScore[1], …, quizScore[9].

This will almost certainly cause you trouble at some point. Don't forget this annoyance and count carefully.

address	value	variable
00000		
00001	9	quiz8
00010	8	quizScore[0]
00011	9	quizScore[1]
00100	10	quizScore[2]
00101	9	quizScore[3]
00110	3	quizScore[4]
00111	9	quizScore[5]
01000	10	quizScore[6]
01001	9	quizScore[7]
01010	8	quizScore[8]
01011	10	quizScore[9]
01100		
01101		
01111		

address	value	variable
10000		
10001		
10010		
10011		
10100		
10101		
10110		
10111		
11000		
11001		
11010		
11011		
11100		
11101		
11111		

Note: If you don't like counting from 0, you can avoid it by defining the array to be one item bigger than needed. Then simply ignore the 0th item and starting counting at 1. So if you had 10 items that you wanted to use, you would define the array as having 11 items (numbered 0 to 10), and then simply use items 1 through 10 only.

This works fine, although you will be wasting memory space. This is probably not a big deal in most cases. But either way, you have to account for the fact that there is always a 0th item in the array.

What makes arrays powerful is that the index itself can be a variable. And so *while* and *for* loops can be used to count through the range of indices, sequentially accessing the items in the array.

The fact that the items are all grouped together in one block of memory is also handy. Instead of using variable names, we can access the items by referring to the memory location of the first item. This is known as using a *pointer*. We will discuss in detail later.

Using arrays and loops together makes our long clunky program compact and scalable.

```c
//  arrays
//
//  Created by Gary Tuttle on 9/15/16.
//  Copyright © 2016 Gary Tuttle. All rights reserved.
//

#include <stdio.h>

int main( void) {

    int i;              //a counter

    int quiz[10];       //the array for our quiz scores, numbered 0 .. 9

    int quizTotal = 0;  //the total score, intialized to zero
    int minQuiz = 10;   //the smallest quiz score, initialized to 10

    for(i = 0; i <= 9; i=i+1){

        printf( "\nEnter score for quiz %d: ", i+1);
        scanf( "%d", &quiz[i]);
    }

    printf("\n\n");

    for(i = 0; i <= 9; i = i + 1)
        quizTotal = quizTotal + quiz[i];

    printf( "The raw quiz score total is %d out of 100 possible.\n\n", quizTotal);

    for(i = 0; i <= 9; i = i + 1){
        if( quiz[i] < minQuiz )
            minQuiz = quiz[i];
    }

    printf( "The minimum quiz score was %d. \n\n", minQuiz);

    printf( "The net quiz score is %d out of 90 possible.\n\n", quizTotal - minQuiz);

    return 0;
}
```

```
Enter score for quiz 1: 8

Enter score for quiz 2: 9

Enter score for quiz 3: 10

Enter score for quiz 4: 9

Enter score for quiz 5: 3

Enter score for quiz 6: 9

Enter score for quiz 7: 10

Enter score for quiz 8: 9

Enter score for quiz 9: 8

Enter score for quiz 10: 10

The raw quiz score total is 85 out of 100 possible.

The minimum quiz score was 3.

The net quiz score is 82 out of 90 possible.

Program ended with exit code: 0
```

Finally, with one small change, we can make program more scalable. By changing one value, the program can handle different numbers of quizzes.

```
//  arrays
//
//  Created by Gary Tuttle on 9/15/16.
//  Copyright © 2016 Gary Tuttle. All rights reserved.
//

#include <stdio.h>

int main( void) {

    int i;                  //a counter

    const int NUM_OF_QUIZZES = 10;

    int quiz[10];           //the array for our quiz scores, numbered 0 .. 9

    int quizTotal = 0;  //the total score, intialized to zero
    int minQuiz = 10;   //the smallest quiz score, initialized to 10

    for(i = 0; i < NUM_OF_QUIZZES; i=i+1){

        printf( "\nEnter score for quiz %d: ", i+1);
        scanf( "%d", &quiz[i]);
    }

    printf("\n\n");

    for(i = 0; i < NUM_OF_QUIZZES; i = i + 1)
        quizTotal = quizTotal + quiz[i];

    printf( "The raw quiz score total is %d out of 100 possible.\n\n", quizTotal);

    for(i = 0; i < NUM_OF_QUIZZES; i = i + 1){
        if( quiz[i] < minQuiz )
            minQuiz = quiz[i];
    }

    printf( "The minimum quiz score was %d. \n\n", minQuiz);

    printf( "The net quiz score is %d out of 90 possible.\n\n", quizTotal - minQuiz);

    return 0;
}
```