Introduction to Arrays

Session 7 14 January 2003
Achieve Target 1

We have focused on program flow control using selection (if-else) and iteration (for-while) structures so far. We will look at a new feature in organizing data in your programs – arrays. Study the following program.

```c
int anArray[1];
printf("Enter an integer: ");
scanf("%d", &anArray[0]);
printf("The integer entered is %d\n", anArray[0]);
...
```

Enter an integer: 23
The integer entered is 23

The above program reads in an integer and prints it out again – a pattern that you have seen a lot before. The key difference is the use of an array element to store the integer.
The program requires only one integer variable to store the integer. The line with a red circle declares an array of integer with the size of one.

Each integer array element can store an integer. The size of an array indicates how many array elements there are in the array.

Remember it. The size of an array indicates how many array elements there are in the array. The line with a red circle declares an array of integer with the size of one. The program requires only one integer variable to store the integer.

Achieve Target 1
Achieve Target 1

The name `anArray` refers to the whole array of integer elements.

Each array element is identified by a number (or index). The first element is numbered 0, the second is numbered 1, and so on.

```c
int anArray[1];

printf("Enter an integer: ");
scanf("%d", &anArray[0]);

printf("The integer entered is %d\n", anArray[0]);
```

Remember it.

We use this index number to refer to an individual integer element, and we enclose the index number within a pair of square brackets.

In the above program, `anArray[0]` represents the first element in the array `anArray`. In other words, `anArray[0]` can be used as an integer variable.
Achieve Target 1

Modify the program below so that instead of printing the entered number, we now want the square of the number be printed.

The square of the number is the number multiplied by itself.

```
... 
int anArray[1];

printf("Enter an integer: ");
scanf("%d", &anArray[0]);

printf("The integer entered is %d\n", anArray[0]);
... 
```

Work on the program to make it happen.
Achieve Target 1

Solution

```c
int anArray[1];

printf("Enter an integer: ");
scanf("%d", &anArray[0]);
anArray[0] = anArray[0] * anArray[0];
printf("Its square is %d\n", anArray[0]);
```

An alternative.

```c
int anArray[1];

printf("Enter an integer: ");
scanf("%d", &anArray[0]);
printf("Its square is %d\n", anArray[0]* anArray[0]);
```
float anArray[2];
float sum;

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

sum = anArray[0] + anArray[1];

printf("The sum is %f\n", sum);
...

Enter first number: 20
Enter second number: 12
The sum is 32.000000
Achieve Target 1A

Modify the program below so that instead of using the variable `sum` to store the sum of `anArray[0]` and `anArray[1]`, the new program uses the third array element. Pay attention to the following issues.

1. Declare the array size correctly – we now need three array elements instead of 2.
2. Refer to the third array element correctly – remember that the first element is indexed 0.

```c
... 
float anArray[2];
float sum;

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

sum = anArray[0] + anArray[1];

printf("The sum is %f\n", sum);
...
```

Work on the program to make it happen.
Achieve Target 1A

Solution

...  

```c
float anArray[3];

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

anArray[2] = anArray[0] + anArray[1];

printf("The sum is %f\n", anArray[2]);
```

...
Each array element is same as a float variable

```c
float anArray[3];

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

anArray[2] = anArray[0] + anArray[1];

printf("The sum is %f\n", anArray[2]);
...
Rewrite the program below so that the program uses only a float array instead of using the three float variables. Pay attention to the following issues.

1. Declare the array size correctly – how many array elements does the program need?
2. Designate each array element for holding the height, weight, and bmi, and follow the designation consistently.

```c
...  
float height, weight, bmi;  
printf("Enter your Height(m): ");  
scanf("%f", &height); 
 printf("Enter your Weight(kg): ");  
scanf("%f", &weight); 
   bmi = weight / (height * height);  
 printf("Your BMI = %f\n", bmi);  
...  
```
Achieve Target 1B

Solution

```c
#include<stdio.h>

void main()
{

    float anArray[3];
    printf("BMI calculator\n");

    printf("Enter your Height(m): ");
    scanf("%f", &anArray[0]);

    printf("Enter your Weight(kg): ");
    scanf("%f", &anArray[1]);

    anArray[2] = anArray[1] / (anArray[0] * anArray[0]);
    printf("Your BMI = %f", anArray[2]);

    fflush(stdin);
    getchar();
}
```
Achieve Target 2

Study the following program.

```c
float anArray[3];
float sum;

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

printf("Enter third number: ");
scanf("%f", &anArray[2]);

sum = anArray[0] + anArray[1] + anArray[2];

printf("The sum is %f\n", sum);
```
Achieve Target 2

The following program behaves exactly as the previous but it uses arrays in a more typical manner – with a loop.

```c
float anArray[3];
float sum;

printf("Enter first number: ");
scanf("%f", &anArray[0]);

printf("Enter second number: ");
scanf("%f", &anArray[1]);

printf("Enter third number: ");
scanf("%f", &anArray[2]);

for (i=0; i<3; i++)
    sum = sum + anArray[i];

printf("The sum is %f\n", sum);
```

...
Achieve Target 2

In a similar manner, the input part can also be replaced by a loop.

```c
...  
float anArray[3];  
float sum = 0;  
int i;  

printf("Enter 3 numbers: \n");  
for (i=0; i<3; i++)  
    scanf("%f", &anArray[i]);

for (i=0; i<3; i++)  
    sum = sum + anArray[i];

printf("The sum is %f\n", sum);
...  
Enter 3 numbers:  
2  
3  
4  
The sum is 9.000000
```
Achieve Target 2

The program on the left side can be rewritten as the program on the right side.

```c
float anArray[3];
float sum = 0;
int i;
printf("Enter 3 numbers:\n");
for (i=0; i<3; i++)
    scanf("%f", &anArray[i]);
    sum = sum + anArray[i];
}
printf("The sum is %f\n", sum);
...
Achieve Target 2

Modify the program below so that 10 numbers are read in and the sum calculated.

```c
float anArray[3];
float sum = 0;
int i;

printf("Enter 3 numbers:\n");

for (i=0; i<3; i++) {
    scanf("%f", &anArray[i]);
    sum = sum + anArray[i];
}

printf("The sum is %f\n", sum);
```

Work on the program to make it happen.
Solution

```c
#include<stdio.h>

void main(){

    float anArray[10];
    float sum = 0;
    int i;

    printf("Enter 10 numbers:\n");
    for (i=0; i<10; i++) {
        scanf("%f", &anArray[i]);
        sum = sum + anArray[i];
    }

    printf("The sum is %f\n", sum);

    fflush(stdin);
    getchar();
}
```

The key is to make modification to all the three places that specifies the size.
int anArray[5];
int i;

printf("Enter 5 integers:\n");
for (i=0; i<5; i++) {
    scanf("%d", &anArray[i]);
}

printf("The integers entered in reverse are:\n");

for (i=4; i>=0; i--) {
    printf("%d ", anArray[i]);
}

Enter 5 integers:
10
5
23
43
12

The integers entered in reverse are:
12 43 23 5 10
Modify the program below so that

1. It prints the integers in the original order (in the order they are entered).
2. It prints the even number integers only, in the original order.
3. It prints the first, third, fifth, elements of the array (remember the first element is indexed 0), but in the reverse order.

```c
... int anArray[5];
int i;

printf("Enter 5 integers:\n");
for (i=0; i<5; i++) {
    scanf("%d", &anArray[i]);
}

printf("The integers entered in reverse are:\n");

for (i=4; i>=0; i--) {
    printf("%d ", anArray[i]);
}
... 
```
Achieve Target 3

Solution

1. In original order

#include <stdio.h>

void main() {
    int anArray[5];
    int i;
    printf("Enter 5 integers:\n");
    for (i=0; i<5; i++) {
        scanf("%d", &anArray[i]);
    }
    printf("The integers entered in original order are:\n");
    for (i=0; i<5; i++) {
        printf("%d ", anArray[i]);
    }
}
Solution

2. Only the even integers in original order

```c
... int anArray[5];
    int i;
    printf("Enter 5 integers:\n");
    for (i=0; i<5; i++) {
        scanf("%d", &anArray[i]);
    }
    printf("The even integers entered in original order are:\n");
    for (i=0; i<5; i++) {
        if (anArray[i] % 2 == 0)
            printf("%d ", anArray[i]);
    }
...```
Achieve Target 3

Solution

3. Only the first, third, fifth, etc in reverse order

```c
#include <stdio.h>

void main() {

    int anArray[5];
    int i;
    printf("Enter 5 integers:\n");
    for (i=0; i<5; i++) {
        scanf("%d", &anArray[i]);
    }
    printf("The 1st, 3rd, 5th, etc entered in reverse are:\n");
    for (i=4; i>=0; i--)
    {
        if (i % 2 == 0)
            printf("%d ", anArray[i]);
    }
}
```
... char charArray[256];
    int length;
    int i;

    printf("Enter a line of text (press Enter at the end): \n");
    for (i=0; i<256; i++) {
        scanf("%c",&charArray[i]);
        if (charArray[i] == '\n') {
            length = i;
            break;
        }
    }

    printf("The text entered is: \n");
    for (i=0; i<length; i++)
        printf("%c", charArray[i]);
    printf("\n");
...

Enter a line of text (press Enter at the end):
Array is a useful data structure
The text entered is:
Array is a useful data structure
Are the following two statements are equivalent?

```c
scanf("%c", &charArray[i]);
charArray[i] = getchar();
```
Modify the program below so that only the lowercase alphabets (‘a’ to ‘z’) are printed. Hint: The statement `ch >= 'a' && ch <= 'z'` can test if a character variable `ch` is lowercase.

```
char charArray[256];
int length;
int i;

printf("Enter a line of text (press Enter at the end):\n");
for (i=0; i<256; i++) {
    scanf("%c", &charArray[i]);
    if (charArray[i] == '\n') {
        length = i;
        break;
    }
}

printf("The text entered is:\n");
for (i=0; i<length; i++)
    printf("%c", charArray[i]);
printf("\n");
```

Work on the program to make it happen.
Achieve Target 3A

Solution

...  
  printf("The lowercase alphabets entered are:\n");
  for (i=0; i<length; i++) {
      if (charArray[i] >= 'a' && charArray[i] <= 'z')
          printf("%c", charArray[i]);
  }
  printf("\n");
...
The program below contains a while loop that reads at most 128 numbers, and it terminates when the entered number is –1.

```c
...  
    float data = 0;  
    int count = 0;  
    
    printf("Enter at most 128 numbers below (-1 to finish)\n");  
    while (count < 128) {  
        scanf("%f", &data);  
        if (data == -1)  
            break;  
        count++;  
    }  
...  
```
Achieve Target 3B

Modify the program below so that the entered numbers are stored in an array of float.

Remember to declare an array with an appropriate size.

... 
float data = 0;
int count = 0;

printf("Enter at most 128 numbers below (-1 to finish)\n");
while (count < 128) {
    scanf("%f", &data);
    if (data == -1)
        break;
    count++;
}
...
float anArray[128];
int count = 0;

printf("Enter at most 128 numbers below
        (-1 to finish)\n");
while (count < 128) {
    scanf("%f", &anArray[count]);
    if (anArray[count] == -1)
        break;
    count++;
}

...
Achieve Target 3C

Write another loop that prints the positive numbers stored in the array.

Remember to declare loop counter variable for the second loop.

```c
float anArray[128];
int count = 0;

printf("Enter at most 128 numbers below (-1 to finish)\n");
while (count < 128) {
    scanf("%f", &anArray[count]);
    if (anArray[count] == -1)
        break;
    count++;
}
```

Work on the program to make it happen.
Solution

...  
float anArray[128];
tint count = 0;
int i;

printf("Enter at most 128 numbers below(-1 to finish)\n");
while (count < 128) {
    scanf("%f", &anArray[count]);
    if (anArray[count] == -1)
        break;
    count++;
}
for (i=0; i<count; i++) {
    if (anArray[i] > 0)
        printf("%f ", anArray[i]);
}
printf("\n");
...
Write another loop that detects and prints the largest number stored in the array.

Remember to declare and initialise a temporary variable for detecting the largest number.

```c
float anArray[128];
int count = 0;

printf("Enter at most 128 numbers below (-1 to finish)\n");
while (count < 128) {
    scanf("%f", &anArray[count]);
    if (anArray[count] == -1)
        break;
    count++;
}
...
```

Work on the program to make it happen.
... 

    float anArray[128];
    int count = 0;
    int i;
    float highest;
...

    highest = anArray[0];
    for (i=1; i<count; i++) {
        if (anArray[i] > highest)
            highest = anArray[i];
    }

    printf("Largest number is %f\n", highest);
...
The size of the array must be declared large enough for use. Trying to refer to array elements beyond the declared size will cause an error when running the program.

In the programs, a guard should be used to prevent accessing an array beyond the specified size.

Learn it.

In the solutions to Target 3C, the while loop serves as the guard; and in the program of Target 3A, the for loop serves as a guard.
Let us look more closely at storing data into array.

```c
... char charArray[256]; int length; int i;

printf("Enter a line of text (press Enter at the end):\n");

for (i=0; i<256; ) {
    scanf("%c",&charArray[i]);

    if (charArray[i] >= 'a' && charArray[i] <= 'z') {
        i++;
    } else if (charArray[i] == '\n') {
        length = i;
        break;
    }
}

/* printing the array elements */
for (i=0; i<length; i++)
    printf("%c", charArray[i]);
...```
Achieve Target 4

Rewrite the above program so that an entered character is stored when,

1. It is a uppercase alphabet with the exception of ‘C’, or;
2. It is a space.

```c
... char charArray[256];
    int length;
    int i;

    printf("Enter a line of text (press Enter at the end):\n");

    for (i=0; i<256; ) {
        scanf("%c", &charArray[i]);

        if (charArray[i] >= 'a' && charArray[i] <= 'z') {
            i++;
        } else if (charArray[i] == '\n') {
            length = i;
            break;
        }
    }

    /* printing the array elements */
    for (i=0; i<length; i++)
        printf("%c", charArray[i]);
...
```

Work on the program to make it happen.
... char charArray[256];
    int length;
    int i;

    printf("Enter a line of text (press Enter at the end):
\n");
    for (i=0; i<256; ) {
        scanf("%c", &charArray[i]);
        if (charArray[i] >= 'A' && charArray[i] <= 'Z') {
            if (charArray[i] != 'C')
                i++;
            else if (charArray[i] == ' ')
                i++;
        } else if (charArray[i] == ' ')
            i++;
    } else if (charArray[i] == '\n') {
        length = i;
        break;
    }
    for (i=0; i<length; i++)
        printf("%c", charArray[i]);
...
Achieve Target 4A

This program stores all entered numbers into the array, but prints out only the positive array elements. Modify the program so that only the positive numbers are stored in the array, instead of storing all numbers.

Note that the variable count serves the same purpose as the variable i previously.

```c
... float anArray[128];
    int count = 0;
    int k;

    printf("Enter at most 128 numbers below (-1 to finish)\n");
    while (count < 128) {
        scanf("%f", &anArray[count]);
        if (anArray[count] == -1)
            break;
        count++;
    }

    for (k=0; i<count; k++) {
        if (anArray[k] > 0)
            printf("%f ", anArray[k]);
    }
    printf("\n");
...
```

Work on the program to make it happen.
... 
float anArray[128];
int count = 0;
int k;

printf("Enter at most 128 numbers below (-1 to finish)\n");
while (count < 128) {
    scanf("%f", &anArray[count]);
    if (anArray[count] == -1)
        break;
    if (anArray[count] > 0)
        count++;
}

for (k=0; k<count; k++) {
    printf("%f ", anArray[k]);
}
printf("\n");
...
Arrays can be initialised in a similar way as initialising other variables.

```c
...  
int intArray[] = {1, 3, 5, 7};
float floatArray[] = {1, 3.2, 5.5};
char charArray[] = {'a', 'e', 'i', 'o', 'u'};
int i = 0;
for (i=0; i<4; i++)
    printf("%d ", intArray[i]);
printf("\n");
for (i=0; i<3; i++)
    printf("%f ", floatArray[i]);
printf("\n");
for (i=0; i<5; i++)
    printf("%c ", charArray[i]);
printf("\n");
...
```

1 3 5 7
1.000000 3.200000 5.500000
a e i o u
Achieve Target 5

Rewrite the above program so that the arrays are initialised with proper values so that the following is printed.
Take care of the execution conditions of the loops.

... int intArray[] = {1, 3, 5, 7};
float floatArray[] = {1, 3.2, 5.5};
char charArray[] = {'a', 'e', 'i', 'o', 'u'};

int i = 0;
for (i=0; i<4; i++)
    printf("%d ", intArray[i]);
printf("\n");
for (i=0; i<3; i++)
    printf("%f ", floatArray[i]);
printf("\n");
for (i=0; i<5; i++)
    printf("%c ", charArray[i]);
printf("\n");
...

Work on the program to make it happen.
int intArray[] = {10, -12, 3, 4, 5};
char charArray[] = {'+', '-', '*', '/'};

int i = 0;
for (i=0; i<5; i++)
    printf("%d ", intArray[i]);
printf("\n");
for (i=0; i<4; i++)
    printf("%c ", charArray[i]);
printf("\n");

...
... char charArray[] = {'a', 'e', 'i', 'o', 'u'};
int numVowels = 0;
int numAlpha = 0;
char data;
int i;

printf("Enter a line of text (Enter to terminate)\n");
while (1) {
    scanf("%c", &data);
    if (data == '
')
        break;
    else if ((data >= 'a' && data <= 'z') || (data >= 'A' && data <= 'Z'))
        numAlpha++;
    for (i=0; i<5; i++) {
        if (data == charArray[i]) {
            numVowels++;
            break;
        }
    }
}
printf("Number of alphabets %d and vowels %d\n", numAlpha, numVowels);
...

Enter a line of text (Enter to terminate):
C is a programming language
Number of alphabets 23 and vowels 9
... while (1) {
    scanf("%c", &data);
    if (data == '\n')
        break;
    else if ((data >= 'a' && data <= 'z') || (data >= 'A' && data <= 'Z'))
        numAlpha++;
    for (i=0; i<5; i++) {
        if (data == charArray[i]) {
            numVowels++;
            break;
        }
    }
}
printf("Number of alphabets %d and vowels %d\n", numAlpha, numVowels);
...

The statements in the red box are equivalent to the following familiar if statement.

if (charArray[i] == 'a' || charArray[i] == 'e' || charArray[i] == 'i' ||
    charArray[i] == 'o' || charArray[i] == 'u')
    numVowels++;
The program only detects vowels in lowercase. Modify the program so that it detects the uppercase vowels (‘A’, ‘E’, ‘I’, ‘O’, ‘U’) as well.

```c
... char charArray[] = {'a', 'e', 'i', 'o', 'u'};
    int numVowels = 0;
    int numAlpha = 0;
    char data;
    int i;

    printf("Enter a line of text (Enter to terminate):\n");
    while (1) {
        scanf("%c", &data);
        if (data == '\n')
            break;
        else if ((data >= 'a' && data <= 'z') || (data >= 'A' && data <= 'Z'))
            numAlpha++;

        for (i=0; i<5; i++) {
            if (data == charArray[i]) {
                numVowels++;
                break;
            }
        }
    }
    printf("Number of alphabets %d and vowels %d\n", numAlpha, numVowels);
...```
char charArray[] = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};
int numVowels = 0;
int numAlpha = 0;

for (i=0; i<10; i++) {
    if (data == charArray[i]) {
        numVowels++;
        break;
    }
}

The array initialization needs modification, and also the for loop for checking against the charArray.
The data set required in this program is the set of prime numbers. The above program uses an `if` statement to detect the number of prime numbers entered.
Modify the above program so that,

1. The program uses an integer array to define a prime number data set from 1 to 20 (which are 2, 3, 5, 7, 11, 13, 17, and 19).
2. The if statement is replaced by a for/while loop that detects for a prime number.

```c
int number;
int numPrimes = 0;

printf("Enter integers from 1 to 10 (-1 to terminate):
");  
while (1) {
    scanf("%d", &number);
    if (number == -1)
        break;
    else if (number == 2 || number == 3 || number == 5 || number == 7)
        numPrimes++;
}
printf("Number of primes = %d\n", numPrimes);
...```

Work on the program to make it happen.
... int number;
int primeArray[] = {2, 3, 5, 7, 11, 13, 17, 19};
int numPrimes = 0;
int i;

printf("Enter integers from 1 to 10 (-1 to terminate):\n");
while (1) {
    scanf("%d", &number);

    if (number == -1)
        break;
    for (i=0; i<8; i++) {
        if (number == primeArray[i]) {
            numPrimes++;
            break;
        }
    }
}
printf("Number of primes = %d\n", numPrimes);
...
Achieve Target 6

In the last 5 targets, you have touched on the following ideas.

Ponder upon these ideas and remember what you have learned.
1. An array is a set of variables/elements of the same variable type. Each array has a size that indicates the number of array elements. The array elements are arranged in a linear sequence, each has an index or numbering. The first array element is always indexed or numbered zero (0).

The following are examples of array declaration.

```c
int intArray[10]; /* an array of 10 int variables/elements */
float floatArray[20]; /* an array of 20 float variables/elements */
char charArray[5]; /* an array of 5 char variables/elements */
```

2. Each array element is no different from an ordinary typed variable. So each `int` array element is the same as an `int` variable. An array element is referred by the array name and its index or numbering. For example, the first element of `intArray` is referred by `intArray[0]`, and the third element of `charArray` is referred by `charArray[2]`. 
3. Arrays are usually used with loops, whereby each array element is visited by each execution of loops. This is usually called **array traversal** – travel through the array. In array traversal loops, the loop counter is often used as the index variable of the array. As the loop counter changes, the index changes as well. Each array element is visited one after the other.

4. Arrays must be declared with a sufficiently large size. Accessing array elements beyond the declared size will cause error. Use **array guard** to prevent this from happening. An array guard is a condition that monitors the index variable of array and makes sure that the index cannot go beyond the declared size. Usually the loop execution condition is used as the array guard, but additional **if statements** could be used as well.
5. Array elements could be **initialised** with values. In this initialisation form, the values are arranged within a pair of **curly brackets**. The **array size** needs no explicitly declared in this form of array declaration.

   ```
   int intArray[] = {1, 2, 5, 8};
   ```

6. An **explicit array size** could also be specified in the initialisation form. The size, however, must be equal to or more than the number of initialising values. For an array size **more than** the number of initialising values, some array elements at the end will not be initialised. For example, in the last array declaration in the following, only the first four elements are initialised. The values of the last two elements are undefined.

   ```
   char charArray[3] = {'a', 'b', 'c'};
   int intArray[6] = {1, 2, 5, 8};
   ```
7. A common application of arrays is to store a pre-defined data set for comparison or matching. For example, we could define the set of vowels or prime numbers for detection purpose. The advantage is that we could make modification of the data set easily.
End of Session 7

Session 8  28 January 2003

All rights reserved. No part of this show may be reproduced, transmitted, or stored in a retrieval system, in any form or by any means, without permission in writing by the author.