LECTURE 07

Array

Outcomes

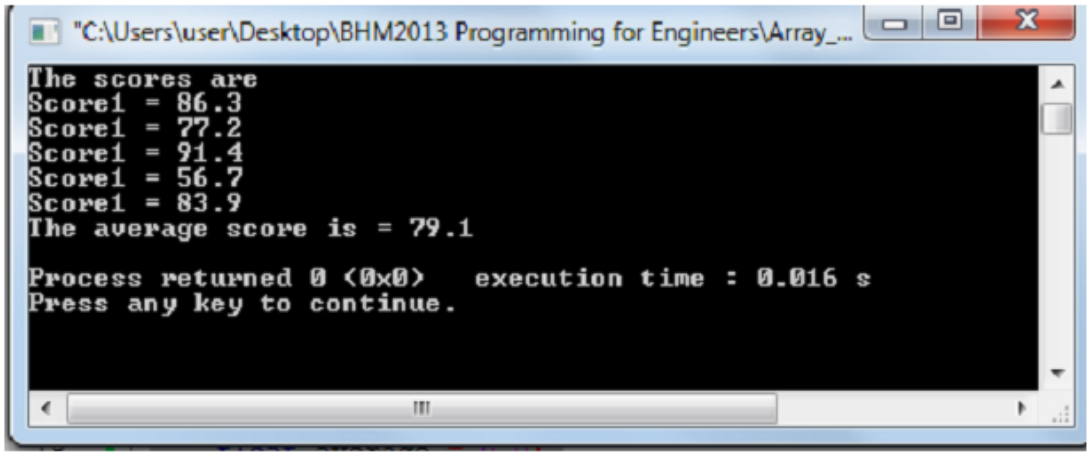
1. To understand basic array concepts.
2. To create simple C program that can manipulate, access and store array of data.

Contents

1. Introduction

Previously, we learn about basic data types such as integer, character and floating numbers. In C programming language, if we have 5 test scores and would like to average the scores, we may code in the following way.

```
#include <stdio.h>
int main()
{
	//declare and initialize values
	float score1 = 86.3;
	float score2 = 77.2;
	float score3 = 91.4;
	float score4 = 56.7;
	float score5 = 83.9;
	float average = 0.0;
	// determine the average
	average = (score1+score2+score3+score4+score5)/5;
	// display the results
	printf("The scores are\n");
	printf("Score1 = %4.1f\n",score1);
	printf("Score1 = %4.1f\n",score2);
	printf("Score1 = %4.1f\n",score3);
	printf("Score1 = %4.1f\n",score4);
	printf("Score1 = %4.1f\n",score5);
	printf("The average score is = %4.1f\n",average);
	return 0;
}
```



This program is manageable if the scores are only 5. What should we do if we have 100,000 scores? In such case, we need an efficient way to represent **a collection of similar data type**. In C programming, we usually use array. Array is a fixed-size sequence of elements of the same data type. In C programming, we declare an array like the following statement:

```
float score[5];
```

The above statement tells the compiler that we want to group five float numbers in one array called score. Thus in computer memory, we organize the data like the following:

score[0]	four byte of a float number
score[1]	four byte of a float number
score[2]	four byte of a float number
score[3]	four byte of a float number
score[4]	four byte of a float number

So the compiler has allocated 5 memory spaces for each float number. Notice that the subscript (the number inside the parenthesis) is from 0 to 4 not from 0 to 5 because we allocate only 5 memory spaces and we start our subscript from 0. Be aware that in an array, there is only one data type, in this case the float data type. Thus, the total bytes reserved for the variable `score[5]` are 20 bytes.

Each data in an array is called an element of an array. In our case, `score[2]` or `score[4]` is **an element** of the array `score[5]`. Similarly, we can declare arrays for other data types.

`int number[100]` - array name "number" with 100 sequence of integer data (4000 bytes).
`char huruf[53]` - array name "huruf" with 53 sequence of character data (53 bytes).
`double points[1000]` - array name "points" with 1000 sequence of double data (12,000 bytes).

The elements of an array can be initialized in several ways:

- Basic initialization - initialize all values in the array.
  `int number[5] = {3,7,1,1,21};`
  In this case, `number[0] = 3`, `number[1] = 7`, `number[2] = 1`, `number[3] = 1` and `number[4] = 21`.
- Partial initialization - initialize some values in the array.
  `int number[5] = {3,7};`

In this example, the first two elements are initialized while the rest is zero. Specifically, `number[0] = 3`, `number[1] = 7`, `number[2] = 0`, `number[3] =0` and `number[4] =0`.

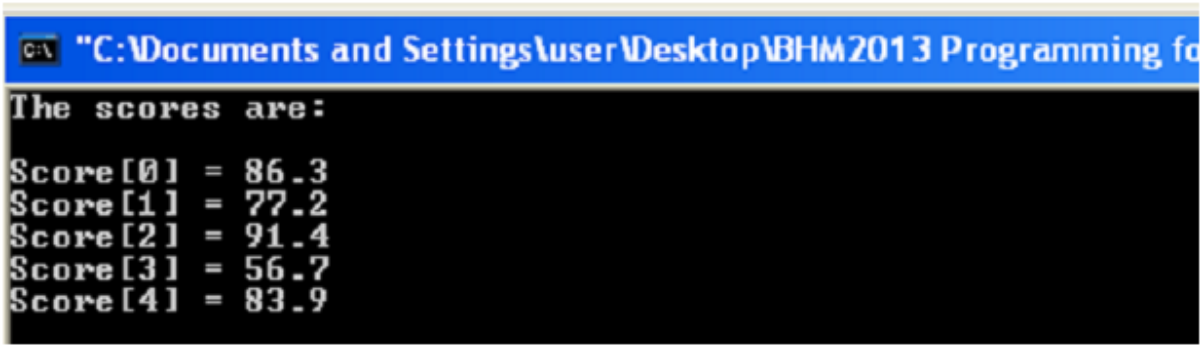
- Initialize to all zero.

```
int number[5] = {0};
```

To display the elements of an array we need to use repetitive structure (loop) such as the for-loop or the while-loop. We write a program using an array data type.

```
#include <stdio.h>

int main()
{
	//declare and initialize and array named "score" with five values
	float score[5] = {86.3,77.2,91.4,56.7,83.9};
	int i;
	//display the array elements
	printf("The scores are:\n\n");
	for (i=0;i<5;i++)
		printf("Score[%d] = %4.1f\n",i,score[i]);
	return 0;
}
```



2. Save Array to File

To be a good engineer, we should know the basic knowledge to save and read output to and from file. This is because, when you dealing with some applications such as MATLAB or CADCAM, we need to know how to manipulate the data given and also save the process data so that it can be read by other program. In this lesson, we will learn how to save array to file. Later on in the next class, we will learn further on how to read data from file to array. Let's say we have the following simple program:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[5], i;
    for (i=0;i<5;i++)
    {
        a[i] = i;
        printf("%d\t",a[i]);
    }
    return 0;
}
```

The output of the above program is:

0 1 2 3 4

So how we are going to save those output into file? C provides way to save those output into file. The basic syntax for this terminology is as follows:

1. `FILE *fptr;`
   - Declare file variable(fptr)
2. `fptr=(fopen("C:\\test.txt","w"));`
   - To open a file, use the `fopen` function
   - `w` for write, `C:\\test.txt` is a file name and location
3. `fprintf(fptr,"%d\t", a[i]);`
   - Add `fprintf` function right after `printf` to copy `a[i]` to file
4. `fclose(fptr);`
   Close file using `fclose` function

The final code for the above example is:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fptr;
    fptr = (fopen("test.txt","w"));
    int a[5], i;
    for (i=0;i<5;i++)
    {
        a[i] = i;
        printf("%d\t",a[i]);
        fprintf(fptr,"%d\t", a[i]);
    }
    fclose(fptr);
    return 0;
}
```

Notice that there are many variant modes for file manipulation (refer no 2.). The allowed modes for `fopen` are as follows:

`r` - open for reading
`w` - open for writing (file need not exist)
`a` - open for appending (file need not exist)
`r+` - open for reading and writing, start at beginning
`w+` - open for reading and writing (overwrite file)
`a+` - open for reading and writing (append if file exists)

We can save output to other file extension if we want. In the above example, I'm using text file (.txt) as a file. We can also save to Microsoft office file (.doc), wordpad file (.rtf) etc.

C has three loop statements which are the `for`, the `while`, and the `do-while`. The first two are pre-test loops and the last one is a post-test loop. So what is pre-test and post-test loops? In a pre-test loop, in each of iteration, the condition is tested first. If it is true, the loop continues, otherwise the loop is terminated. It is vice versa in post-test loop whereby the loop continues and the condition is tested later. If condition is true, a new iteration is started, otherwise the loop is terminated. The below figures illustrate these two loop types.

3. Read File to Array

In C, it is easy to read file line by line. For example, if we have two lines of data in text file, we only use the following code to read data line by line.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char line[1000];
    int j=0;
    int content;
    FILE *infile;
    infile = fopen("test.txt", "r");
    while(fgets(line, sizeof line, infile)!=NULL)
    {
        printf("%s",line);
    }
    fclose(infile);
    return 0;
}
```

Lets say data in text file is:
I love to learn programming
in FKP, UMP

The output of the above code is:

```
I love to learn programming
in FKP, UMP
```

But to manipulate the data from file, we need to store the given data into array. How we going to do this? I have wrote a code to do this process in the next given code. However, this code only useful when we have the same data dimension. I think it is enough for us to know this implementation since to be an engineer, we only need to understand how to retrieve data (most probably in the same dimension).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *file;
    file = fopen("test.txt", "r");
    int a = 0;
    int row = 0, col = 0;
    char output[100][100];
    char c;
    while (c != EOF)
    {
        c = getc(file);
        //printf("\n\nLOOPS %d\n", a);
        if(c == '\n')
        {
            row++;
            col = 0;
            c = 'n';
            //printf("data = %c\n", c);
        }
        else if(c == ' ')
        {
            col++;
            c = 's';
            //printf("data = %c\n", c);
        }
        else if(c == '\t')
        {
            col++;
            c = 't';
            //printf("data = %c\n", c);
        }
        else
        {
            int b = c;
            if(b == -1)
            {
                 //printf("data = %c\n", c);
            }
```

```
            else
            {
                //printf("data = %c\n", c);
                output[row][col] = c;
                //printf("Masuk data = %c\n",  output[row][col]);
                //printf("Rows = %d\n", row);
                //printf("Columns = %d\n", col);
            }
        }
        a++;
    }
    printf("\n\nTotal Rows = %d\n", row+1);
    printf("Total Columns = %d\n\n", col+1);
    fclose(file);
    int i,j;
    for (i=0;i<row+1;i++)
    {
        for (j=0;j<col+1;j++)
        {
            //printf("data [%d][%d] = %c\t", i,j,output[i][j]);
        }
        //printf("\n");
    }
    return 0;
}
```

Exploration

Lets say we have the following text file:

0 2 3 4 9

1 7 4 8 1

What is the output of the above code? Is it all of the data store into array? Can we manipulate all of the data if it is store in array? Try to do some calculation to the given data and print all the data.

Based on the previous lesson, we found that we have trouble to read file to array if we have floating number. Today, we are going to explore more on reading file in floating point number to array. I have wrote the below code for computing the above mentioned problem.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
int main()
{
    char c, d, e, f;
    char str_d[10], str_e[10], str_f[10];
    float val;
    int row = 0, col = 0;
    int temp = 0;                    //use to control number of character use in this case 2.1
    float output[100][100];          //is equal to 3 character
    FILE *file;
    file = fopen("test.txt", "r");   //read file test.txt in the same directory with main.c
    while (c != EOF)
    {
        if (temp == 3)
        {
            temp = 0;
        }
        c = getc(file);
        if(c == '\n')
        {
            row++;
            col = 0;
        }
        else if(c == ' ')
        {
            col++;
        }
       else if(c == '\t')
        {
            col++;
        }
        else
        {
            int b = c;
            if(b == -1)
            {
            }
            else
            {
                if (temp == 0)
                {
                    d = c;
                    sprintf(str_d, "%c\0", d);       //change single char to string
                    //printf("%d\t%s\n",temp,str_d);
                }
                if (temp == 1)
                {
                    e = c;
                    sprintf(str_e, "%c\0", e);       //change single char to string
```

```
                    strcat(str_d, str_e);           //append str_e to str_d
                    //printf("%d\t%s\t%s\n",temp,str_e,str_d);
                }
                if (temp == 2)
                {
                    f = c;
                    sprintf(str_f, "%c\0", f);      //change single char to string
                    strcat(str_d, str_f);           //append str_f to str_d
                    val = atof(str_d);              //change string str_d to float val
                    //printf("%d\t%s\t%s\t%f\t[%d][%d]\n\n",temp,str_f,str_d,val,row,col);
                    output[row][col] = val;         //save text file data into array
                }
                temp++;
            }
        }
    }
    fclose(file);
    //following is for display array
    int i,j;
    for (i=0;i<row+1;i++)
    {
        for (j=0;j<col+1;j++)
        {
            printf("data [%d][%d] = %f\t", i,j,output[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Exploration

Lets say we have the following text file:

0.1	2.1	3.2	4.4	9.5
1.3	7.4	4.3	1.8	9.2

What is the output of the above code? Is it all of the data store into array? Can we manipulate all of the data if it is store in array? Try to do some calculation to the given data and print all the data. How about if we put more data? How about we change the data in text into 2 decimal points?

4. Pass Array to Function

```
#include <stdlib.h>
#include <stdio.h>

/* function declaration */
void printArray(int arr[], int size);
double getAverage(int arr[], int size);
int main()
{
    /* an int array with 5 elements */
    int balance[5] = {1000, 2, 3, 17, 50};
    double avg;

    /* pass pointer to the array as an argument */
    avg = getAverage( balance, 5 ) ;

    /* pass pointer to the array as an argument */
    printArray( balance, 5 ) ;

    /* output the returned value */
    printf( "\nAverage value is: %f ", avg );

    return 0;
}
double getAverage(int arr[], int size)
{

   int i;
   double avg;
   double sum = 0;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = sum / size;

   return avg;
}
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; ++i) {
      printf("%d\t",arr[i]);
    }

}
```

Before this, we have learned how to input and output various data type using C language for example `int`, `float`, `double`, and `char`. Some of you may notice that it is very difficult to handle many `int data`, many `float` data etc. In this case, we need array to group them into specific data type. For instance, if we have two integer, lets say `int age1` and `int age2`, we declared this data into `int age[2]`. This means that we have assigned two memory space (4 bytes x 2) of `int` to `age`.