生態模擬: 以C語言為例

Class 07 (2018/05/10)

Array (陣列)

7.1 What is an array?
7.2 Declaration and initialization of an array
7.3 How to use arrays
7.4 An application: sorting
7.5 Multidimensional arrays
7.6 Homework for using arrays

Function (函數)

7.7 An example of function and keywords
7.8 Separate compilation and makefile
7.9 Function declaration, definition, and call
7.10 Parameter and argument
7.11 Return value
7.12 Local and global variables
7.13 Homework for using functions

Takeshi Miki

三木　健（海洋研究所）

7.1 What is an array? (陣列)

Let's consider the situation when we need to record the score of the grade of 50 students.

```
int test1 = 80;
int test2 = 60;
int test3 = 92;
...
int test50 = 37;
```

Two many variables!!

```
ave_score = (1.0/50.0)*(test1 + test2 + test3 + test4 + .... + test50);
```

An array, e.g. test[50], can memory the multiple values of the same data type.

7.2 Declaration and initialization of an array

How to declare an array

構文(Syntax):

```
Array_Data_Type    Array_name [size] ;
```

```
int test[5];
```

Then, you can use five elements from index 0 to index 4.

```
test[0], test[1], test[2], test[3], test[4]
```

Very important!

You can NOT specify the maximum number of elements by a variable !!.

```
int num = 5;
int test[num];
```

7.2 Declaration and initialization of an array

How to initialize an array

構文(Syntax):

```
陣列的資料型態　陣列的名稱 [ 陣列的大小 ] = {數值1,數值2,... } ;
```

```
int test[5] = {80, 60, 22, 50, 75};
```

Or,

```
int test[] = {80, 60, 22, 50, 75};
```

You can specify the maximum number of elements by a macro.

```
#define NUM 5
...
int main(void) {
  int test[NUM] = {80, 60, 22, 50, 75};
```

7.3 How to use an array

Prepare a new C code file.

```
#include <stdio.h>
#define NUM 5

int main(void)
{
        int test[NUM];
        int j, k;

        printf("Input the scores of 5 students.\n");
        for(j = 0; j < NUM; j++) scanf("%d", &test[j]);

        for(k = 0; k < NUM; k++) {
                printf("The score of student no. %d is %d. \n", k + 1, test[k]);
        }

        return 0;
}
```

Please check what happens if you replace it with "for(k = 0; k <= NUM; k++) { ".

7.4 An application: sorting (“bubble sort”)

Prepare a new C code file.

```
#include <stdio.h>
#define NUM 5
int main(void)
{
        int test[NUM] = {22, 80, 57, 60, 50};
        int j, s, t, tmp;

        for(s = 0; s < NUM - 1; s++) {
                for(t = s + 1; t < NUM; t++) {
                        if(test[t] > test[s]) {
                                tmp = test[t];
                                test[t] = test[s];
                                test[s] = tmp;
                        }//end if
                }//end for t
        }//end for s
        for(j = 0; j < NUM; j++) printf(“No. %d’s score is %d.\n”, j+1, test[j]);

        return 0;
}
```

7.5 Multidimensional arrays

You can declare the multidimensional arrays as follows.

```
#include <stdio.h>
#define SUB 2
#define NUM 5
int main(void)
{
        int test[SUB][NUM];
        ......
        return 0;
}
```

You can initialize the multidimensional arrays as follows.

```
int test[2][5] = {
                {80,60, 22,50, 75}, {90, 55, 68, 72, 58}
};
```

```
int test[][5] = {
                {80,60, 22,50, 75}, {90, 55, 68, 72, 58}
};
```

We will learn a better way to declare arrays in a later week!

7.6 Homework for using arrays

Write a program code, which (1) asks the user to input 5 persons' grades (0-100) from the keyboard, and (2) shows the best, worst, and average grades in the display.

7.7 An example of function and keywords

Prepare a new C code file.

```
#include <stdio.h>
int max(int x, int y);

int main(void)
{
        int num1 = 10;
        int num2 = 5;
        int sum1;

        sum1 = max(num1, num2);
        printf("The maximum value is %d. \n", sum1);

        return 0;
}

int max(int x, int y)
{
        if (x > y) return x;
        else return y;
}
```

function declaration

'Parameter'

Type of 'return value'

'Argument'

function call

function definition

Function declaration before main function

```
int max(int x, int y);
```

Should put semicolon (;)

List of parameter type and name, separated by comma (,).

Function Name

Type of 'return value'

By functional declaration, you can put the function definition after the main function. Without function declaration, you need to write the function definition before main function, resulting in not-easily-readable source file.

Function definition

```
int max(int x, int y)
{
        if (x > y) return x;
        else return y;
}
```

Need to return value with the same type as the definition

You can use 'parameters' in the statements within a function

You need to use {} for the definition of a function

Another example is main function:

```
int main(void)
{
        ....
        return 0;
}
```

No parameters are necessary as input.

7.9 Function declaration, definition, and call

Prepare a new C code file.

```
#include <stdio.h>
int max(int x, int y);

int main(void)
{
        int num1 = 10;
        int num2 = 5;
        int sum1;

        sum1 = max(num1, num2);
        printf("The maximum value is %d. \n", sum1);

        return 0;
}

int max(int x, int y)
{
        if (x > y) return x;
        else return y;
}
```

function declaration

function call

function definition

7.10 Parameter and argument

Modify the code:

```
#include <stdio.h>
int max(int x, int y);

int main(void)
{
        int num1 = 10;
        int num2 = 5;
        int sum1;

        sum1 = max(num1, num2);
        printf("The maximum value is %d and num2 is %d. \n", sum1, num2);
        return 0;
}
int max(int x, int y)
{
        if (x > y) {
            y = x;
            return x;
        }
        else return y;
}
```

'Parameter'

'Argument'

重要！！

You cannot change the values of variables that are used as arguments. Only the values are given to arguments. 'y' is NOT identical to num2.

Return value can be empty

```
void wo_lei_le(void)
{
        printf("I'm tired.\n");
        return;
}
```

How to call this function

```
int main (void)
{
        wo_lei_le();
        return 0;
}
```

7.12 Local and global variables

Prepare a new C code file.

```
#include <stdio.h>
void func(void);
int a = 0;

int main(void)
{
        int b = 1;
        printf("The value of a is %d. \n", a);
        printf("The value of b is %d. \n", b);
        //printf("The value of c is %d. \n", c);
        func();
        return 0;
}

void func(void)
{
        int c = 2;
        a++;
        printf("The value of a is %d. \n", a);
        printf("The value of c is %d. \n", c);
}
```

A global variable, defined outside of any functions

A local variable within main()

You cannot use 'c' because it is a local variable, which is effective only within a function func().

A local variable within func()

I recommend NOT to change global variables, which are not used as parameters.

7.13 Homework for using functions

To make a function, double bmi(double height, double weight), and write a program that can calculate BMI from keyboard inputs.