# Computer Programming 1 Lab

**2020-10-22**

# Outline

- Array Basic
- 2D array
- Exercise

# Array Basic

## Declaration

```
T a[N];
```

- Declares `a` as an array object that consists of N contiguously allocated objects of type T.
- `T`: data type
- `a`: identifier (variable name)
- `N`: amount of elements

# Array Basic - Declaration

## Examples

```
int student_score[100];
float coordinate_x[10];
float coordinate_y[10];
```

# Array Basic - Initialization

## Initialization from brace-enclosed lists

- Declare and initialize at one time

```
int array1[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

- Declare and initialize all elements 0

```
int array2[10] = {0};
```

# Array Basic - Initialization

## Initialization from brace-enclosed lists

- First and second element is 10 and 2; and each of the following elements is zero

```
int array3[5] = {10, 2}; // myArray will stores 10, 2, 0, 0, 0
```

- Array size could be computed from initializer

```
int array4[] = {2, 4, 8};
```

# Array Basic - Initialization

## Initialization from strings (array of `char` type)

- `str` has type char[4] and holds 'a', 'b', 'c', '\0'
  **('\0' is terminating null character)**

```
char str[] = "abc";
```

- If size is known, the array will only take enough elements and ingore terminating character.

```
char str[3] = "abc"; // `str` has type char[3] and holds 'a', 'b', 'c'
```

# Array Basic - Accessing and Modification

- The elements of an array are numbered 0, …, N - 1, and may be accessed with the subscript operator [], as in a[0], …, a[N - 1].

```
int array[10];
for(int i = 0; i < 10; i++){
    array[i] = i;
}
array[8] = 0;
```

# Array Basic - Memory Concept

- Array in memory, e.g. `int array[5] = {1, 2, 3, 4, 5}`:

| Address | Value | Array Form |
|---|---|---|
| 0x00124400 | 1 | array[0] |
| 0x00124404 | 2 | array[1] |
| 0x00124408 | 3 | array[2] |
| 0x0012440C | 4 | array[3] |
| 0x00124410 | 5 | array[4] |

- Array is continuum in memory space.

# Input and Output

- How to assign and output values to an array.

```
int score[5];
for(int i = 0; i < 5; i++){
    scanf("%d", &score[i]);
}
for(int i = 0; i < 5; i++){
    printf("%d\n", score[i]);
}
```

# 2D Array

## Declaration

- `datatype name[ rowSize ][ columnSize ];`

## Example

```
float matrix[5][3];
int map[4][5];
```

# 2D Array - Initialization

- Declare and initialize at one time

```
int myArray1[3][2] = { {1, 2} , {3, 4} , {5, 6} };
// or int myArray1[3][2] = {1, 2, 3, 4, 5, 6}
```

- Declare and initialize all elements 0

```
int myArray[3][2] = {{0}, {0}, {0}};
```

# 2D Array - Memory Space

Take `int array[2][3] = {1, 2, 3, 4, 5, 6}` as an example:

| **array[0][0]** | **array[0][1]** | **array[0][2]** |
|---|---|---|
| array[1][0] | array[1][1] | array[1][2] |

| **Address** | **Value** | **Array Form** |
|---|---|---|
| 0x00124400 | 1 | array[0][0] |
| 0x00124404 | 2 | array[0][1] |
| 0x00124408 | 3 | array[0][2] |
| 0x0012440C | 4 | array[1][0] |

# Input and Output

- How to assign and output values to an array.

```
int map[4][5];
for( int i = 0; i < 4; i++) {
    for( int j = 0; j < 5; j++) {
        scanf(“%d”, &map[i][j]);
    }
}

for( int i = 0; i < 4; i++) {
    for( j = 0; j < 5; j++) {
        printf(“%d\n”, map[i][j]);
    }
}
```

# Exercise 5

- Please find out the sum of contiguous sub-array within a one-dimensional array of numbers which has the largest sum.
- For example: -2, -3, 4, -1, -2, 1, 5, -3
  The maximum sum is 7 = 4 + (-1) + (-2) + 1 + 5
- Input

```
-2 -3 4 -1 -2 1 5 -3
```

- Output

```
Maximum contiguous sum is 7
```

# Any Question?