

Lesson 6.1 Array Declaration.

In this lesson, we will discuss on one dimensional arrays. To do that, let's do a quick review of previous lessons.

REVIEW

What have you learnt so far:

Content Copyright Nanyang Technological University

2

From the previous lessons, you have learnt the following:

REVIEW

What have you learnt so far:

- Basic Sequential Programming (data, operators, simple I/O)

Content Copyright Nanyang Technological University

3

Basic Sequential Programming. We covered data, operators, simple Input, Output.

REVIEW

What have you learnt so far:

- Basic Sequential Programming (data, operators, simple I/O)
- Branching (if-else, switch, conditional operator)

Content Copyright Nanyang Technological University

4

Branching using if-else, switch, conditional operator.

REVIEW

What have you learnt so far:

- Basic Sequential Programming (data, operators, simple I/O)
- Branching (if-else, switch, conditional operator)
- Looping (for, while, do-while)

Content Copyright Nanyang Technological University

5

Looping using for, while, and do, while

REVIEW

What have you learnt so far:

- Basic Sequential Programming (data, operators, simple I/O)
- Branching (if-else, switch, conditional operator)
- Looping (for, while, do-while)
- Functions (function definition and call by value)

Content Copyright Nanyang Technological University

6

Functions where we learnt about function definition and call by value.

REVIEW

What have you learnt so far:

- Basic Sequential Programming (data, operators, simple I/O)
- Branching (if-else, switch, conditional operator)
- Looping (for, while, do-while)
- Functions (function definition and call by value)
- Pointers (call by reference for function communication)

Content Copyright Nanyang Technological University

7

Pointers where we learnt about call by reference for function communication.

REVIEW

- With **Pointers**, you may use **call by reference** to pass more than one value to the calling function.

Content Copyright Nanyang Technological University

8

With **Pointers**, you may use **call by reference** to pass more than one value to the calling function.

REVIEW

- With **Pointers**, you may use **call by reference** to pass more than one value to the calling function.
- Apart from using it to pass more than one parameter to the calling functions, pointers can also be used in other data structures such as **arrays**, **strings** and **structures**.

Content Copyright Nanyang Technological University

9

Apart from using pointers to pass more than one parameter to the calling functions, pointers can also be used in other data structures such as **arrays**, **strings** and **structures**.

WHY LEARNING ARRAYS?

- Most programming languages provide array data structure as built-in data structure.

Content Copyright Nanyang Technological University

10

Why do we need to learn arrays?

It is because most programming languages provide array data structure as built in data structure.

WHY LEARNING ARRAYS?

- Most programming languages provide array data structure as built-in data structure.
- An **array** is a list of values with the **same data type**. If not using array, you will need to define many variables instead of just one array variable.

Content Copyright Nanyang Technological University

11

An **array** is a list of values with the **same data type**. If not using array, you will need to define many variables instead of just one array variable.

WHY LEARNING ARRAYS?

Python provides the **list** structure, two major differences from array:

Python provides the list structure. List has two major differences from array.

WHY LEARNING ARRAYS?

Python provides the **list** structure, two major differences from array:

1. Arrays have only limited operations while lists have a large number of operations;

Arrays have only limited operations while lists have a large number of operations.

WHY LEARNING ARRAYS?

Python provides the **list** structure, two major differences from array:

1. Arrays have only limited operations while lists have a large number of operations;
2. Size of arrays cannot be changed while lists can grow and shrink.

Size of arrays cannot be changed while lists can grow and shrink.

ARRAYS

- Data types: **char, int, float**

Content Copyright Nanyang Technological University

15

In the previous lectures, we have discussed the various data types such as **character, integer, float**, etc.

ARRAYS

- Data types: **char, int, float**
- Define a variable of one of these types

Example:

num

5

Content Copyright Nanyang Technological University

16

When we define a variable of one of these types,

ARRAYS

- Data types: **char, int, float**
- Define a variable of one of these types
- Reserve a memory location for the variable

Example:



Content Copyright Nanyang Technological University

17

the computer will reserve a memory location for the variable.

ARRAYS

- Data types: **char, int, float**
- Define a variable of one of these types
- Reserve a memory location for the variable
- Only one value is stored for each variable at any one time

Example:



Content Copyright Nanyang Technological University

18

Only one value is stored for each variable at any one time.

ARRAYS

- Applications which require storing related data items under one variable name

Content Copyright Nanyang Technological University

19

However, there are applications which require storing related data items under one variable name.

ARRAYS

- Applications which require storing related data items under one variable name
- Example:
 - Exam marks for programming course

Content Copyright Nanyang Technological University

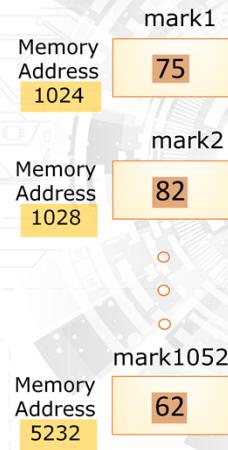
20

For example, if we have different items with similar nature, such as examination marks for the programming course,

ARRAYS

- Applications which require storing related data items under one variable name
- Example:
 - Exam marks for programming course

Example:



Content Copyright Nanyang Technological University

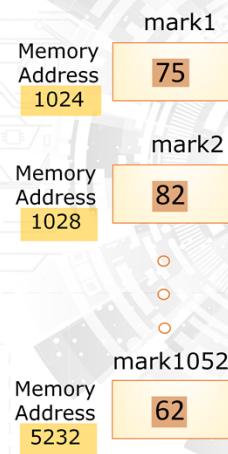
21

we might need to declare different variables such as mark 1, mark 2, etc. to represent the mark for each student.

ARRAYS

- Applications which require storing related data items under one variable name
- Example:
 - Exam marks for programming course
 - Cumbersome if number of students is very large

Example:



Content Copyright Nanyang Technological University

22

This is quite cumbersome if the number of students is very large.

ARRAYS

- Applications which require storing related data items under one variable name
- Example:
 - Exam marks for programming course
 - Cumbersome if number of students is very large
 - Declare a single variable called mark as an array

Example: mark



Content Copyright Nanyang Technological University

23

Instead, we can declare a single variable called mark as an array, and each element of the array can be used to store the mark for each student.

ARRAYS

- Applications which require storing related data items under one variable name
- Example:
 - Exam marks for programming course
 - Cumbersome if number of students is very large
 - Declare a single variable called mark as an array
 - In arrays, we can use a single **variable** to collect a **group** of data items of the **same data type**.

Example: mark

The diagram illustrates a one-dimensional array named 'mark'. It consists of a vertical column of four rectangular boxes, each containing a numerical value. The values are 75, 82, 62, and three empty circles (○). An orange border surrounds the entire column of boxes.

Content Copyright Nanyang Technological University 24

In arrays, we can use a single **variable** to collect a **group** of data items of the **same data type**.

In this lesson, we introduce this important topic on data structure that can be used to organize and store related data items. *Arrays* are used to store related data items of the same data type. In arrays, we can categorize them as one-dimensional arrays and multi-dimensional arrays.

In this lesson, we focus on discussing one-dimensional arrays, and in the next lesson, we will discuss multi-dimensional arrays.

TYPES OF VARIABLES

Data (or values) stored in variables are mainly in two forms:

- **Primitive Variables:**

values. They can be either simple or complex.

such as int, float and double, etc.

Structure,

- **Reference Variables:**

variables, array, pointer, etc.

Content Copyright Nanyang Technological University

25

As discussed before, there are mainly two types of variables: primitive type variables and reference (pointer) variables.

TYPES OF VARIABLES

Data (or values) stored in variables are mainly in two forms:

- **Primitive Variables:** Variables that are used to store **values**. They are mainly variables of primitive data types, such as int, float and char. Later on, you will learn **Structure**, which is used to store a record of data (values).
- **Reference Variables:**

Content Copyright Nanyang Technological University

26

Primitive Variables are variables that are used to store values. They are mainly variables of primitive data types, such as integer, float and character.

Later on, you will learn Structure, which is used to store a record of data (values).

TYPES OF VARIABLES

Data (or values) stored in variables are mainly in two forms:

- **Primitive Variables:** Variables that are used to store **values**. They are mainly variables of primitive data types, such as int, float and char. Later on, you will learn **Structure**, which is used to store a record of data (values).
- **Reference Variables:** Variables that are used to store **addresses**, such as **pointer variables**, **array variables**, **character string variables**.

Content Copyright Nanyang Technological University

27

Reference variables are variables that are used to store addresses, such as pointer variables, array variables, character string variables. The content stored in an array variable is an address, not the actual data.

WHAT IS AN ARRAY?

- An **array** is a **list of values** with the **same data type**.
Each value is stored at a specific, numbered position in the array.

Content Copyright Nanyang Technological University

28

An array is a list of values with the same data type.

Each value is stored at a specific, numbered position in the array.

WHAT IS AN ARRAY?

- An **array** is a **list of values** with the **same data type**.
Each value is stored at a specific, numbered position in the array.
- An array uses an **integer** called **index** to reference an element in the array.

Content Copyright Nanyang Technological University

29

An array uses an integer called index to reference an element in the array.

WHAT IS AN ARRAY?

- An **array** is a **list of values** with the **same data type**.
Each value is stored at a specific, numbered position in the array.
- An array uses an **integer** called **index** to reference an element in the array.
- The **size** of an array is **fixed** once it is created. Could the size be created dynamically? Yes by using malloc(), you will learn that later.

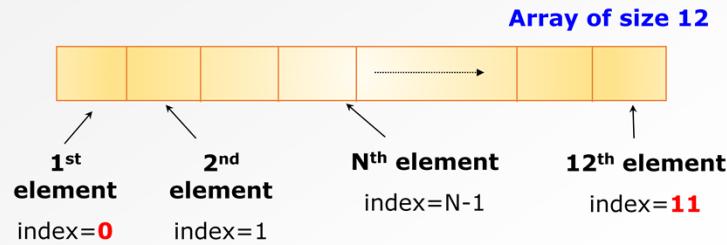
Content Copyright Nanyang Technological University

30

The size of an array is fixed once it is created. Could the size be created dynamically? Yes by using malloc, you will learn that later.

WHAT IS AN ARRAY?

- Index always starts with **0 (zero)**.



Content Copyright Nanyang Technological University

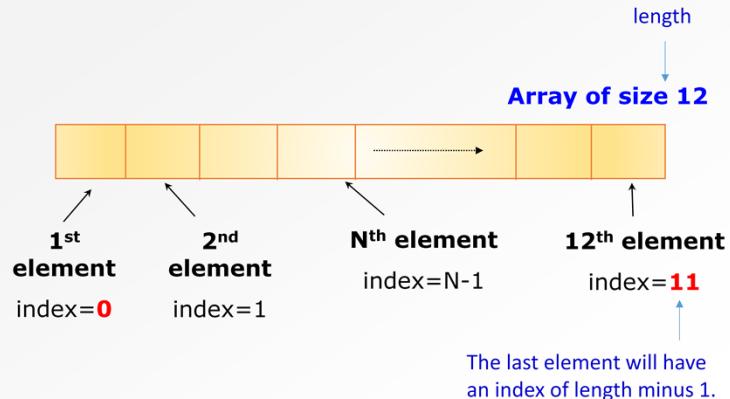
31

Index always starts with zero. Index is used to reference an element in the array.

In this example, there are 12 elements in the array so the array size is 12. The first element is referred by index 0.

WHAT IS AN ARRAY?

- Index always starts with **0 (zero)**.



Content Copyright Nanyang Technological University

32

The last element will have an index of length minus 1.

In this example, the last element, that is the 12th element has an index of 11.

ARRAY DECLARATION

The syntax for an array declaration is as follows:

The syntax for an array declaration is as follows:

ARRAY DECLARATION

The syntax for an array declaration is as follows:

`typeSpecifier`

Type specifier

ARRAY DECLARATION

The syntax for an array declaration is as follows:

`typeSpecifier arrayName`

Content Copyright Nanyang Technological University

35

Array name

ARRAY DECLARATION

The syntax for an array declaration is as follows:

```
typeSpecifier arrayName[arraySize];
```

Square bracket **array size** square bracket

ARRAY DECLARATION

The syntax for an array declaration is as follows:

```
typeSpecifier arrayName[arraySize];
```

where

typeSpecifier specifies the type of data to be stored in the array,

where **type specifier** specifies the type of data to be stored in the array, for example, character, integer, float etc.

ARRAY DECLARATION

The syntax for an array declaration is as follows:

`typeSpecifier arrayName[arraySize];`

where

`typeSpecifier` specifies the type of data to be stored in the array,

`arrayName` is the name given to the array, and

Content Copyright Nanyang Technological University

38

Array name is the name given to the array

ARRAY DECLARATION

The syntax for an array declaration is as follows:

```
typeSpecifier arrayName[arraySize];
```

where

typeSpecifier specifies the type of data to be stored in the array,

arrayName is the name given to the array, and

arraySize specifies the number of elements in the array.

and **array size** specifies the number of elements in the array.

ARRAY DECLARATION

The syntax for an array declaration is as follows:

`typeSpecifier arrayName[arraySize];`

where

`typeSpecifier` specifies the type of data to be stored in the array,

`arrayName` is the name given to the array, and

`arraySize` specifies the number of elements in the array.

For example, the declaration

`char name[12];`

Content Copyright Nanyang Technological University

40

For example, the declaration

character name square bracket 12 square bracket;

ARRAY DECLARATION

The syntax for an array declaration is as follows:

```
typeSpecifier arrayName[arraySize];
```

where

typeSpecifier specifies the type of data to be stored in the array,

arrayName is the name given to the array, and

arraySize specifies the number of elements in the array.

For example, the declaration

```
char name[12];
```

array of 12 elements

Content Copyright Nanyang Technological University

41

defines an array of 12 elements,

ARRAY DECLARATION

The syntax for an array declaration is as follows:

```
typeSpecifier arrayName[arraySize];
```

where

typeSpecifier specifies the type of data to be stored in the array,

arrayName is the name given to the array, and

arraySize specifies the number of elements in the array.

For example, the declaration

```
char name[12];
```



Content Copyright Nanyang Technological University

42

each element of the array stores data of type **character**. The elements are stored sequentially in memory. Each memory location in an array is accessed by a relative address called an *index* (or *subscript*).

ARRAY DECLARATION

Declaration of arrays without initialization:

Content Copyright Nanyang Technological University

43

Arrays can be declared without initialization.

For example,

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];
```

Content Copyright Nanyang Technological University

44

float sales square bracket 365 square bracket means

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */
```

Declaring an **array of 365 floats**.

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];
```

Character name 12 means

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */
```

Declaring an array of 12 characters.

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */  
int states[50];
```

Content Copyright Nanyang Technological University

48

Integer states 50 means

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */  
int states[50];        /* array of 50 integers */
```

Declaring an array of 50 integers

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */  
int states[50];        /* array of 50 integers */  
int *pointers[5];
```

Content Copyright Nanyang Technological University

50

Integer asterisk pointers 5 means

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */  
int states[50];        /* array of 50 integers */  
int *pointers[5];       /* array of 5 pointers to integers */
```

Content Copyright Nanyang Technological University

51

Declaring an array of 5 pointers to integers

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */  
char name[12];         /* array of 12 characters */  
int states[50];        /* array of 50 integers */  
int *pointers[5];       /* array of 5 pointers to integers */
```

The size of array must be **integer constant** or **constant expression** in declaration:

Content Copyright Nanyang Technological University

52

The size of array must be **integer constant** or **constant expression** in declaration.

ARRAY DECLARATION

Declaration of arrays without initialization:

```
float sales[365];      /* array of 365 floats */
char name[12];         /* array of 12 characters */
int states[50];        /* array of 50 integers */
int *pointers[5];       /* array of 5 pointers to integers */
```

The size of array must be **integer** constant or **constant expression** in declaration:

e.g. char name[i]; // i variable ==> illegal
 int states[i*6]; // i variable ==> illegal

Content Copyright Nanyang Technological University

53

Variables or expressions containing a variable cannot be used for the declaration of the size of the array.

The following declarations

character name [i]; and
integer states [i multiply by 6];
are illegal.

In this case, i is a variable, not a constant thus it is ~~not~~ an illegal declaration.

ARRAY DECLARATION

When an array is declared, some **consecutive memory** locations are allocated by the compiler for the whole array
(**2 or 4** bytes for an integer depending on machine)

When an array is declared, **consecutive memory locations** for the number of elements specified in the array are allocated by the compiler for the whole array.

ARRAY DECLARATION

When an array is declared, some **consecutive memory** locations are allocated by the compiler for the whole array (**2 or 4** bytes for an integer depending on machine). For example, if 2 bytes are used, then

```
int h[4];
```

Content Copyright Nanyang Technological University

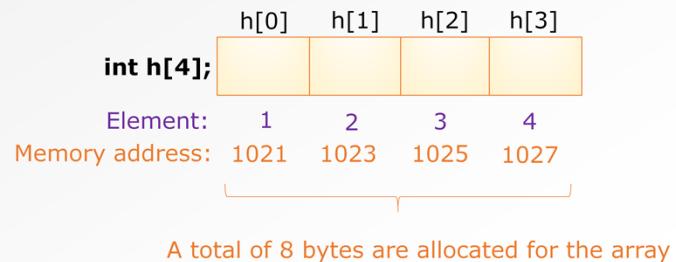
55

For example, if a system uses 2 bytes to store an integer, the declaration for the array

```
integer h[4];
```

ARRAY DECLARATION

When an array is declared, some consecutive memory locations are allocated by the compiler for the whole array (2 or 4 bytes for an integer depending on machine). For example, if 2 bytes are used, then



Content Copyright Nanyang Technological University

56

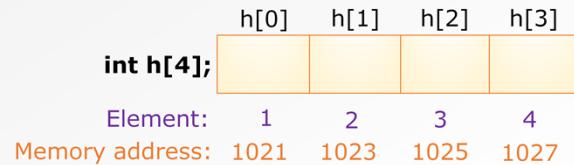
will result in a total of 8 bytes allocated for the array.

The total number of bytes of storage allocated to an array will depend on the size of the array and the type of data items.

The size of memory required can be calculated using the following equation:

ARRAY DECLARATION

When an array is declared, some **consecutive memory** locations are allocated by the compiler for the whole array (**2 or 4 bytes** for an integer depending on machine). For example, if 2 bytes are used, then



`total_memory = sizeof(type_specifier)*array_size;`

Content Copyright Nanyang Technological University

57

total memory equals size of (type specifier) multiply by array size;

where **size of** operator gives the size of the specified data type and **array size** is the total number of elements specified in the array.

SUMMARY

Content Copyright Nanyang Technological University

58

Let's do a quick summary of the important points that have been covered.
We have learnt:

SUMMARY

- Defining array

What is an array? Basically array is a list of values with the same data type.

SUMMARY

- Defining array
- Differences between array and Python list structure

The differences between array and Python list structure.

SUMMARY

- Defining array
- Differences between array and Python list structure
- Using index

Using index to reference an element in the array.

SUMMARY

- Defining array
- Differences between array and Python list structure
- Using index
- Declaration of arrays without initialisation

How to declare arrays without initialisation.

SUMMARY

- Defining array
- Differences between array and Python list structure
- Using index
- Declaration of arrays without initialisation
- Memory size calculation for array declared

Content Copyright Nanyang Technological University

63

And the memory size calculation for array declared.

In the next video, we will look into how to initialize the arrays declared.