Array in C++

Tong Yi

Table of contents

Motivation of Using an Array

- Declare and initialize a variable of primitive type.

```
1 int i = 5;
2 double weight = 6.7;
3 bool bFirst = false;
```

- Declare and instantiate an object of a class.

```
1 std::string greeting("Hello");
2 //same as std::string greeting = "Hello";
3 std::ifstream fin("data.txt");
```

- To read 100 integers from the keyboard or a file to calculate their sum, each input needs to be saved. How to save these integers? Should we use 100 integer variables, say `num1`, `num2`, $\cdots$, `num100`?
  - What if the number of integers is decided by user input?
  - array comes to the rescue!

Definition of Array

An array is a **collection** of **same type** elements residing in **consecutive** memory locations.

- an array of children
  - each child live in a room
  - the children's rooms are adjacent (no gap)

- an array of integers

1	5	2	8	0

- an array of chars

'A'	'B'	'C'	'D'	'F'

- an array of strings

"hi"	"hello"	"how are you?"

An Analog of Array

- Imagine a group of people living in adjacent rooms, with one person per room.
  - The room number of the first person is 0.
  - The room number of the second person is 1, and so on.
  - Suppose there are n person in this group. What is the room number of the last person?
- An array is a **collection** of **same type** elements residing in **consecutive** memory locations.
  - array name is the name of the group
  - room number is index.
  - The first index is 0. *arrayName*[0] is the first element of the array
  - The second index is 1. *arrayName*[1] is the second element of the array
  - Suppose the size of the array, aka, number of elements in the array, is n.
    - What is value of the last index?
    - What is the value of the last element in the array?

Declare an Array without Initialization

An array has three parts

- Type of each single element of the array
- array name

  An array name also implies the initial address of the array, ie, the address of the first element of the array.
- number of elements in the array
  - array name followed by a pair of square brackets [] enclosing a positive integer or a const int variable.
    Without const can cause error "variable-sized object may not be initialized"
  - The elements in the array are not initialized.

```
1 const int SIZE = 10; //initialize a const
      variable exactly once, like a dumbbell
2 int arr[SIZE];
```

The above statements can be written as

```
int arr[10];
```

Declare an Array and Initialize All its Elements

Declare and initialize all elements of the array in one statement.

- Array name followed by a pair of square bracket []. If the [] is empty, list all elements out, separated by comma (,) and enclosed in a pair of curly braces {}.

```
int arr[] = {1, 5, 2, 8, 0}; //✓
//int arr[5] = {1, 5, 2, 8, 0}; //also ok
```

- Declaration of array and initialize its elements – enclosing elements in {} – needs to be put in one statement.

```
1 int arr[5];
2 ~~arr = {1, 5, 2, 8, 0};~~ ✗
3 ~~arr[] = {1, 5, 2, 8, 0};~~ ✗
4 ~~arr[5] = {1, 5, 2, 8, 0};~~ ✗
```

Declare an Array and Initialize Some of its Elements

Declare and initialize some elements of the array in one statement.

- Initialize the first several elements if the size of the array is larger than the elements listed in the curly braces {}.

```
1 int arr[8] = {1, 5, 2, 8, 0};
2 //The array has 8 elements.
3 //The first five elements are 1, 5, 2, 8, 0,
     respectively,
4 //the remaining elements are set to be zero.
```

- If the size of the array is fewer than the number of elements in the curly braces {}, cause error "excess elements in array initializer".

```
1 int arr2[4] = {1, 5, 2, 8, 0}; ✗
```

Capacity vs. Size of an Array

In programming, capacity and size are distinct concepts.

- Imagine a hotel.
  - The capacity is the total number of rooms the hotel was built to hold.
  - The size is the number of rooms that are currently occupied by guests.
  - The capacity of the hotel remains constant, the size (the number of occupied rooms) may change.
- Suppose in a program, we enter no more than 10 integers from console and save them to an array.
  - The capacity of the array is 10.
  - The size of the array is the actual number of integers we've entered and saved. This number can vary.
    - For example, if we haven't entered any integers yet, the size is 0. If we've entered exactly 10 integers, the size is 10.
- size $\leq$ capacity.

Example of declaring and Initialize an Array

Suppose we have the following array of integers.

1	5	2	8	0

- Declare and initialize an array in the same statement.

  `arrayType arrayName[] = {values separated by comma};`

```
int arr[] = {1, 5, 2, 8, 0};
//int arr[5] = {1, 5, 2, 8, 0}; //also ok
```

- Declare an array and initialize later

  `arrayType arrayName[int_const];`

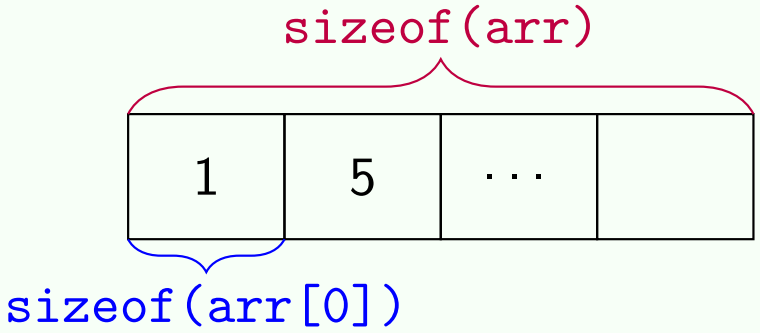
```
1 ~~int arr[];~~ ✗ Need to have an int inside []
```

```
1 int arr[5];
2 arr[0] = 1; //the index of an array starts from 0
3 arr[1] = 5;
4 arr[2] = 2;
5 arr[3] = 8;
6 arr[4] = 0;
```

Print Elements of an Integer Array

```
1  #include <iostream>
2  int main() {
3      const int SIZE = 10;
4      //const variable are all uppercase by convention
5      //without const cause error: variable-sized
           object may not be initialized
6      int arr[SIZE] = {1, 5, 2, 8, 0};
7      //the first five elements 1, 5, 2, 8, 0.
8      //the remaining elements are initialized to be 0.
9
10     for (int i = 0; i < SIZE; i++) {
11         std::cout << arr[i] << " ";
12     }
13     std::cout << std::endl;
14
15     return 0;
16 }
```

Calculate the Capacity of an Array if it Is Not Provided



- sizeof(arr[0]): memory size of the first element of the array, in bytes
- sizeof(arr): memory size of the whole array, in bytes
- Suppose array `arr` has three elements.
  - What is the relationship among sizeof(arr[0]), sizeof(arr[1]), and sizeof(arr[2])?
  - What is the relationship between sizeof(arr) and sizeof(arr[0]) + sizeof(arr[1]) + sizeof(arr[2])
  - How to calculate the number of elements of arr from sizeof(arr) and sizeof(arr[0])?

Calculate the Capacity of an Array if it Is Not Provided: Code

In C and C++, **sizeof**(arr) / **sizeof**(arr[0]) only works in the function where the array arr is declared.

```
#include <iostream>
int main() {
    int arr[] = {1, 2, 5, 8, 0};
    int size = sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < size; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Calculate the Capacity of an Array if it Is Not Provided: Caveat

When an array is passed as an argument to a function, it is not passed by value. Instead, it "decays" into a pointer to its first element. Will cover more in Chapter 5 and 7.

WRONG: arr in function header can only carry the initial address of array. That is, sizeof(arr) is NOT the actual memory size of arr.

```
1 void print(int arr[]) {
2     int size = sizeof(arr) / sizeof(arr[0]); ✗
3 }
```

CORRECT: pass array name and its size to funtion working on an array

```
1 void print(int arr[], int size) {
2     for (int i = 0; i < size; i++) {
3         std::cout << arr[i] << " ";
4     }
5     std::cout << std::endl;
```

Print Elements of a String Array in Reverse Order

"hi"	"hello"	"how are you?"

```
1 std::string greetings[] = {"hi", "hello", "how are
    you?"};
```

Write a code to print the elements of the array in reversed order.

Print Elements of a String Array in Reverse Order

"hi"	"hello"	"how are you?"

```
1 std::string greetings[] = {"hi", "hello", "how are
    you?"};
```

Write a code to print the elements of the array in reversed order.

```
1  #include <iostream>
2  #include <string>
3  int main() {
4      std::string greetings[] = {"hi", "hello", "how
           are you?"};
5      int size = ... //(1) TODO: You fill in the code
6
7      //(2) TODO: print elements of greetings in
           reversed order
8
9      return 0;
10 }
```

Array of Strings

Write a code to print the elements of the array in reversed order.

```
1  #include <iostream>
2  #include <string>
3  int main() {
4      std::string greetings[] = {"hi", "hello", "how
           are you?"};
5      int size = sizeof(greetings) / sizeof(greetings[0]);
6
7      for (int i = size-1; i >= 0; i--) {
8          std::cout << greetings[i] << std::endl;
9      }
10
11     return 0;
12 }
```

Create an Array of Uppercase Letters

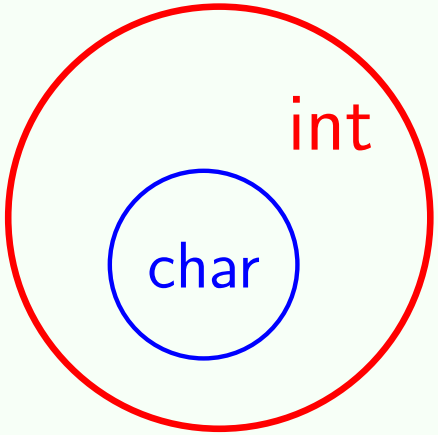
an array of chars

'A'	'B'	'C'	'D'	'F'

```
1 char letterGrades[] = {'A', 'B', 'C', 'D', 'F'};
```

Declare and initialize an array of chars with uppercase letters.

character	ASCII code
'A'	65
'B'	66
...	...
'Z'	90



- ASCII code is in [0, 127], a small set of int.
- What is the result of subtracting 'A' from 'B'?
- What is the result of 1 + 'A'?
- How to convert an ASCII code to the corresponding character?

Create an Array of Uppercase Letters: Array Initialization

Declare and initialize an array of chars with uppercase letters.

```
1  const int SIZE = 26;
2  char alphabet[SIZE];
3  for (int i = 0; i < SIZE; i++) {
4      alphabet[i] = ... //TODO: you fill in the code
5      //0 + 'A' is ASCII code of 'A',
6      //1 + 'A' is ASCII code of 'B',
7      //25 + 'A' is ASCII code of 'Z'.
8      //i + 'A' is ASCII code, an integer
9      //(char)(i + 'A') converts the ASCII code to the
           corresponding char.
10     //For example, (char)(1 + 'A') is (char)(ASCII
           code of 'B'), which is 'B'.
11 }
```

Create an Array of Uppercase Letters: Code

Initialize an array of chars with size 26. Put uppercase letters in this array and print the elements out.

```
1  #include <iostream>
2  int main() {
3      const int SIZE = 26;
4      char alphabet[SIZE];
5
6      for (int i = 0; i < SIZE; i++) {
7          alphabet[i] = (char)(i + 'A');
8      }
9
10     for (int i = 0; i < SIZE; i++) {
11         std::cout << alphabet[i] << " ";
12     }
13     std::cout << std::endl;
14     return 0;
15 }
```