



한국외국어대학교  
HANKUK UNIVERSITY OF FOREIGN STUDIES

# Introduction to Computers & Lab

# Lab 07

2021.04.15

Prof. Muhammad Bilal

TA. Sohee Jang



# Index

---

## 1. Review

- Arrays
- const
- Matrix

## 2. This week's Tasks + Hint



# Arrays

- Array is an aggregated data type that allows access to multiple variables of the same data type through a single identifier.
- Arrays are a series of variables of the same data type, and when combined with repetitive statements, continuous and repetitive values can be easily processed.



# Why are you using an array?

Let's consider the case of recording the scores of 30 students.  
If there is no array, 30 variables must be declared and allocated.

```
// allocate 30 integer variables (each with a different name)
```


```
int score1;
```

```
int score2;
```

```
int score3;
```

```
// ...
```

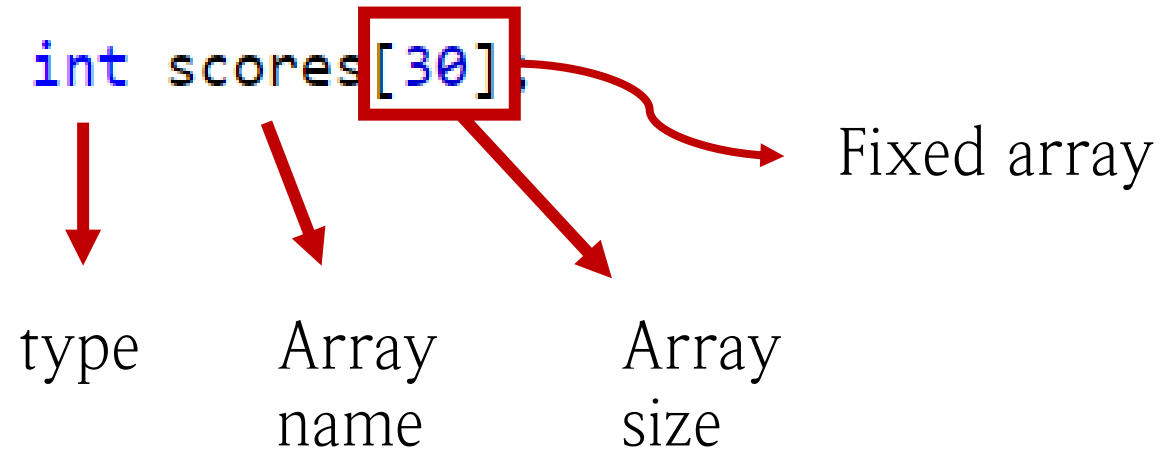
```
int score30;
```



```
int scores[30]; // allocate 30 integer variables in a fixed array
```



# Components of an array





# Array elements & Index

Must be integral type

Each variable in an array is called an element.

The element has no unique name. Instead, to access individual elements of an array, you can use the array name with a sub operator ([]) and a parameter called index that tells the compiler what elements it wants.

`score[0]` → Memory address starting from 0  
`score[1]`  
`score[2]`  
.  
.  
.  
`score[29]`



# Simple Array

```
#include <iostream>
using namespace std;

int main() {

    int prime[5];

    prime[0] = 2;
    prime[1] = 3;
    prime[2] = 5;
    prime[3] = 7;
    prime[4] = 11;

    cout << "The sum of primes is: "
    << prime[0]+prime[1]+prime[2]+prime[3]+prime[4] << endl;

    return 0;
}
```

main					
prime					
array					
	0	1	2	3	4
	int	int	int	int	int
	2	3	5	7	11

The sum of primes is: 28



# Array – initializer list

`int prime[5] = {2, 3, 5, 7, 11};`

[0] [1] [2] [3] [4]



`prime[0] = 2;`

main					
array					
0	1	2	3	4	
int	int	int	int	int	
7	4	5	0	0	

```
#include <iostream>
using namespace std;

int main() {

    int array[5] = {7, 4, 5};

    cout << array[0] << endl;
    cout << array[1] << endl;
    cout << array[2] << endl;
    cout << array[3] << endl;
    cout << array[4] << endl;

    return 0;
}
```





# Passing arrays to functions

However, if the array is large, copying the array can be a very expensive task, so C++ does not copy the array when it is delivered as a function. Instead, the actual arrangement is transferred.

This has side effects that can directly change the values of the array elements.



# Passing arrays to functions - example

```
before passValue : 1
after passValue : 1
before passArray : 2 3 5 7 11
after passArray : 11 7 5 3 2
```

```
#include <iostream>
using namespace std;
```

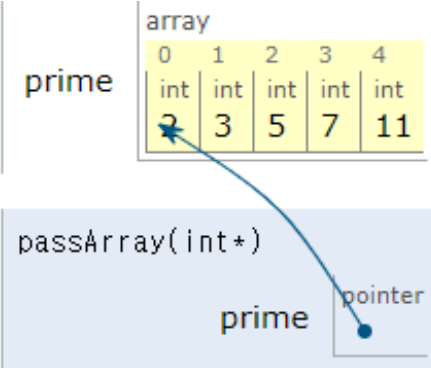
```
void passValue(int value){
    value = 99;
}
```

```
void passArray(int prime[5]){
    prime[0] = 11;
    prime[1] = 7;
    prime[2] = 5;
    prime[3] = 3;
    prime[4] = 2;
}
```

```
int main() {
    int value = 1;
    cout << "before passValue : " << value << endl;
    passValue(value);
    cout << "after passValue : " << value << endl;

    int prime[5] = {2,3,5,7,11};
    cout << "before passArray : " << prime[0] << " " << prime[1] << " "
    << prime[2] << " " << prime[3] << " " << prime[4] << endl;
    passArray(prime);
    cout << "after passArray : " << prime[0] << " " << prime[1] << " "
    << prime[2] << " " << prime[3] << " " << prime[4] << endl;

    return 0;
}
```





# Const

- To prevent the function from modifying passed array elements, you can make the array const.

```
void passArray(const int prime[5]){  
    prime[0] = 11;  
    prime[1] = 7;  
    prime[2] = 5;  
    prime[3] = 3;  
    prime[4] = 2;  
}
```



Compile error

# Matrix – array[row][column]

- To prevent the function from modifying passed array elements, you can make the array const.

```
#include <iostream>
using namespace std;

int main() {

    int array[3][5]=
    {
        {1,2,3,4,5},
        {6,7,8,9,10},
        {11,12,13,14,15}
    };

    return 0;
}
```



Column						
R O W	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	
		?				
				?		



## Task 1 : grade

---

Defines an array and uses an iterative structure to initialize the values of the array elements into random numbers. (grade 0~100)

★ define SIZE 5:

★ Hint. `grade[i] = rand() % 101;`



## Task 2 : average

---

Write a program that determines the average student's grades.  
The size of the array depends on the input. -> initialize SIZE 100

★  $\text{average} = \text{sum} / \text{STUDENTS}$



## Task 3 : Passing arrays to functions

---

Write a code that passes the array to the function's factor.  
Try using the code covered during the class.

- ★ Define the following functions:  
    void passValue(int value);  
    void passArray(int prime[5]);



## Task 4 : 9X9 Array

Write a code that outputs a multiplication table.

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

★ Use “\t”