Parameter Passing by value And by reference

Real life Scenario 01

- Suppose your laptop battery is damaged and you want to buy a new battery.

Real life Scenario 01

- You take your laptop with you to Shop Center and the shop manager gives you a new battery according to your laptop.

Real life Scenario 01

- You bring the battery home, replace the old battery with the new battery and everything starts working fine.

Real life Scenario 02

- Now, again suppose your laptop battery is damaged and you want to buy a new battery.

Real life Scenario 02

- Now, in this scenario you just call the Shop Center and tell them your house address and they come and fix your laptop.

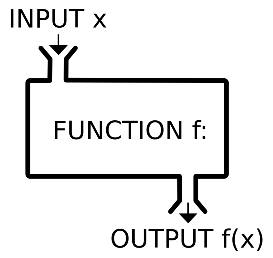
In C++

- Similarly, in C++, we can pass the complete variables to the functions (Pass by Value) or we can just pass the address of the variable (Pass by Reference).

In C++

- Previously, we have already seen how we can pass the complete variables to the functions (Pass by Value).

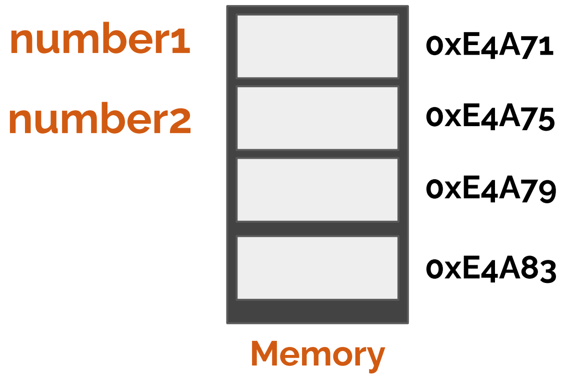
Review: Function in C++



Previously, we have written the function of Addition.

```
int addition(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Variables: Passing by Value

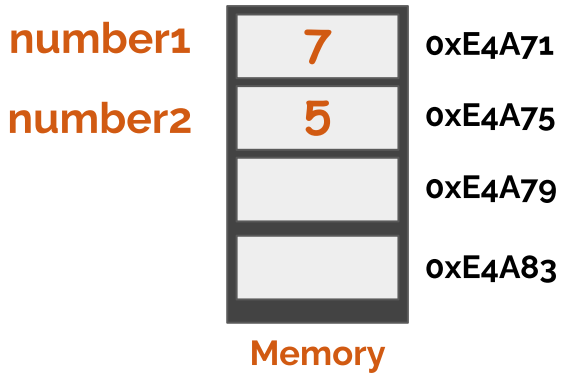


```
int addition(int num1, int num2)
{
     int sum = num1 + num2;
     return sum;
}
```

Function Call

```
main(){
     int number1, number2, result;
     cout << "Enter First Number: ";
     cin >> number1;
     cout << "Enter Second Number: ";
     cin >> number2;
     result = addition(number1, number2);
     cout << "Sum is: " << result;
}
```

Variables: Passing by Value

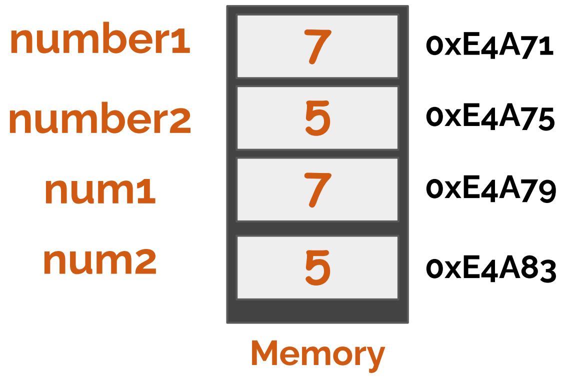


```
int addition(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Function Call

```
main(){
    int number1, number2, result;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    result = addition(number1, number2);
    cout << "Sum is: " << result;
}
```

Variables: Passing by Value

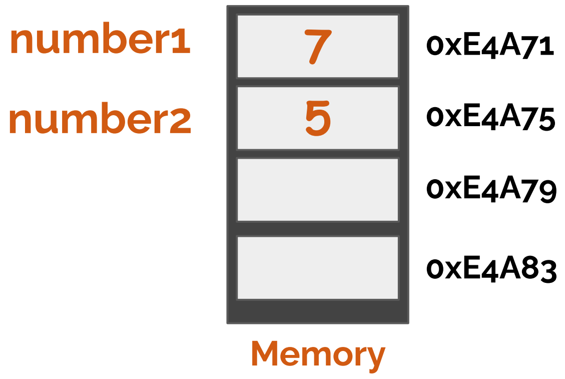


```cpp
int addition(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Function Call

```cpp
main(){
    int number1, number2, result;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    result = addition(number1, number2);
    cout << "Sum is: " << result;
}
```

Variables: Passing by Value



After the function call, the local variables of the addition function will be destroyed and only the variables of the main function will remain.

```
main(){
    int number1, number2, result;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    result = addition(number1, number2);
    cout << "Sum is: " << result;
}
```

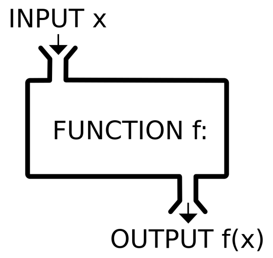
```
int addition(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Function Call

In C++

- Let's see how we can just pass the address of the variable (Pass by Reference).

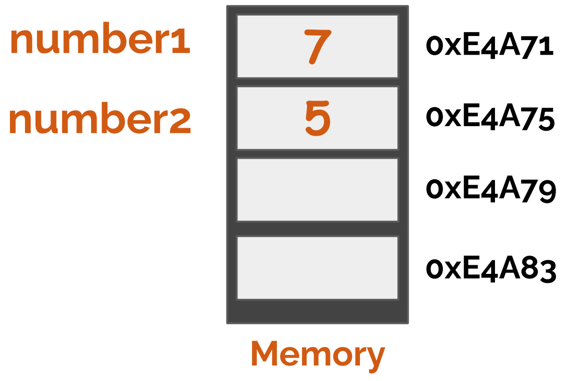
Function in C++



Now, the Addition function is only receiving the address of the variables.

```
int addition(int &num1, int &num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Variables: Passing by Reference

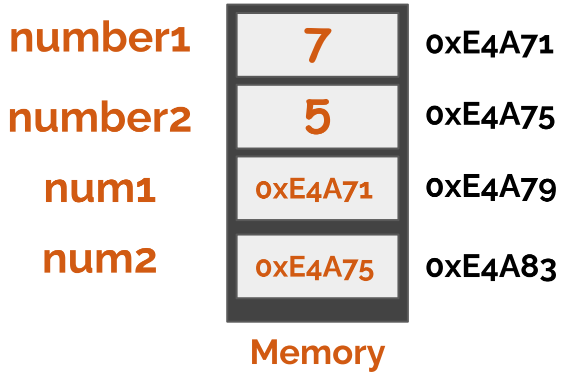


```cpp
int addition(int &num1, int &num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Function Call

```cpp
main(){
    int number1, number2, result;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    result = addition(number1, number2);
    cout << "Sum is: " << result;
}
```

Variables: Passing by Reference



```cpp
int addition(int &num1, int &num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Function Call

```cpp
main(){
    int number1, number2, result;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    result = addition(number1, number2);
    cout << "Sum is: " << result;
}
```

Variable passing by Value VS by Reference

- Both functions did the same job, but what was the benefit of passing the variable by reference?

```
int addition(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Pass by Value

```
int addition(int &num1, int &num2)
{
    int sum = num1 + num2;
    return sum;
}
```

Pass by Reference

Variable passing by Reference

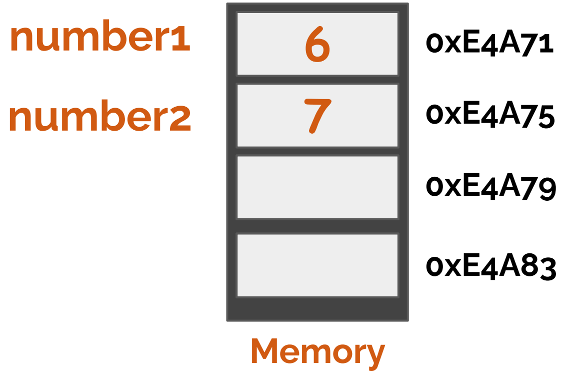
- If we want to return multiple values from the function then we must pass the parameters by reference.

Variable passing by Reference

- Let's make our own Swap function by passing the parameters by reference.

```
void swapped(int &num1, int &num2)
{
     int temp = num1;
     num1 = num2;
     num2 = temp;
}
```

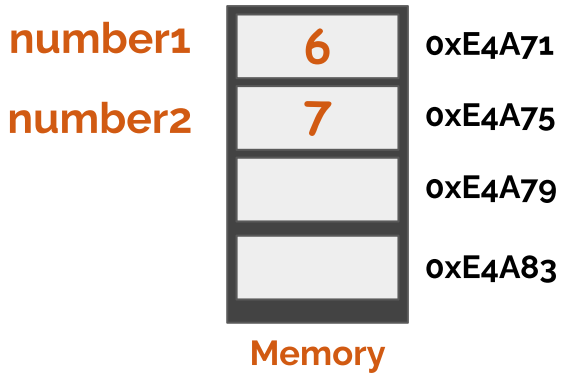
Variable passing by Reference

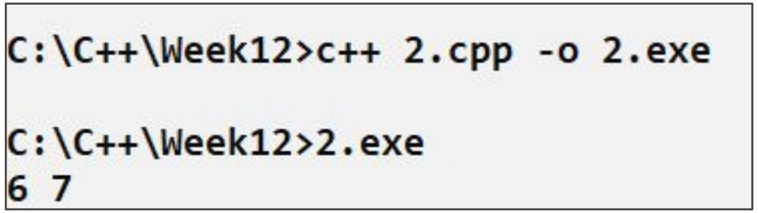


```
void swapped(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

```
main()
{
    int number1 = 6, number2 = 7;
    cout << number1 << " " << number2 << endl;
    swapped(number1, number2);
    cout << number1 << " " << number2;
}
```

Variable passing by Reference



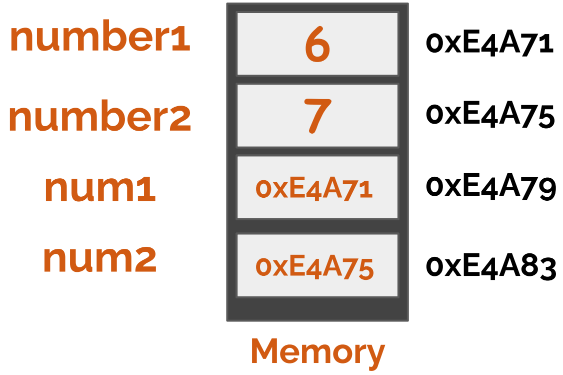


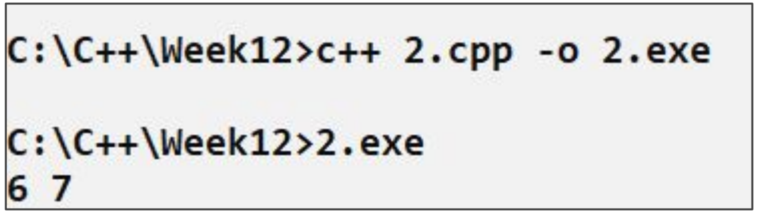
```
C:\C++\Week12>c++ 2.cpp -o 2.exe

C:\C++\Week12>2.exe
6 7
```

```
void swapped(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

```
main()
{
    int number1 = 6, number2 = 7;
    cout << number1 << " " << number2 << endl;
    swapped(number1, number2);
    cout << number1 << " " << number2;
}
```

Variable passing by Reference



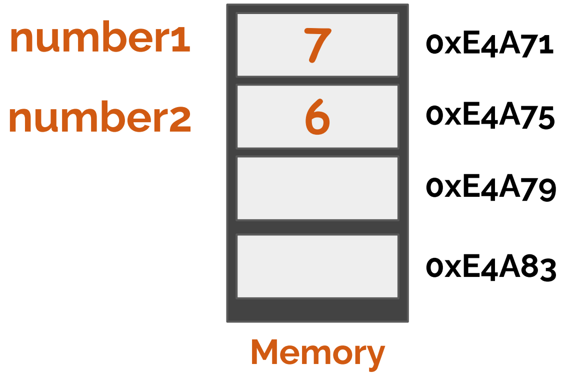


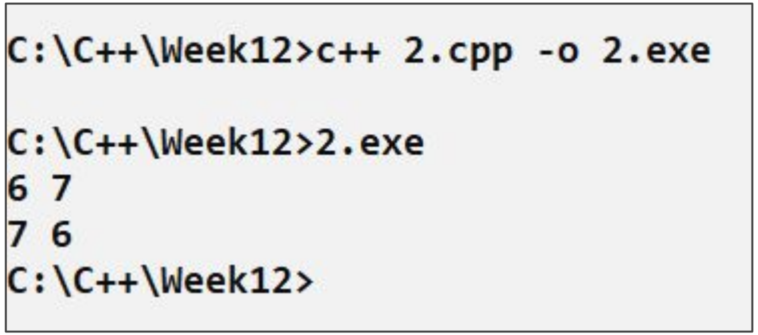
```
C:\C++\Week12>c++ 2.cpp -o 2.exe

C:\C++\Week12>2.exe
6 7
```

```
void swapped(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

```
main()
{
    int number1 = 6, number2 = 7;
    cout << number1 << " " << number2 << endl;
    swapped(number1, number2);
    cout << number1 << " " << number2;
}
```

Variable passing by Reference





```
C:\C++\Week12>c++ 2.cpp -o 2.exe

C:\C++\Week12>2.exe
6 7
7 6
C:\C++\Week12>
```

```
void swapped(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

```
main()
{
    int number1 = 6, number2 = 7;
    cout << number1 << " " << number2 << endl;
    swapped(number1, number2);
    cout << number1 << " " << number2;
}
```

Real life Scenario 03

- Now, suppose your room *AC* breaks down and it is not working.

Real life Scenario 03

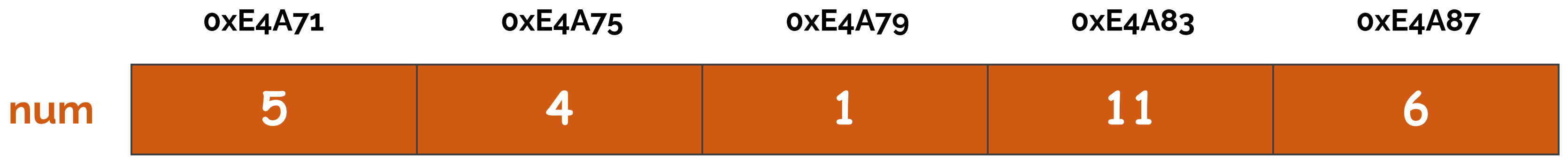
- Since it is a bigger item, therefore you do not take your AC to the Electrician.

Real life Scenario 03

- You call the electrician and tell him the address of your house so he can come and repair your AC.

In C++

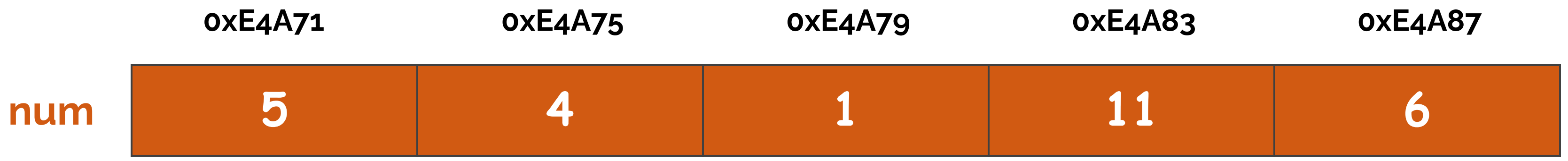
- Since, in C++, arrays take large memory space and they are difficult to pass to the functions therefore we only pass their address i.e. we pass them by Reference.



Arrays: Passing by Reference

Let's see first what happens if we just cout the name of the array.

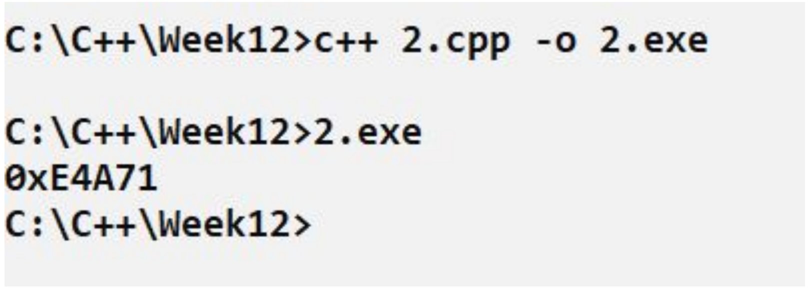
```
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    cout << num;
}
```



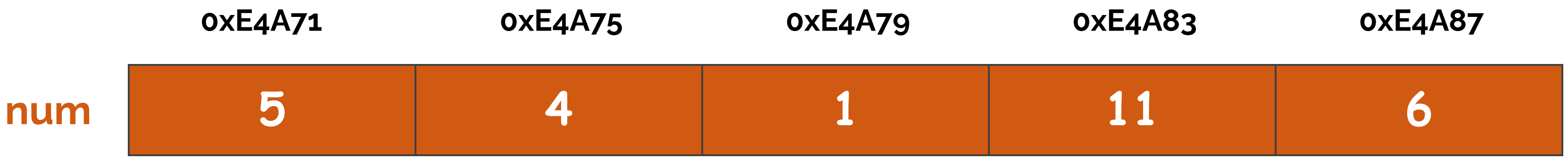
Arrays: Passing by Reference

Let's see first what happens if we just cout the name of the array.

```cpp
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    cout << num;
}
```



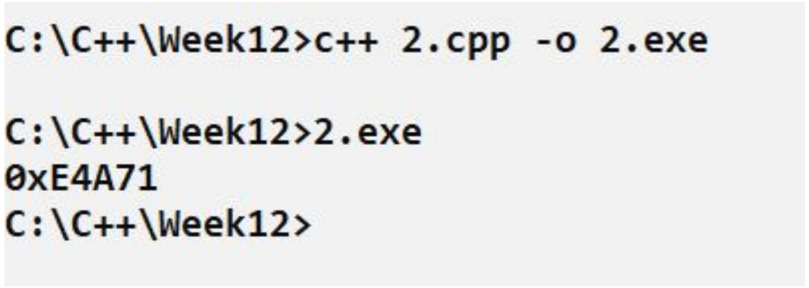
```
C:\C++\Week12>c++ 2.cpp -o 2.exe

C:\C++\Week12>2.exe
0xE4A71
C:\C++\Week12>
```



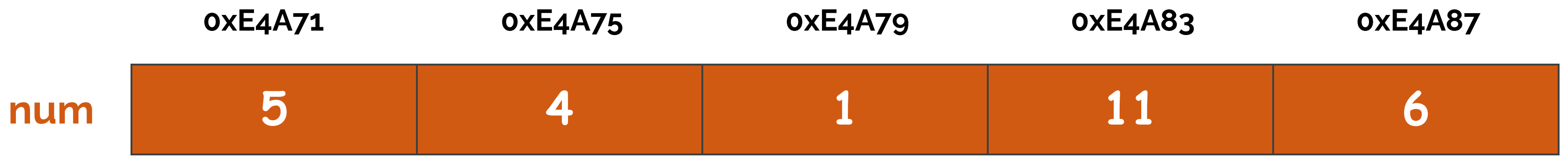
Arrays: Passing by Reference

It means if we just write the name of the array then we get the starting address of the array.

```cpp
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    cout << num;
}
```



```
C:\C++\Week12>c++ 2.cpp -o 2.exe

C:\C++\Week12>2.exe
0xE4A71
C:\C++\Week12>
```



Arrays: Passing by Reference

Now, lets print this array after passing to printArray(int arr[], int size) function.

```cpp
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    printArray(num, 5);
}
```

```cpp
void printArray(int arr[], int size)
{
    for(int x = 0; x < size; x++)
    {
        cout << arr[x] << " ";
    }
}
```

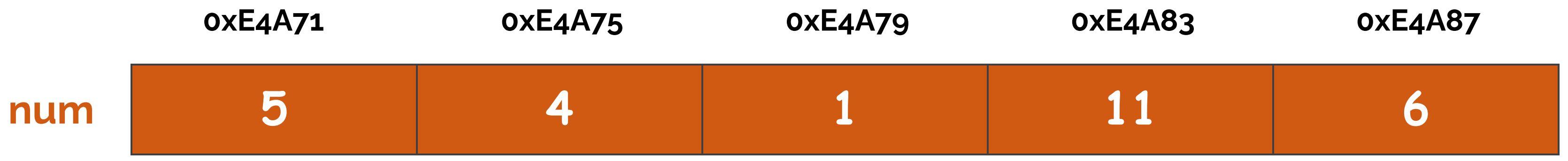
	0xE4A71	0xE4A75	0xE4A79	0xE4A83	0xE4A87
num	5	4	1	11	6

Arrays: Passing by Reference

We just passed the starting address of the array to the function instead of passing the complete array.

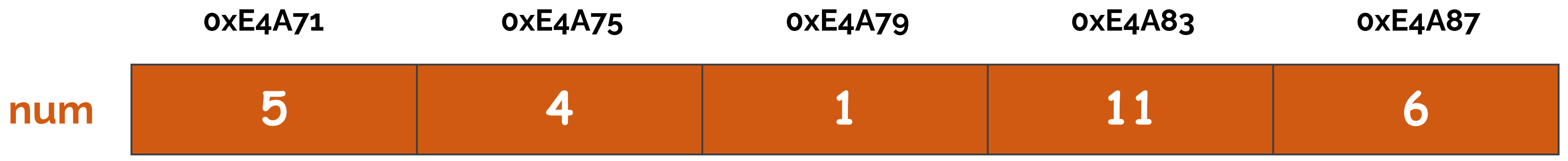
```
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    printArray(num, 5);
}
```

```
void printArray(int arr[], int size)
{
    for(int x = 0; x < size; x++)
    {
        cout << arr[x] << " ";
    }
}
```



Arrays: Passing by Reference

If we change something in the array (in the function), its gets changed in the main function as well.

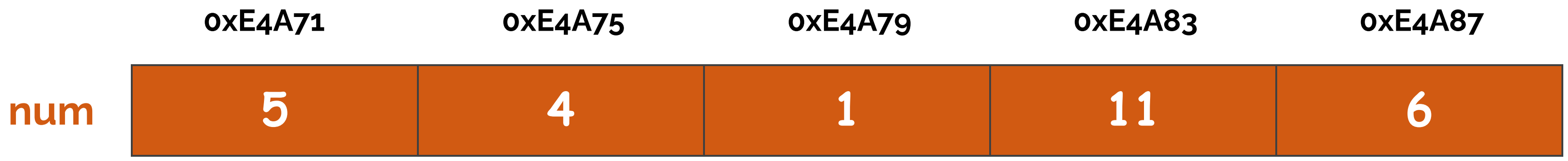


Arrays: Passing by Reference

If we change something in the array (in the function), its gets changed in the main function as well.

```
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    changeArray(num, 5);
}
```

```
void changeArray(int arr[], int size)
{
    for(int x = 0; x < size; x++)
    {
        arr[x] = x;
    }
}
```

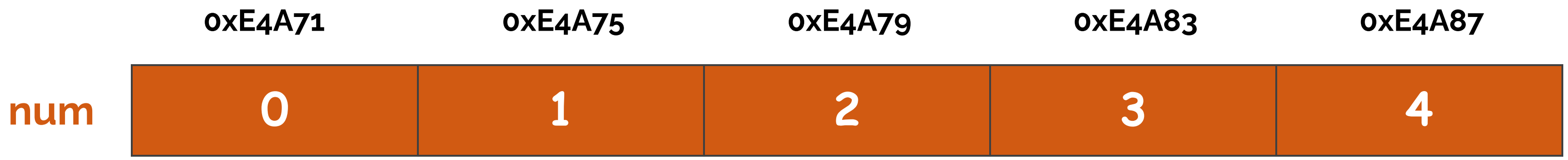


Arrays: Passing by Reference

If we change something in the array (in the function), its gets changed in the main function as well.

```
#include <iostream>
using namespace std;
main(){
    int num[5] = {5, 4, 1, 11, 6};
    changeArray(num, 5);
}
```

```
void changeArray(int arr[], int size)
{
    for(int x = 0; x < size; x++)
    {
        arr[x] = x;
    }
}
```



Arrays: Passing by Reference

Similarly, we can pass the 2D array by reference to any function as well.

```
main()
{
    int cars[5][5] = {{10, 7, 12, 10, 4},
                      {18, 11, 15, 17, 2},
                      {23, 19, 12, 16, 14},
                      {7, 12, 16, 0, 2},
                      {3, 5, 6, 2, 1}};
    printArray(cars, 5);
}
```

```
void printArray(int arr[][5], int rowSize)
{
    for (int row = 0; row < rowSize; row++)
    {
        for (int col = 0; col < 5; col++)
        {
            cout << arr[row][col] << "\t";
        }
        cout << endl;
    }
}
```

Arrays: Passing by Reference

It is mandatory to tell the column size of the 2D array when passed to the function. Row size can vary.

```
main()
{
    int cars[5][5] = {{10, 7, 12, 10, 4},
                      {18, 11, 15, 17, 2},
                      {23, 19, 12, 16, 14},
                      {7, 12, 16, 0, 2},
                      {3, 5, 6, 2, 1}};
    printArray(cars, 5);
}
```

```
void printArray(int arr[][5], int rowSize)
{
    for (int row = 0; row < rowSize; row++)
    {
        for (int col = 0; col < 5; col++)
        {
            cout << arr[row][col] << "\t";
        }
        cout << endl;
    }
}
```

Learning Objective

Differentiate between passing the parameters **by value** and **by reference**.

Conclusion

Pass by Value	Pass by Reference
Mechanism of copying function parameter value to another variable	Mechanism of passing the actual parameters to the function
Changes made inside the function are not reflected in the original value	Changes made inside the function are reflected in the original value
Makes a copy of the actual parameter	Address of the actual parameter passes to the function
Function gets a copy of the actual content	Function accesses the original variable's content
Requires more memory	Requires less memory
Requires more time as it involves copying values	Requires a less amount of time as there is no copying

Self Assessment:

1. What is the **output** of the following code?

```
main()
{
    int i = 10, j = 20;
    swapThemByVal(i, j);
    cout << i << "  " << j << endl;
    swapThemByRef(i, j);
    cout << i << "  " << j << endl;
}
```

```
void swapThemByVal(int num1, int num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

```
void swapThemByRef(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

Self Assessment: (Video Profile Activity)

2. You are stuck in a 3-storey car parking lot. Your task is to exit the car park using only the staircases. Exit is always at the bottom right of the ground floor and you are always on the 2nd floor.

Create a function that takes a two-dimensional array as input where:

1. Free car parking spaces are represented by a 0
2. Staircases are represented by a 1
3. Your starting position is represented by a 2 and can be at any level of the car park.
4. Exit is always at the bottom right of the ground floor.
5. You must use the staircases (1) to go down a level.
6. Each floor will have only one staircase apart from the ground floor which will not have any staircases.

Self Assessment:

Your Task is to display the quickest route out of the car park.

Test Cases:

Input	Output	Explanation
parking_exit([[1, 0, 0, 0, 2], [0, 0, 0, 0, 1], [0, 0, 0, 0, 0],])	Left: 4 Down: 1 Right: 4 Down: 1	// Starting from 2, move to left 4 times = Left: 4 // Go down from stairs 1 step = Down: 1 // Move to right 4 times to exit from right bottom corner = Right 4 // Go down from stairs 1 step = Down: 1

Self Assessment:

Your Task is to display the quickest route out of the car park.

Test Cases:

Input	Output	Explanation

parking_exit(
[
  [2, 0, 0, 1, 0],
  [0, 0, 0, 1, 0],
  [0, 0, 0, 0, 0]
])
	Right: 3
	Down: 2
	Right: 1
		// Starting from 2, move to right 3 times = "R3"
		// Go down from stairs 2 steps = "D2"
		// Move to right 1 step to exit from right bottom corner = "R1"