한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Introduction to Computers & Lab

Lab 10

2021.05.13

Prof. Muhammad Bilal
TA. Sohee Jang

Index

Pointer and Array

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

* Similarities between pointers and fixed arrays

```
//declare a fixed array of 5 integers
int array[5] = {9, 7, 5, 3, 1};
```

The above is an array of five integers, but the compiler has an array of int[5] variables. Each has values of array[0], array[1], and array[4]. But what value does the array itself have?

=> An array variable has the address of the first element of the array as if it were a pointer.

Pointer & Array Example


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

int main() {
  int array[5] = {9, 7, 5, 3, 1};

  cout << "The array has address: " << array << endl;
  cout << "Element 0 has address: " << &array[0] << endl;

  return 0;
}
```

```
The array has address: 0xfff000bc0
Element 0 has address: 0xfff000bc0
```

-> array : int[5]
-> pointer : int*

Pointer and Array

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

* Differences between pointers and fixed arrays

There are several instances of differences between fixed arrays and pointers.

The main difference occurs when the size of() operator is used. In a fixed array, the sizof() operator returns the total size. The sizeof() operator returns the size of the memory address in bytes when used on the (array length * element size) pointer.

The second difference occurs when the address operator & is used. Taking the pointer's address yields the memory address of the pointer variable. Selecting an array address returns the pointer to the entire array. The pointer also points to the first element of the array, but has different type information.

Pointer & Array Example


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

int main() {
  int array[5] = {9, 7, 5, 3, 1};
  cout << sizeof(array) << endl;

  int *ptr = array;
  cout << sizeof(ptr) << endl;

  return 0;
}
```

```
20
8
```

-> will print sizeof(int) * array length

-> will print the size of a pointer

A fixed array knows the length to which the array points, but does not know the pointer to the array.

Revisiting to functions

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

void printSize(int *array){
  cout << sizeof(array) << endl;
}

int main() {
  int array[] = {1,1,2,3,5,8,13,21};
  cout << sizeof(array) << endl;
  printSize(array);

  return 0;
}
```

```
32
8
```

=

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

void printSize(int array[]){
  cout << sizeof(array) << endl;
}

int main() {
  int array[] = {1,1,2,3,5,8,13,21};
  cout << sizeof(array) << endl;
  printSize(array);

  return 0;
}
```

-> It is recommended to use pointer syntax.

Pointer arithmetic I

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

The C++ language allows you to add or subtract integers from pointers. If ptr refers to an integer, ptr+1 is the integer address of the memory after ptr. ptr-1 is the address of an integer prior to ptr.

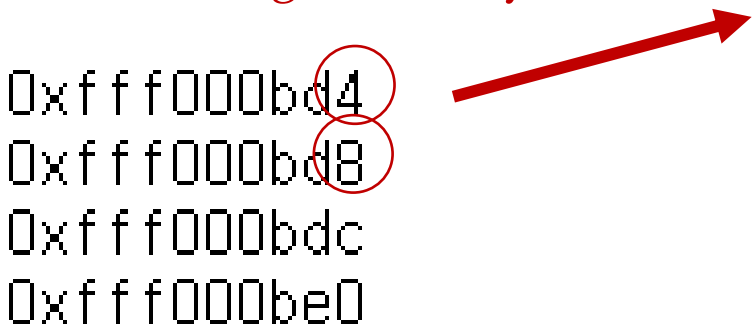
```
#include <iostream>
using namespace std;

int main() {
  int value = 7;
  int *ptr = &value;

  cout << ptr << endl;
  cout << ptr + 1 << endl;
  cout << ptr + 2 << endl;
  cout << ptr + 3 << endl;

  return 0;
}
```

-> The printed addresses differ by 4 bytes each. (The computer tested is an environment with an integer of 4 bytes).)



Pointer arithmetic II

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

int main() {
  int array[5] = {9,7,5,3,1};

  cout << &array[1] << endl;
  cout << array + 1 << endl;
  cout << array[1] << endl;
  cout << *(array + 1) << endl;

  return 0;
}
```

```
0xfff000bc4
0xfff000bc4
7
7
```

array[n] = *(array + n)

Thus, adding 1 to an array indicates that it refers to the second element of the array (index 1).

Const pointer I

The pointer itself can be made a constant. A constant pointer is a pointer that cannot change the address it points to after initialization.

To declare a constant pointer, use the const keyword after the datatype.

```
int value = 5;
int* const ptr = &value;
```

```
int value1 = 5;
int value2 = 6;

int* const ptr = &value1;
ptr = &value2;
```

Constant pointers always refer to the same address.

Const pointer II

Since the pointer is only a constant, and the variable it points to is not a constant, it is possible to reverse-reference the pointer to change the value.

```
int value = 5;
int* const ptr = &value;
*ptr = 6;
```

Memory Allocation

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Static
memory allocation

Auto
memory allocation

★

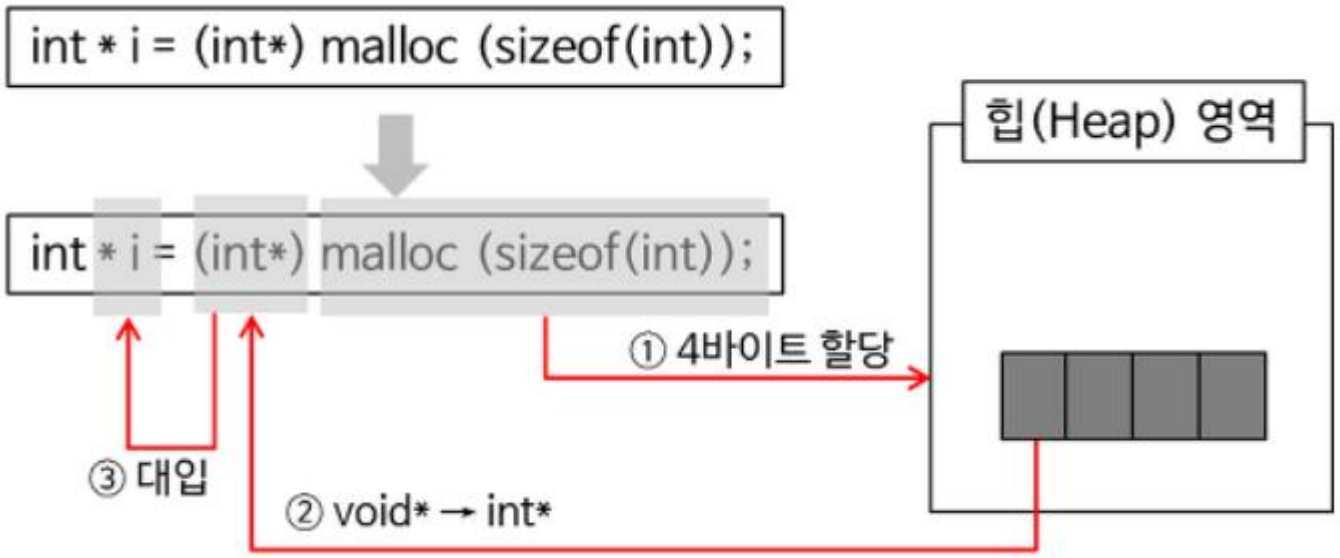
Dynamic
memory allocation

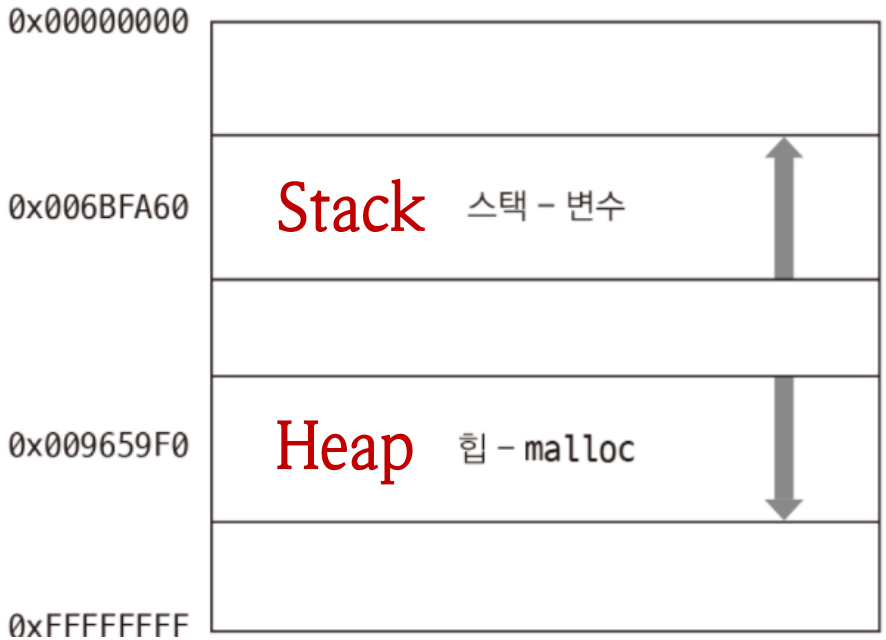
-> Dynamic memory allocation is a method of requesting the operating system for memory required during program execution.

Stack

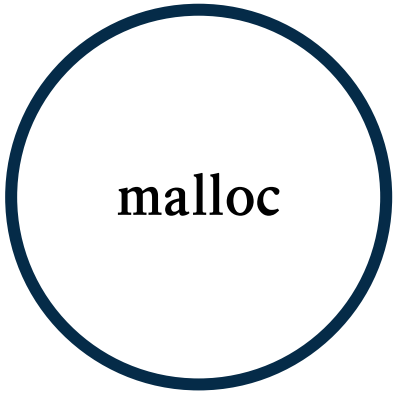
Heap

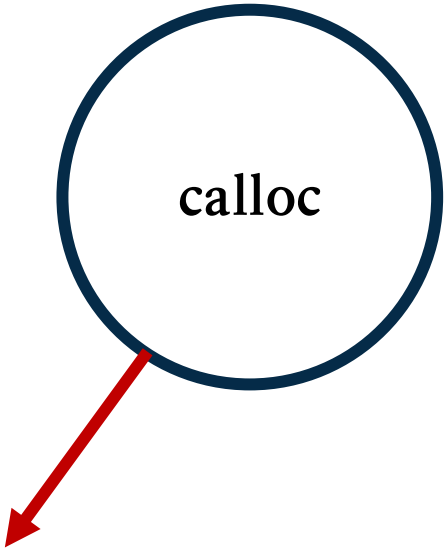
Memory Allocation



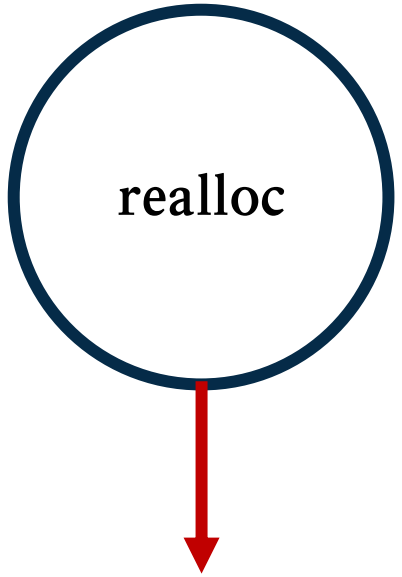


Memory Allocation





It functions similar to a malloc function, but is characterized by zero initialization to the allocated space.



Change the size of the space already allocated.

Memory Allocation Example

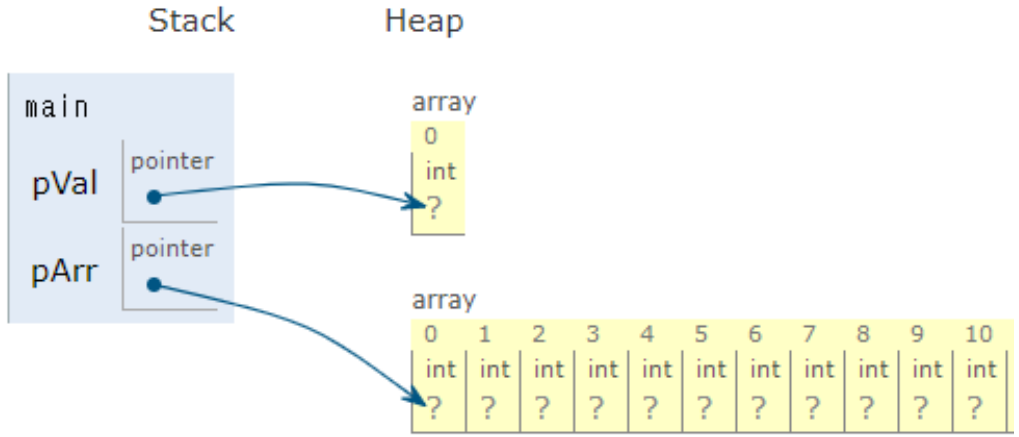
```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main() {

  int* pVal = (int*)malloc(sizeof(int));
  int* pArr = (int*)malloc(sizeof(int) * 1024);

  free(pVal);
  free(pArr);

  return 0;
}
```



Since the malloc function returns the address value to void pointer (void*), it must transform the type.

Task 1 : Swap

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Use pointers to create swap function functions

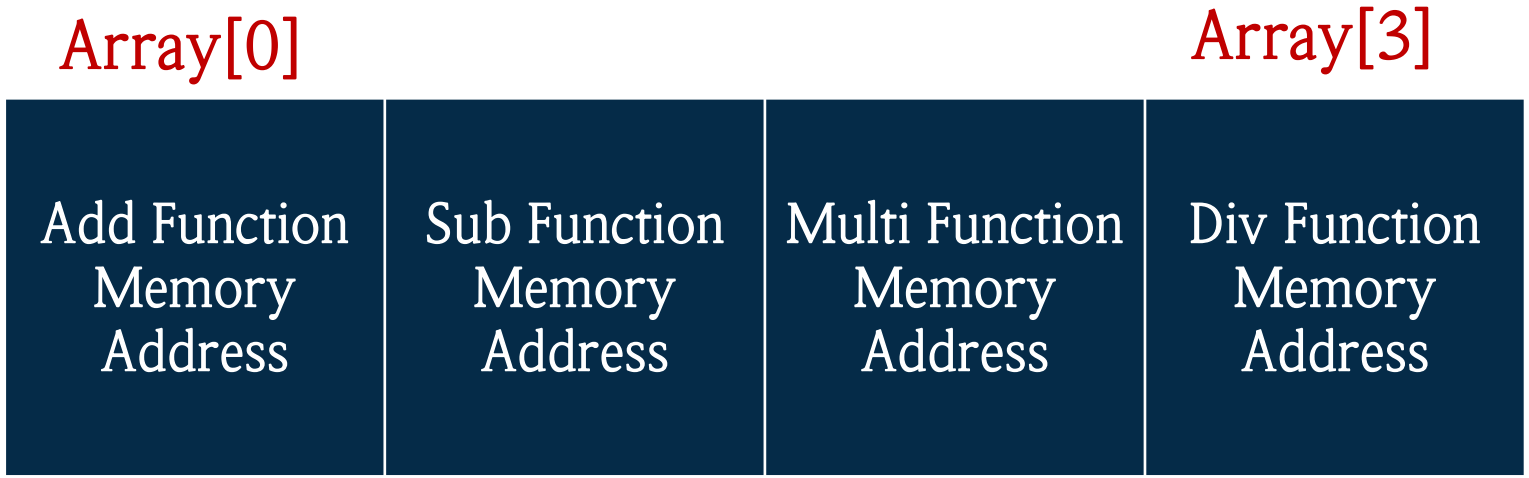
★ Input: 3 Integers
Ex) 60 50 2

★ Output: Output in ascending order
Ex) 2 50 60

Task 2 : Array of function pointers

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Create addition, subtraction, multiplication, and division functions,
and write a program that stores the memory address of each function in the index of each array by declaring an array of function pointers.



Task 3 : Copy an array

Write a program that copies an array of sizes n.

★ Use malloc

```
int* pArr;
pArr = (int*)malloc(sizeof(int) * N);
```

★ Use free

Task 4 : Average score

Write a program that scores an average score on a subject.

★ Determine how many subjects to save by input value.
-> It is going to proceed with dynamic memory allocation as many subjects as possible.

★ The average score is output by discarding the decimal point.