C Programming – Day 3

2018.05.08

JunGu Kang
Whols

아주대학교 Whols

배열과 포인터

배열과 포인터

배열의 이름은 배열의 주소를 가리키는 포인터

배열과 포인터

- 배열과 포인터
  - 3_1.c
- 포인터를 활용한 배열 원소 접근
  - 3_2.c

배열과 포인터

*(ptr + i)와 ptr(i)는 같다.

다차원배열과 포인터

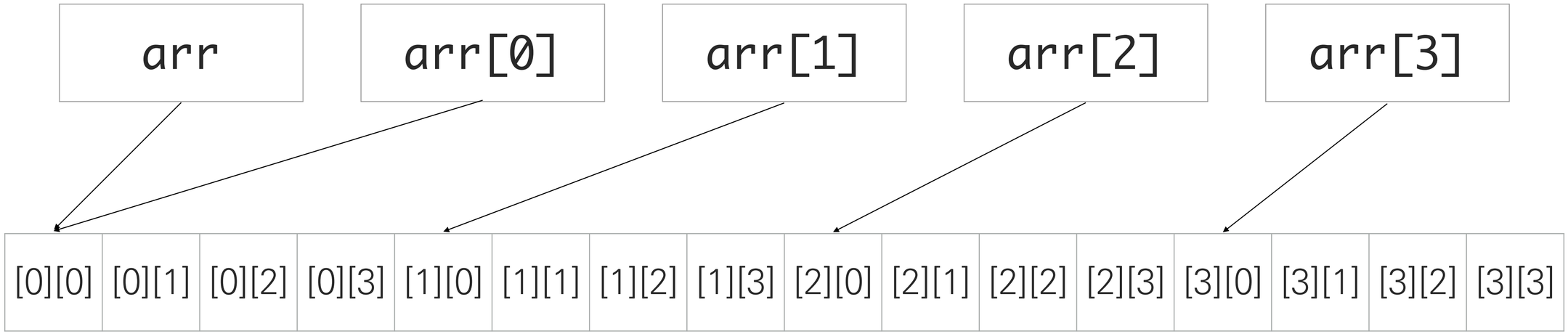
배열의 이름은 배열의 주소를 가리키는 포인터

다차원배열과 포인터

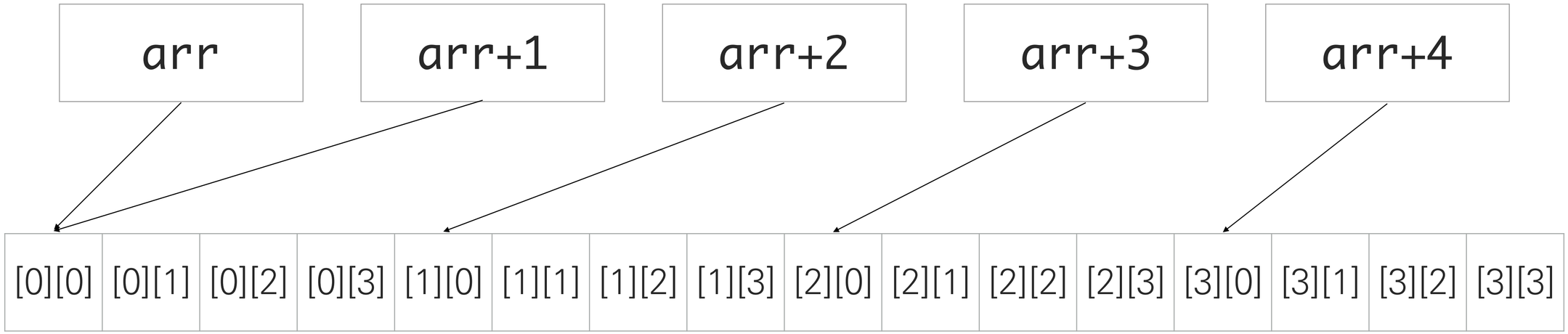
다차원배열이라 하더라도 메모리에는 이렇게 저장된다.

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

다차원배열과 포인터



다차원배열과 포인터



다차원배열과 포인터

- 주소
  - 3_3.c
- 크기
  - 3_4.c

다차원배열과 포인터

arr와 arr[0]는 가리키는 주소는 같지만 의미는 다르다.

함수 포인터

함수 포인터

메모리 주소를 가리키는게 포인터라고 했다.

함수 포인터

프로그램 코드도 메모리에 저장된다.(Von Neumann Architecture)

함수 포인터

그렇다면 프로그램 코드를 가리키는 포인터도 있을 수 있다.

함수 포인터

```
return_type (* name) (argument declaration);
```

함수 포인터

- 함수 포인터
  - 3_5.c
- 함수 포인터로 계산기 만들기
  - 3_6.c

void 포인터

void 포인터

가리키는 주소의 자료형이 void인(없는) 포인터

void 포인터

주소를 저장하기만 할 수 있다.
가리키는 대상을 역참조할 수 없다.

void 포인터

- void 포인터 역참조
  - 3_7.c

문자열 포인터

문자열 포인터

문자열도 메모리에 저장되므로 포인터로 가리킬 수 있다.

문자열 포인터

- 배열에 저장된 문자열 바꾸기
  - 3_8.c
- 문자열 포인터가 가리키는 문자열 바꾸기
  - 3_9.c
- 배열의 원소 바꾸기
  - 3_10.c
- 문자열 포인터가 가리키는 문자 바꾸기
  - 3_11.c

다중 포인터

문자열 포인터

포인터도 메모리에 저장되므로 포인터로 가리킬 수 있다.

문자열 포인터

- 다중포인터
  - 3_12.c

포인터 배열

포인터 배열

포인터 변수도 이어붙여서 배열로 만들 수 있다.

포인터 배열

- 포인터 배열
  - 3_13.c

함수의 인자

배열을 인자로 전달하기

main 함수로도 인자를 전달할 수 있다.

배열을 인자로 전달하기

- 두 배열의 원소 합과 차 출력하기
  - 3_14.c

다차원배열을 인자로 전달하기

- 행렬의 합과 차 구하기
  - 3_15.c

main 함수의 인자

main 함수로도 인자를 전달할 수 있다.

main 함수의 인자

```
int main(int argc, char * argv[]) {
    // statements
    return 0;
}
```

main 함수의 인자

argv는 더블 포인터

main 함수의 인자

Strings

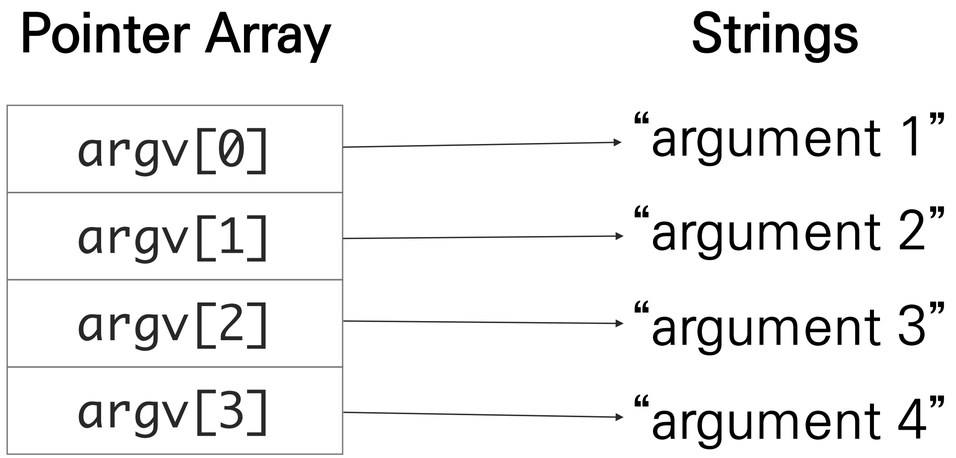
“argument 1”

“argument 2”

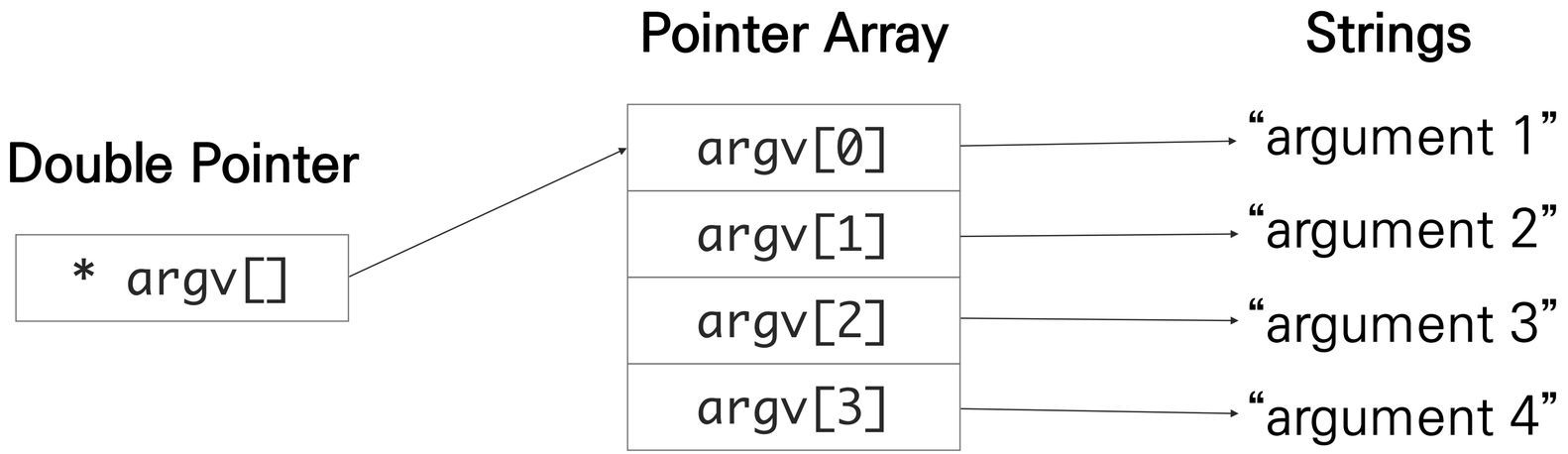
“argument 3”

“argument 4”

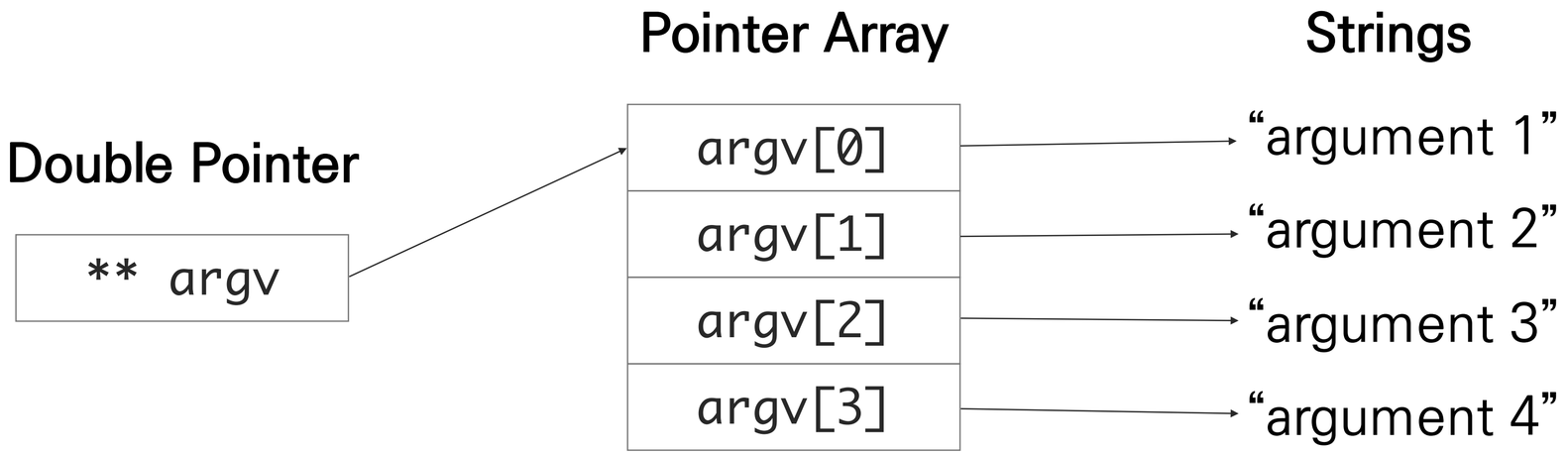
main 함수의 인자



main 함수의 인자



main 함수의 인자



main 함수의 인자

- main 함수의 인자들 출력하기
  - 3_16.c

사용자 정의 자료형

사용자 정의 자료형

필요로 하는 자료형을 만들어 쓸 수 있다.

사용자 정의 자료형

정의한 자료형에 새로운 이름을 붙인다.

사용자 정의 자료형

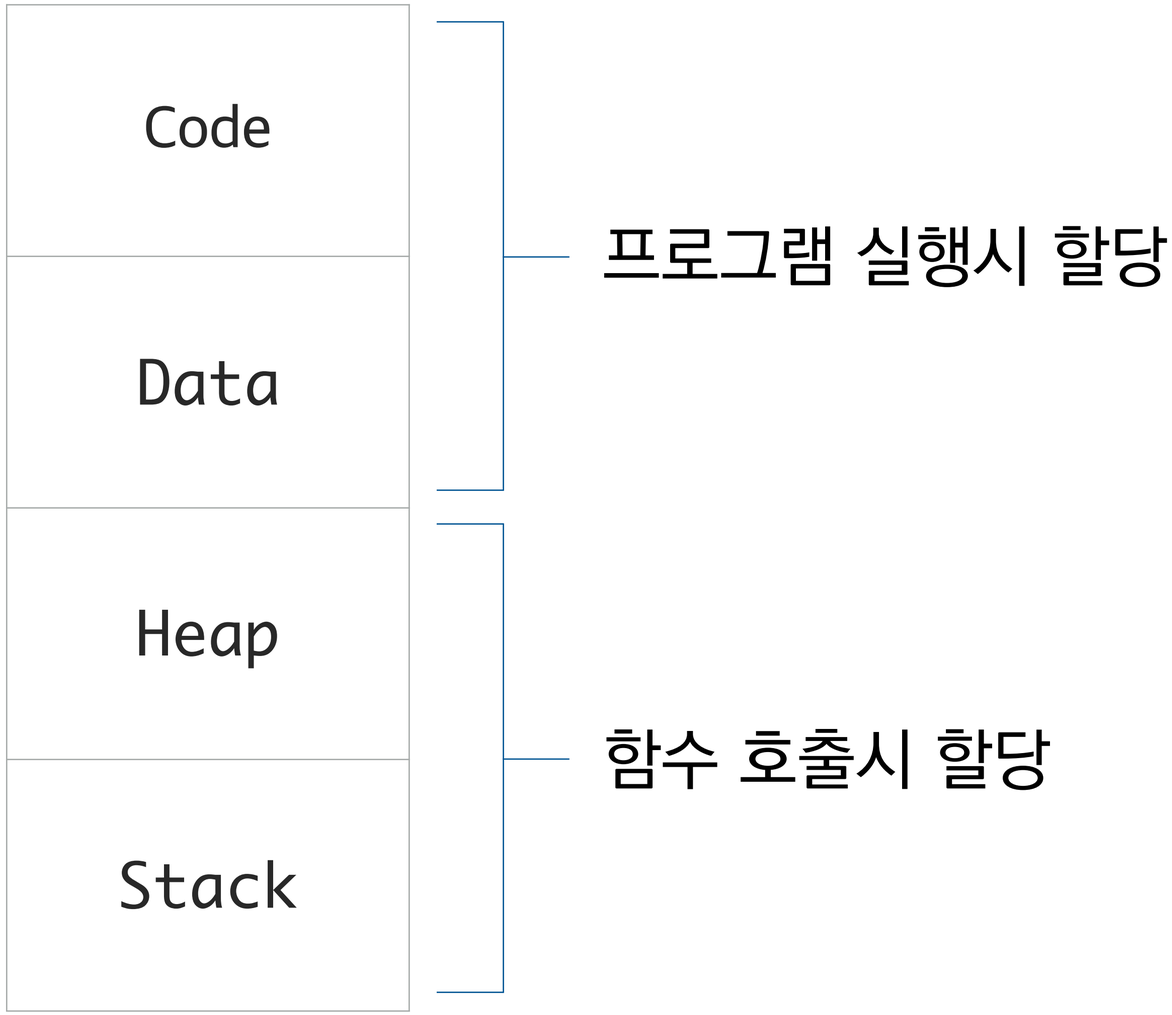
```
typedef type_name new_name;
```

사용자 정의 자료형

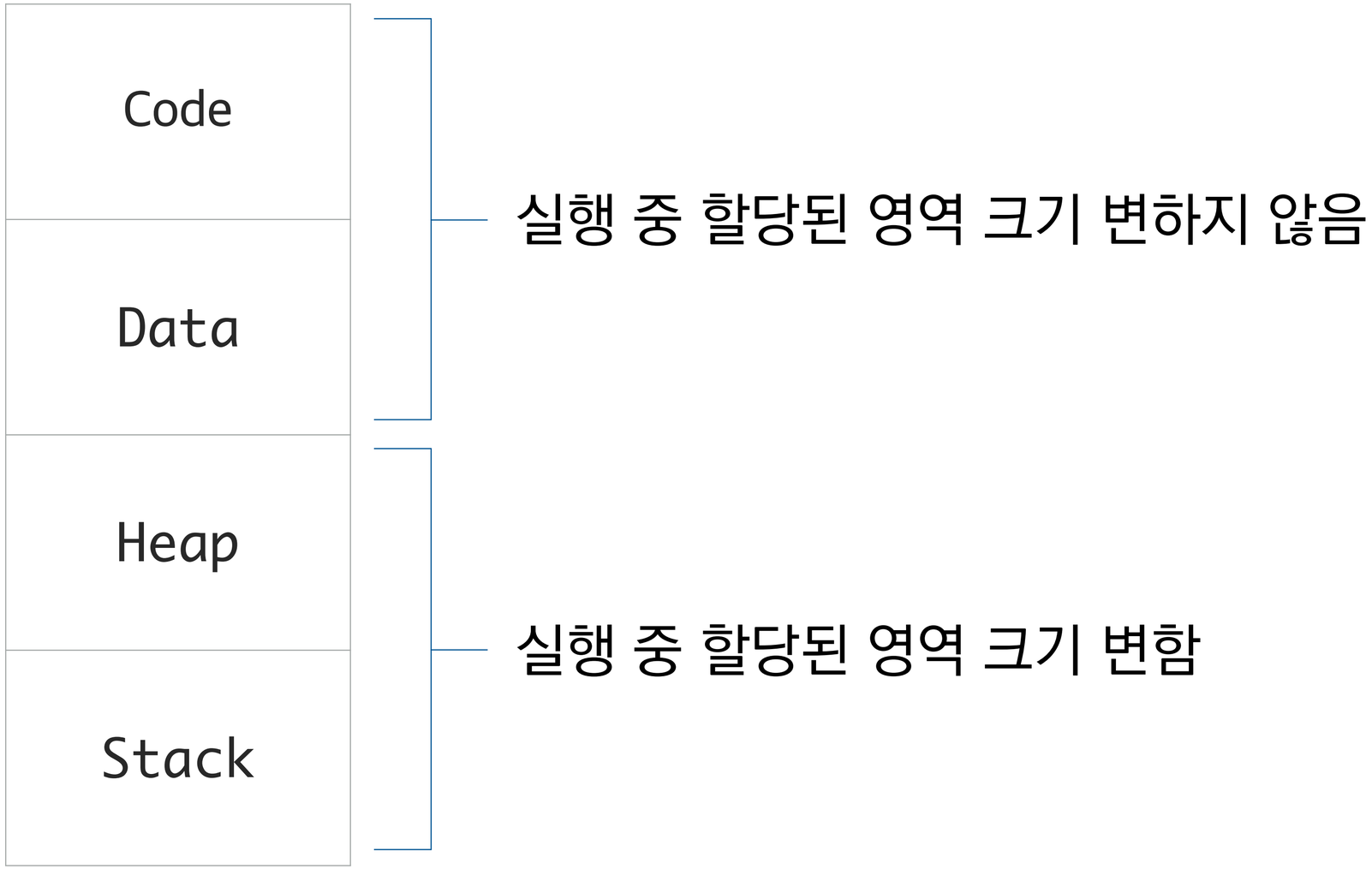
- typedef
  - 3_17.c

동적 메모리 할당

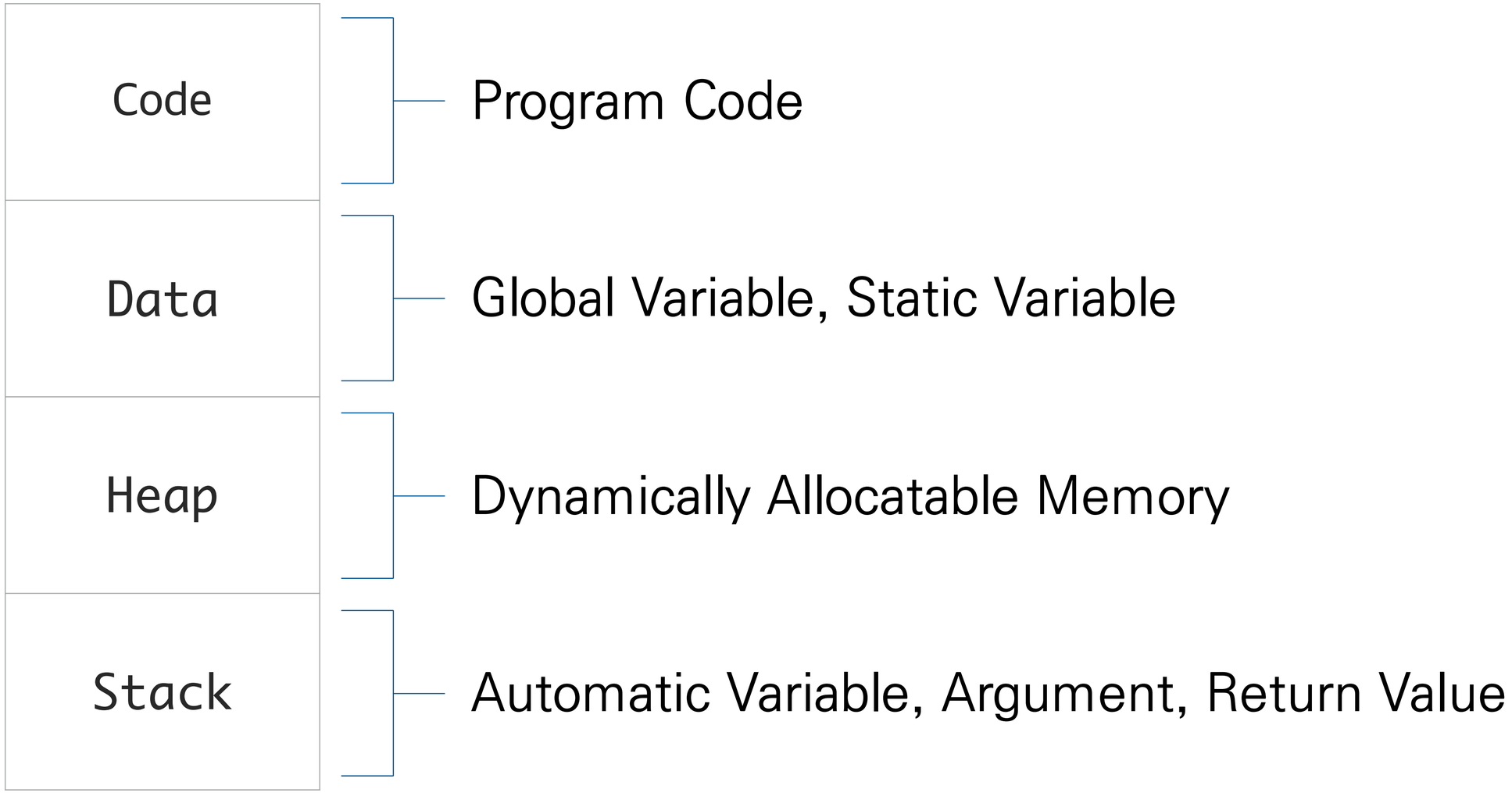
메모리 구조



메모리 구조



메모리 구조



malloc / calloc

heap 영역에 지정된 크기의 메모리를 할당한다.

malloc / calloc

```
#include <stdlib.h>

void * malloc(size_t);
// argument: 할당할 메모리 공간 크기
// return: 성공하면 할당된 메모리 주소, 실패하면 NULL

void * calloc(size_t, size_t);
// argument: 블록의 개수, 블록의 크기(둘을 곱한 만큼 할당)
// return: 성공하면 할당된 메모리 주소, 실패하면 NULL
// calloc은 할당시 공간을 0으로 초기화한다
```

free

할당된 메모리 영역을 해제한다.

free

```
#include <stdlib.h>

void free(void *);
```

동적 할당

- malloc
  - 3_18.c
- calloc
  - 3_19.c

realloc

할당된 공간의 크기를 바꿔 다시 할당한다.

realloc

```
#include <stdlib.h>

void * realloc(void *, size_t);
```

동적 할당

- realloc
  - 3_20.c

memset

지정한 메모리 영역을 지정한 값으로 초기화.
다른 메모리 관련 함수와 다르게 string.h에 선언되어 있다.

memset

```
#include <string.h>

void * memset(void *, int, size_t);
```

memset

- 배열 초기화
  - 3_21.c
- 동적 할당된 공간 초기화
  - 3_22.c

C Programming – Day 3

2018.05.08

JunGu Kang
Whols

아주대학교 Whols