

# Stack Frame1

- High level -

안태진([taejin@codecure.smuc.ac.kr](mailto:taejin@codecure.smuc.ac.kr))

상명대학교 보안동아리 CodeCure

# 목 차

---

- StackFrame의 필요성

- 목표
- 계획

- Stack이란

- 변수의 종류

- 지역, 전역, 정적

- 호출 스택

- Stack Frame이란

# StackFrame의 필요성

- 목표

- BOF 실습

- BOF를 위해서 알아야 함

- bof (Buffer Overflow)

- 버퍼의 크기보다 더 큰 문자열을 입력하여 사용자가 원하는 코드흐름을 탈취하는 공격기법

- StackFrame

- Memory Structure

- Reversing

날짜	활동
9/9	OT, C언어 복습
9/16 (축제 주)	C언어 문제
9/23	C언어 문제 풀이
9/30	WarGame(FTZ)
10/7	WarGame(FTZ)
10/14	시험 전 휴식
10/21	중간고사
10/28	StackFrame1
11/4	StackFrame2
11/11	Reversing1
11/18	Reversing2
11/25	BOF 실습
12/2	WarGame(Protostar, picoCTF)
12/9	시험 전 휴식
12/16	기말고사

# StackFrame의 필요성

---

- 계획 (1/2)
  - Stack Frame
    - Stack Frame1 (Medium)
      - 스택(Stack)에 대해 전체적인 이해
      - 코드 수준에서의 스택의 동작 과정 이해
    - Stack Frame2 (Hard)
      - 스택이 실제 프로세스 수준에서 동작되는 과정 이해
      - gdb를 이용한 실제 동작 과정 확인 이해
  - 메모리 구조
    - Stack Frame에서 전체적인 구조 설명

# StackFrame의 필요성

---

- 계획 (2/2)
  - Reversing
    - Reversing1
      - OllyDbg를 통한 리버싱에 대한 전체적인 이해
      - 워게임을 통해 Reversing 이해
    - Reversing2
      - 좀 더 심화된 Reversing
      - 후에 BOF 실습1으로 대체될 수도 있음

# 목 차

---

- StackFrame의 필요성

- 목표
- 계획

- Stack이란

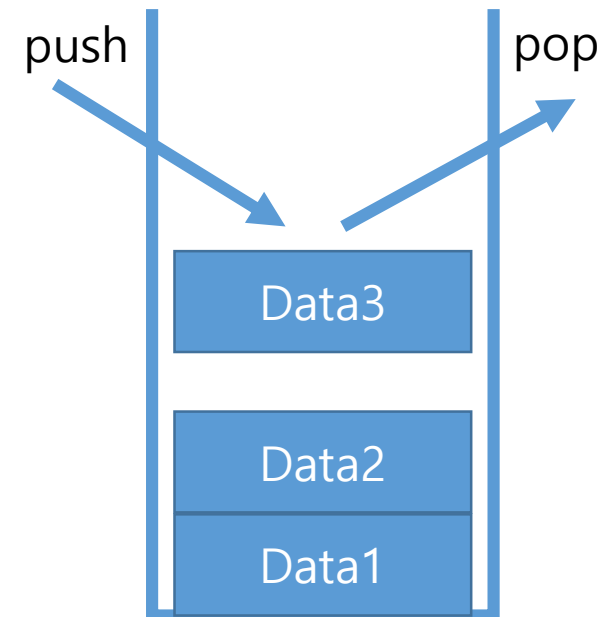
- 변수의 종류

- 지역, 전역, 정적
- 호출 스택

- Stack Frame이란

# Stack이란

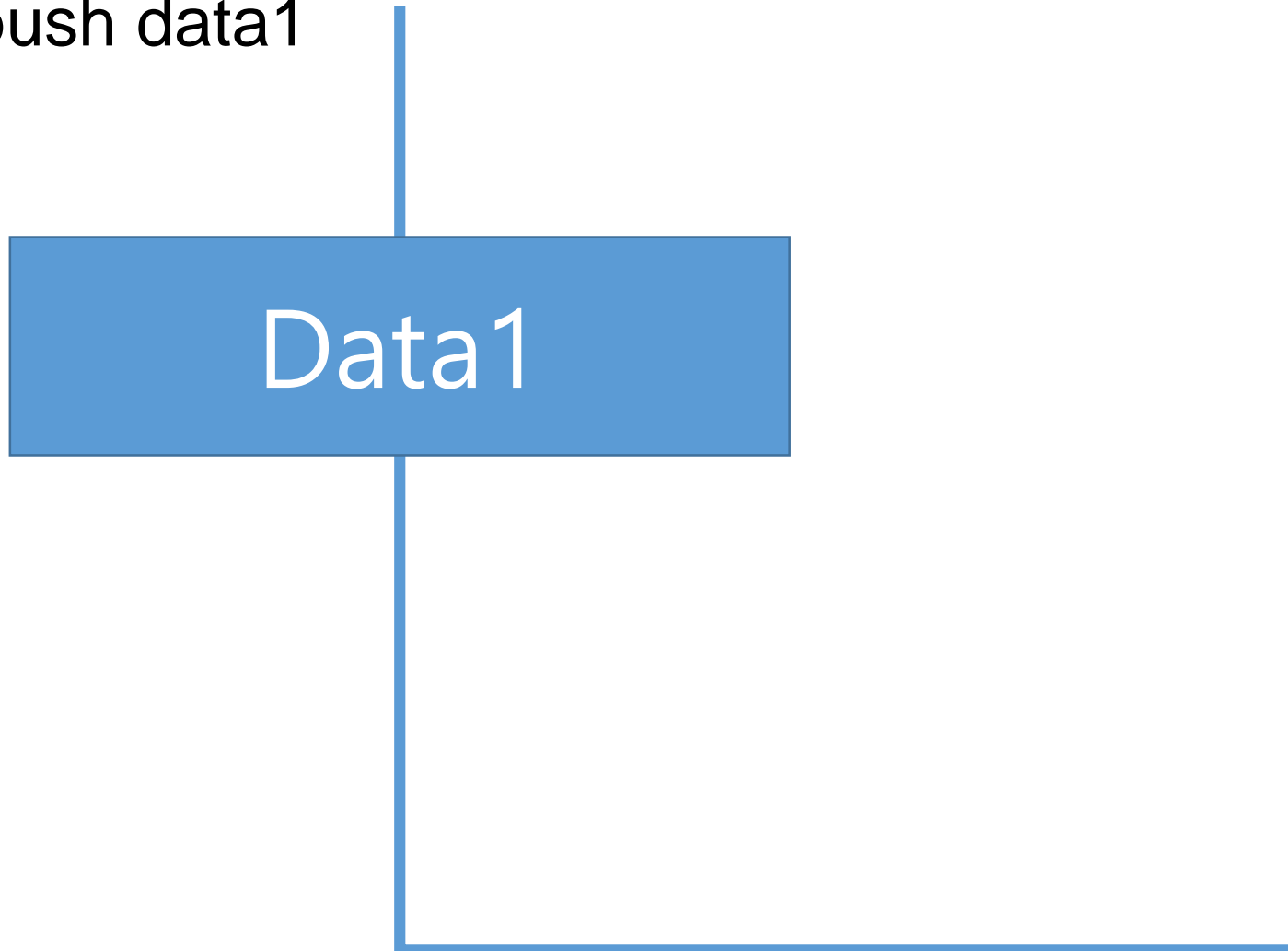
- Stack의 개념
  - LIFO방식의 자료구조
    - LIFO(Last In First Out, 후입선출)
      - 가장 나중에 들어온 데이터가 가장 먼저 나가는 형식
  - 프로그램 내에서 호출하는 함수들이 Stack의 형식으로 저장
- push
  - Stack에 데이터를 넣는 명령어
- pop
  - Stack에서 데이터를 인출하는 명령어



# Stack이란

---

- 스택의 동작 과정 (1/10)
- push data1





# Stack이란

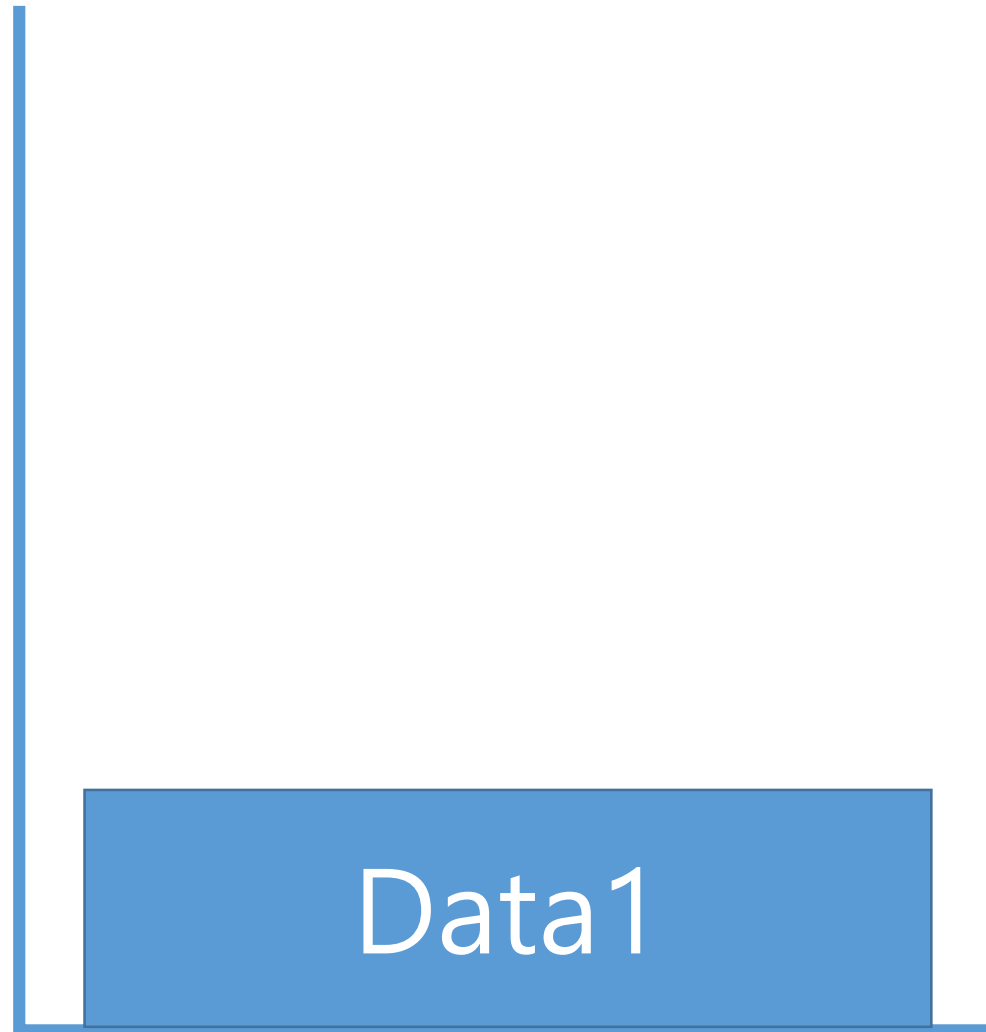
- 스택의 동작 과정 (1/10)
- push data1



# Stack이란

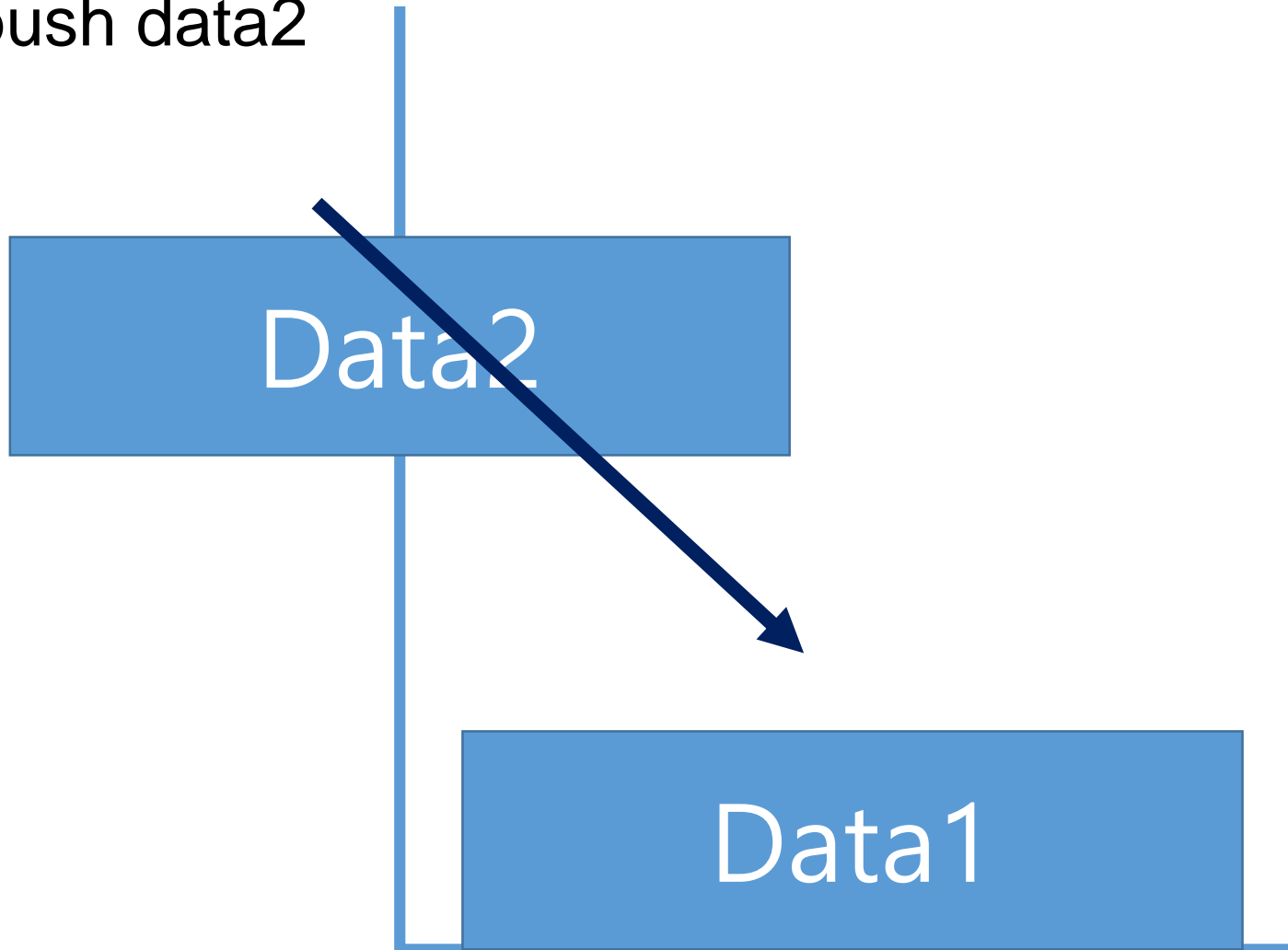
---

- 스택의 동작 과정 (1/10)
- push data1



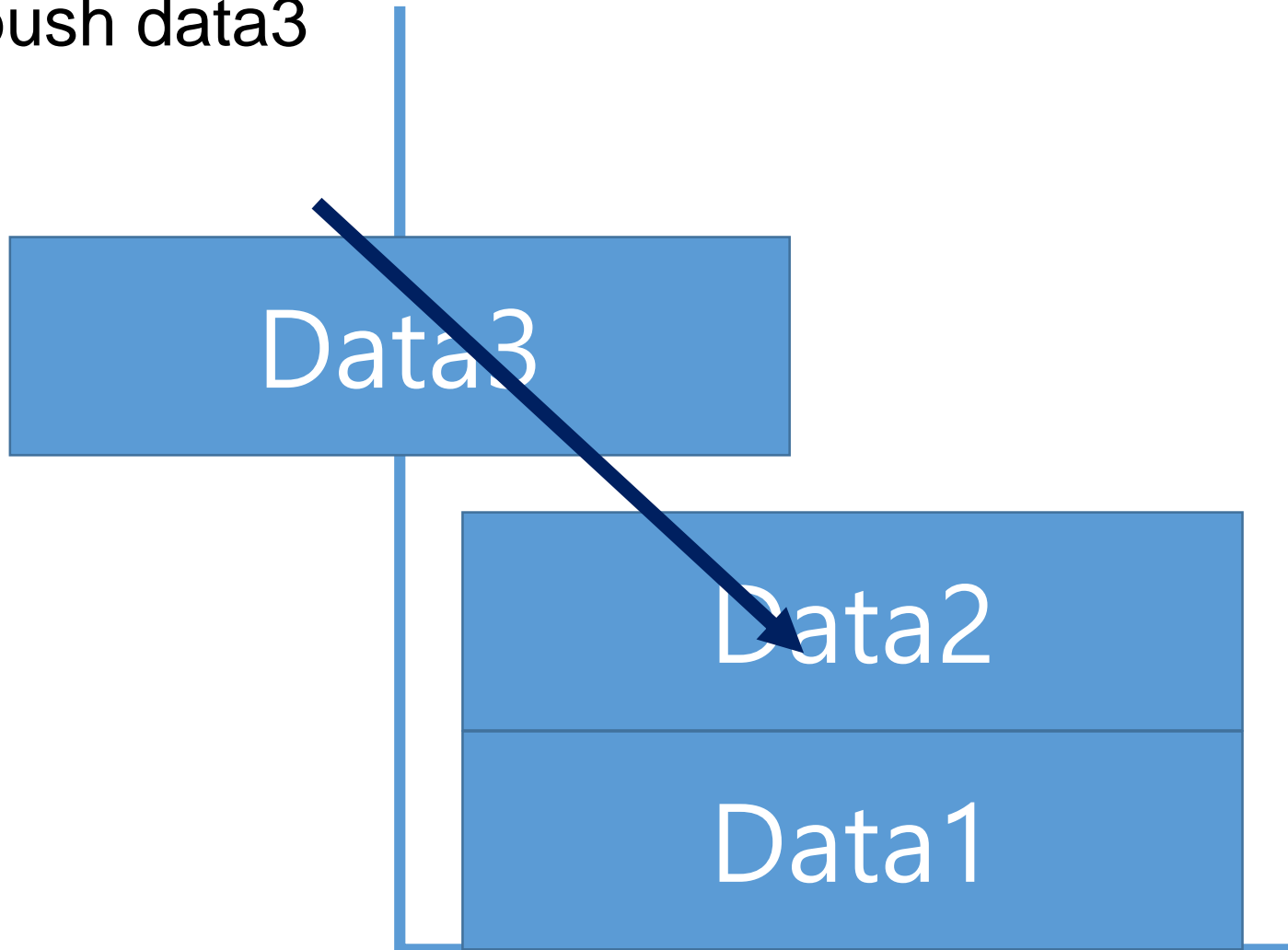
# Stack이란

- 스택의 동작 과정 (2/10)
- push data2



# Stack이란

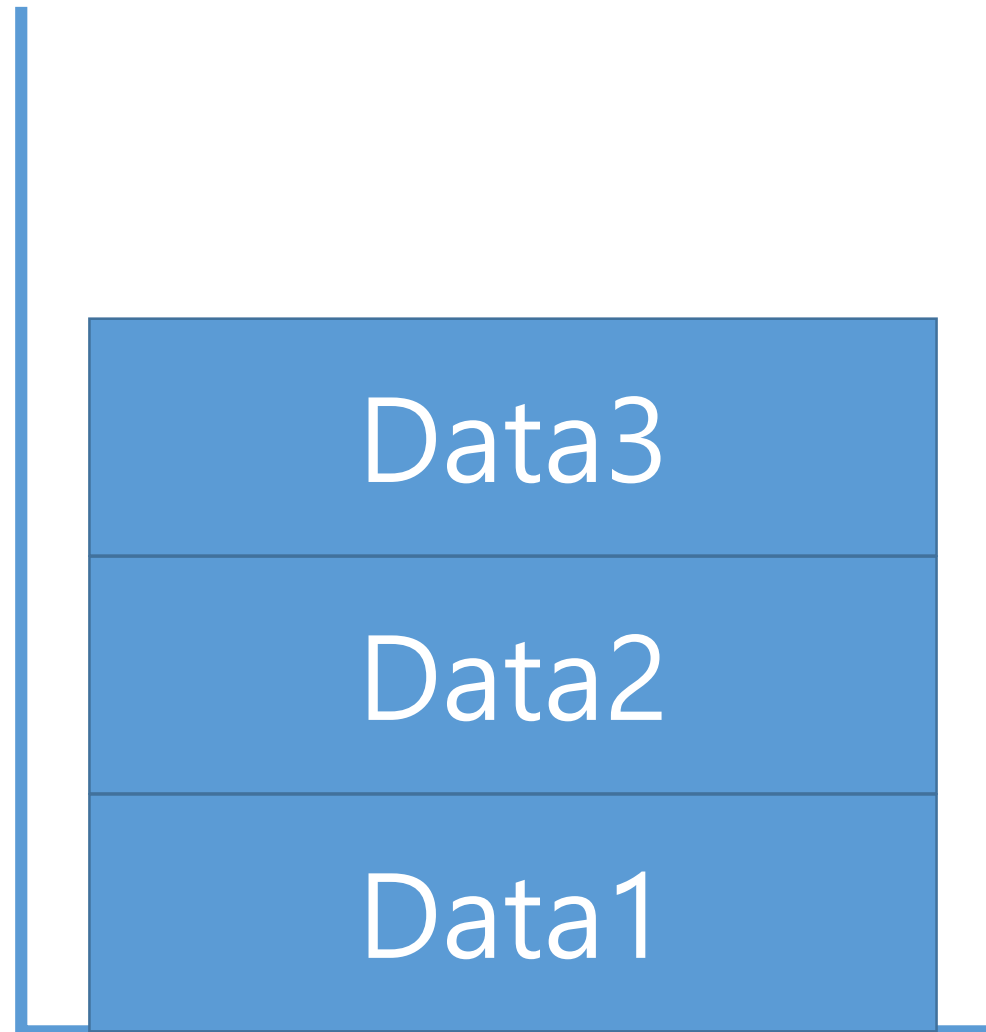
- 스택의 동작 과정 (3/10)
- push data3



# Stack이란

---

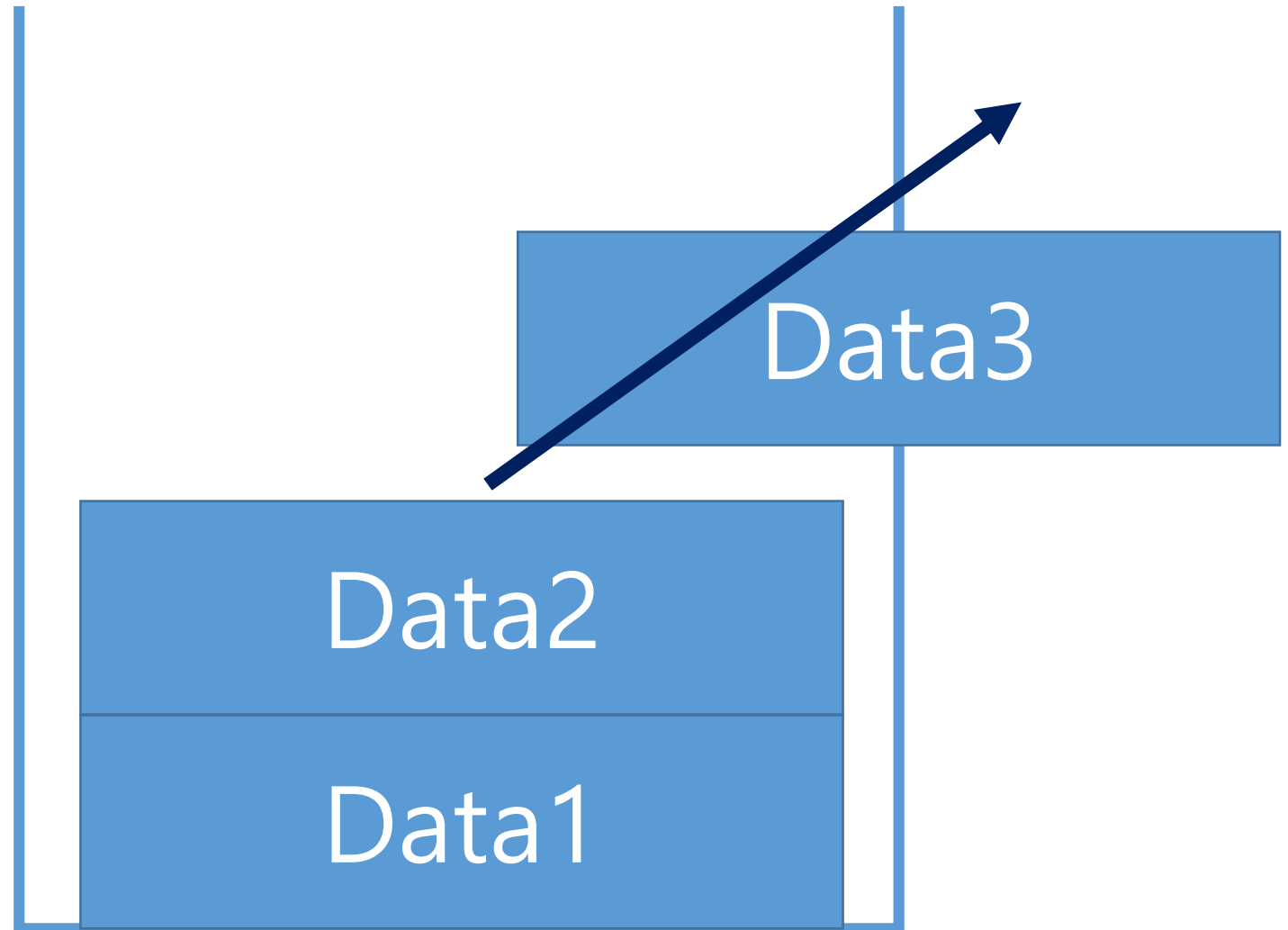
- 스택의 동작 과정 (4/10)
- 현재 Stack



# Stack이란

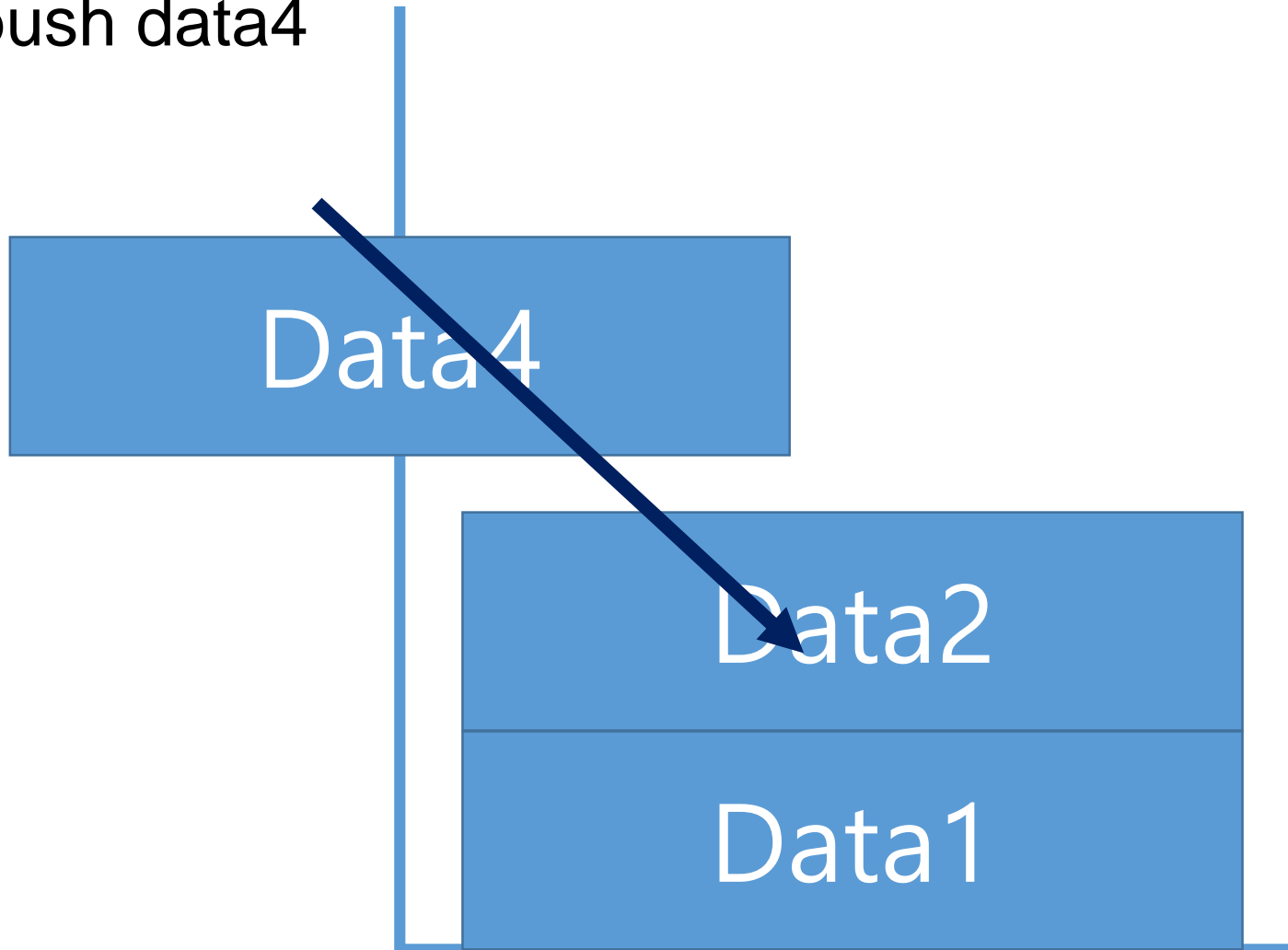
- 스택의 동작 과정 (5/10)

- pop



# Stack이란

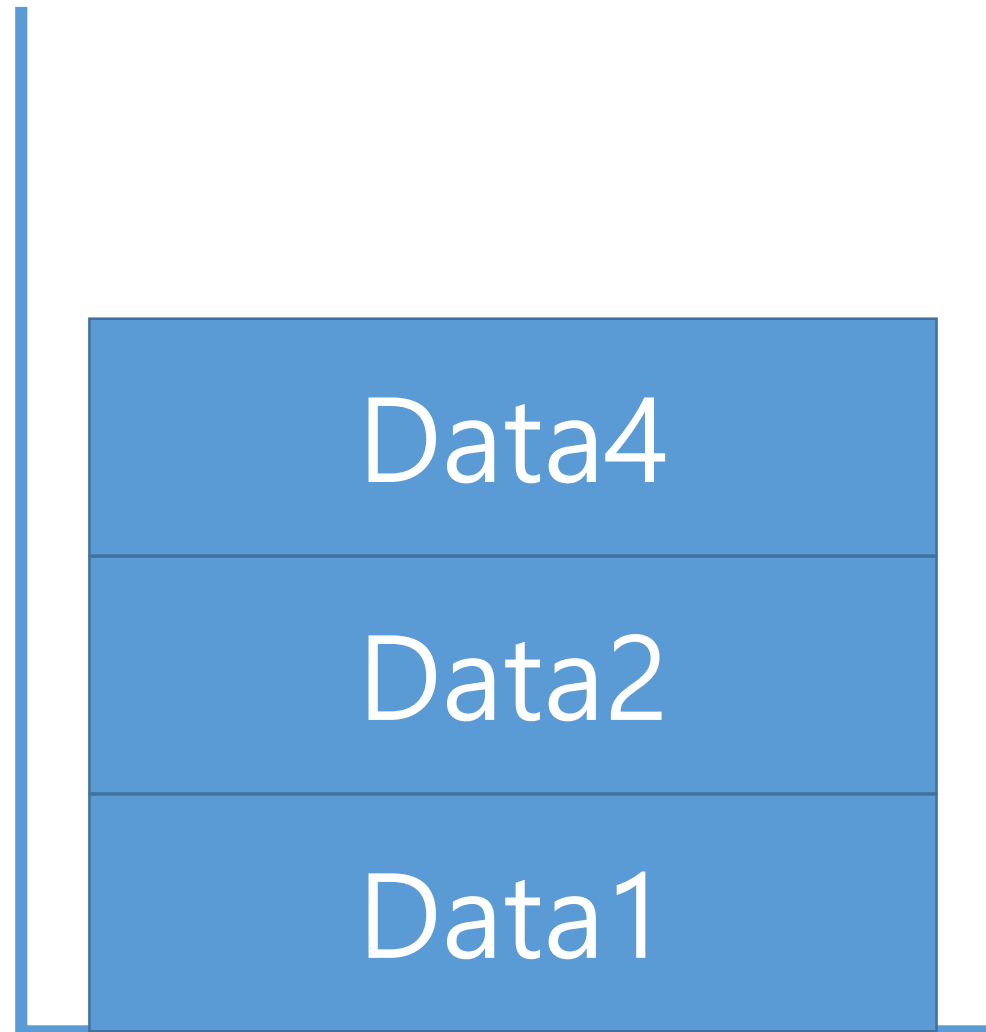
- 스택의 동작 과정 (6/10)
- push data4



# Stack이란

---

- 스택의 동작 과정 (7/10)
- 현재 Stack

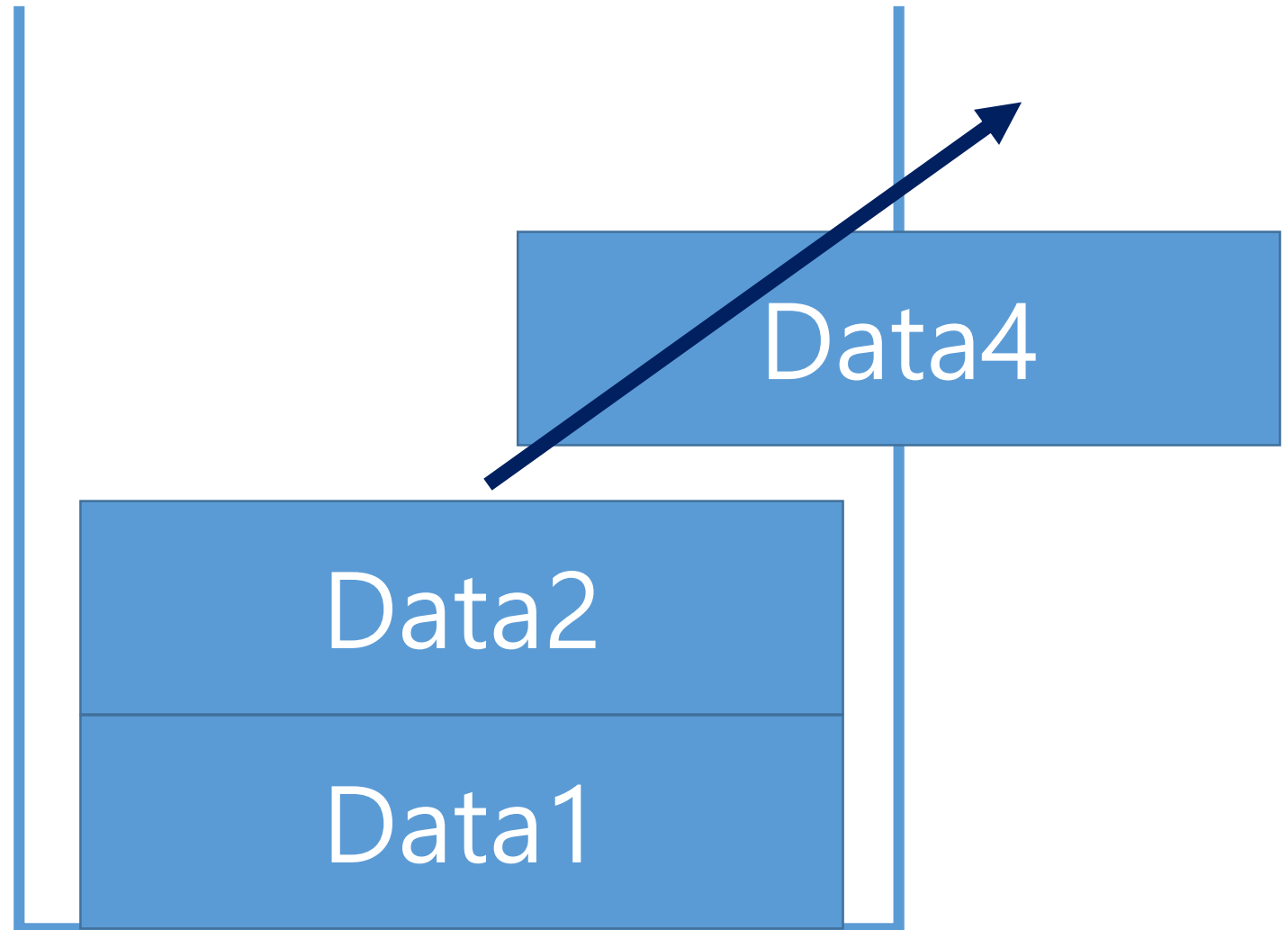




# Stack이란

- 스택의 동작 과정 (8/10)

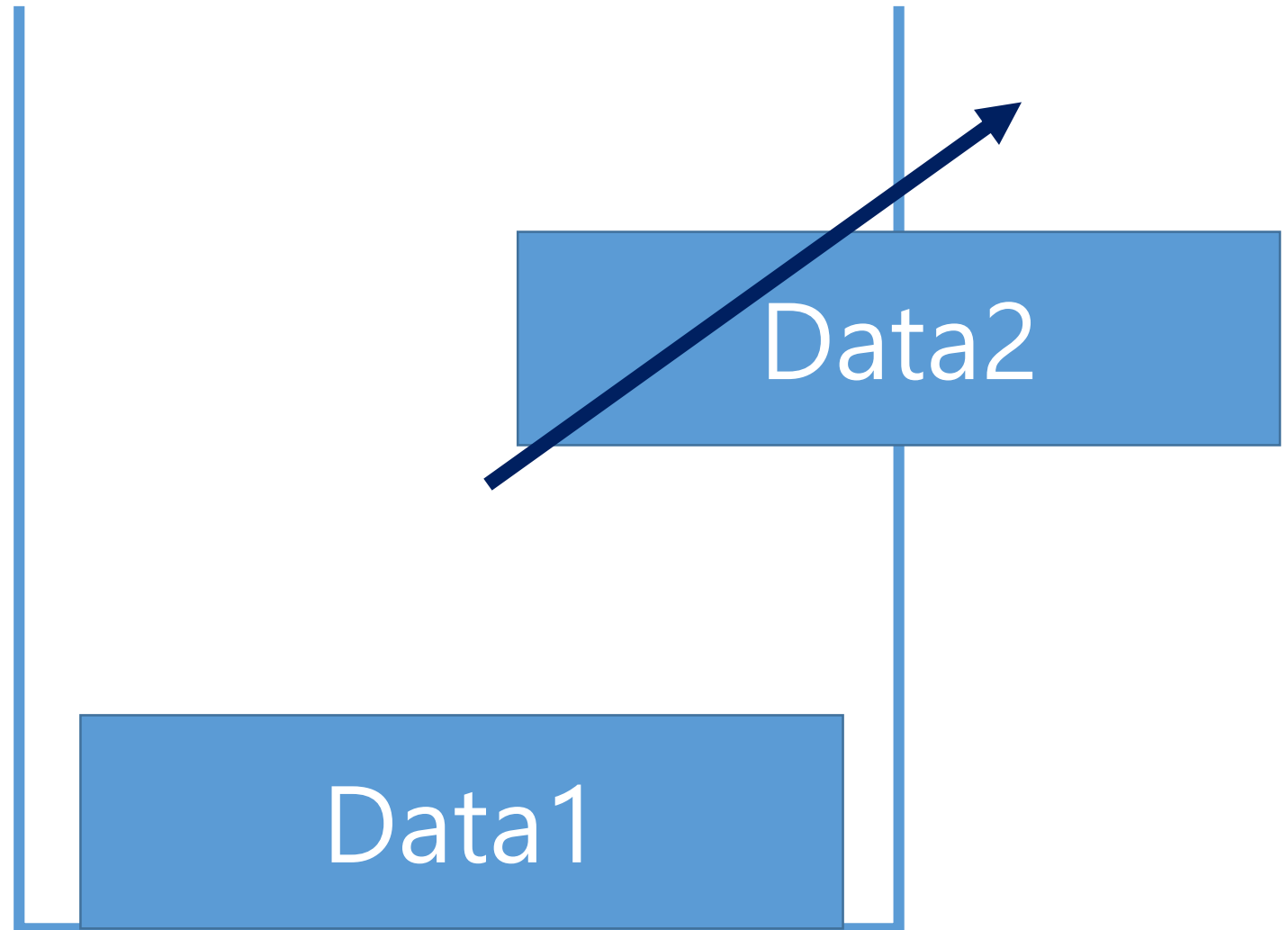
- pop



# Stack이란

- 스택의 동작 과정 (9/10)

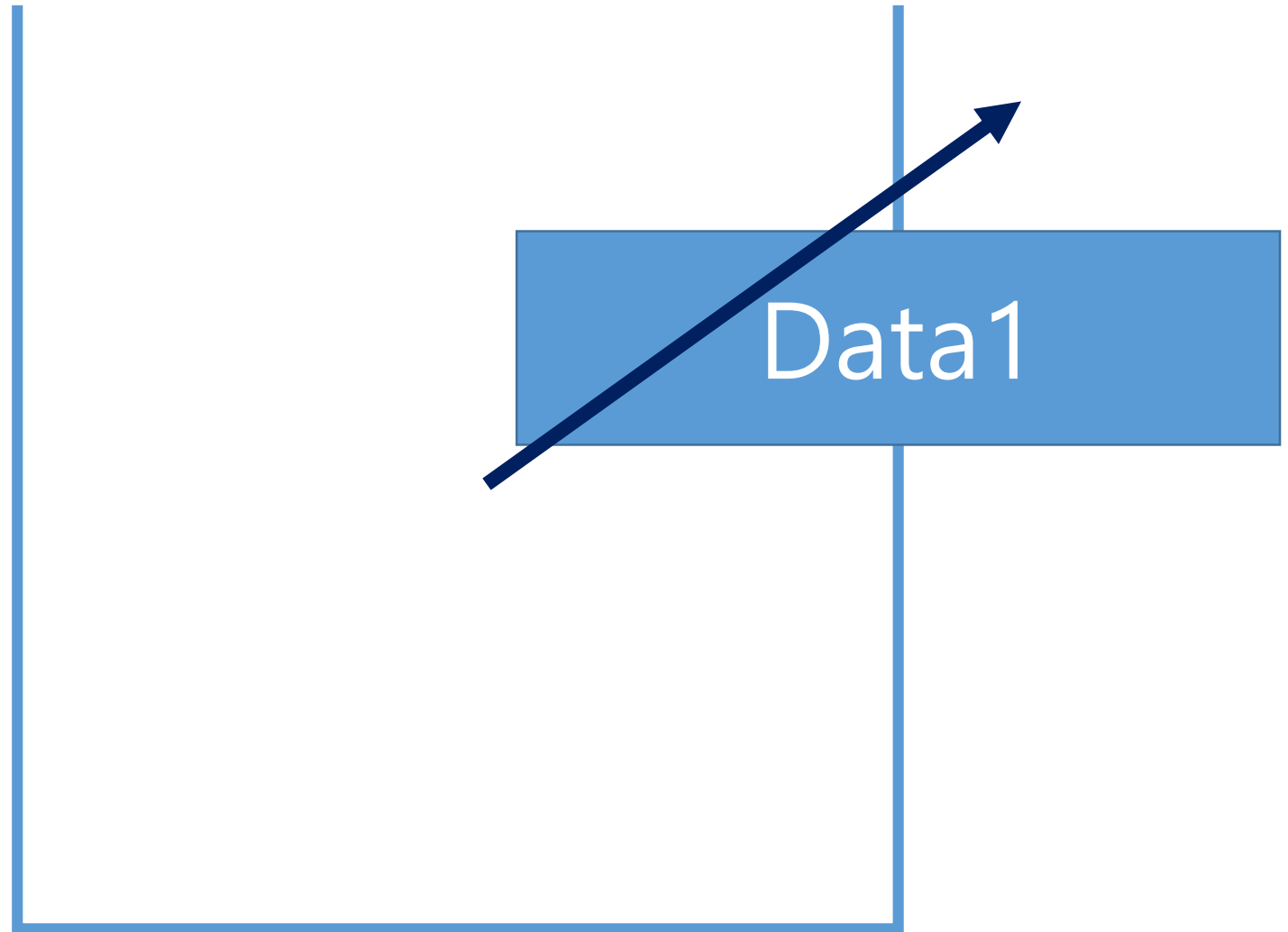
- pop



# Stack이란

- 스택의 동작 과정 (10/10)

- pop



# 목 차

---

- StackFrame의 필요성

- 목표
- 계획

- Stack이란

- 변수의 종류

- 지역, 전역, 정적
- 호출 스택

- Stack Frame이란

# 변수의 종류

- 지역 변수 (Local Variable)
  - 특정한 지역에 진입 했을 때 생성되며,  
그 지역을 벗어나면 메모리에서 삭제되는 변수
- 우측 그림에서
  - main 지역
    - num1, num2
    - nums1
  - for 지역
    - i

```
int glob1, glob2;
int globs1[10], globs2[10];

static int stat1, stat2;

int main(void) {
    int num1, num2;
    int nums1[10];

    static int stat3, stat4;

    for (int i = 0; i < 10; i++)
        nums1[i] = i;
}
```

# 변수의 종류

- 전역 변수 (Global Variable)

- 프로그램이 실행되면 메모리에 생성되며,  
프로그램이 종료시에 메모리에서 삭제되는 변수

- 우측 그림에서

- glob1, glob2
- globs1, globs2

```
int glob1, glob2;
int globs1[10], globs2[10];

static int stat1, stat2;

int main(void) {
    int num1, num2;
    int nums1[10];

    static int stat3, stat4;

    for (int i = 0; i < 10; i++)
        nums1[i] = i;
}
```

# 변수의 종류

- 정적 변수(Static Variable)

- 지역 변수 + 전역 변수
- 프로그램 실행시 메모리에 생성되며, 프로그램 종료시에 메모리에서 삭제됨
- 단, 생성된 지역에서만 접근 가능

- 우측 그림에서

- stat1, stat2
- stat3, stat4

```
int glob1, glob2;
int globs1[10], globs2[10];

static int stat1, stat2;

int main(void) {
    int num1, num2;
    int nums1[10];

    static int stat3, stat4;

    for (int i = 0; i < 10; i++)
        nums1[i] = i;
}
```

# 변수의 종류

## • 정적 변수의 특성 (StaticTest.c)

```
/* StaticTest.c */
#include <stdio.h>

int time;

int func1(void) {
    static int num1 = 0;
    printf("%d: %d\n", time, num1++);
}

int func2(void) {
    int num1 = 0;
    printf("%d: %d\n", time, num1++);
}

int main(void) {
    for (time = 0; time < 10; time++)
        func1();

    for (time = 0; time < 10; time++)
        func2();

    // printf("%d: %d\n", time, num1);

    return 0;
}
```

주석 제거

```
jinn-desk@JIN-DESK:/mnt/c/Users/Jin/Desktop$ gcc -o StaticTest StaticTest.c
StaticTest.c: In function 'main':
StaticTest.c:23:27: error: 'num1' undeclared (first use in this function)
    printf("%d: %d\n", time, num1);
                           ^~~~~
StaticTest.c:23:27: note: each undeclared identifier is reported only once for each function it appears in
```

```
jinn-desk@JIN-DESK:/mnt/c/Users/Jin/Desktop$ gcc -o StaticTest StaticTest.c
jinn-desk@JIN-DESK:/mnt/c/Users/Jin/Desktop$ ./StaticTest
0: 0
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
0: 0
1: 0
2: 0
3: 0
4: 0
5: 0
6: 0
7: 0
8: 0
9: 0
```

정적 변수

지역 변수



# 변수의 종류

- 변수의 위치 (AddressTest.c)

```
jin-desk@JIN-DESK:~$ gcc -o at AddressTest.c
jin-desk@JIN-DESK:~$ ./at
glob1: 0x7f6156801015
glob2: 0x7f6156801016
stat1: 0x7f6156801011
stat2: 0x7f6156801012
local1: 0x7fffca2f8fae
local2: 0x7fffca2f8faf
stat3: 0x7f6156801013
stat4: 0x7f6156801014
locals1: 0x7fffca2f8fb0
```

```
/* AddressTest.c */
#include <stdio.h>

char glob1, glob2; // Global Variable
static char stat1, stat2; // Static Global Variable

int main(void) {
    char local1, local2; // Local Variable

    static char stat3, stat4; // Static Local Variable

    char locals1[16]; // Local Array

    printf("glob1: %p\n", &glob1);
    printf("glob2: %p\n", &glob2);
    printf("stat1: %p\n", &stat1);
    printf("stat2: %p\n", &stat2);
    printf("local1: %p\n", &local1);
    printf("local2: %p\n", &local2);
    printf("stat3: %p\n", &stat3);
    printf("stat4: %p\n", &stat4);
    printf("locals1: %p\n", locals1);

    return 0;
}
```

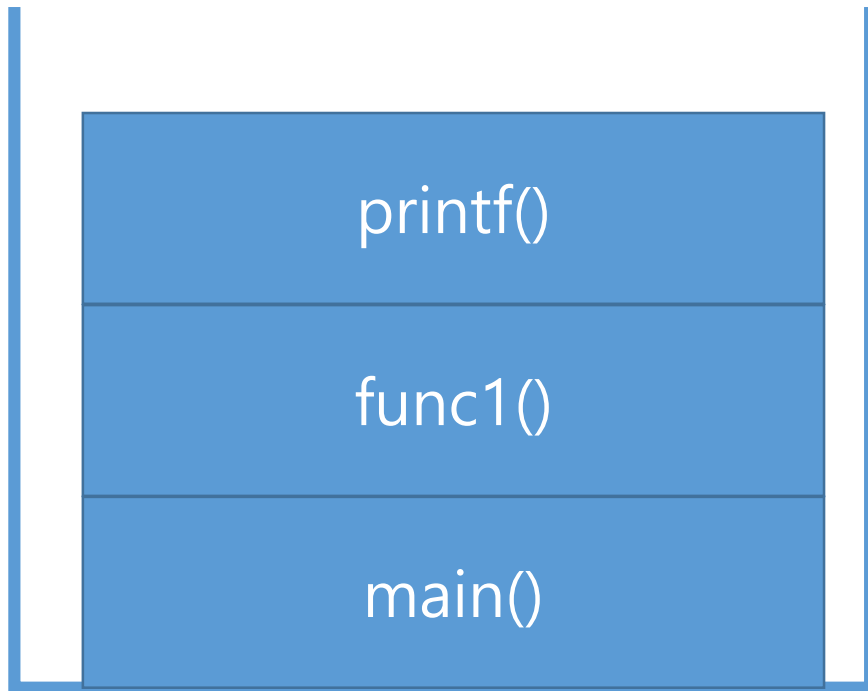
# 목 차

---

- StackFrame의 필요성
  - 목표
  - 계획
- Stack이란
- 변수의 종류
  - 지역, 전역, 정적
- 호출 스택
- Stack Frame이란

# 호출 스택

- 호출 스택 (Call Stack)
- 함수가 호출되면 스택의 형식으로 메모리에 적재
- 적재된 함수들을 나타낸 그림이 호출 스택



```
void func1(void) {  
    printf("This is func1\n");  
}  
  
int main(void) {  
    func1();  
  
    return 0;  
}
```

# 호출 스택

- 호출 스택 동작 과정 (1/13) (CallStack.c)

```
jln-desk@JIN-DESK:~$ gcc -o cs CallStack.c
```

```
jln-desk@JIN-DESK:~$ ./cs
```

```
This is main  
This is func1  
This is func2  
This is func3  
This is func4  
This is func5
```

```
/* CallStack.c */  
#include <stdio.h>
```

```
void func1(void);  
void func2(void);  
void func3(void);  
void func4(void);  
void func5(void);
```

```
void main(void) {  
    puts("This is main");  
    func1();  
    func5();  
}
```

```
void func1(void) {  
    puts("This is func1");  
    func2();  
    func3();  
}
```

```
void func2(void) {  
    puts("This is func2");  
}
```

```
void func3(void) {  
    puts("This is func3");  
    func4();  
}
```

```
void func4(void) {  
    puts("This is func4");  
}
```

```
void func5(void) {  
    puts("This is func5");  
}
```



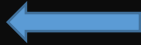
# 호출 스택

## • 호출 스택 동작 과정 (2/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
```

```
jin-desk@JIN-DESK:~$ ./cs
```

```
This is main  
This is func1  
This is func2  
This is func3  
This is func4  
This is func5
```



```
/* CallStack.c */  
#include <stdio.h>
```

```
void func1(void);  
void func2(void);  
void func3(void);  
void func4(void);  
void func5(void);
```

```
void main(void) {  
    puts("This is main");  
    func1();  
    func5();  
}
```



```
void func1(void) {  
    puts("This is func1");  
    func2();  
    func3();  
}
```

```
void func2(void) {  
    puts("This is func2");  
}
```

```
void func3(void) {  
    puts("This is func3");  
    func4();  
}
```

```
void func4(void) {  
    puts("This is func4");  
}
```

```
void func5(void) {  
    puts("This is func5");  
}
```

main()

# 호출 스택

## • 호출 스택 동작 과정 (3/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1 ←
This is func2
This is func3
This is func4
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

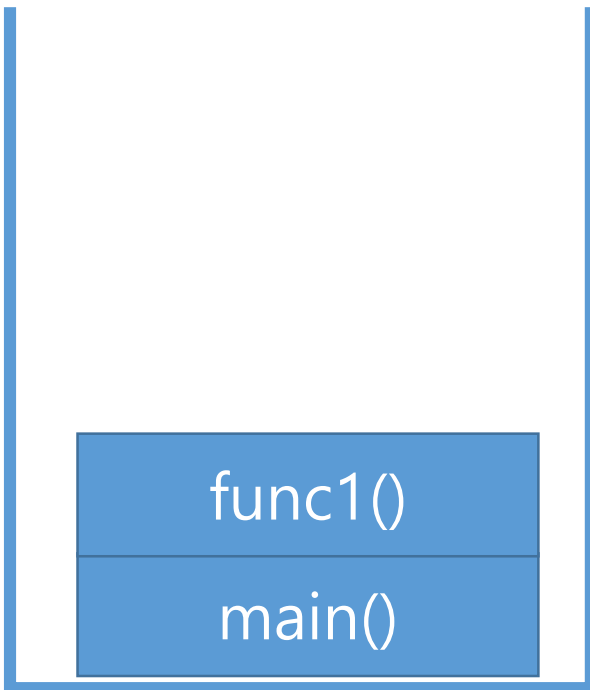
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (4/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2 ←
This is func3
This is func4
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

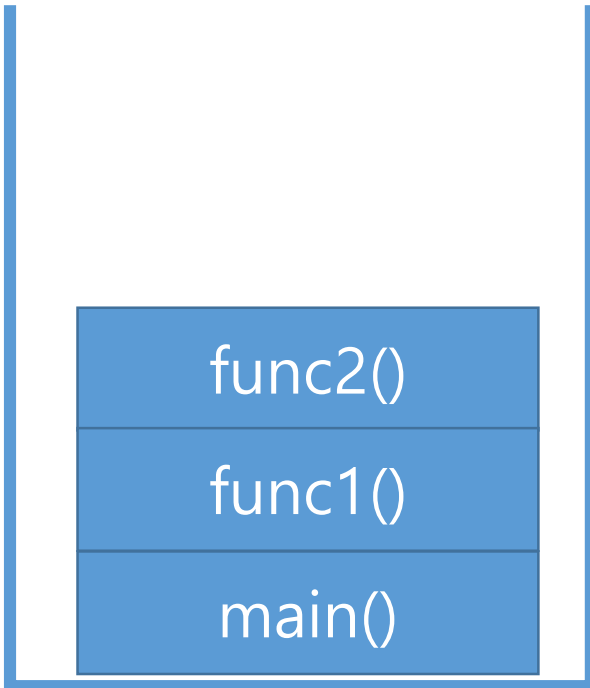
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (5/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2 ←
This is func3
This is func4
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

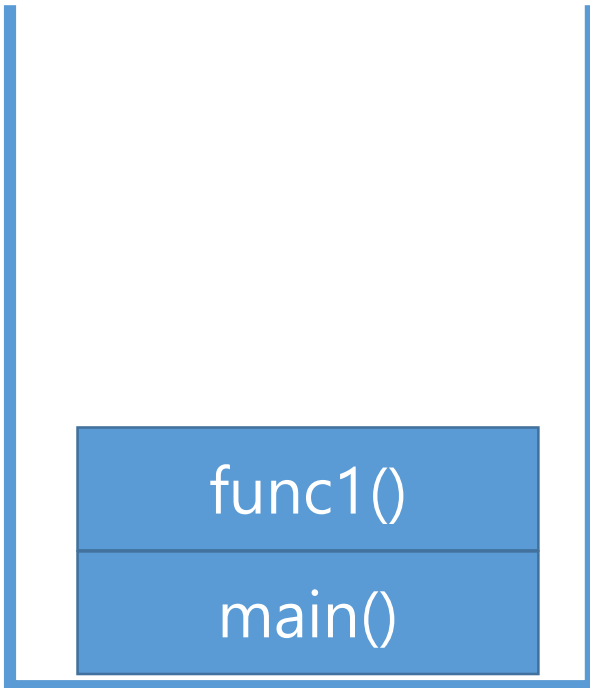
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```





# 호출 스택

## • 호출 스택 동작 과정 (6/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3 ←
This is func4
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

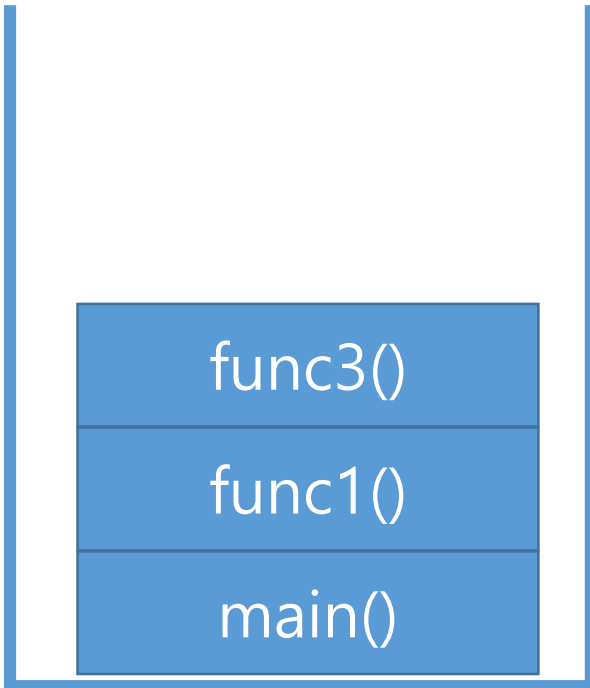
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (7/13)

```
jln-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jln-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3
This is func4 ←
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

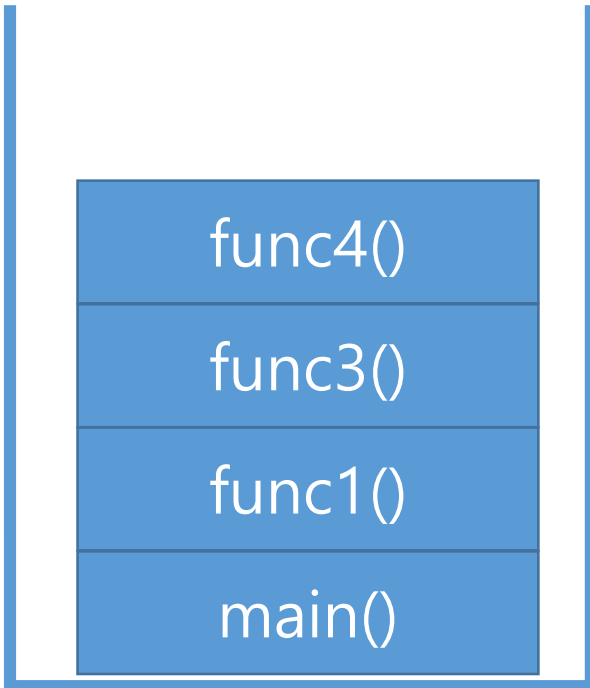
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (8/13)

```
jln-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jln-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3
This is func4 ←
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void func2(void) {
    puts("This is func2");
}

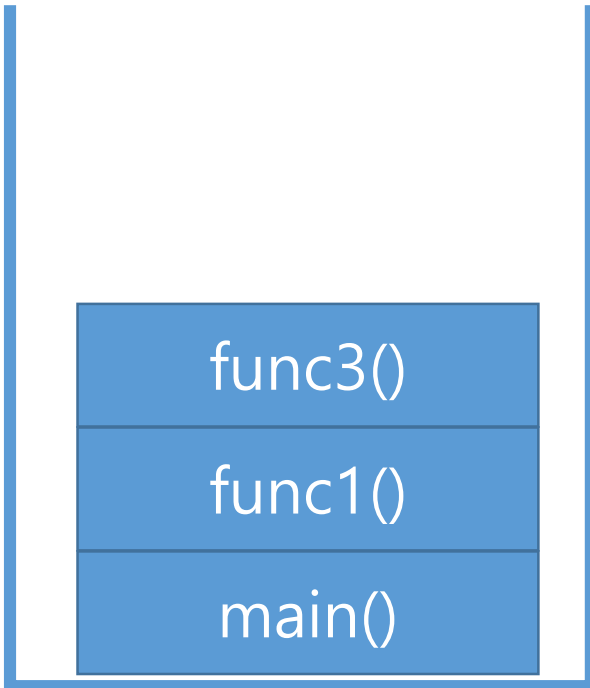
void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}

void main(void) {
    puts("This is main");
    func1();
    func5();
}

void func1(void) {
    puts("This is func1");
    func2();
    func3();
}
```



# 호출 스택

## • 호출 스택 동작 과정 (9/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3
This is func4 ←
This is func5
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

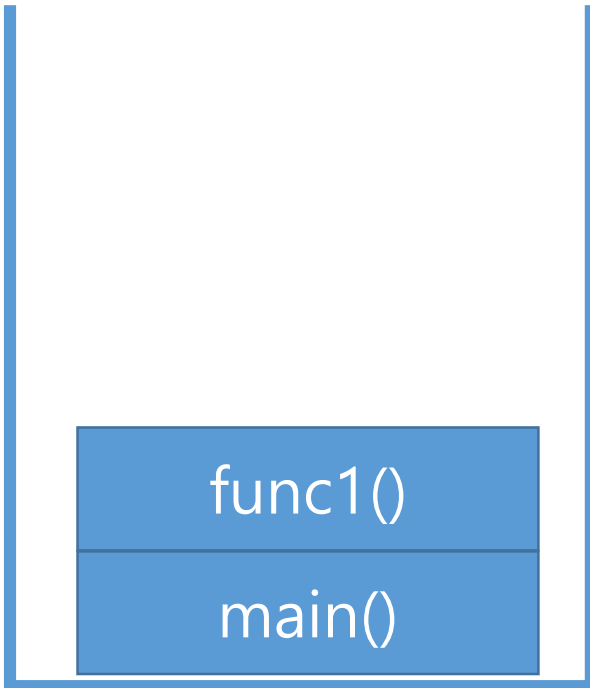
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (10/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3
This is func4 ←
This is func5
```

```
/* CallStack.c */
#include <stdio.h>
```

```
void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);
```

```
void main(void) {
    puts("This is main");
    func1();
    func5();
}
```

```
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}
```

```
void func2(void) {
    puts("This is func2");
}
```

```
void func3(void) {
    puts("This is func3");
    func4();
}
```

```
void func4(void) {
    puts("This is func4");
}
```

```
void func5(void) {
    puts("This is func5");
}
```



main()



# 호출 스택

## • 호출 스택 동작 과정 (11/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
jin-desk@JIN-DESK:~$ ./cs
This is main
This is func1
This is func2
This is func3
This is func4
This is func5 ←
```

```
/* CallStack.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

void main(void) {
    puts("This is main");
    func1();
    func5();
}

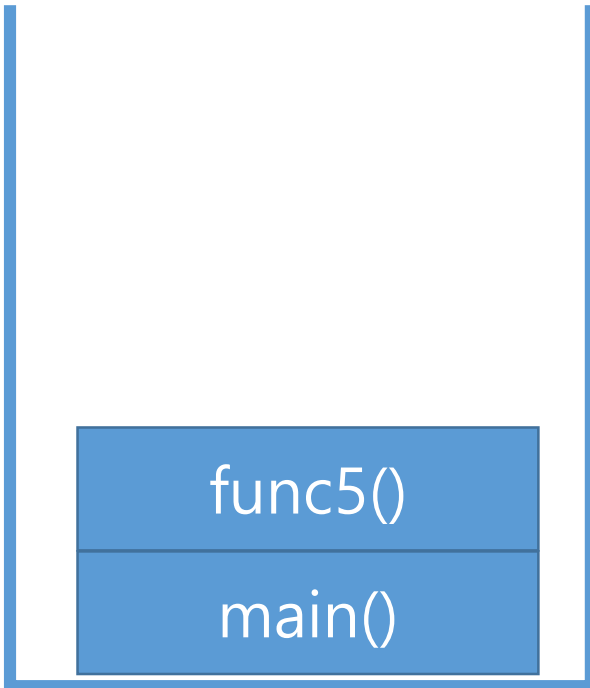
void func1(void) {
    puts("This is func1");
    func2();
    func3();
}

void func2(void) {
    puts("This is func2");
}

void func3(void) {
    puts("This is func3");
    func4();
}

void func4(void) {
    puts("This is func4");
}

void func5(void) {
    puts("This is func5");
}
```



# 호출 스택

## • 호출 스택 동작 과정 (12/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
```

```
jin-desk@JIN-DESK:~$ ./cs
```

```
This is main  
This is func1  
This is func2  
This is func3  
This is func4  
This is func5 ←
```

```
/* CallStack.c */  
#include <stdio.h>
```

```
void func1(void);  
void func2(void);  
void func3(void);  
void func4(void);  
void func5(void);
```

```
void main(void) {  
    puts("This is main");  
    func1();  
    func5();  
}
```

```
void func1(void) {  
    puts("This is func1");  
    func2();  
    func3();  
}
```

```
void func2(void) {  
    puts("This is func2");  
}
```

```
void func3(void) {  
    puts("This is func3");  
    func4();  
}
```

```
void func4(void) {  
    puts("This is func4");  
}
```

```
void func5(void) {  
    puts("This is func5");  
}
```



main()



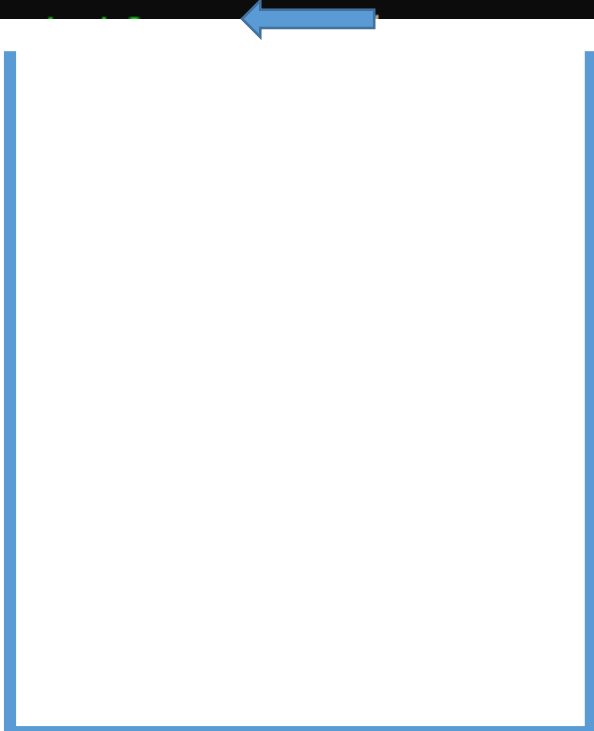
# 호출 스택

## • 호출 스택 동작 과정 (13/13)

```
jin-desk@JIN-DESK:~$ gcc -o cs CallStack.c
```

```
jin-desk@JIN-DESK:~$ ./cs
```

```
This is main  
This is func1  
This is func2  
This is func3  
This is func4  
This is func5
```



```
/* CallStack.c */  
#include <stdio.h>
```

```
void func1(void);  
void func2(void);  
void func3(void);  
void func4(void);  
void func5(void);
```

```
void main(void) {  
    puts("This is main");  
    func1();  
    func5();  
}
```

```
void func1(void) {  
    puts("This is func1");  
    func2();  
    func3();  
}
```

```
void func2(void) {  
    puts("This is func2");  
}
```

```
void func3(void) {  
    puts("This is func3");  
    func4();  
}
```

```
void func4(void) {  
    puts("This is func4");  
}
```

```
void func5(void) {  
    puts("This is func5");  
}
```



# 호출 스택

- 호출 스택 문제

- Synology의 CallStackProb.c 출력 결과 예측해보기

```
/* CallStackProb.c */
#include <stdio.h>

void func1(void);
void func2(void);
void func3(void);
void func4(void);
void func5(void);

int main(void) {
    func1();

    return 0;
}

void func1(void) {
    puts("Start of func1");

    puts("This is func1");
    func2();
    func3();

    puts("End of func1");
}

void func2(void) {
    puts("Start of func2");

    func4();
    puts("This is func2");
    func5();

    puts("End of func2");
}

void func3(void) {
    puts("Start of func3");

    puts("This is func3");

    puts("End of func3");
}

void func4(void) {
    puts("Start of func4");

    func3();
    puts("This is func4");

    puts("End of func4");
}

void func5(void) {
    puts("Start of func5");

    puts("This is func5");

    puts("End of func5");
}
```

# 호출 스택

- 호출 스택 문제
  - Synology의 CallStackProb.c 출력 결과 예측해보기

```
jinn-desk@JIN-DESK:~$ gcc -o csp CallStackProb.c
jinn-desk@JIN-DESK:~$ ./csp
Start of func1
This is func1
Start of func2
Start of func4
Start of func3
This is func3
End of func3
This is func4
End of func4
This is func2
Start of func5
This is func5
End of func5
End of func2
Start of func3
This is func3
End of func3
End of func1
```

# 목 차

---

- StackFrame의 필요성

- 목표
- 계획

- Stack이란

- 변수의 종류

- 지역, 전역, 정적

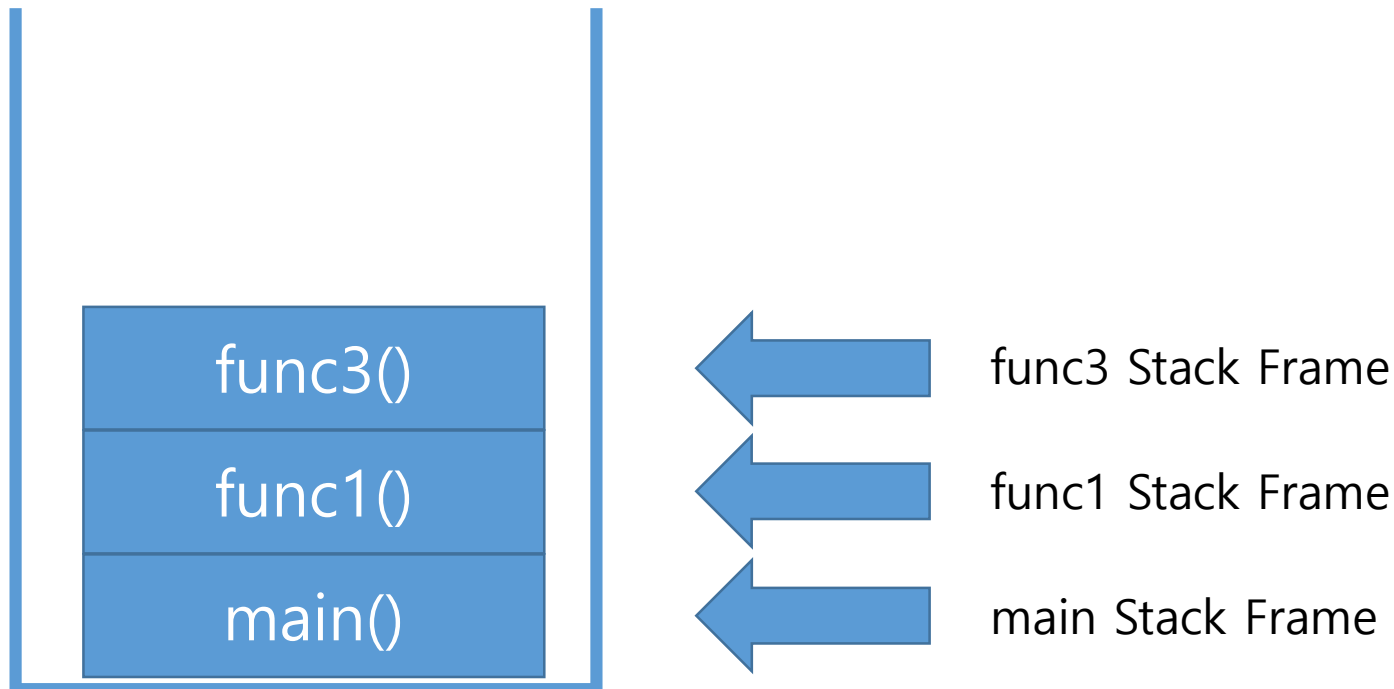
- 호출 스택

- Stack Frame이란

# Stack Frame이란

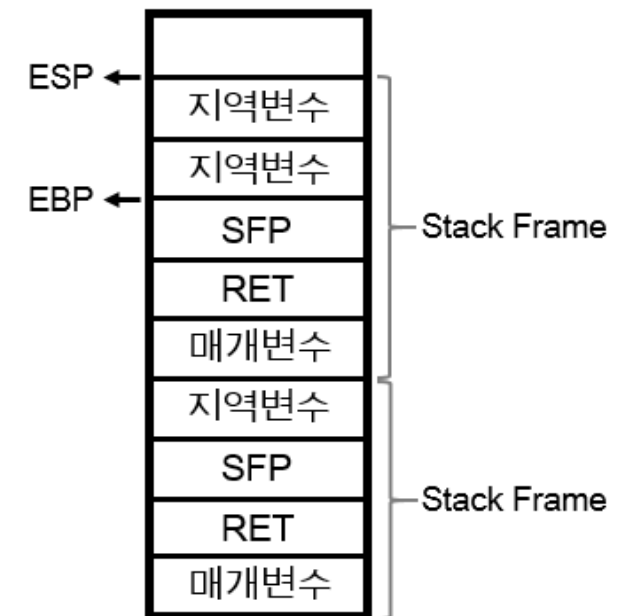
- Stack Frame이란

- 함수를 호출 할 때 정보를 담고 있는 한 단위
  - 함수의 정보
    - 현재 함수의 지역 변수, rtn address



# Stack Frame이란

- 스택 프레임(Stack Frame)
  - 함수 호출 시 사용되는 스택 공간
- ESP(Extend Stack Pointer)
  - 스택 프레임의 가장 최상부, 현재위치
- EBP(Extend Base Pointer)
  - 스택 프레임이 시작된 주소
- RET
  - 복귀할 함수의 주소
- SFP(Saved Frame Pointer)
  - EBP 복귀 주소 저장



---

감사합니다!