Chapter 3 Pointers and Functions

Stack

- stack영역은 function을 실행하는데 필요한 메모리다.
- heap과 memory를 같은 영역에 한다.
- stack은 memory에 낮은 영역을 사용하고, heap은 memory에 높은 영역부터 사용한다.

Stack과 Stack frame의 차이점 ⭐

1. Stack (스택)

- **정의**: 스택은 메모리의 한 영역으로, 데이터를 LIFO(Last In, First Out) 방식으로 저장하고 관리합니다. 즉, 마지막에 저장된 데이터가 가장 먼저 읽혀집니다.
- **용도**: 주로 함수 호출 시의 지역 변수 저장, 함수 호출 및 복귀 주소 저장, 함수 호출 순서 관리 등에 사용됩니다.
- **구조**: 스택은 메모리의 연속적인 블록으로 구성되어 있으며, `push`와 `pop` 연산을 통해 데이터를 추가하거나 제거합니다.

2. Stack Frame (스택 프레임)

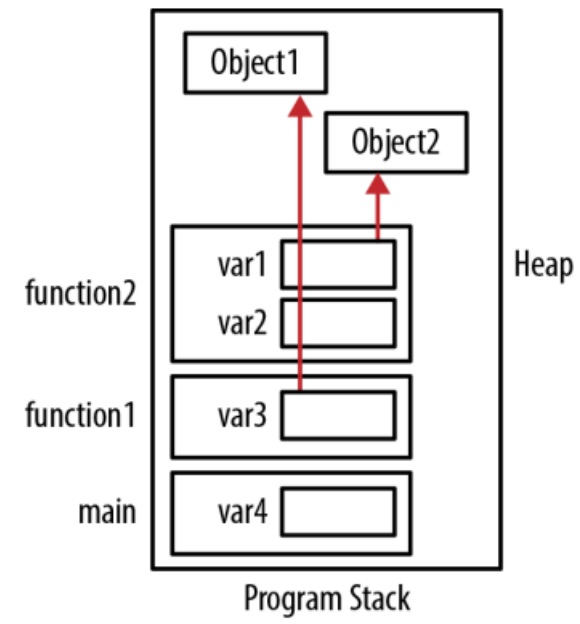
- **정의**: 스택 프레임은 함수 호출 시 스택에 할당되는 메모리 블록을 의미합니다. 각 스택 프레임은 함수의 실행 컨텍스트를 담고 있으며, 함수 호출 시 생성되고 함수 반환 시 소멸합니다.
- **구성 요소**: 스택 프레임은 보통 다음과 같은 요소를 포함합니다:
  - **지역 변수**: 함수 내에서 선언된 변수.
  - **매개변수**: 함수에 전달된 인자.
  - **복귀 주소**: 함수 호출 후 돌아갈 주소.
  - **프레임 포인터**: 현재 스택 프레임의 시작 위치를 가리키는 포인터.
- **작동**: 함수가 호출될 때 새로운 스택 프레임이 스택에 추가되고, 함수가 반환되면 해당 스택 프레임이 제거됩니다.

스택과 스택 프레임의 관계

- **스택**은 메모리의 전체 영역을 의미하며, 여러 스택 프레임이 스택 위에 쌓이게 됩니다.
- **스택 프레임**은 스택에 저장되는 개별적인 데이터 블록으로, 각 함수 호출마다 새로운 스택 프레임이 생성됩니다.

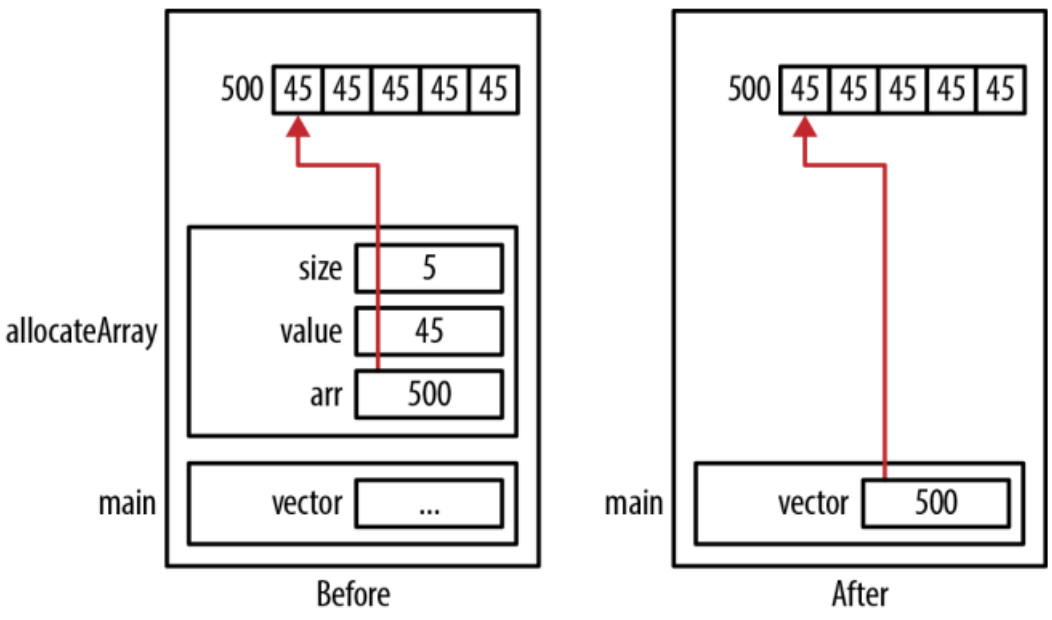
Code Example

```c
#include <stdio.h>
#include <stdlib.h>

void function2()
{
    int *ptr = (int*)malloc(sizeof(int));
    int var2;
    printf("Program Stack Example\n");
}
void function1()
{
    int *ptr2 = (int*)malloc(sizeof(int));
    function2();
}

int main()
{
    int var4;
    function();
    return (0);
}
```



Returning allocated pointer

```c
#include <stdio.h>
#include <stdlib.h>

int *allocateArray(int size, int value)
{
    int *arr = (int*)malloc(sizeof(int)*size);
    for (int i = 0; i < size; i++)
        arr[i] = value;
    return (arr);
}

int main(void)
{
    int *vec = allocateArray(5, 45);
    for(int i =0; i <  5; i++)
        printf("%d ", vec[i]);

    return (0);
}
```

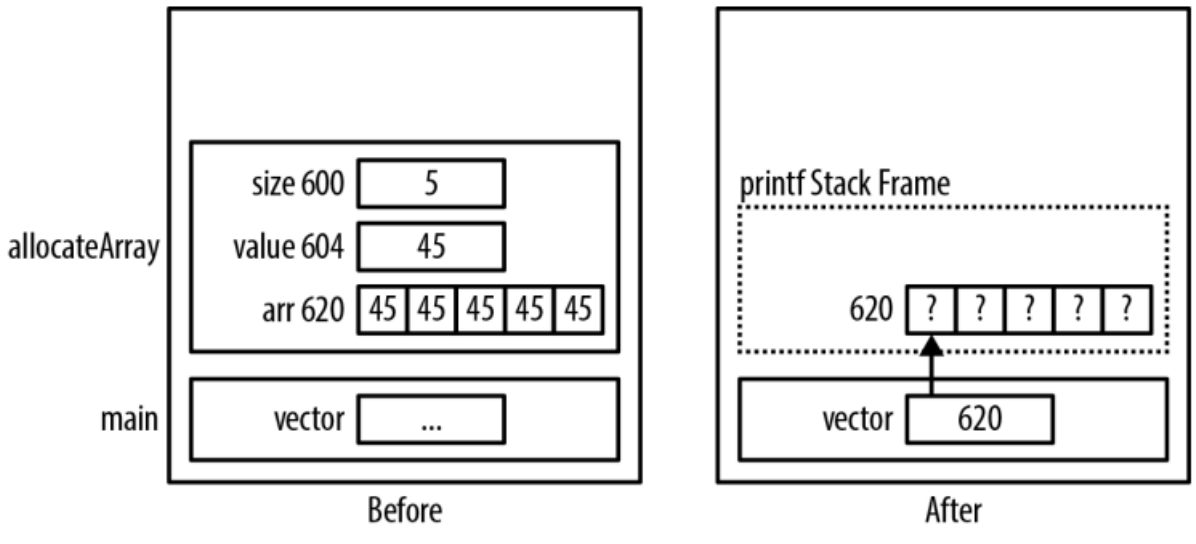


- stackframe에서 `malloc` function을 사용해서 heap에 메모리를 요청을 한 뒤에 그 시작점은 main stackframe에 있는 vector포인터 변수가 받는다.

Return Local Data

```c
#include <stdio.h>
#include <stdlib.h>

int *allocateArray(int size, int value)

{
    int arr[size];
    for (int i = 0; i < size; i++)
        arr[i] = value;
    return (arr);
}

int main(void)
```

```
{
    int *vec = allocateArray(5, 45);
    for(int i =0; i <  5; i++)
        printf("%d ", vec[i]);

    return (0);
}
```



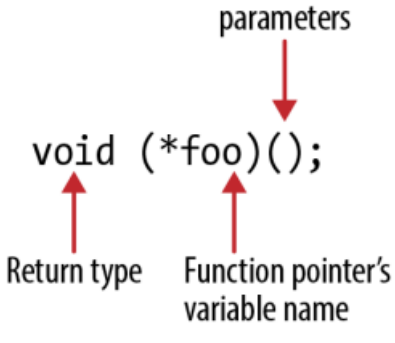
- Function stackframe이 stack에서 pop이 된 뒤라서 arr를 return하는 것은 의미가 없다.
- static으로 선언하면 괜찮아진다!

Function Pointer

정의 : Function의 address를 가지고 있는 포인터를 말한다.
장점

- 함수를 실행하는 다양한 방법을 제공한다.
  단점
- program의 실행을 느리게 하는 원인이 된다.
  - processor가 파이프라이닝을 통한 branch prediction을 하지 못하게 한다.

Declaration Function Pointer



주의점

📌 function pointer를 사용할 때는, C가 정확한 parameter가 들어갔는지 확인하지 않기 때문에 조심히 사용해야한다.