CAPS

C언어 스터디
3.0주차

<함수 && 포인터>

CAPS

함수

```
출력형식 이름(입력형식){
        내용
        return 출력;
}
```

출력 형식 : 함수의 값을 어떻게 뱉을 것인지 (void, int, double ..)

입력 형식 : 함수의 입력 값을 어떻게 뱉을 것인지 (void, int, double)

(int x, double y, char c,int k) 와 같이 여러 개를 받아도 됨.

return : 함수값을 뱉음

출력형식이 void이면 return; , 출력이 있다면 return 출력;

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int add(int x, int y) { // 함수 선언 및 정의(함수 선언은 무조건 main 위여야 합니다.)
        int tmp = x + y;
        return tmp;
        printf("H");
}
int main(void) {
        int a, b;
        scanf("%d %d", &a, &b);
        int k = add(a, b);
        printf("%d\n", k);
        printf("%d", add(a, b));
}
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int add(int x, int y); // 함수 선언 (함수 선언은 무조건 main 위여야 합니다.)
int main(void) {
        int a, b;
        scanf("%d %d", &a, &b);
        int k = add(a, b);
        printf("%d\n", k);
        printf("%d", add(a, b));
}
int add(int x, int y) { // 함수 정의
        int tmp = x + y;
        return tmp;
        printf("H");
}
```

함수를 배우는 이유

1. 편의성, 안정성, 재사용성
2. %%재귀를 쓸 수 있음%%

```
#include<stdio.h>
int add(int a, int b) {
	return a+b;
	printf("나는 출력이 될까요?\n"); //출력이 안 됩니다. return이 되는 순간 해당 함수는 꺼집니다. main 함수도 return하면 바로 꺼집니다.
}
void print_function(void) {
	printf("Hello\n");
	return; // 없어도 됩니다.
}
int main(void) {
	int n,m;
	n=100;
	m=10;
	printf("%d",add(n,m));
	while(m--)print_function(); //입력이 void이면 소괄호를 붙여야 합니다.
}
```

전역변수

중괄호를 벗어나면 변수는 사라집니다.!!!!!!!!!!!!!!!

```
#include<stdio.h>
int N = 1;
void func(void) {
	printf("%d", N);
}
int main(void) {
	func();
}
```

```
#include<stdio.h>
void func(void) {
	printf("%d", M);
}
int main(void) {
	int M = 2;
	func();
}
```

CAPS

재귀함수 (나중에 더 심도있게)

```c
#include<stdio.h>
void func(int x) {
	printf("A");
	func(x);
}
int main(void) {
	func(10);
} // 어떻게 될까요?
```

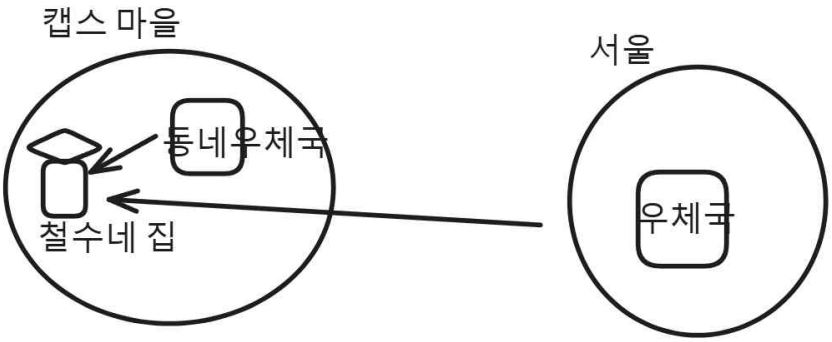
```c
#include<stdio.h>
void func(int x) {
	func(x);
	printf("A");
}
int main(void) {
	func(10);
} // 어떻게 될까요?
```

```c
#include<stdio.h>
void func(int x) {
	if (x == 0)return; //종단점이 있어야 하구요
	printf("%d ", x);
	func(x - 1); //재귀함수로 넘길 때 적어도 인자를 증감해야합니다.
}
int main(void) {
	func(10);
} // 어떻게 될까요?
```

```c
#include<stdio.h>
void func(int x) {
	if (x == 0)return; //종단점이 있어야 하구요
	func(x - 1); //재귀함수로 넘길 때 적어도 인자를 증감해야합니다.
	printf("%d ", x);
}
int main(void) {
	func(10);
} // 어떻게 될까요?
```

위의 출력 : 10 9 8 7 6 5 4 3 2 1
아래 출력 : 1 2 3 4 5 6 7 8 9 10
왜인지는 집에 가서 생각해 봅시다.

포인터



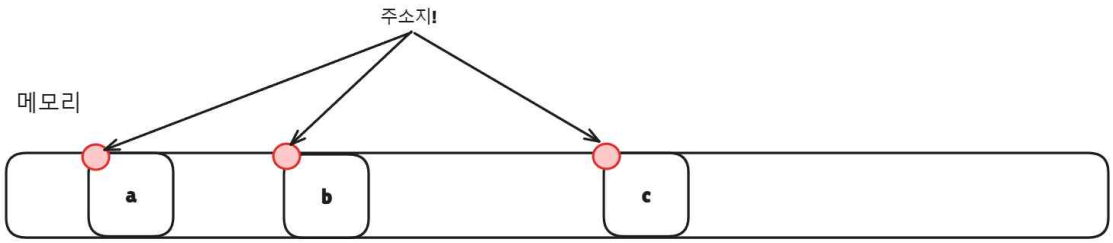
같은 캡스 마을에서
동네 우체국 : “철수네 집에 배달해주세요”
서울 : “철수네 집에 배달해주세요”
이러면 안되겠죠? 포인터는 “도로번주소”입니다.

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
void func(void) {
	a = 100;
}
int main(void) {
	int a = 2;

	func();

	printf("%d", a);
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
void func(int x) {
	x = 100;
}
int main(void) {
	int a = 2;

	func(a);

	printf("%d", a);
}
```

Call by Value 문제

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
void func(int* x) {
	*x = 100;
}
int main(void) {
	int a = 2;

	func(&a);

	printf("%d", a);
}
```



포인터 연산

1. &

가. 주소지를 뱉게 하는 "함수"

2. *

가. 주소지에 뛰어들게 하는 "함수"

나. 주소지를 담는 "자료형"

int, double과 같이 *도 '자료형'입니다.

```
int a=0;
double b=0;
* c=2;
```

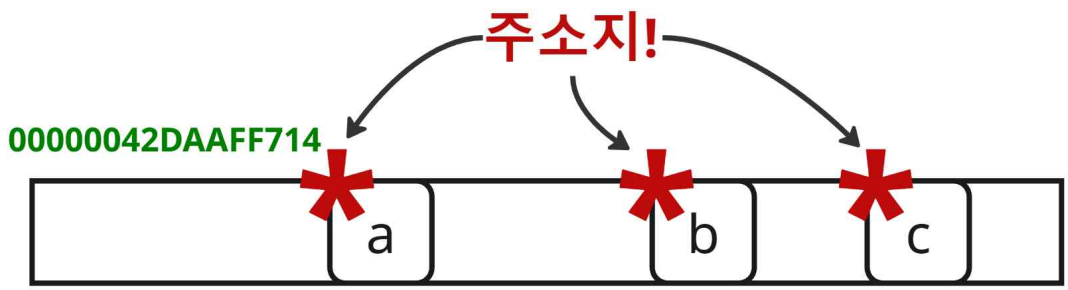
```
void* a  = 00000042DAAFF714; // "주소지"이다.
int* b   = 00000042DAAFF714; // "주소지"인데, int 크기만큼만 쓸 거다.
double* c = 00000042DAAFF714; // "주소지"인데, double 크기만큼만 쓸 거다.
```

int* 의 주인공은 int가 아니라 *입니다

&

주소지를 뱉게 하는 “함수”

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int main(void) {
	int a = 2;
	printf("%p", &a); // 00000042DAAFF714 (계속 바뀜)
}
```



*

1. 주소지에 뛰어들게 하는 “함수”

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int main(void) {
	int a = 2;
	printf("%p", &a); // 0000002441D9FC54 (계속 바뀜)
	printf("\n");
	printf("%d", *(&a)); // 2
}
```

2. 주소지를 담는 “자료형”

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int main(void) {
	int a = 2;
	printf("%p\n", &a); // 0000009D14D4FA24
	int* pa = &a; // a의 주소지를 pa에 저장
	printf("%p\n", pa); // 0000009D14D4FA24 a의 주소값을 저장 중
	printf("%d", *pa); // 2
}
```

Call by Value 문제

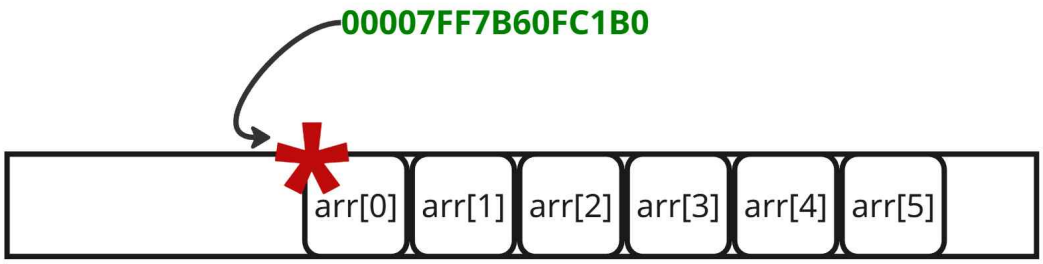
```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int main(void) {
	int a = 2;
	int b = 100;
	a += b;
	printf("%d", a);
}
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
void add(int x, int y) {
	x = x + y;
}
int main(void) {
	int a = 2;
	int b = 100;
	add(a, b);
	printf("%d", a);
}
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
void add(int* x, int* y) {
	printf("%x\n", x);
	*x = *x + *y;
}
int main(void) {
	int a = 2;
	int b = 100;
	add(&a, &b);
	printf("%d", a);
}
```

배열의 이름은 포인터다.

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int arr[10];
int main(void) {
	printf("%p\n", arr); // 00007FF7B60FC1B0
	printf("%p", &(arr[0])); // 00007FF7B60FC1B0
	// 배열의 이름은 배열의 첫번째 원소의 포인터 값입니다.
}
```




```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
int arr[10];
int main(void) {
	printf("%p\n", arr); // 00007FF62BD6C1B0
	printf("%p\n", arr + 1); // 00007FF62BD6C1B4
	// int의 크기는 '4'이기 때문에 '4' 가 차이나는 모습입니다.
	arr[1] = 100;
	printf("%d\n", *(arr + 1)); // 100

	double d_arr[100];
	printf("%p\n", d_arr); // 00000054859CF5C0
	printf("%p\n", d_arr + 1);// 00000054859CF5C8
}
```