상 명 튜 터 링

5 주 차 함 수

201721166 박의성

함 수 기본 적인 함수 구조

```
반환형 함수명(인수 목록){
    //함수 코드 내용
}
```

제약 조건

함수명: 함수의 기능을 잘 들어낼 수 있는 이름을 사용한다.

인수 목록: 함수의 정의에서 전달받은 인수를 함수
내부로 전달하기 위해 사용하는 변수이다.

반환형: void, int, float, char 등이 올 수 있고 이때 void는
반환값이 없는 함수를 작성할 때 사용된다.

반 복 문 while문을 중심으로

```c
#include <stdio.h>

int sum(int a, int b, int c){
	int sum = a + b + c;
	return sum;
}

float average(int a, int b, int c){
	float sum = a + b + c;
	return sum / 3;
}

void detect_bigger_number(int a, int b){
	if (a > b){
		printf("첫번째 인자 숫자가 더 큽니다.\n");
	}
	else{
		printf("두번째 인자 숫자가 더 큽니다.\n");
	}
}
```

```c
void printf_fn(int a){
	printf("숫자 %d가 입력되었습니다.\n", a);
}

void method_in_method(int a, int b, int c){
	printf("%d, %d, %d의 합은 %d 입니다.\n", a, b, c, sum(a,b,c));
}

int main(){
	int num_a=1, num_b=2, num_c=3;
	printf("%d, %d, %d의 합은 %d 입니다.\n", num_a, num_b, num_c,
	sum(num_a,num_b,num_c));
	printf("%d, %d, %d의 평균은 %.2f 입니다.\n", num_a, num_b, num_c,
	average(num_a,num_b,num_c));
	method_in_method(num_a, num_b, num_c);
	printf_fn(num_c);
	detect_bigger_number(num_a, num_c);
return 0;
}
```

반 복 문 while문을 중심으로

```c
#include <stdio.h>

int sum(int a, int b, int c){
	int sum = a + b + c;
	return sum;
}

int main(){
	int num_a=1, num_b=2, num_c=3;
	printf("%d, %d, %d의 합은 %d 입니다.\n",
		num_a, num_b, num_c, sum(num_a,num_b,num_c));
return 0;
}
```

```
1, 2, 3의 합은 6 입니다.
```

< 결 과 >

sum 함수에 매개변수로 1,2,3을 넣어주니

1,2,3의 합인 6이 반환되는 것을 알 수 있다.

반 복 문 while문을 중심으로

```c
#include <stdio.h>

float average(int a, int b, int c){
	float sum = a + b + c;
	return sum / 3;
}

int main(){
	int num_a=1, num_b=2, num_c=3;
	printf("%d, %d, %d의 평균은 %.2f 입니다.\n",
		num_a, num_b, num_c, average(num_a,num_b,num_c));
return 0;
}
```

```
1, 2, 3의 평균은 2.00 입니다.
```

< 결 과 >

average 함수에 매개변수로 1,2,3을 넣어주니 1,2,3의 평균인 2가 반환되는 것을 알 수 있다.

반 복 문 while문을 중심으로

```c
#include <stdio.h>

void method_in_method(int a, int b, int c){
	printf("%d, %d, %d의 합은 %d 입니다.\n", a, b, c, sum(a,b,c));
}


int main(){
	int num_a=1, num_b=2, num_c=3;
	method_in_method(num_a, num_b, num_c);
return 0;
}
```

```
1, 2, 3의 합은 6 입니다.
```

< 결 과 >

이처럼 함수 안에 함수를 사용하여 계산을 처 리 할 수 있 다 .

반 복 문 while문을 중심으로

```c
#include <stdio.h>

void printf_fn(int a){
	printf("숫자 %d가 입력되었습니다.\n", a);
}


int main(){
	int num_a=1, num_b=2, num_c=3;
	printf_fn(num_c);
return 0;
}
```

```
숫자 3이(가) 입력되었습니다.
```

< 결 과 >

매개변수로 입력받은 값을 그대로 출력해주는 것을 확인할 수 있다.

반 복 문 while문을 중심으로

```c
#include <stdio.h>

void detect_bigger_number(int a, int b){
	if (a > b){
		printf("첫번째 인자 숫자가 더 큽니다.\n");
	}
	else{
		printf("두번째 인자 숫자가 더 큽니다.\n");
	}
}

int main(){
	int num_a=1, num_b=2, num_c=3;
	detect_bigger_number(num_a, num_c);
return 0;
}
```

```
두번째 인자 숫자가 더 큽니다.
```

< 결 과 >

함수에서는 조건문과 반복문 모두를 실 행 시 킬 수 있 습 니 다 .