KG 아이티뱅크

C언어

VISUAL STUDIO

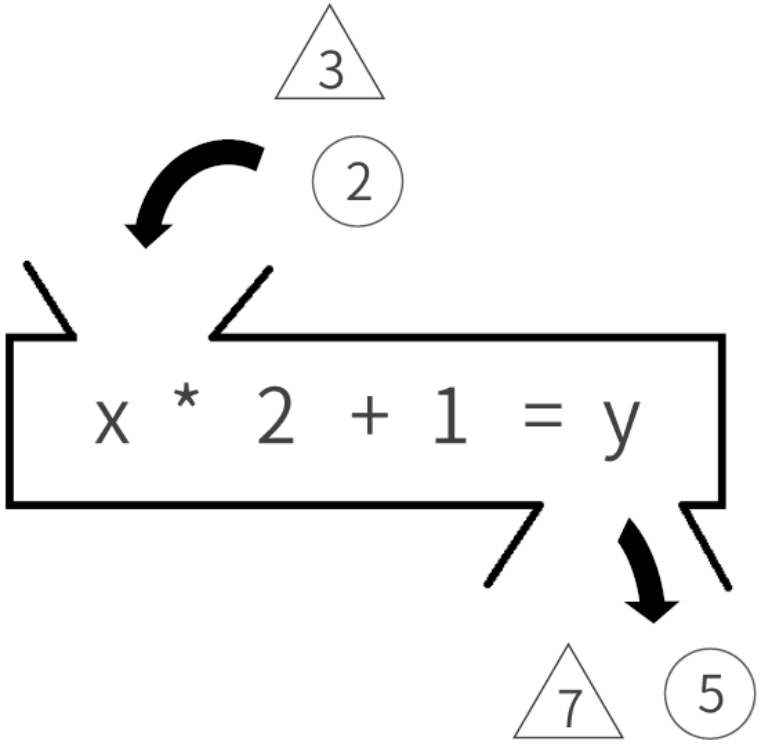
함수

함수

❖ 함수(Function) : 함수는 특정한 작업을 처리하기 위한 코드. 프로그램에서 동일하게 반복되는 부분을 독립된 프로그램(코드 블록)으로 한 번만 만들어 두고 필요할 때마다 호출해서 사용

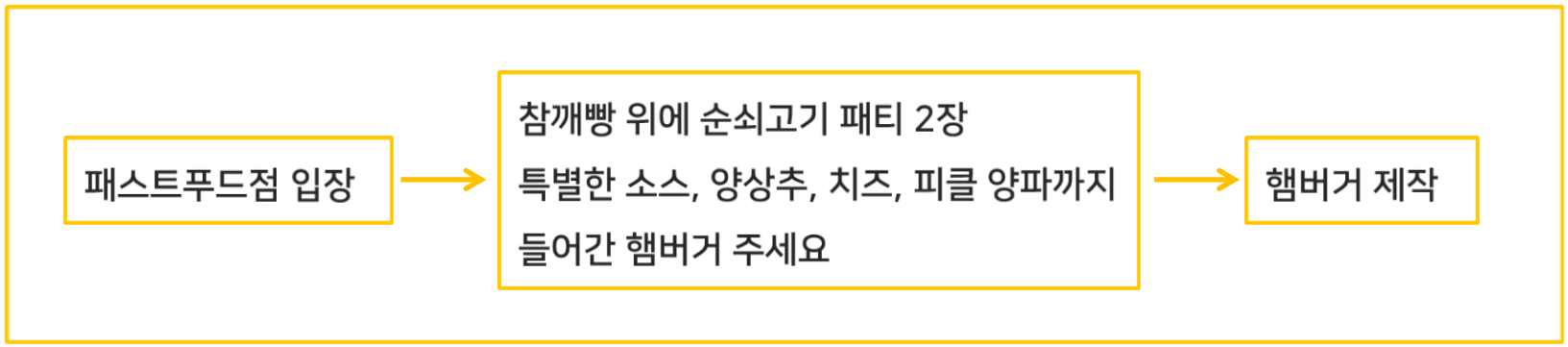
- 수학에서의 함수

2x + 1 = y

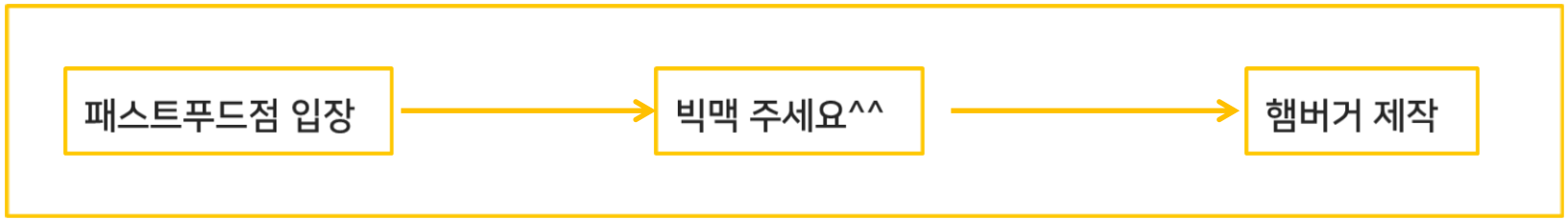


함수

- 패스트푸드점에서 햄버거 주문



- 빅맥이라는 이름으로 함수화



즉, 자세한 코드를 몰라도 함수의 이름만 가지고도 똑같은 기능을 수행할 수 있음

ex) printf(), scanf() 등

함수란?

■ 자판기에서의 함수

- 구입이라는 '함수의 이름'
  함수의 이름이 있어야 호출

- 물건을 사기 위한 돈 '입력'
  함수 호출 시 데이터 입력

- 선택한 제품 구매를 위한 '기능'
  데이터에 따른 함수의 동작방식

- 작업완료에 따른 '출력'
  함수 호출 완료 시 출력

함수란?

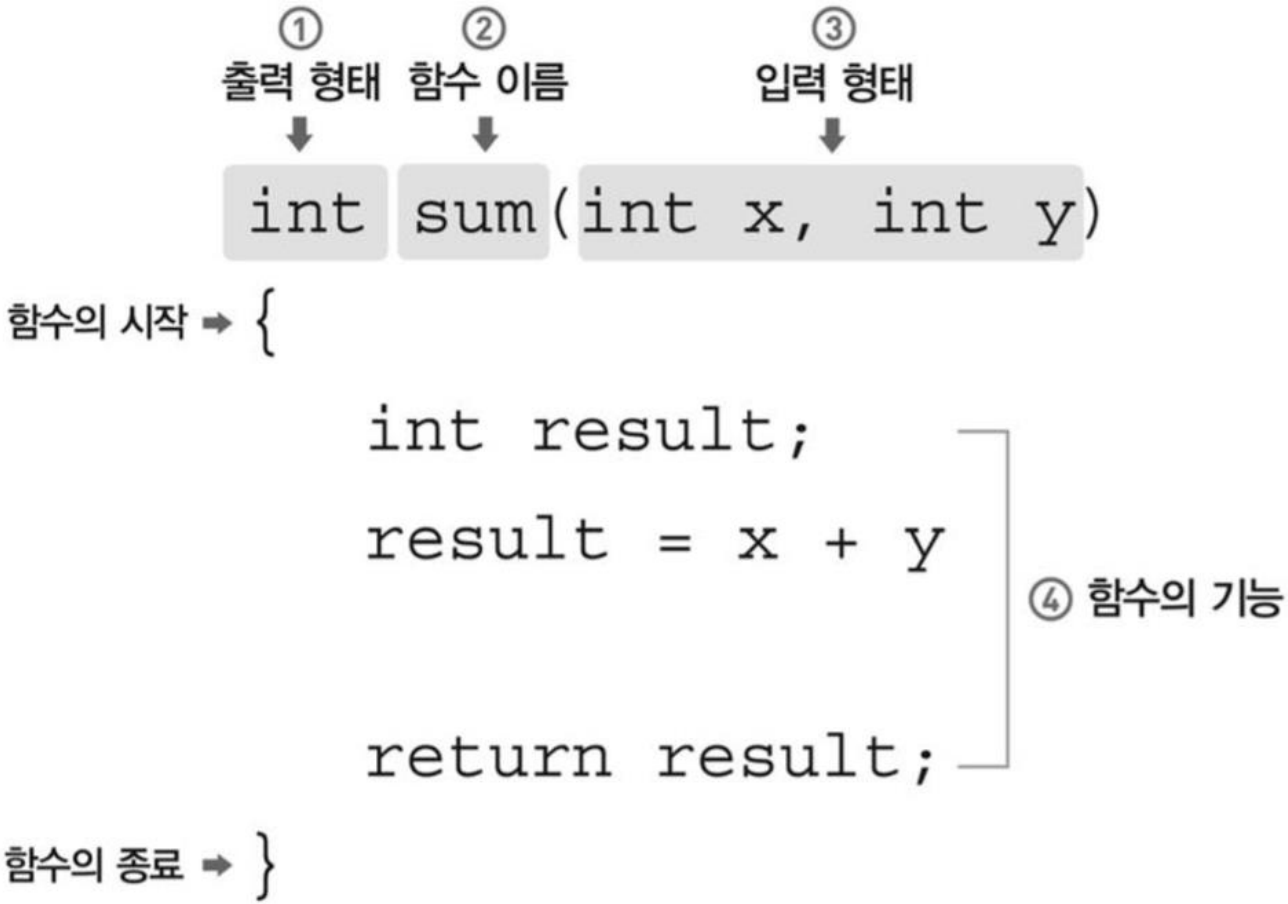
- 독립적으로 특정한 기능을 가지는 프로그램

- 함수 사용의 장점
  - 언제나 재사용 가능
  - 프로그램의 가독성 향상
  - 에러검출 용이
  - 기능별 프로그램 관리가 용이

- 함수의 종류
  - 표준 라이브러리 함수 ex> printf(), scanf()
  - 사용자 정의 함수

함수의 형태

함수의 기본 형태



함수의 형태

함수의 4가지 형태

- .int sum (int x, int y) → '출력 값' 과 '입력 값' 이 있다
- void sum (int x, int y) → '출력 값' 은 없고, '입력 값' 은 있다
- .int sum (void) → '출력 값' 은 있고, '입력 값' 은 없다
- void sum (void) → '출력 값' 과 '입력 값' 이 없다

함수의 1형태

입력 값(매개변수)과 출력 값(반환형)이 있는 함수

```
int sum(int x, int y)
{
    int result;
    result = x + y;

    return result;
}
```

함수의 2형태

입력 값(매개변수)은 있고 출력 값(반환형)이 있는 함수

```
void sum(int x, int y)
{
    int result;
    result = x + y;

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

함수의 3형태

입력 값(매개변수)이 없고 출력 값(반환형)만 있는 함수

```
int sum(void)
{
    int x,y,result;
    scanf("%d%d",&x,&y);
    result = x + y;

    return result;
}
```

함수의 4형태

입력 값(매개변수)과 출력 값(반환형)이 없는 함수

```
void sum(void)
{
    int x,y,result;
    scanf("%d%d",&x,&y);
    result = x + y;

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

함수의 호출

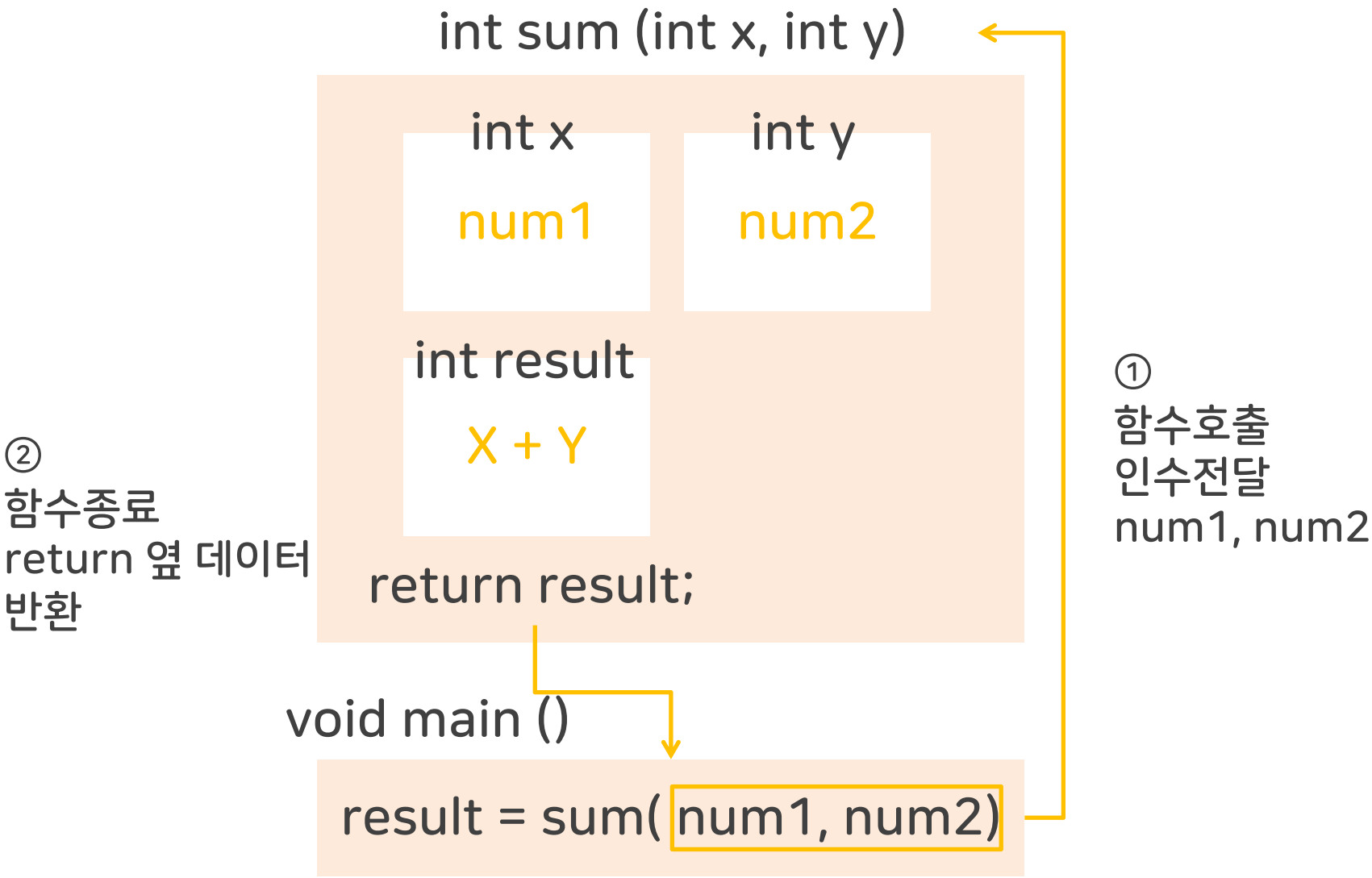
<파일이름 : 01.함수.c>

```
#include<stdio.h>

int sum(int x, int y) {
    int result;
    result = x + y;

    return result;
}

void main() {
    int num1, num2,result;
    printf("숫자 두 개를 입력하세요 : ");
    scanf("%d%d", &num1, &num2);
    result = sum(num1, num2);
    printf("합계 : %d₩n", result);
}
```

sum 함수 반환

sum 함수 호출

함수의 호출


int sum (int x, int y)
int x
num1
int y
num2
int result
X + Y
return result;
②
함수종료
return 옆 데이터
반환
①
함수호출
인수전달
num1, num2
void main ()
result = sum(num1, num2)

함수의 호출

<파일이름 : 02.함수.c>

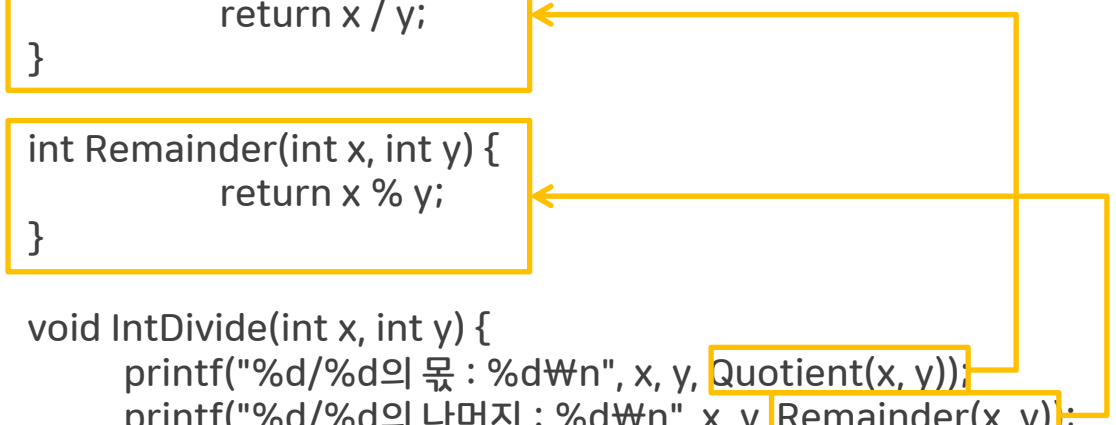
```
#include<stdio.h>

void mul(int x, int y)
{
	int result = x * y;
	printf("%d * %d = %d₩n", x, y,result);
}

void main()
{
	int num1, num2;
	while (1) {
		printf("숫자를 두 개 입력하세요 (0을 두 번 입력하면 종료) : ");
		scanf("%d%d", &num1, &num2);
		if (num1 == 0 && num2 == 0) {
			printf("종료합니다.₩n");
			break;
		}
	mul(num1, num2);
	}
}
```

함수의 재사용이 가능하다
한 번 선언한 함수는 계속 사용가능

함수의 호출

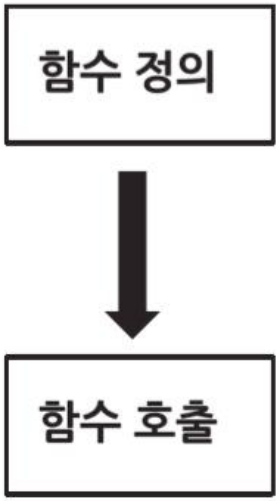
<파일이름 : 03.함수.c>

```
#include<stdio.h>

int Quotient(int x, int y) {
        return x / y;
}

int Remainder(int x, int y) {
        return x % y;
}

void IntDivide(int x, int y) {
    printf("%d/%d의 몫 : %d₩n", x, y, Quotient(x, y));
    printf("%d/%d의 나머지 : %d₩n", x, y, Remainder(x, y));
}

void main()
{
    int num1, num2;
    printf("나누기 할 두 수를 입력 : ");
    scanf("%d%d", &num1, &num2);
    IntDivide(num1, num2);
    printf("₩n");
}
```



호출한 함수안에서도 다른 함수를 호출가능

함수의 호출



정상

컴파일 방향

```
#include <stdio.h>

int sum(int x, int y)
{
    int result=0;
    result=x+y;
    return result;
}

int main(void)
{
    int answer= 0;
    answer=sum(3, 4);
    printf("%d ₩n", answer);

    return 0;
}
```

위에 sum( ) 함수가 있나? 있네~

함수의 호출

정상

컴파일 방향

```
#include <stdio.h>

int sum(int x, int y);

int main(void)
{
     int answer= 0;
     answer=sum(3, 4);
     printf("%d ₩n", answer);

     return 0;
}

int sum(int x, int y)
{
     int result=0;
     result=x+y;
     return result;
}
```

위에 sum( ) 함수가 있나? 있네~

함수의 호출

<파일이름 : 04.함수.c>

```
#include<stdio.h>
float avg(int x, int y);
void show_avg(int, int, float);//미리 함수의 선언만 할 때는 자료형만 적어줘도 됨

void main() {
	int num1, num2; float f;
	printf("숫자를 두 개 입력하세요 : ");
	scanf("%d%d",&num1, &num2);
	f = avg(num1, num2);
	show_avg(num1, num2, f);
}

float avg(int x, int y)
{
	float f;
	f = (float)(x + y) / 2;
	return f;
}

void show_avg(int x, int y, float f)
{
	printf("%d와 %d의 평균값은 %.2f입니다.₩n", x, y, f);
}
```

위에서 먼저 함수선언 후 아래쪽에 함수의 내용을 적어도 된다.

함수의 호출

<파일이름 : 05.함수.c>

```
#include<stdio.h>
char compare(int x, int y);

void main() {
	int num1, num2;
	char size;
	printf("숫자를 두 개 입력하세요 : ");
	scanf("%d%d", &num1, &num2);
	size = compare(num1, num2);

	if (size == 'Y')
		printf("%d는 %d보다 크다.₩n", num1, num2);
	else if (size == 'N')
		printf("%d는 %d보다 작다.₩n", num1, num2);
	else if (size == '=')
		printf("%d는 %d와 같다.₩n", num1, num2);
}

char compare(int x, int y) {
	if (x > y)
		return 'Y';
	else if (x < y)
		return 'N';
	else
		return '=';
}
```

문제

<파일이름 : 06.문제.c>, <파일이름 : 07.문제.c>, <파일이름 : 08.문제.c>

[문제1] 두 수를 입력 받아 큰 수를 출력하는 함수를 만드시오
<변수선언> 정수형 변수 num1,num2,result 선언
<함수선언> 정수형을 반환하고 정수형 변수 x와 y를 매개변수로 하는 compare 함수를 선언 (함수의 1형태)

[문제2] 입력 받은 값이 짝수인지 홀수인지 판별하는 함수 제작
<변수선언> 정수형 변수 num1선언
<함수선언> 반환형은 없고 정수형 변수 x를 매개변수로 하는 EvenOdd함수 선언 (함수의 2형태)

[문제3] 절대값을 구하는 함수를 정의하시오
<변수선언> 정수형 변수 x를 Avalue함수 안에 선언
<함수선언> 정수형을 반환하고 매개변수가 없는 Avalue함수 선언
(함수의 3형태)

문제

<파일이름 : 09.문제.c>, <파일이름 : 10.문제.c>

[문제4] 1 부터 입력한 숫자까지 중에서 3의 배수만 출력하시오.
<변수선언> ThreeMul 함수 안에 정수형 변수 x, y를 선언
<함수선언> 반환형이 없고 매개변수도 없는 ThreeMul함수를 선언
While문을 이용하여 1부터 입력 받은 숫자까지의 3의 배수를 출력

[문제5] main에서 숫자를 입력 받고 그 수를 거꾸로 반환하지 않는 함수를 만드시오.
<변수선언> 정수형 변수 num선언
<함수선언> 반환형이 없고 정수형 변수 x를 매개변수로 하는 Reverse함수 선언
While문을 이용하여 입력 받은 숫자를 거꾸로 뒤집어서 출력하는 함수 선언
%와 / 연산자 사용

문제

<파일이름 : 11.문제.c>

[문제6] main에서 두 수를 입력 받아 그 두 수로 사칙연산을 하는 함수 네 개를 만드시오.
<변수선언>정수형 변수 num1,num2,menu선언
<함수선언>정수형을 반환하는 sum, sub, multi, divi 함수를 main지역에 선언
While반복문과 switch선택문, system함수를 이용하여 한 번 실행 후 다시 다른 수를 입력 받을 수 있도록 정의

<출력>
숫자를 두 개 입력하세요 : 5 10
계산할 방식을 선택하세요 :
1.더하기
2.빼기
3.곱하기
4.나누기
5 * 10 = 50
계속하려면 아무 키나 누르십시오 . . .

지역변수

<파일이름 : 01.지역변수.c>

```
#include<stdio.h>
void func_A(void);
int func_B(void);

void main(void)
{
	int aa = 10;
	printf("함수 호출 전 main()함수의 aa 값 : %d₩n", aa);
	func_A();
	printf("func_A() 함수 호출 후 main()함수의 aa 값 : %d₩n", aa);		//aa는 메인함수의 지역변수
	printf("func_B() 함수 호출 후 main()함수의 aa 값 : %d₩n", func_B());//func_B()함수의 리턴 값 aa
}
void func_A()
{
	int aa = 20;
	int bb = 30;
	printf("func_A()함수의 aa값 : %d₩n", aa);
	printf("func_A()함수의 bb값 : %d₩n", bb);
}
int func_B()
{
	int aa = 35;		//func_B()함수의 지역변수 aa 선언
	return aa;		//aa는 35로 리턴
}
```

지역변수 : 각 함수가 끝나면 사라지는 변수

지역변수

<파일이름 : 02.지역변수.c>

```
#include<stdio.h>

void main()
{
	for (int i = 1; i < 3; i++)//for반복문 내부의 지역변수 total선언
	{
		int total = 0;
		total += i;
		printf("for의 total값은 %d입니다.₩n", total);
	}
	if (total < 10)
	{
		printf("if의 total값은 %d입니다.₩n", total);
	}
}
```

for함수 안에서 total을 선언하면 for문이 끝나는 순간 없어짐

전역변수

<파일이름 : 03.전역변수.c>

```
#include<stdio.h>
int num;  //전역변수 선언, 초기화하지 않아도 0이 된다.
void grow();

void main()
{
	printf("함수 호출 전 num의 값 : %d₩n", num);
	grow();
	printf("함수 호출 후 num의 값 : %d₩n", num);
}
void grow()
{
	num = 60;
	printf("grow 함수 내부 num의 값 : %d₩n", num);
}
```

전역변수 선언 시 전 지역에서 사용 가능

정적변수

<파일이름 : 04.정적변수.c>

```
#include<stdio.h>

void sv() {
	static int a = 10;//정적변수, 함수 실행 시 단 한번만 초기화(계속 축적)
	int b = 10;		//지역변수, 함수 실행 시 매번 초기화
	a++;
	b++;
	printf("static 변수 a의 값 : %d, 지역변수 b의값 : %d₩n", a, b);

}

void main()
{
	int i;
	for (i = 0; i < 5; i++) {
		sv();
	}
	//printf("%d", a);//정적변수는 함수내에서만 사용가능
}
```

정적변수는 단 한번만 초기화가 되고 함수 내에서는 안 사라짐
하지만 전역변수처럼 다른 함수에도 남아있는게 아니라 그 함수안에서만 존재

정적변수

<파일이름 : 05.정적변수.c>

```
#include<stdio.h>

void count();
void main()
{
	int num = 0;
	while (num < 3)
	{
		count();
		num++;
	}
}
void count(void)
{
	static int x = 0;
	int y = 0;
	x += 1;
	y += 1;
	printf("x값 : %d, y값 : %d₩n", x, y);
}
```