C언어 강의자료

문정욱

C언어 맛 보기 3

프로그램의 구조 표현: 인덴트(indent)

구조 파악이 어려운 소스코드

```
#include <stdio.h>

        int main(void)
        {
    int a;

scanf("%d", &a);

            if(a == 1)
           printf("a == one\n");
printf("a == one\n");

                          else
     printf("a == other\n");

               printf("after if-stat\n");
return 0;
    }
```

구조를 파악하기 힘들어서 오류를 찾기가 어렵다.

구조 파악이 쉬운 소스코드

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
        printf("a == one\n");

    else
        printf("a == other\n");

    printf("after if-stat\n");
    return 0;
}
```

a if-statement

a statement

else statement ???

Syntax Error !!!

구조를 파악하기 쉬워서 오류를 찾기가 쉽다.

프로그램의 구조 표현: 인덴트(indent)

함수 내부 선언 및 문장들

```
#include <stdio.h>

int main(void)
{
    int a;

    printf("Input a integer number: ");
    scanf("%d", &a);
    printf("a = %d\n", a);
    return 0;
}
```

Beginning of function: 1st column

Ending of function: 1st column

If문 안의 선언 및 문장들

```
#include <stdio.h>

int main(void)
{
    int a;

    printf("Input a integer number: ");
    scanf("%d", &a);
    if(a%2 == 1) {
        printf("a is a odd number\n");
        printf("a is a odd number\n");
        printf("a is a odd number\n");
    }
    return 0;
}
```

Beginning of if: last column

Ending of if: same column

프로그램의 구조 표현: 인덴트(indent)

For문 안의 선언 및 문장들

```
#include <stdio.h>

int main(void)
{
    int i,a;

    printf("Input a integer number: ");
    scanf("%d", &a);
    for(i=0;i<a;++i) {
        printf("i = %d\n", i);
        printf("a = %d\n", a);
    }
    return 0;
}
```

Beginning of for: last column

Ending of for: same column

for문 안의 if문

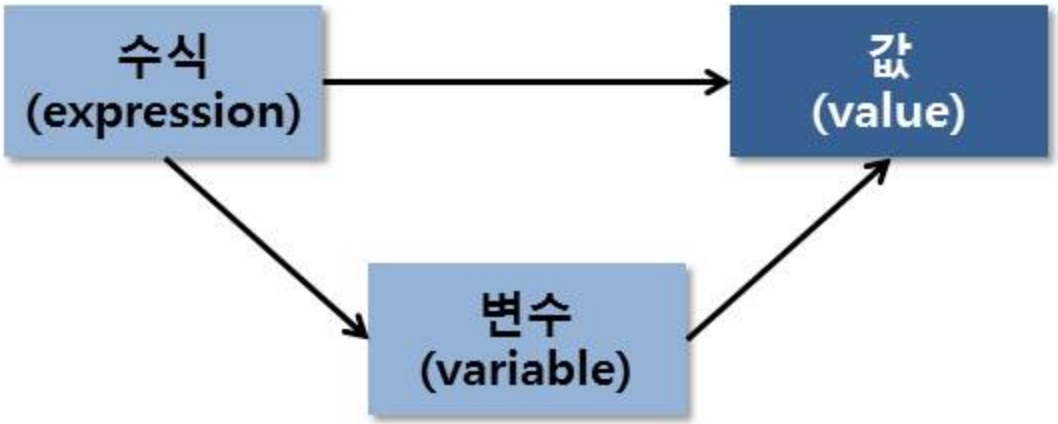
```
#include <stdio.h>

int main(void)
{
    int i,a;

    printf("Input a integer number: ");
    scanf("%d", &a);
    for(i=0;i<a;++i) {
        printf("i = %d\n", i);
        if(a%2 == 0) {
            printf("a is a even number\n");
            printf("a is a even number\n");
        }
    }
    return 0;
}
```

K & R Style

수식(expression)

- 수식(expression)의 개념
  - **연산자(operator)와 피연산자(operand)의 나열(sequence)**

- 수식의 결과
  - **수식은 반드시 값(결과값)이 된다.**
  - 연산자의 종류에 따라 수식은 변수(variable)가 되기도 한다.
    - 변수(variable)을 만드는 연산자: 배열 연산자, 구조체 참조 연산자, 포인터 연산자
    - 값(value)을 만드는 연산자: 그 외 모든 연산자



수식(expression)

산술 연산자의 피연산자와 결과값

산술 연산	정수의 연산	실수의 연산
+	정수 + 정수 → **정수**	실수 + 실수 → **실수**
-	정수 - 정수 → **정수**	실수 - 실수 → **실수**
*	정수 * 정수 → **정수**	실수 * 실수 → **실수**
/	정수 / 정수 → **정수**	실수 / 실수 → **실수**
%	정수 % 정수 → **정수**	없음

수식(expression)

관계 연산자의 피연산자와 결과값

관계 연산	정수의 연산	실수의 연산
==	정수 == 정수 → 진리값	실수 == 실수 → 진리값
!=	정수 != 정수 → 진리값	실수 != 실수 → 진리값
<	정수 < 정수 → 진리값	실수 < 실수 → 진리값
<=	정수 <= 정수 → 진리값	실수 <= 실수 → 진리값
>	정수 > 정수 → 진리값	실수 > 실수 → 진리값
>=	정수 >= 정수 → 진리값	실수 >= 실수 → 진리값

※ 진리값: 참(true) 또는 거짓(false)

수식(expression)

논리 연산자의 피연산자와 결과값

논리 연산	진리값의 연산
&&	진리값 && 진리값 → **진리값**
||	진리값 || 진리값 → **진리값**
!	! 진리값 → **진리값**

※ 진리값: 참(true) 또는 거짓(false)

수식(expression)

- 산술/관계/논리 연산자
  - 피연산자의 타입이 같아야 한다. ☞ **CPU 설계 효율성 때문**

연산자	피연산자	결과값
산술 연산자	정수	정수
	실수	실수
관계 연산자	정수	진리값
	실수	진리값
논리 연산자	진리값	진리값

* CPU가 계산할 수 있는 값은? 정수와 실수 뿐
* 진리값을 CPU가 이해하나? 안됨. 정수로 대체함.

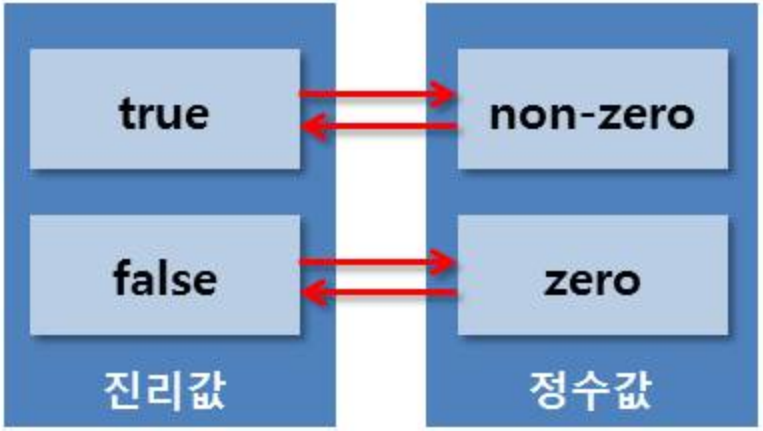
```c
#include <stdio.h>

int main(void)
{
    int a=3, b=2;
    double c=3.3, d=2.2;

    printf("%d, %f\n", a / b, c * d);
    printf("%d, %d\n", a < b, c != d);
    printf("%d, %d\n", a && b, ! a);
    return 0;
}
```

입출력 결과

```
1, 7.260000
0, 1
1, 0
계속하려면 아무 키나 누르십시오 . . .
```

수식(expression)

- 진리값의 타입(type)
  - C언어에서는 진리값의 타입은 int형이다.
  - **진리값은 정수값으로 변환되고 정수값은 진리값으로 인식된다.**




```
#include <stdio.h>

int main(void)
{
    int a=0, b=2;

    printf("%d\n", a<b );
    printf("%d\n", a>b );
    if( a ) printf("a is true\n");
    if( b ) printf("b is true\n");
    return 0;
}
```

입출력 결과

```
1
0
b is true
계속하려면 아무 키나 누르십시오 . . .
```

수식(expression)

- 진리 값의 변환
  - false는 반드시 0으로 변환
  - **true는 1로 변환되지만 반드시 1이 되다고 단정말 것.**

입출력 결과

```
a<b is true
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a=0, b=2, c;

    c = a<b;
    if( c==1 ) {  // c==1 ➔ c!=0
        printf("a<b is true\n");
    }
    return 0;
}
```

수식(expression)

- 대입 연산
  - 왼쪽 피연산자: 반드시 변수
  - 오른쪽 피연산자: 값
  - 피연산자의 타입이 같아야 한다
  - 대입연산자의 결과값: **왼쪽 피연산자에 대입된 값**

- 대입 연산의 결과값

대입 연산	피연산자 및 결과값
=	정수변수 = 정수 → **정수**
	실수변수 = 실수 → **실수**

```c
#include <stdio.h>

int main(void)
{
    int a;
    double b;

    printf("%d\n", a=3);
    printf("%f\n", b=3.3);
    return 0;
}
```

입출력 결과

```
3
3.300000
계속하려면 아무 키나 누르십시오 . . .
```

수식(expression)

- 대입연산의 잘못된 사용
  - if 문의 조건 수식에 대입연산자 사용한 경우

```
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);
    if( a = 1 )    // a = 1 → a == 1
        printf("a == one\n");
    else
        printf("a == other\n");
    return 0;
}
```

not syntax error

logical error

입출력 결과

```
1
a == one
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
a == one
계속하려면 아무 키나 누르십시오 . . .
```

Logical Error

수식(expression)

- 대입 연산자(assignment operator)

대입연산자	사용 예	의미
=	x=3;	변수 x에 3대입
+=	x+=3;	x=x+3;
-=	x-=3;	x=x-3;
*=	x*=3;	x=x*3;
/=	x/=3;	x=x/3;
%=	x%=3;	x=x%3;

```
#include <stdio.h>

int main(void)
{
    int a;

    printf("%d\n", a=3 );
    printf("%d\n", a+=1 );
    printf("%d\n", a-=a+5 );
    return 0;
}
```

a = a + 1

a = a - (a+5)

a = -5

입출력 결과

```
3
4
-5
계속하려면 아무 키나 누르십시오 . . .
```

증감 연산자

- 개념

증감연산	전치	후치	의미
++	++x	x++	x=x+1
--	--x	x--	x=x-1

- 장점: 덧셈/뺄셈 연산보다 계산 속도가 빠르다.
- 주의: **피연산자는 변수이어야 함.**

입출력 결과

```
4
4
2
2
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    int a=3, b=3, c=3, d=3;

    ++a; // means that a=a+1
    printf("%d\n", a);

    b++; // means that b=b+1
    printf("%d\n", b);

    --c; // means that c=c-1
    printf("%d\n", c);

    d--; // means that d=d-1
    printf("%d\n", d);
    return 0;
}
```

중첩 if 문과 연속 if 문

```c
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
    else {
        if(a == 2)
            printf("a == two\n");
        else
            printf("a == other\n");
    }
    return 0;
}
```

입출력 결과

```
1
a == one
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2
a == two
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
3
a == other
계속하려면 아무 키나 누르십시오 . . .
```

중첩 if 문과 연속 if 문

```c
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
    else {
        if(a == 2)
            printf("a == two\n");
        else
            printf("a == other\n");
    }
    return 0;
}
```

하나의 if-statement이므로
block을 만들 필요가 없다.

```c
#include <stdio.h>

int main(void)
{
    int a;

    scanf("%d", &a);

    if(a == 1)
        printf("a == one\n");
    else if(a == 2)
        printf("a == two\n");
    else
        printf("a == other\n");


    return 0;
}
```

중첩 if 문과 연속 if 문

```c
#include <stdio.h>

int main(void)
{
    int a;

    scanf(" %d",&a);
    if( a == 1 )
        printf("a==one\n");
    else
        if( a == 2 )
            printf("a==two\n");
        else
            if( a == 3 )
                printf("a==three\n");
            else
                if( a == 4 )
                    printf("a==four\n");
                else
                    if( a == 5 )
                        printf("a==five\n");
                    else
                        printf("a==other\n");
    return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
    int a;

    scanf(" %d",&a);
    if( a == 1 )
        printf("a==one\n");
    else if( a == 2 )
        printf("a==two\n");
    else if( a == 3 )
        printf("a==three\n");
    else if( a == 4 )
        printf("a==four\n");
    else if( a == 5 )
        printf("a==five\n");
    else
        printf("a==other\n");
    return 0;
}
```

for문의 분석

```
#include <stdio.h>

int main(void)
{
    int s,i;

    s=0;
    for(i=1; i<4; i=i+1) {
        s=s+i;
    }
    printf("%d\n",s);
    return 0;
}
```

- for문 분석 테이블

variable (i)	condition (i<4)	statement (s = s + i)

- for문 분석 결과

Analysis	Value
The range of i in for-stat.	
The value of i after for-stat.	
The number of iteration	

for문의 분석

```
#include <stdio.h>

int main(void)
{
    int s,i;

    s=0;
    for(i=1; i<=4; i=i+1) {
        s=s+i;
    }
    printf("%d\n",s);
    return 0;
}
```

- for문 분석 테이블

variable (i)	condition (i<=4)	statement (s = s + i)

- for문 분석 결과

Analysis	Value
The range of i in for-stat.	
The value of i after for-stat.	
The number of iteration	

for문의 분석

```
#include <stdio.h>

int main(void)
{
    int s,i;

    s=0;
    for(i=0; i<4; i=i+1) {
        s=s+i;
    }
    printf("%d\n",s);
    return 0;
}
```

- for문 분석 테이블

variable (i)	condition (i<4)	statement (s = s + i)

- for문 분석 결과

Analysis	Value
The range of i in for-stat.	
The value of i after for-stat.	
The number of iteration	

for문의 분석

```
#include <stdio.h>

int main(void)
{
    int s,i;

    s=0;
    for(i=0; i<=4; i=i+1) {
        s=s+i;
    }
    printf("%d\n",s);
    return 0;
}
```

- for문 분석 테이블

variable (i)	condition (i<=4)	statement (s = s + i)

- for문 분석 결과

Analysis	Value
The range of i in for-stat.	
The value of i after for-stat.	
The number of iteration	

배열의 초기화

```c
#include <stdio.h>

int main(void)
{
    int i;
    int a[3];

    a[0]=2;
    a[1]=9;
    a[2]=8;

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={2,9,8};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

배열의 초기화

초기화 예제 1

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={2,9};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

원소 중 일부만 주어진 값으로 초기화되고, 나머지는 0으로 초기화 됨.

초기화 예제 2

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={0,};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

첫 번째 원소가 0으로 초기화되고 나머지도 0으로 초기화 됨.
즉, 모두 0으로 초기화 됨.

배열의 초기화

초기화 예제 3

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[ ]={2,9,8};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

배열 initializer가 있으면
배열 길이는 생략 가능.

배열 길이는 initializer의
원소 개수와 동일하다.

초기화 예제 4

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[ ];

    for(i=0; i<3; i=i+1) {
        a[i]=0;
    }
    return 0;
}
```

배열 initializer가 없으면
배열 길이가 생략될 경우
구문 오류 발생

배열의 초기화

구문 오류

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={2,9,8,7};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

Initializer의 값의 개수가 배열의 길이를 초과하여 구문 오류.

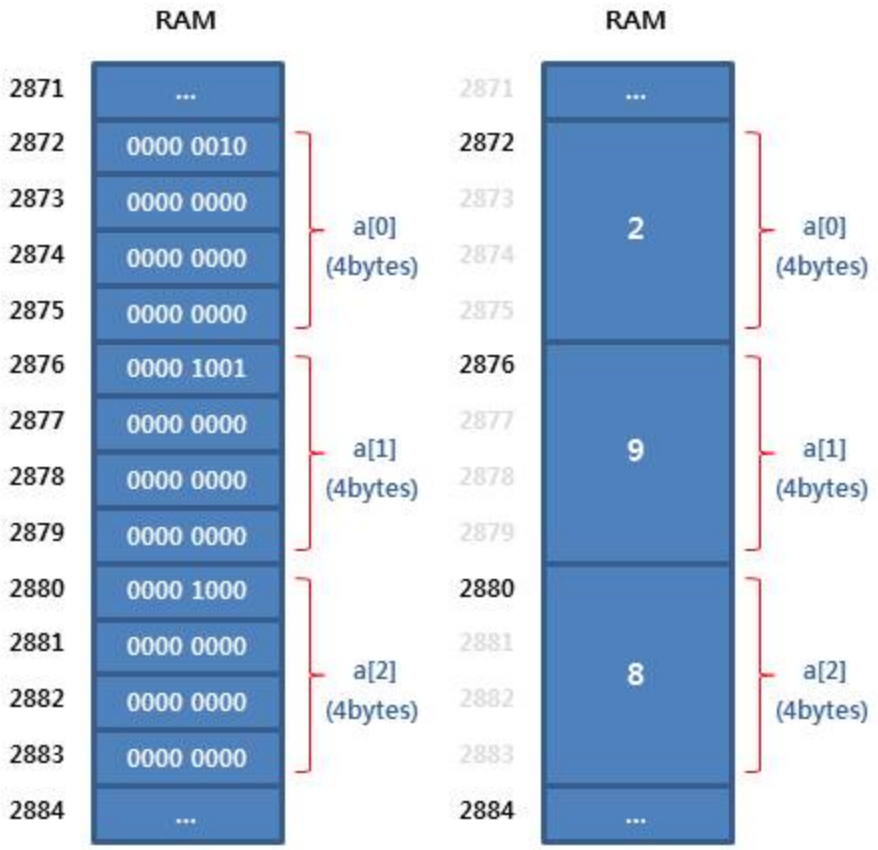
실행 오류

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={2,9,8};

    for(i=0; i<=3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

배열 index 범위 밖의 원소를 참조하여 실행 오류

배열의 메모리 구조

```
#include <stdio.h>

int main(void)
{
    int i;
    int a[3]={2,9,8};

    for(i=0; i<3; i=i+1) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```



함수의 형태

- 입력이 없는 함수
  - 함수 정의부
    - 괄호 안에 void라고 기술한다.
    - 괄호 안에 입력 내용을 기술하지 않아도 되지만, void라고 기술하는 것이 입력이 없음을 명확히 한다.
  - 함수 호출부
    - 괄호 안에 입력 내용을 기술하지 않는다.
    - 호출부에서 함수의 반환 값은 활용하지 않아도 무방하다.

```
#include <stdio.h>

int my_main(void) // void type
{
    printf("Hello, World.\n");
    return 0;
}

int main(void)
{
    my_main(  );  // no input
    return 0;
}
```

입출력 결과

```
Hello, World.
계속하려면 아무 키나 누르십시오 . . .
```

함수의 형태

- 문장만 수행하는 함수
  - main 함수 처럼 다양한 작업을 수행할 수 있다.
  - 함수의 반환 값에 의미가 없을 수도 있다.

입출력 결과

```
7
7 is odd.
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
8
8 is even.
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int even_odd(int n)
{
    if( n%2 == 0 )
        printf("%d is even.\n", n);
    else
        printf("%d is odd.\n", n);
    return 0;    // meaningless
}

int main(void)
{
    int value;

    scanf("%d", &value);
    even_odd(value);
    return 0;
}
```

함수의 형태

- 다양한 반환값
  - 조건에 따라 다른 반환값을 반환할 수 있다.

입출력 결과

```
9 7
9 > 7
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
3 7
3 < 7
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
2 2
2 == 2
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int compare(int a,int b)
{
    if(a>b) return 1;    // positive: a>b
    if(a<b) return -1;   // negative: a<b
    return 0;            // zero: a==b
}

int main(void)
{
    int x, y;
    int ret_value;

    scanf("%d%d", &x, &y);

    ret_value = compare(x,y);
    if( ret_value > 0 )
        printf("%d > %d\n", x, y);
    else if( ret_value < 0 )
        printf("%d < %d\n", x, y);
    else
        printf("%d == %d\n", x, y);
    return 0;
}
```

함수의 형태

조건에 따른 값의 반환

```
#include <stdio.h>

int compare(int a,int b)
{
    if(a>b) return 1;   // positive: a>b
    if(a<b) return -1;  // negative: a<b
    return 0;           // zero: a==b
}

int main(void)
{
    int x, y;
    int ret_value;

    scanf("%d%d", &x, &y);

    ret_value = compare(x,y);
    if( ret_value > 0 )
        printf("%d > %d\n", x, y);
    else if( ret_value < 0 )
        printf("%d < %d\n", x, y);
    else
        printf("%d == %d\n", x, y);
    return 0;
}
```

산술연산 식으로 반환값 계산

```
#include <stdio.h>

int compare(int a,int b)
{
    return a-b;
}

int main(void)
{
    int x, y;
    int ret_value;

    scanf("%d%d", &x, &y);

    ret_value = compare(x,y);
    if( ret_value > 0 )
        printf("%d > %d\n", x, y);
    else if( ret_value < 0 )
        printf("%d < %d\n", x, y);
    else
        printf("%d == %d\n", x, y);
    return 0;
}
```

함수의 형태

- 다양한 반환값
  - 조건에 따라 다른 반환값을 반환할 수 있다.

입출력 결과

```
7
7 is odd.
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
8
8 is even.
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;     // non-zero: even
    return 0;         // zero: odd
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( is_even(value) != 0 )
        printf("%d is even.\n", value);
    else
        printf("%d is odd.\n", value);
    return 0;
}
```

함수의 형태

- 진리값 반환
  - 반환값을 진리값으로 간주할 수 있다.
  - 이때 반환값의 타입은 int를 사용한다.

입출력 결과

```
7
7 is odd.
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
8
8 is even.
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;    // return true;
    return 0;        // return false;
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( is_even(value) ) // no rel. op.
        printf("%d is even.\n", value);
    else
        printf("%d is odd.\n", value);
    return 0;
}
```

함수의 형태

반환 값 비교

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;
    return 0;
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( is_even(value) == 1 )
        printf("%d is even.\n", value);
    else
        printf("%d is odd.\n", value);
    return 0;
}
```

반환 값을 진리 값으로 간주

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;    // return true;
    return 0;        // return false;
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( is_even(value) )
        printf("%d is even.\n", value);
    else
        printf("%d is odd.\n", value);
    return 0;
}
```

함수의 형태

반환 값 비교

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;
    return 0;
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( is_even(value) == 0 )
        printf("%d is odd.\n", value);
    else
        printf("%d is even.\n", value);
    return 0;
}
```

반환 값을 진리 값으로 간주

```c
#include <stdio.h>

int is_even(int n)
{
    if( n%2 == 0 )
        return 1;   // return true;
    return 0;       // return false;
}

int main(void)
{
    int value;

    scanf("%d", &value);
    if( ! is_even(value) )
        printf("%d is odd.\n", value);
    else
        printf("%d is even.\n", value);
    return 0;
}
```

함수의 형태

- 함수의 리턴
  - 함수는 return 문을 수행할때 값을 반환하고 함수의 수행을 종료한다.

입출력 결과

```
0
n==1
2
2
3
3
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int f(int n)
{
    if(n<=0) return 0;
    if(n<=1) printf("n==1\n");
    if(n<=2) return 2;
    return 3;
}

int main(void)
{
    printf("%d\n", f(0) );
    printf("%d\n", f(1) );
    printf("%d\n", f(2) );
    printf("%d\n", f(3) );
    printf("%d\n", f(99) );
    return 0;
}
```

함수의 형태

- void 리턴 타입 함수
  - 반환 값이 없다.
  - 반환 값이 없는 return 문을 사용하여 함수를 종료한다.
  - 함수의 끝에 도달하면 함수가 종료된다.

입출력 결과

```
Hello, World.
Hello, World.
Hello, World.
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

void hello(int n)
{
    int i;

    if(n<0) return; // no value.

    for(i=0;i<n;++i)
        printf("Hello, World.\n");

} // terminate hello() function.

int main(void)
{
    hello(3);
    return 0;
}
```

함수를 사용한 모듈화

- 모듈화
  - 1부터 n까지의 합 구하기 (n=100일 때)

$$S_n = \sum_{i}^{n} i$$

$$S_{100} = 1 + 2 + 3 + \cdots 100$$

입출력 결과

```
sum == 5050
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    int i,s;
    int n;

    n=100;

    s=0;
    for(i=1;i<=n;++i)
        s = s + i;

    printf("sum == %d\n",s);
    return 0;
}
```

함수를 사용한 모듈화

모듈화

```c
#include <stdio.h>

int main(void)
{
    int i,s;
    int n;

    n=100;

    s=0;
    for(i=1;i<=n;++i)
        s = s + i;

    printf("sum == %d\n",s);
    return 0;
}
```

```c
#include <stdio.h>


int sum( int n )
{
    int i,s;

    s=0;
    for(i=1;i<=n;++i)
        s = s + i;
    return s;
}


int main(void)
{
    printf("sum=%d\n", sum(100) );
    return 0;
}
```

1. 함수 이름 및 틀
2. 입력 타입
3. 입력 변수
4. 반환 타입
5. 반환 식
6. 구현
7. 변수 정리