C언어 문법

printf로 데이터 출력하기

```
# include <stdio.h>

int main(void){
    int age = 12;

    printf("%d\n", age); // %d는 정수형 데이터를 출력할 때에 사용
    return 0;
}
```

```
#include <stdio.h>

int main(void)
{
    int i, j;
    int line;

    printf("몇 줄? ");
    scanf("%d", &line);
    for(i=0; i<line; i++)
    {
        for(j=0; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

변수형과 값의 범위

- 문자형(char) : -128 ~ +127
- 정수형(int) : -2,147,483,648 ~ +2,147,483,647
- 실수형(double) : 2.2E-308 ~ 1.8E308

함수의 사용 장점

- 코드가 간결해짐, 직관적임
- 파일의 크기가 작아짐
- 함수는 만들어 두고 언제나 사용 가능
- 프로그램의 모듈화 가능

함수 사용의 단점

- 실행시간이 더 걸림
- 프로그램을 복잡하게 만들 수도 있음

#include

- 헤더파일을 불러옴
  - 프로젝트가 존재하는 폴더에서 헤더파일을 찾을 때는 `<>` 안에 작성
  - 직접 만든 헤더인 경우 `""` 안에 작성

변수명 선언 규칙

- 대문자, 소문자, 숫자, 언더바(_)를 사용
- 숫자로 시작할 수 없음
- 기본 예약어, 함수명을 사용할 수 없음
- 변수명의 길이는 ANSI C가 31자, MS-C는 247자까지 가능
- 대소문자를 다르게 구분
- 한글을 사용할 수 없음

전역변수와 지역변수

- 우선순위는 지역변수가 높음
- 초기화 하지 않았을 때
  - 전역변수 : 0
  - 지역변수 : 임의의 쓰래기 값

상수 선언

```
// #define을 사용. 뒤에 ;는 사용 X.
#include <stdio.h>

int main(void)
{
    #define AGE 24 // 매크로 상수

    printf("My age is : %d \n", AGE);

    return 0;
}
// ############
// const를 사용한 상수. 정확한 데이터 타입을 사용하여 정의
#include <stdio.h>

int main(void)
{
    const int AGE = 24;

    printf("My age is : %d \n", AGE);
    return 0;
}
```

제어 문자(Escape sequence)

제어 문자	설명
\n	다음 줄의 첫 행으로 이동
\r	현재 줄의 맨 앞으로 이동
\a	내부 스피커를 통해 소리가 남
\t	탭 키를 누른 것과 같은 효과
\\	\를 표현
\’	‘를 표현
\”	“를 표현

서식 문자

문자	타입	출력 형식
c	char	문자
d, i	short, int	부호 있는 10진수
ld	long	부호 있는 10진수
u	unsigned int	부호 없는 10진수

문자	타입	출력 형식
o	int	8진수
x	int	16진수, 출력 시 0_9와 a~f 사용
X	int	16진수, 출력 시 0_9와 A~F 사용
f	double	고정 소수점 표기법 사용
e	double	부동 소수점 표기법 사용, 지수문자 e 사용
E	double	부동 소수점 표기법 사용, 지수문자 E 사용
g	double	고정 또는 부동 소수점 표기법 중 길이가 짧은 서식 사용
G	double	‘g’와 같으며, 지수 사용 시 E 사용
s	string	문자열

삼항 조건 연산자(? :)

(조건식) ? (참인 경우 실행할 수식) : (거짓일 겨우 실행할 수식)

비트 연산자

연산자	의미	사용 결과
&	AND	1과 1을 연산 하면 1이 되고, 하나라도 0이면 0이 된다.
|	OR	둘 중 하나라도 1이면 1이 되고, 둘 다 0이면 0이 된다.
~	NOT	1은 0으로, 0은 1로 바뀐다.
^	XOR	둘 중 하나만 1이면 1이 되고, 둘 다 1 또는 둘 다 0이면 0이 된다.
<<	SHIFTL	비트값을 왼쪽으로 이동시킨다.
>>	SHIFTR	비트값을 오른쪽으로 이동시킨다.

포인터

- 주소값(&)은 변수에 대입할 수 없다.
  - 대입을 위해서는 포인터를 정의한다(`int* pi = &i`).
  - 만약 포인터의 주소를 받는 포인터를 선언하려면 `int** pi`처럼 선언한다.

```
#include <stdio.h>

int main()
{
    int i = 5;
    int* pi;

    pi = &i;

    printf("i : %d\n", i);
    printf("i address : %d\n", &i);
    printf("pi : %d\n", pi);
    return 0;
}
```

- 포인터를 통해 배열에 값을 넣을수도 있다.

```
#include <stdio.h>

int main()
{
    int array[5];
    int* parray = array;

    parray[1] = 5;

    printf("%d", array[1]);
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    int array[2][5];
    int (*parray)[5];

    parray = &array[0];

    parray[0][1] = 5;

    printf("%d", array[0][1]);
    return 0;
}
```

```
#include <stdio.h>

int main()
{
```

```
    int array[5];
    int *parray = array;


    parray = parray + 1;

    parray[0] = 5;

    printf("%d, %d, %d", *(array), parray, array[1]);
    return 0;
}
```