유인물(C언어 Cheat sheet)

1. 기본 - Hello World 출력하기

```
#include <stdio.h>
int main() {
printf("Hello world!");
 return 0;
}
```

---

2. 입출력

printf() - 출력 함수

- C언어의 표준 **출력 함수**로, **여러 종류의 데이터(data)**를 다양한 서식에 맞춰 **출력**할 수 있게 해준다

```
printf("정수를 입력하시오"); //큰따움표 안에있는 문장 출력
```

scanf() - 입력 함수

- C언어의 표준 **입력 함수**로, 사용자로부터 **다양한 데이터**를 다양한 서식에 맞춰 **입력** 받을 수 있게 해줍니다.

```
#include <stdio.h>
int main(void) {
    int num01, num02;
    printf("첫 번째 정수를 입력하세요 : ");
    scanf("%d", &num01);
    printf("두 번째 정수를 입력하세요 : ");
    scanf("%d", &num02);
    printf("입력하신 두 정수의 합은 %d입니다.\n", num01 + num02);
    return 0;
}
```

---

3. 조건문

if

- if문은 주어진 조건식의 결과에 따라 { }(중괄호)안에 코드를 실행하는 조건문입니다. 조건이 참일 경우 if 내의 코드 블록이 실행되며, 그렇지 않을 경우 else if나 else의 코드 블록이 실행됩니다.

```
#include <stdio.h>
int main() {
    int score;
```

```
    // 사용자에게 점수 입력 요청
    printf("당신의 점수를 입력하세요: ");
    scanf_s("%d", &score);
    // 점수에 따른 등급 결정
    if (score >= 90) {
        printf("당신의 등급은 A입니다.\n");
    }
    else if (score >= 80) {
        printf("당신의 등급은 B입니다.\n");
    }
    else if (score >= 70) {
        printf("당신의 등급은 C입니다.\n");
    }
    else {
        printf("당신의 점수는 C 등급 미만입니다.\n");
    }
    return 0;
}
```

switch case

- 제어식의 값에 따라서 여러 경로 중에서 하나를 선택할 수 있는 제어 구조
- 경로를 계속하여 나가던 중 break를 만날시 프로그램을 종료한다

```
#include <stdio.h>
int main() {
    int score;
    // 사용자에게 점수 입력 요청
    printf("당신의 점수를 입력하세요: ");
    scanf_s("%d", &score);

    switch (score / 10) {  // 점수를 10으로 나눈 몫을 사용하여 10점 단위로 구분
    case 10:
    case 9:
        printf("당신의 등급은 A입니다.\n"); //나눈 몫이 9이면 당신의 등급은 A입니다 출력
        break;
    case 8:
        printf("당신의 등급은 B입니다.\n"); //나눈 몫이 8이면 당신의 등급은 B입니다 출력
        break;
    case 7:
        printf("당신의 등급은 C입니다.\n"); //나눈 몫이 7이면 당신의 등급은 C입니다 출력
        break;
    default:
        printf("당신의 점수는 C 등급 미만입니다.\n");  나눈 몫이 9, 8, 7 이 아니라면 당신의 점수는 C 등급 미만입니다 출
    }

    return 0;
}
```

If ↔ Switch, Switch ↔ If

1. if → switch-case

단계:

1. 비교하는 변수나 표현식을 **switch()** 내부에 넣습니다.
2. 각 **if** 또는 **else if** 조건의 결과값을 **case**로 변환합니다.
3. 각 조건 블록의 코드를 해당 **case** 아래에 넣습니다.
4. 각 **case** 블록의 끝에 **break;**를 추가합니다.
5. 마지막 **else** 블록은 **default**로 변환합니다.

```
// if-else 구조
if (number == 1) {
        printf("하나\n");
    } else if (number == 2) {
        printf("둘\n");
    } else if (number == 3) {
        printf("셋\n");
    } else {
        printf("유효하지 않은 숫자입니다.\n");
    }

// switch-case 구조
 switch (number) {
        case 1:
            printf("하나\n");
            break;
        case 2:
            printf("둘\n");
            break;
        case 3:
            printf("셋\n");
            break;
        default:
            printf("유효하지 않은 숫자입니다.\n");
    }
```

2. switch-case → if

단계:

1. **switch()** 내부의 변수나 표현식을 기준으로 첫 번째 **if** 조건을 작성합니다.
2. 각 **case** 값을 사용하여 **if** 또는 **else if** 조건을 작성합니다.
3. **default** 블록은 **else**로 변환합니다.

예시:

```
// switch-case 구조
 switch (number) {
        case 1:
            printf("하나\n");
            break;
        case 2:
            printf("둘\n");
            break;
        case 3:
```

```
            printf("셋\n");
            break;
        default:
            printf("유효하지 않은 숫자입니다.\n");
    }

// if-else 구조
if (number == 1) {
        printf("하나\n");
    } else if (number == 2) {
        printf("둘\n");
    } else if (number == 3) {
        printf("셋\n");
    } else {
        printf("유효하지 않은 숫자입니다.\n");
    }
```

---

4. 반복문

while문

- 주어진 조건이 참인 동안 내부의 코드 블록을 반복해서 실행

```
#include <stdio.h>
int main() {
    int score;
    int count = 0;
    while (count < 5) {
        printf("%d", count);
        count++;
    }
    return 0;
}
```

- 출력 결과

```
01234
```

변수 이름 조건

- 영문자와 숫자, 밑줄 문자_로 이루어진다
- 변수 중간에 공백이 들어가면 안 된다
- 첫 글자는 반드시 영문자 또는 밑줄 기호 _ 이여야 한다. 숫자로 시작할 수 없다
- 대문자와 소문자는 구별된다. 따라서 index,Index, INDEX은 모두 서로 다른 변수이다
- C언어의 키워드와 똑같은 식별자는 허용되지 않는다.

연산자

- 산술 연산자
  - 컴퓨터의 가장 기본적인 연산
  - 덧셈, 뺄셈, 곱셈, 나눗셈 등의 사칙 연산을 수행하는 연산

연산자	기호	사용예	결과값
덧셈	+	7 + 4	11
뺄셈	–	7 – 4	3
곱셈	*	7 * 4	28
나눗셈	/	7 / 4	1
나머지	%	7 % 4	3

- 증감 연산자
  - 변수의 값을 하나 증가시키거나 감소시키는 연산자
  - 예)++x, --x

증감 연산자	차이점
++x	수식의 값은 증가된 x값이다.
x++	수식의 값은 증가되지 않은 원래의 x값이다.
--x	수식의 값은 감소된 x값이다.
x--	수식의 값은 감소되지 않은 원래의 x값이다.

- 복합 대입 연산자
  - +=처럼 대입연산자 =와 산술연산자를 합쳐 놓은 연산

복합 대입 연산자	의미	복합 대입 연산자	의미
x += y	x = x + y	x &= y	x = x & y
x -= y	x = x - y	x |= y	x = x | y
x *= y	x = x * y	x ^= y	x = x ^ y
x /= y	x = x / y	x >>= y	x = x >> y
x %= y	x = x % y	x <<= y	x = x << y

- 관계 연산자
  - 두개의 피연산자를 비교하는 연산

연산	의미	연산	의미
x == y	x와 y가 같은가?	x < y	x가 y보다 작은가?
x != y	x와 y가 다른가?	x >= y	x가 y보다 크거나 같은가?
x > y	x가 y보다 큰가?	x <= y	x가 y보다 작거나 같은가?

- 논리 연산자
  - 비교 연산자가 여러 번 필요할 때 사용

연산	의미
x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

- 비트 연산자
  - 비트 단위로 논리 연산을 할 때 사용하는 연산자

연산자	연산자의 의미	예
&	비트 AND	두개의 피연산자의 해당 비트가 모두 1이면 1, 아니면 0
|	비트 OR	두개의 피연산자의 해당 비트중 하나만 1이면 1, 아니면 0
^	비트 XOR	두개의 피연산자의 해당 비트의 값이 같으면 0, 아니면 1
<<	왼쪽으로 이동	지정된 개수만큼 모든 비트를 왼쪽으로 이동한다.
>>	오른쪽으로 이동	지정된 개수만큼 모든 비트를 오른쪽으로 이동한다.
~	비트 NOT	0은 1로 만들고 1은 0로 만든다.

- 비트 이동 연산자
  - 정수 데이터의 비트를 왼쪽 또는 오른쪽으로 이동시키는 연산

연산자	기호	설명
왼쪽 비트 이동	<<	x ⟨⟨ y x의 비트들을 y 칸만큼 왼쪽으로 이동
오른쪽 비트 이동	>>	x ⟩⟩ y x의 비트들을 y 칸만큼 오른쪽으로 이동

우선순위

우선순위	연산자	설명	결합성
1	++ --	후위 증감 연산자	→ (좌에서 우)
	()	함수 호출	
	[]	배열 인덱스 연산자	
	.	구조체 멤버 접근	
	->	구조체 포인터 접근	
	(type){list}	복합 리터럴(C99 규격)	
2	++ --	전위 증감 연산자	← (우에서 좌)
	+ -	양수, 음수 부호	
	! ~	논리적인 부정, 비트 NOT	
	(type)	형변환	
	*	간접 참조 연산자	
	&	주소 추출 연산자	
	sizeof	크기 계산 연산자	
	_Alignof	정렬 요구 연산자 (C11 규격)	
3	* / %	곱셈, 나눗셈, 나머지	→ (좌에서 우)
4	+ -	덧셈, 뺄셈	
5	<< >>	비트 이동 연산자	
6	< <=	관계 연산자	
	> >=	관계 연산자	
7	== !=	관계 연산자	
8	&	비트 AND	
9	^	비트 XOR	
10	|	비트 OR	
11	&&	논리 AND 연산자	
12	||	논리 OR 연산자	
13	?:	삼항 조건 연산자	← (우에서 좌)
14	=	대입 연산자	
	+= -=	복합 대입 연산자	
	*= /= %=	복합 대입 연산자	
	<<= >>=	복합 대입 연산자	
	&= ^= |=	복합 대입 연산자	
15	,	콤마 연산자	→ (좌에서 우)

자료형

- 데이터의 타입(종류)

자료형			설명	바이트	범위
정수형	부호있음	short	short형 정수	2	-32768 ~ 32767
		int	정수	4	-2147483648 ~ 2147483647
		long	long형 정수	4	-2147483648 ~ 2147483647
	부호없음	unsigned short	부호없는 short형 정수	2	0 ~ 65535
		unsigned int	부호없는 정수	4	0 ~ 4294967295
		unsigned long	부호없는 long형 정수	4	0 ~ 4294967295
문자형	부호있음	char	문자 및 정수	1	-128 ~ 127
	부호없음	unsigned char	문자 및 부호없는 정수	1	0 ~ 255
부동소수점형		float	단일정밀도 부동소수점	4	1.2E-38 ~ 3.4E38
		double	두배정밀도 부동소수점	8	2.2E-308 ~ 1.8E308

아스키 코드 표

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
0	0X00	NULL	16	0X10	DLE	32	0x20	SP	48	0x30	0
1	0X01	SOH	17	0X11	DC1	33	0x21	!	49	0x31	1
2	0X02	STX	18	0X12	SC2	34	0x22	"	50	0x32	2
3	0X03	ETX	19	0X13	SC3	35	0x23	#	51	0x33	3
4	0X04	EOT	20	0X14	SC4	36	0x24	$	52	0x34	4
5	0X05	ENQ	21	0X15	NAK	37	0x25	%	53	0x35	5
6	0X06	ACK	22	0X16	SYN	38	0x26	&	54	0x36	6
7	0X07	BEL	23	0X17	ETB	39	0x27	'	55	0x37	7
8	0X08	BS	24	0X18	CAN	40	0x28	(	56	0x38	8
9	0X09	HT	25	0x19	EM	41	0x29	)	57	0x39	9
10	0X0A	LF	26	0x1A	SUB	42	0x2A	*	58	0x3A	:
11	0X0B	VT	27	0x1B	ESC	43	0x2B	+	59	0x3B	;
12	0X0C	FF	28	0x1C	FS	44	0x2C	,	60	0x3C	<
13	0X0D	CR	29	0x1D	GS	45	0x2D	-	61	0x3D	=
14	0X0E	SO	30	0x1E	RS	46	0x2E	.	62	0x3E	>
15	0X0F	SI	31	0x1F	US	47	0x2F	/	63	0x3F	?