3장. 연산자 & 입력처리

Visualstudio 2019

항과 연산자

- **항(operand)**
  - 연산에 사용되는 값
- **연산자(operator)**
  - 연산에 사용되는 기호

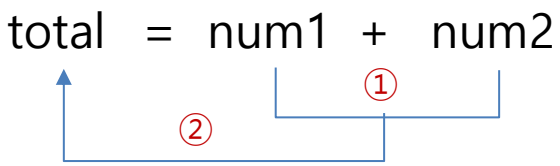
  예) 3 + 7 (3과 7은 항, '+'는 연산자)

- **항의 개수에 따른 연산자 구분**

연산자	설명	연산 예
단항 연산자	항이 한 개인 연산자	++num
이항 연산자	항이 두 개인 연산자	num1 + num2
삼항 연산자	항이 세 개인 연산자	(5>3) ? 1 : 0

대입 및 부호 연산자

- **대입 연산자**
  - 변수에 값을 대입하는 연산자
  - 연산의 결과를 변수에 대입
  - 우선 순위가 가장 낮은 연산자
  - 왼쪽 변수(lvalue)에 오른쪽 값(rvalue)를 대입



대입 연산자 연습문제

변수 값 교환하기

변수 blue에 1이 저장되어 있고, red에 2가 저장되어 있을때 새로운 변수 yellow를 사용하여 값을 교환해 보세요

```
=== 교환전 ===
blue = 1, red = 2
=== 교환후 ===
blue = 2, red = 1
```

```c
int blue = 1;
int red = 2;
int yellow;

printf("=== 교환전 ===");
printf("blue = %d, red = %d\n", blue, red);

//변수 바꾸기
yellow = blue;
blue = red;
red = yellow;

printf("=== 교환전 ===");
printf("blue = %d, red = %d\n", blue, red);
```

산술 및 증감 연산자

- **산술 연산자**

연산자	기 능	연산 예
+	두 항을 더합니다.	5+3
-	앞 항에서 뒤 항을 뺍니다.	5-3
*	두 항을 곱합니다.	5*3
/	앞 항에서 뒤 항을 나누어 몫을 구합니다.	5/3
%	앞 항에서 뒤 항을 나누어 나머지를 구합니다.	5%3

산술 및 증감 연산자

- **증가 감소 연산자**

1만큼 더하거나 1만큼 뺄 때 사용하는 연산자

연산자	기 능	연산 예

++	항의 값에 1을 더합니다.	val = ++num; // num = num+1;
		val = num++;

--	항의 값에서 1을 뺍니다.	val = --num // num = num-1;
		val = num--;

산술 및 증감 연산자

```c
int a = 99;
int b = 2;
printf(" a + b의 결과 : %d\n", a + b);
printf(" a - b의 결과 : %d\n", a - b);
printf(" a * b의 결과 : %d\n", a * b);
printf(" a / b의 결과 : %d\n", a / b);
printf(" a %% b의 결과 : %d\n", a % b);

printf("a++의 값은 : %d\n", a++);  //99
printf("a의 값은 : %d\n", a);      //100

printf("++a의 값은 : %d\n", ++a);  //101
printf("a의 값은 : %d\n", a);      //101
```

비교 및 논리 연산자

- **관계(비교) 연산자**

연산의 결과가 참(1), 거짓(0)으로 반환됨

연산자	기 능	연산 예
>	왼쪽 항이 크면 참을, 아니면 거짓을 반환합니다.	num > 3;
<	왼쪽 항이 작으면 참, 아니면 거짓을 반환합니다.	num < 3;
>=	왼쪽 항이 크거나 같으면 참, 아니면 거짓을 반환합니다.	num >= 3;
<=	왼쪽 항이 작거나 같으면 참, 아니면 거짓을 반환합니다.	num <= 3;
==	두 개의 항 값이 같으면 참, 아니면 거짓을 반환합니다.	num == 3;
!=	두 개의 항 값이 다르면 참, 아니면 거짓을 반환합니다.	num != 3

비교 및 논리 연산자

- **논리 연산자**

연산자	기 능	연산 예
&& (논리 곱)	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	(7<3) && (5>2)
|| (논리 합)	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	(7>3) || (5<2)
! (부정)	단항 연산자, 참인 경우는 거짓으로, 거짓인 경우는 참으로 바꿉니다.	!(7>3)

비교 및 논리 연산자

```c
//참이면 0, 거짓이면 1
printf("10 == 10의 값은 %d입니다.\n", 10 == 10);
printf("10 != 10의 값은 %d입니다.\n", 10 != 10);
printf("10 > 5의 값은 %d입니다.\n", 10 > 10);
printf("10 < 10의 값은 %d입니다.\n", 10 <10);
printf("10 >= 10의 값은 %d입니다.\n", 10 >= 10);
printf("10 <= 10의 값은 %d입니다.\n", 10 <= 10);

//논리 연산자
int a = 5;
int b = 3;
int c = 2;

printf("0 && 0 = %d\n", (a < b) && (b < c)); //0
printf("0 && 1 = %d\n", (a < b) && (b > c)); //0
printf("1 && 1 = %d\n", (a > b) && (b > c)); //1

printf("0 || 0 = %d\n", (a < b) || (b < c)); //0
printf("1 || 0 = %d\n", (a > b) || (b < c)); //1

printf("!0 = %d\n", !(a < b)); //1
printf("!1 = %d\n", !(b > c)); //0
```

복합대입 및 조건 연산자

- **복합대입 연산자**

연산자	기 능	연산 예
+=	두 항의 값을 더해서 왼쪽 항에 대입합니다.	num += 2; num=num+2
-=	왼쪽 항에서 오른쪽 항을 빼서 그 값을 왼쪽 항에 대입합니다.	num -= 2; num=num-2
*=	두 항의 값을 곱해서 왼쪽 항에 대입합니다.	num *= 2; num=num*2
/=	왼쪽 항을 오른쪽 항으로 나누어 그 몫을 왼쪽 항에 대입합니다.	num /= 2; num=num
%=	왼쪽 항을 오른쪽 항으로 나누어 그 나머지를 왼쪽 항에 대입합니다.	num %= 2; num=num%2

복합대입 및 조건 연산자

- **조건 연산자**

조건 연산자 -> 제어문중 조건문을 간단히 표현할 때 사용할 수 있음

연산자	기 능	연산 예
조건식?결과1:결과2;	조건식이 참이면 결과1, 조건식이 거짓이면 결과2가 선택됩니다.	int num = (5>3)?10:20;

복합대입 및 조건 연산자

- **조건 연산자**

```c
//조건 연산자
int value;
value = (3 > 4) ? 10 : 20;
printf("%d\n", value);

int fatherAge = 35;
int motherAge = 38;

char ch = (fatherAge < motherAge) ? 'T' : 'F';
printf("%c\n", ch);
```

비트 연산자

- **비트 연산자**

연산자	기 능	연산 예
&	a & b	1 & 1 -> 1을 반환, 그 외는 0
|	a | b	0 | 0 -> 0을 반환, 그 외는 1
~	~a	a가 1이면 0, 0이면 1을 반환
<<	a<<2	a를 2비트 만큼 왼쪽으로 이동
>>	a>>3	a를 2비트 만큼 오른쪽으로 이동

비트 연산자

- **비트 논리연산자**

```
int num1 = 5;
int num2 = 10;
int result = num1 & num2;
```

```
  num1 : 0 0 0 0 0 1 0 1
& num2 : 0 0 0 0 1 0 1 0
  result : 0 0 0 0 0 0 0 0
```

- **비트 이동 연산자**

```
int num = 5;
num << 2;
```

```
num     : 0 0 0 0 0 1 0 1
num<<2 : 0 0 0 1 0 1 0 0
```

비트 연산자

```c
//비트 논리 연산자
int num1 = 5;    //00000101
int num2 = 10;   //00001010
int result = num1 & num2;  //00000000
printf("result = %d\n", result);

result = num1 | num2;      //00001111
printf("result = %d\n", result);

//비트 이동 연산자
int num3 = 2;   //00000010  -> 10진수 2
printf("result = %d\n", num3 << 1);  //00000100 -> 10진수 4
printf("result = %d\n", num3 << 2);  //00001000 -> 10진수 8
printf("result = %d\n", num3);
printf("result = %d\n", num3 >> 1);  //00000001  -> 10진수 1
```

```c
int a = 0xF0; //2진수 : 11110000
int b = 0x0F; //2진수 : 00001111

int c = 0x0A; //2진수 : 00001010
int d = 0x0B; //2진수 : 00001011

int e = 0x02; //2진수 : 00000010
int f = 0x08; //2진수 : 00001000

printf(" a & b의 결과는 : %x\n", a & b);
printf(" a | b의 결과는 : %x\n", a | b);
printf(" c ^ d의 결과는 : %x\n", c ^ d);
printf(" ~a의 결과는 : %x\n", ~a);
```

연산자 우선 순위

- **연산자 우선 순위**

우선순위	형태	연산자
1	일차식	( ) [ ]
2	단항	++ -- !
3	산술	% * / + -
4	비트이동	<< >>
5	관계	< > == !=
6	비트 논리	& | ~
7	논리	&& || !
8	조건	? :
9	대입	= += -= *=

키보드로 데이터 입력 받기

1. 수 자료형

scanf_s(입력 형식, 데이터 저장 변수)

ex) int n;

scanf_s("%d", &n);

2. 문자 자료형

scanf_s(입력 형식, 데이터 저장 변수, 데이터의 크기)

ex) char str[100]

scanf_s("%s", str, sizeof(str));

키보드로 데이터 입력 받기

```c
#include <stdio.h>
#define _CRT_SECURE_NO_WARNINGS

int main() {
    int i;
    float f;
    char str[100];

    printf("정수 입력 : ");
    scanf_s("%d", &i);
    printf("입력된 정수 : %d\n", i);
    printf("입력된 정수의 주소 : %x\n", &i);

    printf("실수 입력 : ");
    scanf_s("%f", &f);
    printf("입력된 정수 : %f\n", f);
    printf("입력된 정수의 주소 : %x\n", &f);

    printf("문자열 입력 : ");
    scanf_s("%s", str, sizeof(str));
    printf("입력된 문자열 : %s\n", str);
    printf("입력된 문자열의 주소 : %x\n", str);

    return 0;
}
```

```
수 입력 : 11
정입입력된 정수 : 11
입력된 정수의 주소 : defb6c
실입입력수 입력 : 3.5
입력된 정수 : 3.500000
입력된 정수의 주소 : defb60
문자열 입력 : kim
입력된 문자열 : kim
입력된 문자열의 주소 : defaf4
```

덧셈과 뺄셈 계산기

```c
#include <stdio.h>
#define _CRT_SECURE_NO_WARNINGS

int main() {
    int n1, n2, total;
    char code = '+';

    printf("첫째 수 입력 :");
    scanf_s("%d", &n1);
    printf("둘째 수 입력 :");
    scanf_s("%d", &n2);

    total = (code == '+') ? (n1 + n2) : (n1 - n2);
    printf("합계 : %d\n", total);

    return 0;
}
```

```c
int main() {
    char name[20];
    int age;

    printf("이름 입력 : ");
    scanf_s("%s", name, sizeof(name));
    printf("이름 : %s\n", name);

    printf("나이 입력 : ");
    scanf_s("%d", &age);
    printf("나이 : %d\n", age);


    return 0;
}
```

구속(球速)의 단위 변환 프로그램

메이저리그는 점점 더 빠른 구속을 추구하고 있다. 올 시즌 메이저리그 포심 패스트볼 평균 구속은 시속 93.2마일(150.0km)에 달한다. 이제는 100마일(160.9km)이 넘는 공도 어렵지 않게 볼 수 있게 됐다.

투수에게 있어 구속이 가장 중요한 요소는 아니다. 구종, 제구, 구위 등 수 많은 요소들이 어우러져야 비로소 뛰어난 투구를 할 수 있다. 하지만 같은 조건이라면 당연히 구속이 빠를수록 유리하다. 구속이 빠를수록 타자들이 공에 대처할 수 있는 물리적인 시간이 줄어들기 때문이다.

구속(球速)의 단위 변환 프로그램

```c
#include <stdio.h>
#define _CRT_SECURE_NO_WARNINGS
#define RATE_KPH_MPH 1.609344

int main() {
    //KPH(킬로미터)를 MPH(마일)로 변환
    int kph;
    double mph;
    //const double RATE_KPH_MPH = 1.609344;

    printf("당신의 구속을 입력하세요[km/H]: ");
    scanf_s("%d", &kph);
    mph = kph / RATE_KPH_MPH;
    printf("당신의 구속은 %.2lf[MPH]입니다.\n", mph);

    return 0;
}
```