operator

# operator

## 연산자와 피연산자

1) 연산자(Operator)란 연산을 수행하는 기호를 의미함.
2) 피연산자(Operand)란 연산에 포함되는 변수나 상수를 의미함.
3) A + B에서 A와 B는 피연산자에 해당하며 +는 연산자에 해당함.

| 피연산자 | 연산자 | 피연산자 |
|---|---|---|
| A | + | B |

# operator

연산자의 종류

1) C언어에는 다양한 연산자가 존재함.

| 대입 연산자 | = |
|---|---|
| 산술 연산자 | +, -, *, /, % |
| 관계 연산자 | ==, !=, >, <, >=, <= |
| 논리 연산자 | !, &&, \|\| |
| 증감 연산자 | ++, -- |
| 삼항 연산자 | ? : |
| 비트 연산자 | !, ~, &, ^, >>, << |

# operator

대입 연산자

1) ‘=’ 등호(Equals Sign)를 이용해서 우변항을 좌변항에 넣을 수 있음.
2) 자료형에 부합하는 값을 좌변항에 있는 변수에 넣음.

# operator

## 사칙연산

1) C언어에서는 기본적인 사칙 연산을 사용할 수 있음.
2) 나머지를 구하기 위해 모듈러(Modular) 연산을 사용함.

| | |
|---|---|
| + | 더하기 |
| - | 빼기 |
| * | 곱하기 |
| / | 나누기 |
| % | 나머지 |

# operator

## 사칙연산

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

int main(void) {
  int a, b;
  scanf("%d %d", &a, &b);
  printf("%d + %d = %d\n", a, b, a + b);
  printf("%d - %d = %d\n", a, b, a - b);
  printf("%d * %d = %d\n", a, b, a * b);
  printf("%d / %d = %d\n", a, b, a / b);
  printf("%d %% %d = %d\n", a, b, a % b);
}
```

# operator

## 이스케이프 시퀀스(Escape Sequence)

1) C언어에서 특정한 표현을 출력하기 위해 사용하는 문법임.

| | |
|---|---|
| ₩n | 줄 바꾸기 |
| ₩t | 수평 탭 넣기 |
| ₩₩ | 백슬래시 넣기 |
| ₩" | 큰 따옴표 넣기 |
| ₩b | 백 스페이스 넣기 |

# operator

## 이스케이프 시퀀스(Escape Sequence)

```c
#include <stdio.h>

int main(void)
  { printf("\"A\tB\tC\tD\"\n";
  printf("\"A\tB\tC\tD\"\n");
  printf("\"A\tB\tC\tD\"\n");
  }
```

# operator

## 관계 연산자

| | |
|---|---|
| > | 크다 |
| < | 작다 |
| == | 같다 |
| != | 다르다 |
| >= | 크거나 같다 |
| <= | 작거나 같다 |

# operator

## 관계 연산자

1) 컴퓨터는 0을 거짓(False)로 받아들이며, 그 외의 숫자를 참(True)으로 받아들임.
2) 컴퓨터가 거짓(False)을 출력할 때는 0을, 참(True)을 출력할 때는 1을 출력함.

# operator

## 관계 연산자

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

int main(void)
  { int a, b;
  scanf("%d %d", &a, &b);
  printf("%d₩n", a > b);
}
```

# operator

## 논리 연산자

| | |
|---|---|
| ! | 부정 |
| && | 그리고 |
| \|\| | 또는 |

# operator

## 논리 연산자

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

int main(void)
  { int a, b, c;
  scanf("%d %d %d", &a, &b, &c);
  printf("%d\n", !a);
  printf("%d\n", a && b);
  printf("%d\n", (a > b) && (b > c));
  return 0;
}
```

# operator

## 증감 연산자

| | |
|---|---|
| ++(변수) | i의 값을 1 증가시킨 후에 증가된 값을 반환 |
| (변수)++ | i의 값을 1 증가시킨 후에 증가되기 전의 값을 반환 |
| --(변수) | i의 값을 1 감소시킨 후에 감소된 값을 반환 |
| (변수)— | i의 값을 1 감소시킨 후에 감소되기 전의 값을 반환 |

# operator

## 삼항 연산자

1) 세 개의 항을 이용해서 조건문을 수행할 수 있는 연산자임.

# operator

## 삼항 연산자

```c
#include <stdio.h>

int main(void)
  { int a = 7, b =7;
  printf("%d\n", (a == b) ? 100 : -100);
  return 0;
}
```

# operator

## 비트 연산자

1) 비트 단위의 연산을 수행할 수 있음.

| | |
|---|---|
| ~ | 부정: $\sim(11000011)_2 = (00111100)_2$ |
| & | 그리고: $(00001101)_2 \ \& \ (00000011)_2 = (00000001)_2$ |
| \| | 또는: $(11001100)_2 \mid (00110000)_2 = (11111100)_2$ |
| ^ | 배타적: $(11001111)_2 \wedge (00000011)_2 = (11001100)_2$ |
| << | 왼쪽 시프트: $(00001111)_2 \ll 3 = (01111000)_2$ |

# operator

## 비트 연산자

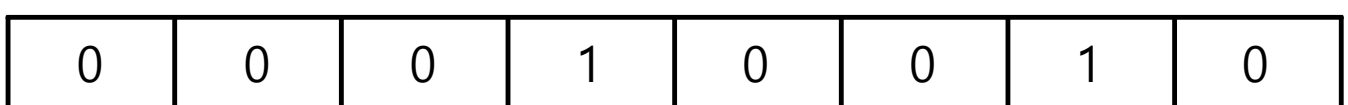
1) 시프트(Shift) 연산자는 2의 배수를 처리하고자 할 때 효과적임.
2) 9를 표현하면 다음과 같음.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

# operator

## 비트 연산자

1) 왼쪽 시프트(Shift)를 수행하면 9에서 18로 **2배**가 증가함.



# operator

## 비트 연산자

1) 오른쪽 시프트(Shift)를 수행하면 2로 나눈 값이 반환됨.

# operator

## 연산자 우선순위

1) C언어의 연산자 우선순위는 기본적으로 수학에서의 우선순위와 흡사함.

# operator

연산자 우선순위

| 우선순위 | 연산자 |
|---|---|
| 1 | ++, -- |
| 2 | !, ~ |
| 3 | *, /, % |
| 4 | +, - |
| 5 | <<, >> |
| 6 | <, <=, >, >= |
| 7 | ==, != |
| 8 | 비트, 논리, 삼항 연산자 |
| 9 | 삼항 연산자 |

# operator 정리

연산자

| 대입 연산자 | = |
| --- | --- |
| 산술 연산자 | +, -, *, /, % |
| 관계 연산자 | ==, !=, >, <, >=, <= |
| 논리 연산자 | !, &&, \|\| |
| 증감 연산자 | ++, -- |
| 삼항 연산자 | ? : |
| 비트 연산자 | !, ~, &, ^, >>, << |