

개정3판

Visual
Studio
2017

쉽게 풀어쓴

C언어
EXPRESS




천인국 지음

제4장 변수와 자료형



이번 장에서 학습할 내용

- 
- * 변수와 상수의 개념
 - * 자료형
 - * 정수형
 - * 실수형
 - * 문자형
 - * 기호 상수 사용
 - * 오버플로우와 언더플로우





변수

Q) 변수(variable)이란 무엇인가?

A) 프로그램에서 일시적으로 데이터를 저장하는 공간

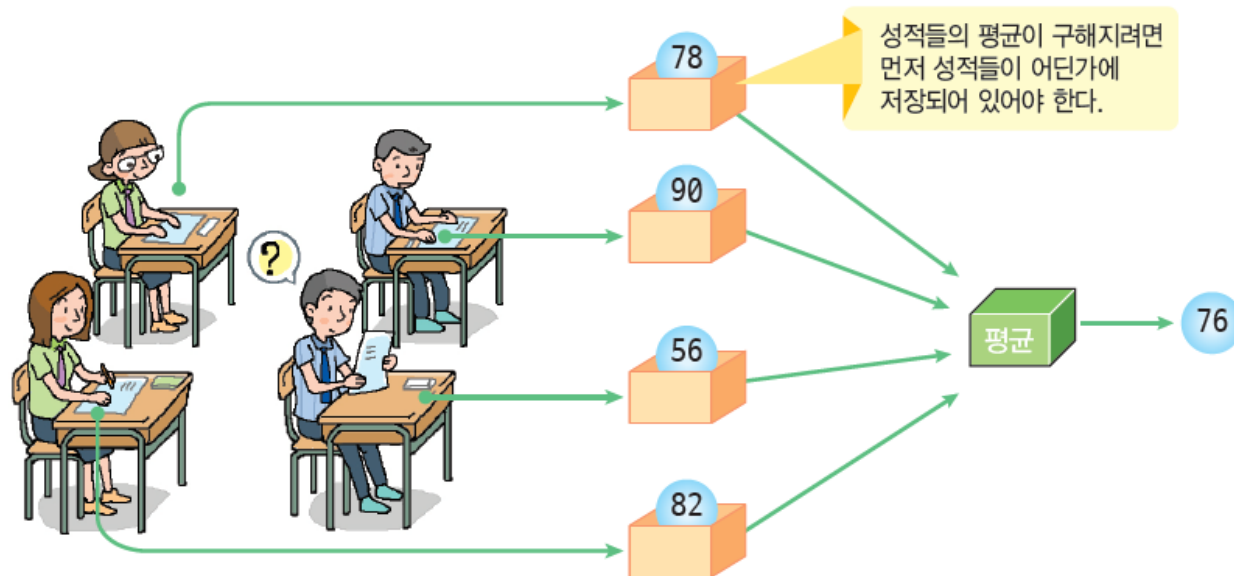
```
sum = x + y;
```

위 프로그램 문장에서 sum, x, y가 변수



변수를 사용하는 이유

- 프로그램 실행 과정에 사용되는 데이터가 저장되는 공간이 반드시 필요
 - ▣ 사용자로부터 입력 받은 데이터, 프로그램 연산에 의해 생성된 데이터 등





변수를 사용하는 이유

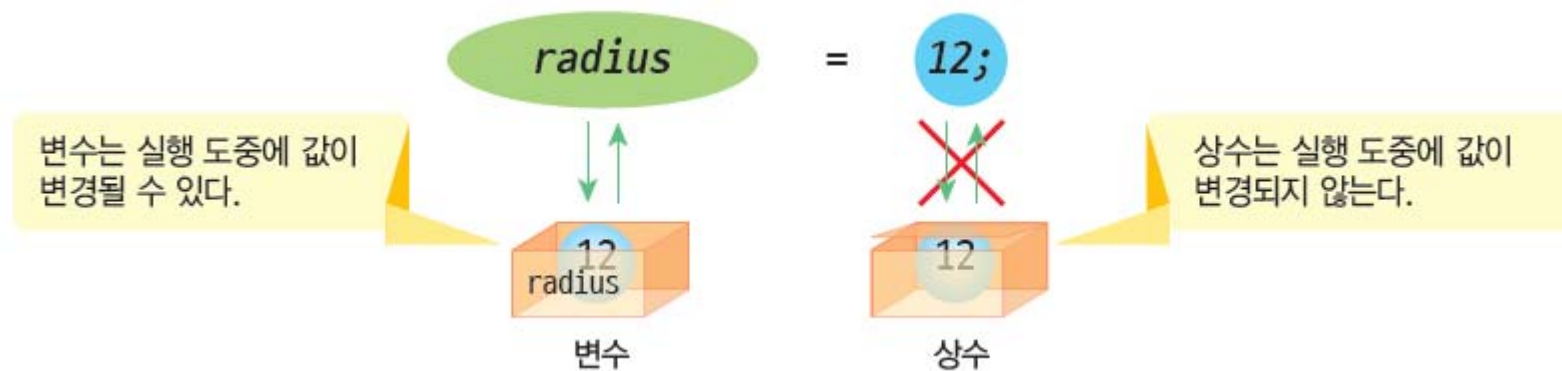
- 변수를 사용하면 프로그램 코드를 더 유연하게 만들 수 있음
 - ▣ 왼쪽 코드: 크기가 100 x 200인 사각형의 넓이만 계산 가능
 - ▣ 오른쪽 코드: width, height 값만 변경하면 어떤 사각형의 넓이도 계산 가능

변수를 사용하지 않는 코드	변수를 사용하는 코드
<pre>// 크기가 100x200인 사각형의 면적 area = 100 * 200;</pre>	<pre>// 크기가 widthxheight인 사각형의 면적 width = 100; height = 200; area = width * height;</pre>



변수와 상수

- **변수(variable)**: 저장된 값의 변경이 가능한 공간
- **상수(constant)**: 저장된 값의 변경이 불가능한 공간
 - ▣ (예) 3.14, 100, 'A', "Hello World!"





예제: 변수와 상수

```
/* 원의 면적을 계산하는 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
float radius;
```

```
float area;
```

변수

```
// 원의 반지름
```

```
// 원의 면적
```

```
printf("원의 면적을 입력하세요:");
```

```
scanf("%f", &radius);
```

상수

```
area = 3.141592 * radius * radius;
```

```
printf("원의 면적: %f \n", area);
```

```
return 0;
```

```
}
```



자료형

-
- Three orange cups with blue 'NG' logos and straws, increasing in size from left to right.





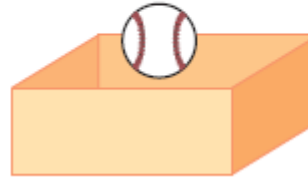
다양한 자료형이 필요한 이유

- 상자에 물건을 저장하는 것과 같다.

물건이 상자보다 크면
들어가지 않을 것이다.



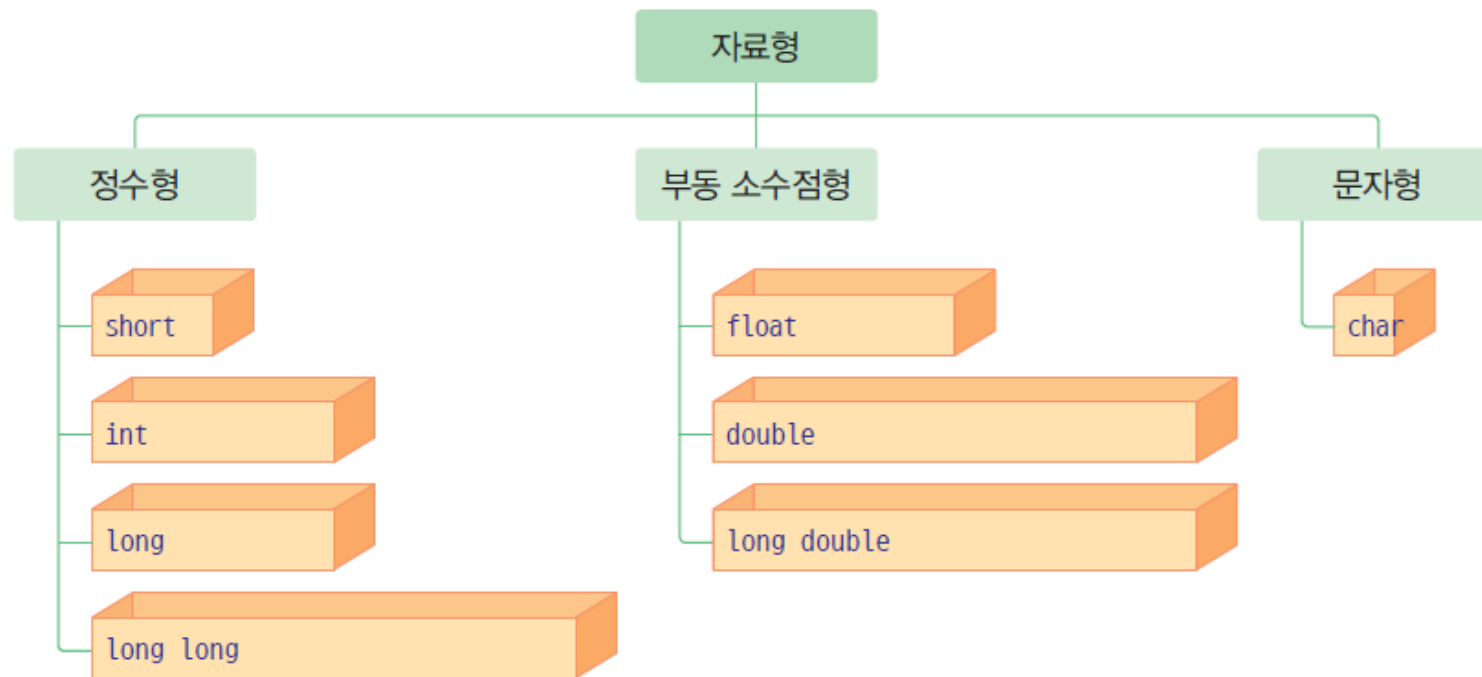
물건이 상자보다 너무 작으면
공간이 낭비될 것이다.



- 저장되는 데이터의 크기에 따라 메모리의 필요량이 다름



자료형



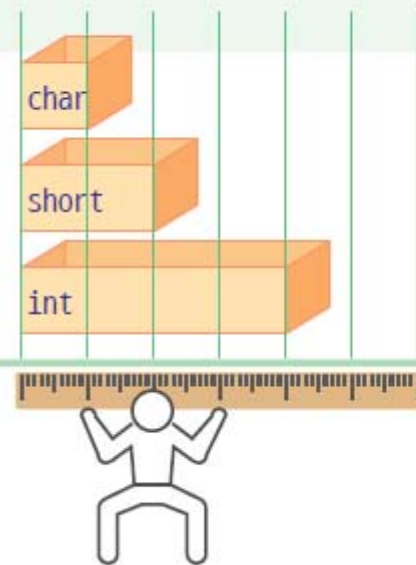


자료형의 크기

Syntax: sizeof()

예

<code>sizeof(x)</code>	// 변수
<code>sizeof(10)</code>	// 값
<code>sizeof(int)</code>	// 자료형
<code>sizeof(double)</code>	// 자료형



sizeof 연산자는 변수나 데이터 타입의 크기를 바이트 단위로 반환합니다.





예제: 자료형의 크기

```
#include <stdio.h>

int main(void)
{
    int x;
    printf("변수 x의 크기: %d\n", sizeof(x));
    printf("char형의 크기: %d\n", sizeof(char));
    printf("int형의 크기: %d\n", sizeof(int));
    printf("short형의 크기: %d\n", sizeof(short));
    printf("long형의 크기: %d\n", sizeof(long));
    printf("long long형의 크기: %d\n", sizeof(long long));
    printf("float형의 크기: %d\n", sizeof(float));
    printf("double형의 크기: %d\n", sizeof(double));
    return 0;
}
```



실행 결과

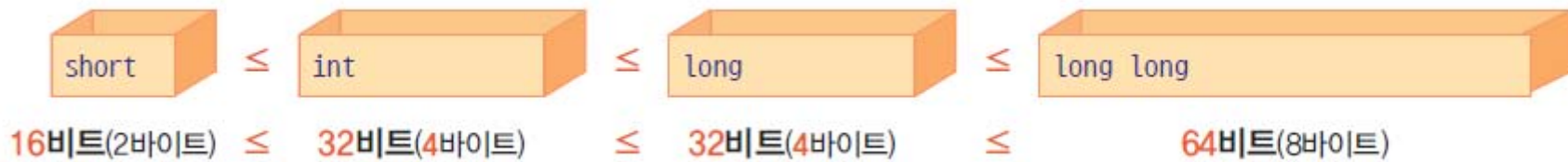
↓ 실행결과

변수 x의	크기: 4
char형의	크기: 1
int형의	크기: 4
short형의	크기: 2
long형의	크기: 4
long long형의	크기: 8
float형의	크기: 4
double형의	크기: 8

- 바이트(byte) 단위: 1byte는 8 bits



정수형



□ int형

$-2^{31}, \dots, -2, -1, 0, 1, 2, \dots, 2^{31} - 1$
(-2147483648 ~ +2147483647)

약 -21억에서
+21억

□ short형

$-2^{15}, \dots, -2, -1, 0, 1, 2, \dots, 2^{15} - 1$
(-32768 ~ +32767)

□ long형

▣ 보통 int형과 같음



정수형 선언의 예

- `short grade;` // short형의 변수를 생성한다.
- `int count;` // int형의 변수를 생성한다.
- `long distance;` // distance형의 변수를 생성한다.



예제

/* 정수 자료형을 사용하는 프로그램*/

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

```
int main(void)
```

```
{
```

```
    short year = 0;           // 0으로 초기화한다.
```

```
    int sale = 0;             // 0으로 초기화한다.
```

```
    long total_sale = 0;      // 0으로 초기화한다.
```

```
    long long large_value;
```

```
    year = 10;                // 약 3만2천을 넘지 않도록 주의
```

```
    sale = 200000000;         // 약 21억을 넘지 않도록 주의
```

```
    total_sale = year * sale; // 약 21억을 넘지 않도록 주의
```

```
    printf("total_sale = %d \n", total_sale);
```

```
    return 0;
```

```
}
```

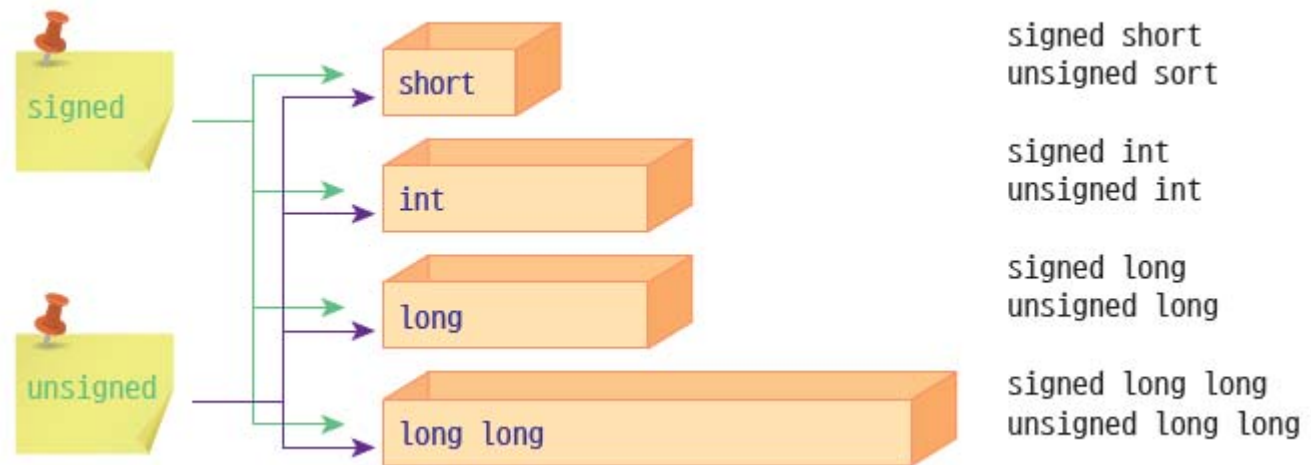
🔍 실행결과

```
total_sale = 2000000000
```




signed, unsigned 수식자

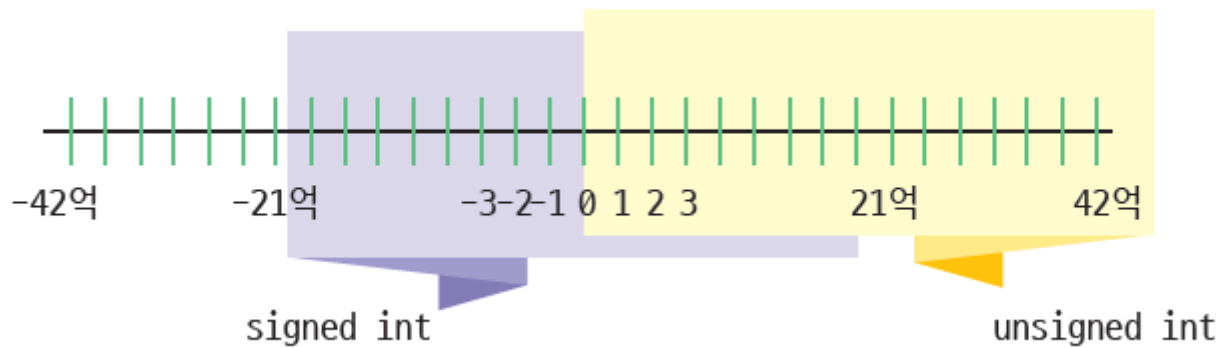
- unsigned
 - ▣ 음수가 아닌 값만을 나타냄을 의미
 - ▣ unsigned int
- signed
 - ▣ 부호를 가지는 값을 나타냄을 의미
 - ▣ 흔히 생략





unsigned int

$0, 1, 2, \dots, 2^{32} - 1$
(0 ~ +4294967295)





unsigned 수식자

- `unsigned int` `speed;` // 부호없는 `int`형
- `unsigned` `distance;` // `unsigned int` `distance`와 같다.
- `unsigned short` `players;` // 부호없는 `short`형
- `unsigned long` `seconds;` // 부호없는 `long`형



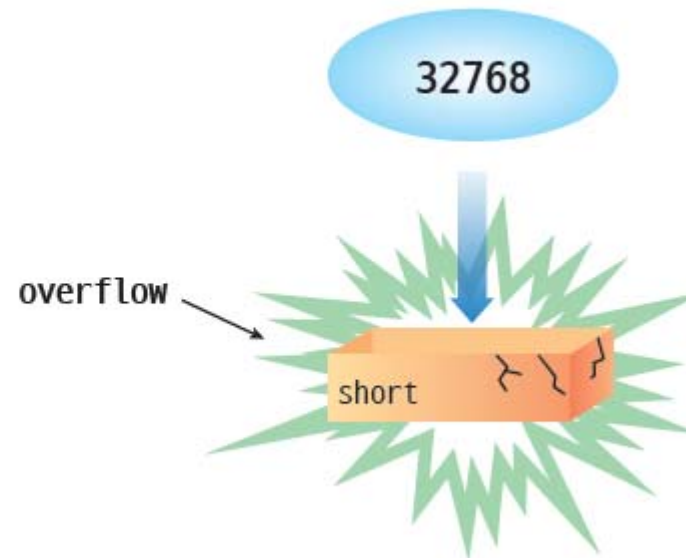
정수형

자료형			비트	범위
정수형	short	부호있는 정수	16비트	-32768~32767
	int		32비트	-2147483648~2147483647
	long			-2147483648~2147483647
	long long		64비트	-9,223,372,036,854,775,808 ~9,223,372,036,854,775,807
	unsigned short	부호없는 정수	16비트	0~65535
	unsigned int		32비트	0~4294967295
	unsigned long			0~4294967295
	unsigned long long		64비트	0~18,446,744,073,709,551,615



오버플로우

- **오버플로우(overflow)**: 변수가 나타낼 수 있는 범위를 넘는 숫자를 저장하려고 할 때 발생





오버플로우

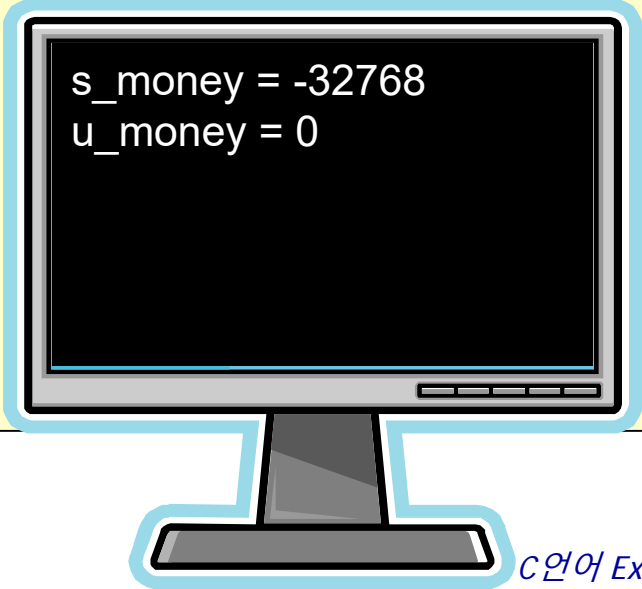
```
#include <stdio.h>
#include <limits.h>

int main(void)
{
    short s_money = SHRT_MAX;           // 최대값으로 초기화한다. 32767
    unsigned short u_money = USHRT_MAX; // 최대값으로 초기화한다. 65535

    s_money = s_money + 1;
    printf("s_money = %d", s_money);

    u_money = u_money + 1;
    printf("u_money = %d", u_money);
    return 0;
}
```

오버플로우 발생!!



```
s_money = -32768
u_money = 0
```



오버플로우

- 규칙성이 있다.
 - ▣ 수도 계량기나 자동차의 주행거리계와 비슷하게 동작



short의 경우



unsigned short의 경우



정수 상수

- 숫자를 적으면 기본적으로 int 형이 된다.
 - ▣ `sum = 123;` // 123은 int 형
- 상수의 자료형을 명시하려면 다음과 같이 한다.
 - ▣ `sum = 123L;` // 123은 long 형

접미사	자료형	예
u 또는 U	unsigned int	123u 또는 123U
l 또는 L	long	123l 또는 123L
ul 또는 UL	unsigned long	123ul 또는 123UL



정수 상수 표현: 10진법, 8진법, 16진법

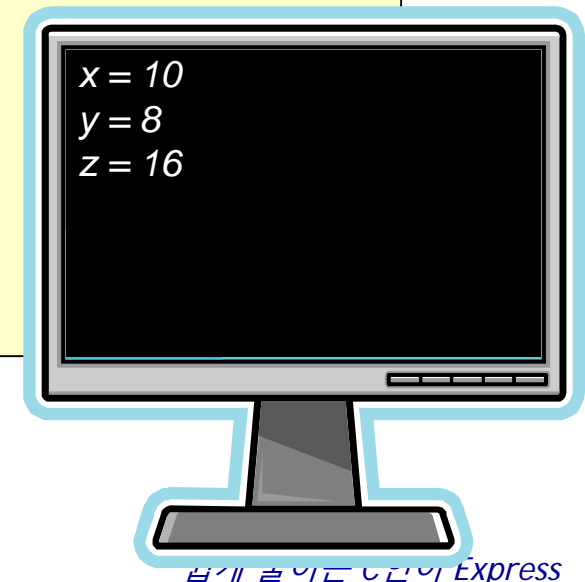
- 정수 상수는 10진법 뿐만 아니라 8진법이나 16진법으로도 표기 가능
- 8진법: 앞에 0을 붙임
 - ▣ $012_8 = 1 \times 8^1 + 2 \times 8^0 = 10$
- 16진법: 앞에 0x를 붙임
 - ▣ $0xA_{16} = 10 \times 16^0 = 10$

10진수	8진수	16진수
0	00	0x0
1	01	0x1
2	02	0x2
3	03	0x3
4	04	0x4
5	05	0x5
6	06	0x6
7	07	0x7
8	010	0x8
9	011	0x9
10	012	0xa
11	013	0xb
12	014	0xc
13	015	0xd
14	016	0xe
15	017	0xf
16	020	0x10
17	021	0x11
18	022	0x12



예제

```
/* 정수 상수 프로그램*/  
#include <stdio.h>  
  
int main(void)  
{  
    int x = 10; // 10은 10진수이고 int형이고 값은 십진수로 10이다.  
    int y = 010; // 010은 8진수이고 int형이고 값은 십진수로 8이다.  
    int z = 0x10; // 010은 16진수이고 int형이고 값은 십진수로 16이다.  
  
    printf("x = %d", x);  
    printf("y = %d", y);  
    printf("z = %d", z);  
  
    return 0;  
}
```





기호 상수

- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것
- (예)
 - ▣ `area = 3.141592 * radius * radius;`
 - ▣ `area = PI * radius * radius;`

 - ▣ `income = salary - 0.15 * salary;`
 - ▣ `income = salary - TAX_RATE * salary;`
- 기호 상수의 장점
 - ▣ 가독성이 높아진다.
 - ▣ 값을 쉽게 변경할 수 있다.



기호 상수의 장점

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

```
int main(void)
{
    ...
    won1 = 1120 * dollar1;
    won2 = 1120 * dollar2;
    ...
}
```

1050

1050

리터럴 상수를 사용하는 경우:
등장하는 모든 곳을 수정하여야 한다.

```
#include <stdio.h>
#define EXCHANGE_RATE 1120
```

1050

```
int main(void)
{
    ...
    won1 = EXCHANGE_RATE * dollar1;
    ...
    won2 = EXCHANGE_RATE * dollar2;
    ...
}
```

기호 상수를 사용하는 경우:
기호 상수가 정의된 곳만 수정하면 한다.

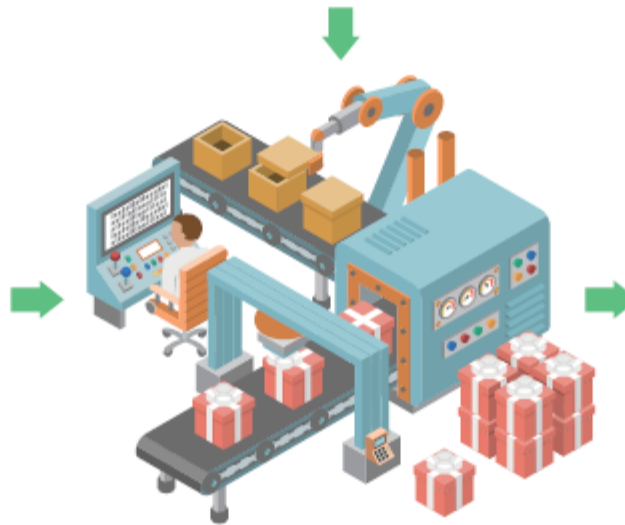


기호 상수를 만드는 방법 #1

EXCHANGE_RATE이라는 기호를 1120으로 정의

#define EXCHANGE_RATE 1120

```
...  
...  
w = EXCHANGE_RATE * 100;  
...  
...
```



전처리기

```
...  
...  
w = 1120 * 100;  
...  
...
```



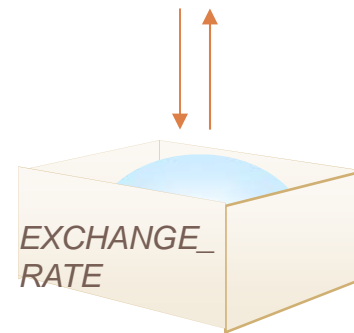
기호 상수를 만드는 방법 #2

변수가 값을 변경할 수 없게 한다.

const int EXCHANGE_RATE = 1120;



const



변수



예제: 기호 상수

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

```
#define TAX_RATE 0.2
```

기호상수

```
int main(void)
```

```
{
```

```
    const int MONTHS = 12;
```

```
    int m_salary, y_salary;    // 변수 선언
```

```
    printf( "월급을 입력하시요: "); // 입력 안내문
```

```
    scanf("%d", &m_salary);
```


```
    y_salary = MONTHS * m_salary; // 순수입 계산
```

```
    printf("연봉은 %d입니다.", y_salary);
```

```
    printf("세금은 %f입니다.", y_salary*TAX_RATE);
```

```
    return 0;
```

```
}
```



```
월급을 입력하시요: 200  
연봉은 2400입니다.  
세금은 480.000000입니다.
```



중간 점검

- 정수형에 속하는 자료형을 모두 열거하라.
- 왜 정수를 하나의 타입으로 하지 않고 int, short, long 등의 여러 가지 타입으로 복잡하게 분류하여 사용하는가?
- 부호가 없는 unsigned int 형의 변수에 음수를 넣으면 어떤 일이 벌어지는가?
- 변수가 저장할 수 있는 한계를 넘어서는 값을 저장하면 어떻게 되는가? 구체적인 예로 short 형의 변수에 32768을 저장하면 어떻게 되는가?
- 숫자 값을 직접 사용하는 것보다 기호 상수를 사용하는 것의 이점은 무엇인가?





Q & A

