작성자 : 김기희 (cupid072@naver.com)

변수와 자료형

- 학습목표
  - C 언어 자료형의 종류를 기술할 수 있다.
  - 자료형에 따른 데이터의 크기를 기술할 수 있다.
  - 10진수, 16진수 데이터 표현을 설명할 수 있다.
  - 정수 데이터의 저장 방식을 설명할 수 있다.
  - 실수 데이터의 저장 방식을 설명할 수 있다.
  - 리터럴의 종류와 표기법을 설명할 수 있다.
  - 변수를 선언하고 초기화를 할 수 있다.
  - scanf() 함수를 사용하여 프로그램을 작성할 수 있다.

작성자 : 김기희 (cupid072@naver.com)

변수와 자료형

● 학습목표

번호	내용	잘함	보통	못함
1	C 언어 자료형의 종류를 기술할 수 있다.			
2	자료형에 따른 데이터의 크기를 기술할 수 있다.			
3	10진수, 16진수 데이터 표현을 설명할 수 있다			
4	정수 데이터의 저장 방식을 설명할 수 있다.			
5	실수 데이터의 저장 방식을 설명할 수 있다.			
6	리터럴의 종류와 표기법을 설명할 수 있다.			
7	변수를 선언하고 초기화를 할 수 있다.			
8	scanf() 함수를 사용하여 프로그램을 작성할 수 있다.			

자료형

- 기본 자료형 (primitive data type)
  프로그램 수행에 필요한 메모리를 할당 받기 위해 컴파일러 차원에서 기본적으로 제공되는 데이터 형

- 복합 자료형 (compound data type)
  기본 데이터형을 응용해서 사용자가 다양한 형태로 만들어 사용하는 데이터 형

<table>
<tr><td rowspan="5">기본자료형</td><td rowspan="4">산술형</td><td rowspan="3">정수형</td><td rowspan="2">문자형</td><td>signed</td><td>char</td></tr>
<tr><td>unsigned</td><td>char</td></tr>
<tr><td rowspan="2">수치형</td><td>signed</td><td>short, int, long, long long</td></tr>
<tr><td>unsigned</td><td>short, int, long, long long</td></tr>
<tr><td>실수형</td><td colspan="3">float, double, long double</td></tr>
<tr><td>무치형</td><td colspan="4">void</td></tr>
<tr><td>복합자료형</td><td colspan="5">array, pointer, struct, union, enum</td></tr>
</table>

자료형에 따른 값의 범위

구분		자료형	범위		Byte
문자		signed char	$-2^{8-1} \sim 2^{8-1}-1$	-128 ~ 127	1
		unsigned char	$0 \sim 2^{8}-1$	0 ~ 255	1
수치형	정수형	signed short	$-2^{16-1} \sim 2^{16-1}-1$	-32768 ~ 32767	2
		unsigned short	$0 \sim 2^{16}-1$	0 ~ 65535	2
		signed int	$-2^{32-1} \sim 2^{32-1}-1$	-2147483648 ~ 2147483647	4
		unsigned int	$0 \sim 2^{32}-1$	0 ~ 4294967295	4
		signed long	$-2^{32-1} \sim 2^{32-1}-1$	-2147483648 ~ 2147483647	4
		unsigned long	$0 \sim 2^{32}-1$	0 ~ 4294967295	4
		singed long long	$-2^{64-1} \sim 2^{64-1}-1$	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	8
		unsigned long long	$0 \sim 2^{64}-1$	0 ~ 18,446,744,073,709,551,615	8
	실수형	float	$\pm 1.18 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$	유효 자리수 6 ~ 9, 일반적으로 7	4
		double	$\pm 2.23 \times 10^{-308}$ to $\pm 1.80 \times 10^{308}$	유효자리수 15 ~ 18, 일반적으로 16	8
		long double	$\pm 3.36 \times 10^{-4932}$ to $\pm 1.18 \times 10^{4932}$	유효자리수 33 ~ 26	16

※ 위의 기준은 GNU C 기준이며 컴파일러에 따라 자료의 크기가 다를 수 있다.

리터럴 #1

- 리터럴은 변하지 않는 데이터, 데이터 그 자체를 의미한다.

- 정수 리터럴은 다음의 접미사를 사용하여 표현한다.

접미사	자료형	예	비고
없음	int	10	default
u, U	unsigned	10U	
l, L	long	10L	
ul, UL	unsigned long	10UL	
ll, LL	long long	10LL	
ull, ULL	unsigned long long	10ULL	

※ long형을 나타내는 접미사 소문자(l)은 사용하지 말 것을 권장 한다.

- 문자상수

  'A' : 단일 따옴표로 감싸서 표현하며 4바이트 크기의 ASCII 코드 값의 표현이다.

리터럴 #2

- 실수형

접미사	자료형	예	비고
없음	double	3.14, 0.314E+1	default
f, F	float	3.14F, 0.314E+1F	
l, L	long double	3.14L, 0.314E+1L	

- 문자열

"ABC" : 이중 따옴표로 감싸서 표현하며 문자열이 저장된 메모리의 선두 주소를 의미한다.

자료형의 크기 확인하기

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("sizeof(char) : %I64u\n", sizeof(char));
    printf("sizeof(short) : %I64u\n", sizeof(short));
    printf("sizeof(int) : %I64u\n", sizeof(int));
    printf("sizeof(long) : %I64u\n", sizeof(long));
    printf("sizeof(long long) : %I64u\n", sizeof(long long));
    printf("sizeof(float) : %I64u\n", sizeof(float));
    printf("sizeof(double) : %I64u\n", sizeof(double));
    printf("sizeof(long double) : %I64u\n", sizeof(long double));
    printf("sizeof('A') : %I64u\n", sizeof('A');                    // 문자 상수는 int형으로 4Byte ASCII 코드 값으로 표현
    printf("sizeof('한') : %I64u\n", sizeof("한"));                 // 문자열 상수 한글 하나의 문자 크기는 2Byte
    char ch[] = "한";                                               // UTF-8 문자 하나를 저장하기 위한 배열의 크기 3
    printf("sizeof(ch) : %I64u\n", sizeof(ch));                     // UTF-8 문자 + NULL 문자

    return 0;
}
```

◆ gcc에서 8바이트 long long 형의 출력 포멧은 다음과 같다.

- ❖ I64d : long long
- ❖ I64u : unsigned long long

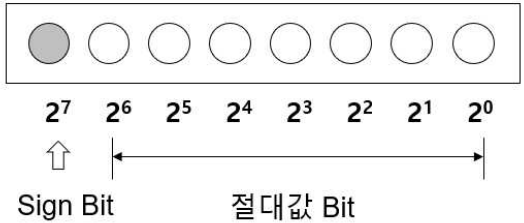
◆ gcc에서의 출력 포멧은 inttypes.h 헤더를 통해 확인할 수 있다.

signed vs unsigned 형 데이터 표현

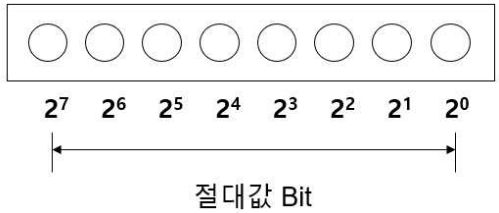
- signed 자료형은 제일 앞 선두 한 개의 비트를 사용하여 부호를 표현한다.
  - char 형 : 부호비트 1개, 데이터 저장비트 7개
  - int 형 : 부호비트 1개, 데이터 저장비트 31개

- unsigned 자료형은 모든 비트를 데이터 저장용으로 사용한다.
  - unsigned char 형 : 데이터 저장비트 8개
  - unsigned int 형 : 데이터 저장비트 32배

[signed char 형 기억공간의 형태]



[unsigned char 형 기억공간의 형태]



정수 값의 저장 방식 (양의 값)

- 정수 값은 2진수로 변환되어 저장된다.
- 2진수 변환은 더 이상 나눌 수 없을 때 까지 2로 나누며 나머지를 열거한 후 부족한 자리 수를 0으로 채운다.
- 10진수 46의 2진수 변환

			1%2	2 % 2	5 % 2	11 % 2	23 % 2	46 % 2
몫			0	1	2	5	11	23
나머지	0	0	1	0	1	1	1	0

- 2진수 46의 10진수 변환

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
몫	0	0	1	0	1	1	1	0
나머지	0	0	32	0	8	4	2	0

정수 값의 저장 방식 (음의 값)

- 음수에 대한 2진수 변환은 절대 값을 2진수로 변환하여 2의 보수를 취한다.

- 10진수 -46의 2진수 변환

① 절대 값을 2진수로 변환
00101110

② 1의 보수를 취한다.
11010001

③ 1을 더한다. (2의 보수)
11010010

- 2진수 음수의 10진수 변환

① -46의 2진수
11010010

② 1의 보수를 취한다.
00101101

③ 1을 더 한 값을 10진수로 변환 후 부호를 붙인다.
00101110

보수(complement)

- 보수란?
  각 자리수에 대하여 특정 값이 되기 위해 보충을 해주어야 하는 수를 의미한다.

- 예) 10의 보수
  6에 대한 10의 보수 값 : 4
  즉 6의 값에 4를 더해야 10 값이 됨

  예) 1의 보수 (two's complement)
  0에 대한 1의 보수 값 : 1
  1에 대한 1의 보수 값 : 0

  이진수 101001011 의 1의 보수 값은
  010110100

16진수와 8진수

- 46의 2진수 00101110
- 16진수 한 자리는 2진수 4자리에 대응한다.

0010	1110
2	E
0x2E	

- 8진수 한 자리는 2진수 3자리에 대응한다.

00	101	110
0	5	6
056		

Over flow와 Under flow

- Over flow : 연산 수행 시 변수에 저장할 수 있는 최대 값 이상을 저장할 때 발생되는 현상

```
char c = 127;          // 127의 비트배열은 01111111
c += 1;                // 비트배열 10000000이 된다.
printf("%d\n", c);     // 변수 c의 값은 -128
```

※ unsigned char 형은 255에 1을 더하면 0이 된다.

- Under flow : 연산 수행 시 변수에 저장할 수 있는 최소 값 이하의 값을 저장할 때 발생되는 현상

```
char c = -128;         // 128의 비트배열은 10000000
c -= 1;                // 비트배열 10000000은  01111111이 된다.
printf("%d\n", c);     // 변수 c의 값은 127이 된다.
```

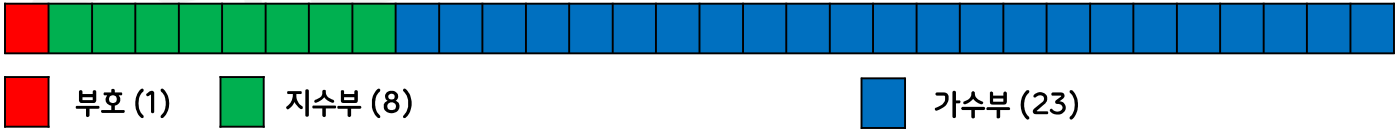
※ unsgiend char 형은 0에 1을 빼면 255가 된다.

실수 값의 저장 방식 #1

- 실수의 표기는 고정 소수점(Fixed Point) 방식과 부동 소수점(Floating Point) 방식이 있다.
- 부동 소수점 방식은 지수부와 가수부로 분리하여 데이터를 저장한다.
- IEEE 754 기준으로 부동 소수점은 다음과 같이 저장된다.

자료형	부호(Sign)	지수부(Exponent)	가수부(Fraction)
float	1	8	23
double	1	11	52

- 부동소수점 표현 방식은 컴파일러나 시스템 아키텍처에 따라 다를 수 있다.



실수 값의 저장 방식 #2

- 실수의 값은 정규화를 통해 저장되며 정규화 과정은 다음과 같다.

① 저장하고자 하는 값을 2진수로 변환한다.
- ❖ 소수점 이상의 값은 몫이 0이 될 때 까지 2로 나누며 나머지를 떨군다.
- ❖ 소수점 이하의 값은 정수부와는 반대로 소수점 이하의 값이 0이 될 때 까지 2를 곱하며 소수점 이상의 값을 떨군다.

② 2진수로 변환 후 소수점 앞의 값이 1이 될 때 까지 소수점을 이동시킨다. 이때 소수점이 이동한 수 가 지수가 되며 앞으로 이동 시 양의 지수, 뒤로 이동 시 음의 지수가 된다.

③ 부호 비트에 실수의 부호를 저장한다. 양수이면 0, 음수이면 1을 저장

④ 지수의 값에 Bias 값을 더한 후 2진수로 변환하여 저장한다. Bias 값을 더하는 이유는 양의 지수와 음의 지수 모두를 표현해야 하기 때문이다. 뿐만 아니라 지수 값 0(지수부의 전체 비트가 0)과 257(지수부의 전체비트가 1)은 특수 용도로 예약이 되어 있기 때문이다.

⑤ 소수점 이하의 값을 가수부에 저장한다.

실수 값의 저장 방식 #3

- 12.20의 2진수 변환 > 1100.00110011001100110011.....

1 / 2	3 / 2	6 / 2	12 / 2	0.2 * 2	0.4 * 2	0.8 * 2	0.6 * 2	0.2 * 2
0	1	3	6	0.4	0.8	1.6	1.2	0.4
1	1	0	0	0	0	1	1	0

- 변환된 2진수에서 소수점 앞의 값이 1이 되도록 소수점을 이동시킨다.
  1.10000110011001100110011... < 앞으로 3번 이동하였으므로 지수 값은 +3

- 지수 값에 127을 더한 값 130을 2진수로 변환한 값 10000010을 지수부에 저장한다.

- 소수점 이하의 10000110011001100110011 값을 가수부에 저장한다.

- 최종 저장 값은 0100 0001 0100 0011 0011 0011 0011 0011

진법 표현

- 정수형 데이터 표현에 사용되는 진법

진법	설명	예
10 (DEC)	0이 아닌 수로 시작	1, 15, 300
8 (OCT)	0으로 시작하며 한 자리를 0 ~ 7까지의 수로 표현	01, 020, 012
16 (HEX)	0x로 시작하며 한 자리를 0 ~ 15까지의 수로 표현 0 ~ 15의 수에 대응하는 0 ~ 9, a ~ f의 문자로 표현	0x30, 0xFF00, 0x123c

- 컴퓨터가 데이터 저장에 사용하는 2진수 표기는 지원되지 않는다.
- 정수형의 기본 자료형은 int 형이다.
- 8진법은 되도록이면 사용하지 않는다.
- long 형 정수 값은 영문자 L을 접미어로 붙여 표현한다.
- long long 형 정수 값은 영문자 LL을 접미어로 붙여 표현한다.
- long 형을 나타내기 위한 접미어 L은 소문자는 되도록이면 사용하지 않는다.

변수의 선언과 초기화

- 변수는 데이터를 저장하기 위한 공간이다.
- 프로그램 실행 도중 변수의 값은 언제든지 변경될 수 있으므로 변수라고 한다.

- 변수의 선언 형식

  자료형 변수명;

```
int a;
```

- 변수 선언과 동시에 초기화

  자료형 변수명 = 초기값;

```
int a = 10;
```

변수 선언 시 signed vs unsigned

- 명시적으로 signed 또는 unsigned를 기술하지 않을 경우 signed 자료형이다.

```
int a = 10;                 // 변수 a는 signed형(부호가 없는 정수형)
long int a = 10;            // 변수 a는 signed long ing형(부호가 없는 long 정수형)
```

- signed 또는 unsigned 뒤에 명시적인 자료형을 기술하지 않을 경우 int 자료형이다.

```
signed a = 10;              // 변수 a는 int형
unsigned long a = 10;       // 변수 a는 long 형
```

- signed, unsigned 접두어 사용시, long 또는 long long 형은 int 를 생략할 수 있다.

```
long a = 10;                // long int a = 10; 과 동일
long long a = 10;           // long long int a = 10; 과 동일
unsigned a = 10;            // 변수 a는 unsigned int 형
signed a = 10;              // 변수 a는 signed int 형
```

변수의 선언

- 변수의 선언은 데이터를 저장하기 위한 메모리 공간의 확보이다.
- 초기화 되지 않은 자동변수는 쓰레기(알 수 없는) 값을 가진다.
- 변수 선언 시 자료형은 메모리 공간을 확보하기 위한 크기를 지정한다.
- 변수명은 개체를 구분하기 위해 프로그래머에 의해 부여된 이름이다.
- 파일 내에서 선언된 변수는 프로그램 전체에서 사용 가능이 가능하다.
- 함수나 중괄호 블록 내에서 선언된 변수는 해당 함수나 블록 내에서 사용이 가능하다.
- 컴파일러에 따라 변수 선언의 위치가 제한될 수 있다.
  - C90 까지는 함수의 선두에 변수의 선언을 두어야 함
  - C99 부터는 필요한 위치에서 변수를 선언할 수 있음
- 동일한 자료형을 가지는 변수는 동시에 선언이 가능하다. (권장하지 않음)

변수명 명명 규칙

① 알파벳 대문자 A ~ Z, 소문자 a ~ z, 숫자 0 ~ 9, 밑줄 _ 을 조합하여 생성한다.

② 숫자로 시작할 수 없다.

③ 대문자와 소문자를 구분한다.

④ 예약어를 식별자로 사용할 수 없다.

- 예약어(Keyword)

  예약어는 Keyword 또는 Reserved word라고 하며, 컴파일러(Compiler)에 의해 그 용도가 약속된 단어이다.

  int a;

  자료형을 나타내는 int 은 예약어이다.

  변수를 구분하기 위한 변수명 a는 식별자이다.