3.1 : C 프로그램 구조와 프로그램 실행

실습예제 3-1: 01comments.c

```
#include <stdio.h>

// 운영체계가 호출하는 함수, void로 매개변수 없음을 표시
int main(void)
{
    puts("3장 첫 c 프로그램!\n");

    printf("키워드: int void return 등\n");
    printf("식별자: main puts 등\n");

    return 0;
}
```

【생각해 보기】 출력은 어떤 형태일까요?

3장 첫 c 프로그램! 키워드: int void return 등 식별자: main puts 등	3장 첫 c 프로그램! 키워드: int void return 등 식별자: main puts 등

☞ 제어문자 출력

제어문자	의미	제어문자	의미
\n	줄 바꿈	\a	벨 소리
\t	탭 위치로 이동	\b	왼쪽으로 한 칸 이동
\r	맨 앞으로 이동		

【도전과제 - 1】 다음 프로그램 출력은?

```
#include <stdio.h>

int main(void)
{
        printf("Goot\bd\tchance\n");
        printf("Cow\rW\n");

        return 0;
}
```

예상 출력

3.2 : 자료형(Data Type)과 변수(Variable) 선언

□ 변수(Variable)

- 프로그램은 문제 해결 **시나리오**이며, 프로그램은 문제 해결을 위해 **데이터**를 처리

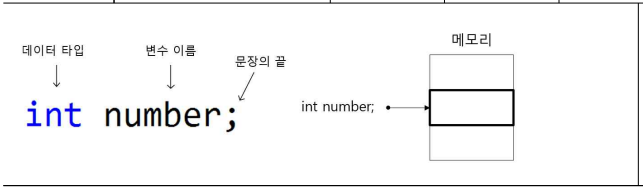
구분	정의	값 변경	메모리 역할	예시코드
변수	실행 중 값이 바뀔 수 있는 데이터 공간	가능	값을 저장할 공간 제공	int age = 20; age = 25;
	age = 25	20 -> 25 (내용을 언제든지 바꿀 수 있음)		

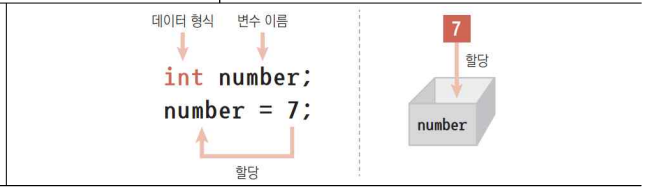
□ 변수(Variable)는 왜 필요할까?

- 변수 : 프로그램에서 일시적으로 데이터를 저장하는 공간
- 데이터가 입력되면 어딘가에 저장해야 다음에 사용할 수 있음
- 변수의 저장공간은 **메인 메모리**에 만들어 짐
- 변수로 메모리의 주소를 사용할 수 있지만 기억하기 불편 - 변수 이름으로 사용
  (예: 메모리 100번지에 20을 대입하라 / 변수 sum에 20을 대입하라)

□ 변수(Variable) 선언 및 활용

자료	설명	자료형	크기 (Byte)	선언 예시	사용 예시/출력 예시
정수	정수 저장 (소수점 없음)	int	4	int age = 20;	printf("%d", age); → 20 출력
실수	실수 저장 (소수점 포함)	double	8	double pi = 3.14;	printf("%.2lf", pi); → 3.14 출력
문자	문자 1개 저장	char	1	char grade = 'A';	printf("%c", grade); → A 출력
문자열	여러 문자 저장	char[]	가변	char name[] = "Tom";	printf("%s", name); → Tom 출력



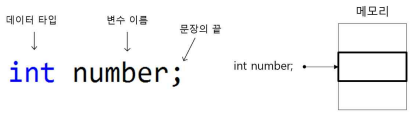


- 변수는 중괄호의 블록({})에 선언하여 선언한 위치부터 블록 끝까지 사용 가능
- 변수의 자료형이 같으면 동시에 여러 개 선언 가능 (예: int b, c)
- 대입 연산자는 왼쪽의 변수에 오른쪽의 값을 저장함 (예: a = 10;)
- 변수는 대입 연산자 왼쪽에서는 **저장공간**, 오른쪽에서는 **값이** 됨 (예: a = 10; b = a;)

※ l-value(left value) : 저장공간으로 사용하는 변수, r-value(left value) : 값으로 사용하는 변수

□ 변수(Variable) 이름 짓기

- 메모리 시작 주소의 이름 (앞으로 number라고 불러 주세요)

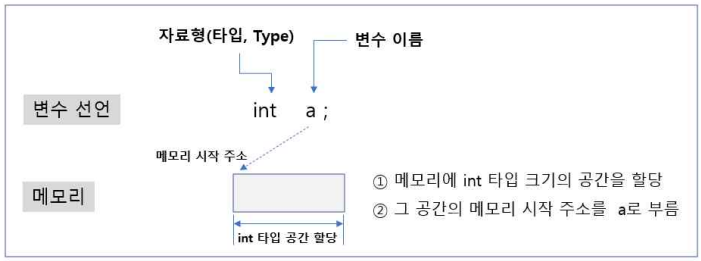


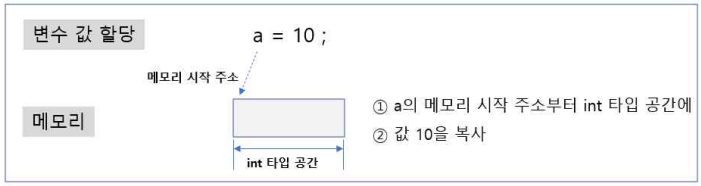
- 변수 이름 만드는 규칙

■ 알파벳 문자와 숫자, 밑줄 문자 _로 구성 ■ 첫 번째 문자는 반드시 알파벳 또는 밑줄 문자 _ ■ 대문자와 소문자를 구별 ■ C 언어의 키워드와 똑같은 이름은 허용되지 않음	sum	o
	_count	o
	dongyang3	o
	dong_yang	o
	2nd_try	x
	Dongyang#	x
	int	x

- 좋은 변수 이름 만들기 : 변수의 역할을 잘 설명하는 이름
  - ■ 밑줄 방식 : bank_account
  - ■ 단어의 첫 글자를 대문자 : BankAccount

□ 변수(Variable) 타입이 왜 필요할까?

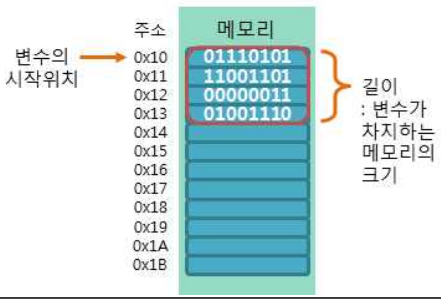




변수 타입이 왜 필요할까?

■ 변수 타입은 **메모리 크기**를 나타냄
- 변수 이름으로 메모리 시작 주소를 알아냄
- 메모리 크기를 알아야 메모리 주소에서 얼마만큼 읽을지 결정할 수 있음

■ 컴퓨터는 변수 타입으로 **데이터 해석**
- 메모리에 2진수 '0100 0001' 값은 int 경우 65로, char인 경우 'A'로 해석



실습예제 3-2: 02variables.c

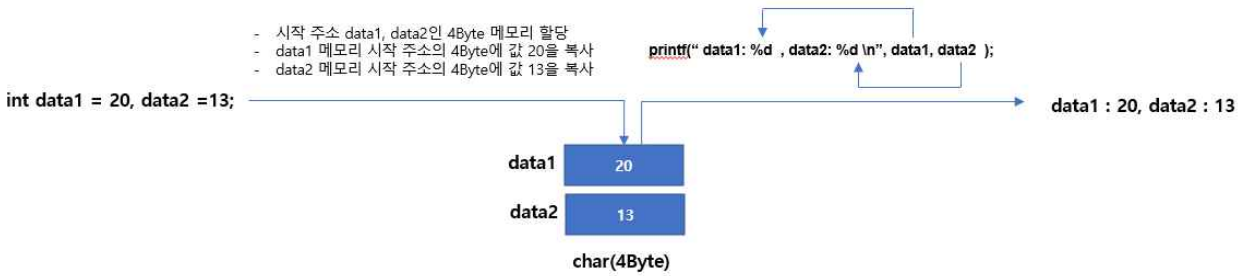
```
/**
 * 소스 : 02variables.c
 * 버전: V1.0
 **/
#include <stdio.h>

int main(void)
{
        int year = 2022;     //선언과 동시에 변수 초기화
        int credits;

        credits = 15;        // 선언된 변수에 초기화

        printf("%d년도\n", year);
        printf("이수학점: %d학점\n", credits);

        return 0;
}
```

【생각해 보기】 실행 결과는 ?

실습예제 3-3: 03addsub.c

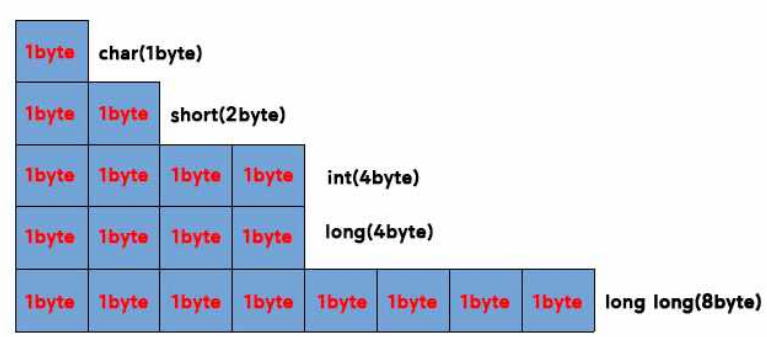
```
/**
 * 소스 : 03addsub.c
 * 버전: V1.0
 **/
#include <stdio.h>

int main(void)
{
        int data1 = 20, data2 = 13;

        // 대입 연산자의 왼쪽과 오른쪽에서의 변수의 의미 해석
        int diff = data1 - data2;
        int sum = data1 + data2;

        printf("data1: %d, data2: %d\n", data1, data2);
        printf("차 : %d, 합: %d\n", diff, sum);

        return 0;
}
```

【생각해 보기】 실행 결과는 ?



3.3 : 기본 자료형(Data Type)

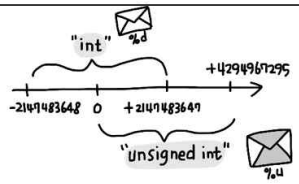
□ 기본 자료형 - signed 정수 자료형



자료형 (Data Type)	할당되는 메모리 크기 (Byte)	표현 가능 데이터 범위	출력 서식 문자
char	1	-128 ~ 127	%c (문자로 출력) 또는 %d
short	2	-32768 ~ 32767	%d
int	4	-214748 ~ 2147483647	%d
long	4	-214748 ~ 2147483647	%ld
long long	8	-2^{63} ~ 2^{63} - 1	%lld

※ 자료형의 int(4byte)와 long(4byte)의 크기가 같은 이유
자료형 int는 운영체제에 따라서 2byte와 4byte로 사용할 수 있는 메모리 크기가 달라짐 - 16bit의 운영체제에서 int의 크기는 2byte, long의 크기는 4byte - 32bit와 64bit 운영체제에서는 int의 크기는 4byte, long의 크기는 4byte - 윈도우가 아닌 다른 운영체제에서는 long의 메모리 크기가 8byte인 경우도 있음

□ 기본 자료형 - unsigned 정수 자료형



- 정수 자료형은 양수와 음수 모두 저장함
- **unsigned** 정수 자료형은 양수만 저장함 - 2배 넓은 범위의 값을 저장할 수 있음

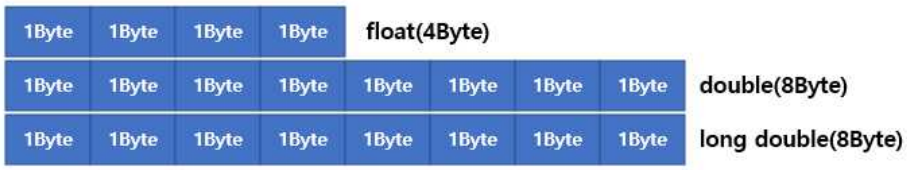
자료형 (Data Type)	할당되는 메모리 크기 (Byte)	표현 가능 데이터 범위	출력 서식 문자
unsigned char	1	0 ~ 255	%u
unsigned short	2	0 ~ 65535	%u
unsigned int	4	0 ~ 4294967295	%u
unsigned long	4	0 ~ 4294967295	%lu
unsigned long long	8	0 ~ 2^{64} - 1	%llu

실습예제 3-4: 04integer.c

【생각해 보기】 실행 결과는 ?

3.3 : 기본 자료형(Data Type)

□ 기본 자료형 - 부동소수(실수) 자료형



자료형 (Data Type)	할당되는 메모리 크기 (Byte)	표현 가능 데이터 범위	출력 서식 문자
float	4	$1.17549E^{-38}$ ~ $3.40282E^{38}$	%f
double	8	$2.22507E^{-308}$ ~ $1.79769E^{308}$	%f
long double	8	$2.22507E^{-308}$ ~ $1.79769E^{308}$	%f

※ long double은 운영체제와 플랫폼마다 크기가 다름

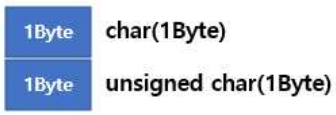
운영체제	CPU(플랫폼)	바이트 크기	비트 크기
MS Windows	X86(32bit)	8	64
	X86-64(64bit)	8	64
Linux	X86(32bit)	12	96
	X86-64(64bit)	16	128

실습예제 3-5: 05floatdouble.c

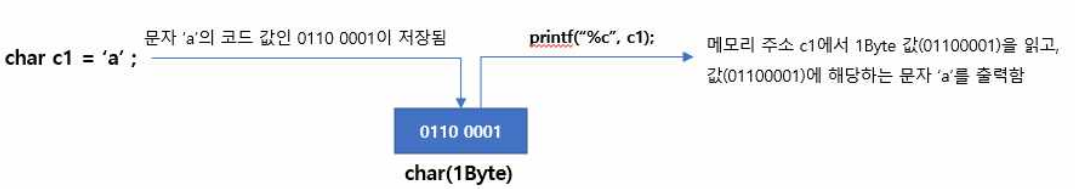
【생각해 보기】 실행 결과는 ?

3.3 : 기본 자료형(Data Type)

□ 기본 자료형 - 문자 자료형



자료형 (Data Type)	할당되는 메모리 크기 (Byte)	표현 가능 데이터 범위	출력 서식 문자
char	1	-128 ~ 127 (문자는 실제 0~127까지 사용)	%c
unsigned char	1	0 ~ 255	%c



※ 표3-6 아스키 코드표 참조 (1Byte인 char로는 한글 문자 저장 불가)

실습예제 3-6: 06char.c

```
```

【생각해 보기】 실행 결과는 ?

```
```

3.3 : 기본 자료형(Data Type)

□ 자료형의 크기 알아내기 - sizeof 연산자

실습예제 3-7: 07sizeflow.c

【생각해 보기】 실행 결과는 ?

3.4 : 상수 표현 방법

□ 상수(Constant)의 개념과 표현 방법

구분	정의	값 변경	메모리 역할	예시코드
상수	프로그램 실행 중 변하지 않는 값	**불가능**	고정된 값 저장	#define PI 3.14 const int MAX = 100;
	#define PI 3.14	3.14 (이름은 있지만 변경은 안 됨)		

□ 상수(Constant)

- 데이터는 어딘가 저장되어야 다음에 사용할 수 있음
- 상수 : 저장된 값의 변경이 불가능한 공간 (예: 3.14, 10, 'A', "Hello World")
- **보통 상수는 이름이 없음**, 이런 상수를 **리터럴(literal) 상수**라고 함 (예: 3.14, 10)
- 이름을 붙일 수 있음, 이를 **기호(symbolic) 상수**라 함 (#define PI 3.14)

□ 상수(Constant) 표현법 (정수, 실수, 문자, 문자열)

자료형	설명	상수 예시	코드 예시	메모리 저장 예시
정수	소수점 없는 수	**10, -25**	20 const int NUM = 10; #define YEAR 2025	이진 정수로 저장됨
	• 정수 상수는 **10진수, 8진수, 16진수**로 표현 가능 ▸정수 10 표현 방법: 10(10진수), 0xA(16진수). 012(8진수)			
실수	소수점 있는 수	**3.14, -0.5**	3.33 const float DISTANCE = 2.89; #define PI 3.14	부동소수점 방식으로 저장
	• 실수 상수는 **소수점 형태, 지수 형태로** 표현 가능 ▸**소수점 형태**: 3.14, -1.5, **-10.** ▸**지수 형태** : 3.14e-5, 0.314E-4			
문자	한 개 문자 (작은 따옴표)	**'A', '1'**	'A' const char grade = 'A'; #define TOP 'A'	(문자 1개) → 아스키코드 65
	• 아스키(ASCII) 코드 : 문자를 메모리에 표현하기 위한 약속 ▸문자 'A'의 아스키 코드 값 : 65 (0x41, 01000001)			
문자열	문자들 묶음 (큰 따옴표)	**"Hello", "AB"**	"DONGYANG" const char str[] = "AB"; #define NAME "DONGYANG"	'A' + 'B' + '\0' (널문자 포함)

실습예제 3-10: 10const.c

【생각해 보기】 실행 결과는 ?