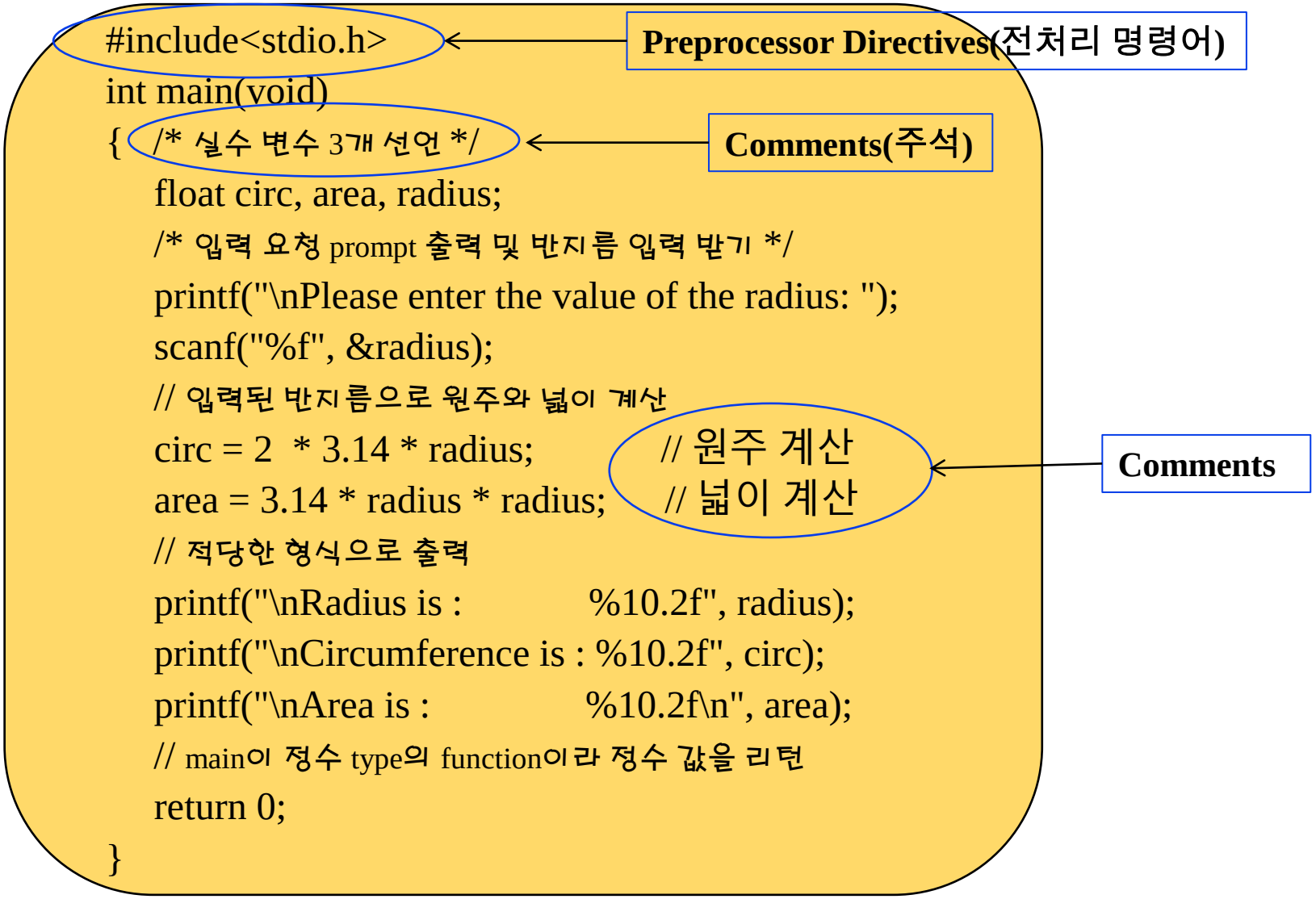
컴퓨터 프로그래밍 I
(CSE2003-3)

Mon/Wed 16:30-17:45 pm
Week 1

Introduction to the C Language
Structure of a C Program

C 프로그램 구조 (source file)



C 프로그램 기본 구조 이해

#include <헤더파일이름>

◆ 전처리기 지시자(Preprocessor directives)

- #으로 시작하는 문장으로 **include, define** 등의 명령(지시자)을 포함
- 컴파일 하기 전에 먼저 처리해야 할 일을 수행하도록 하는 명령문장이기 때문에, 프로그램 제일 앞쪽에 선언
- 문장 #include는 다음에 나오는 헤더 파일 stdio.h라고 하는 파일을 이 문장이 있는 부분에 삽입하는 역할을 수행. stdio.h 파일에는 scanf()와 printf()와 같은 입출력 관련 함수들이 있어서, 이 파일을 포함시켜야 여러가지 입출력 함수를 프로그램에 사용할 수 있다.

C 프로그램 기본 구조 이해

Comments

- 주석은 프로그램 언어의 문법과는 관계없이 프로그램을 설명하는 설명문을 표현하기 위한 방법

```
/* 실수 변수 3개 선언 */

// 원주 계산
// 넓이 계산
```

두 가지 방법

- 원래 /* 주석 부분 */로 주석을 표현
  - 주석의 시작 부분인 /와 * 사이, / 와 * 사이에 공백이 없어야 함.
- 주석 표시 //는 //이후부터 그 줄 끝까지 주석으로 간주

중요성

- 적절한 주석이 없는 프로그램은 이해하기 어렵고, 수정이 어려움
- 주석은 프로그램 유지 보수에 매우 중요하며, 따라서 프로그램에 적절히 주석을 붙이는 것은 매주 중요한 습관이며, 소프트웨어 공학적으로 볼 때 실제 정확한 프로그램을 작성하는 것만큼 중요

C 프로그램 구조 (source file)

```
#include<stdio.h>
int main(void)
{   /* 실수 변수 3개 선언 */
    float circ, area, radius;
    /* 입력 요청 prompt 출력 및 반지름 입력 받기 */
    printf("\nPlease enter the value of the radius: ");
    scanf("%f", &radius);
    // 입력된 반지름으로 원주와 넓이 계산
    circ = 2  * 3.14 * radius;              // 원주 계산
    area = 3.14 * radius * radius;      // 넓이 계산
    // 적당한 형식으로 출력
    printf("\nRadius is :            %10.2f", radius);
    printf("\nCircumference is : %10.2f", circ);
    printf("\nArea is :              %10.2f\n", area);
    // main이 정수 type의 function이라 정수 값을 리턴
    return 0;
}
```

main function

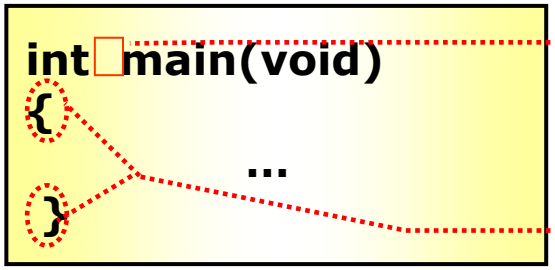
Data type 선언: 실수 형식 변수 3개 선언

출력 function

입력 function

프로그램의 이해 1

- **main 함수**
  - 함수 main()은 C 언어에서 프로그램이 시작되는 함수
  - 프로그램이 실행되려면 프로그램은 반드시 main() 함수를 가져야 함
  - 함수 main()에서 기술되는 단어인 int, main, void는 모두 소문자로 대소문자를 구분하여 정확히 기술



int 와 **main** 사이의 공백은 단어를 구별하는 중요한 의미가 있으므로, 하나 이상의 공백은 반드시 필요.

중괄호 **{ }**는 반드시 있어야 하며, 이는 **main()** 함수의 시작과 끝을 의미한다.

프로그램의 이해 2

- **printf 함수**
  - 표준출력으로 출력을 하기 위한 함수

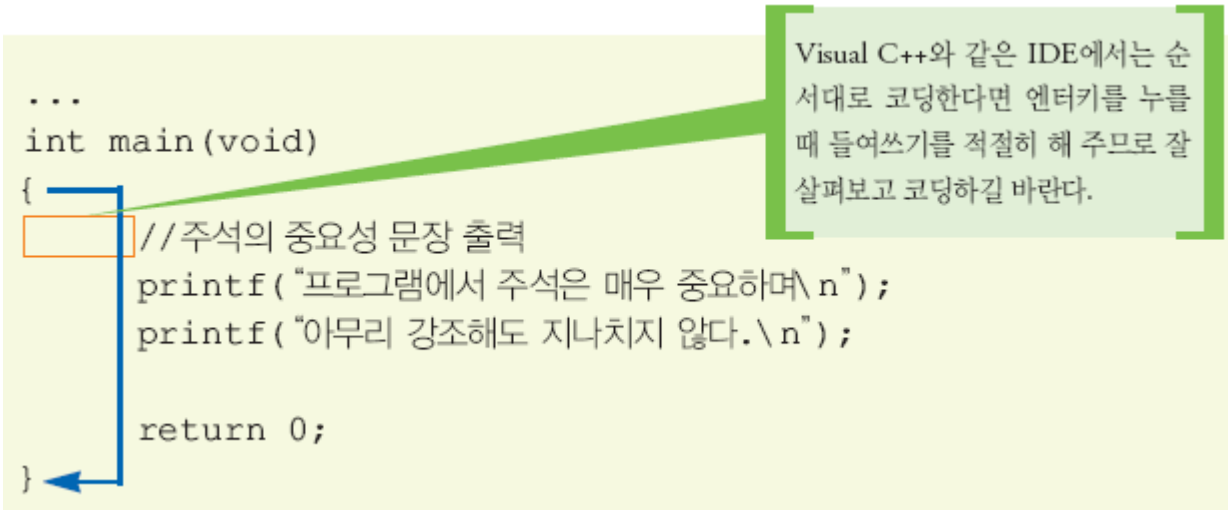
```
printf("나는 홍 길동이고 기초공학설계를 수강합니다.");
```

  - 이 함수는 함수 이름인 printf 이후의 ( ) 사이에 큰 따옴표 "..." 로 둘러싸인 문자열을 출력하는 역할을 담당
  - 세미콜론은 C 언어에서 문장의 마지막에 반드시 나와야 하는 문자
- **문자열(String)**
  - 문자열을 구성하려면 문자열을 큰 따옴표로 앞과 뒤를 둘러싸야 함
- **문자열에서의 \n**
  - \n 의 의미는 \n이 나타나는 이후에는 새로운 줄에 출력하라는 의미

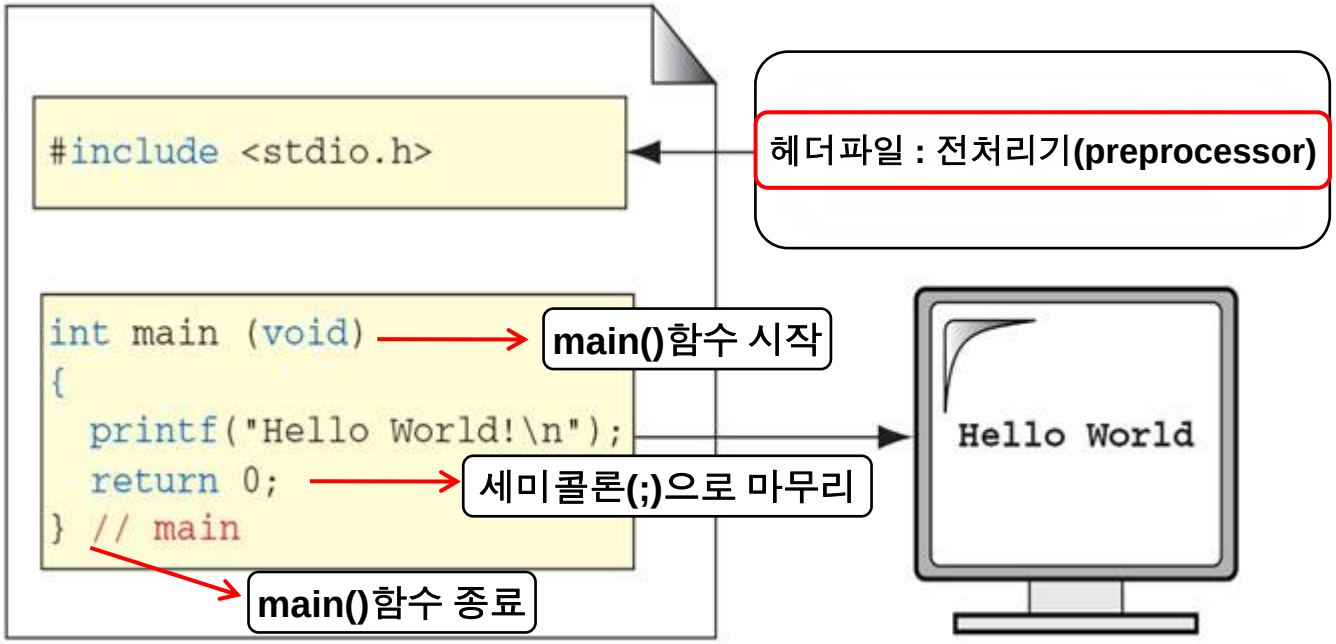
```
printf("나는 홍 길동이고 ");
printf(" 기초공학설계를 수강합니다.\n");
```

들여쓰기(indentation)

- **Indentation**
  - 중괄호로 {...} 표현하는 함수 정의 부분이나 블록 내에서, 각 코드문장의 시작을 여러 개의 공백이나 탭을 이용하여 여백을 일정하게 만들고 코딩하는 방법
- **중요**
  - 들여쓰기는 문법과는 관련이 없으나 사용자가 프로그램을 볼 때 이해를 돕기 위하여 반드시 지켜야 할 코딩 요령



C 프로그램 구조

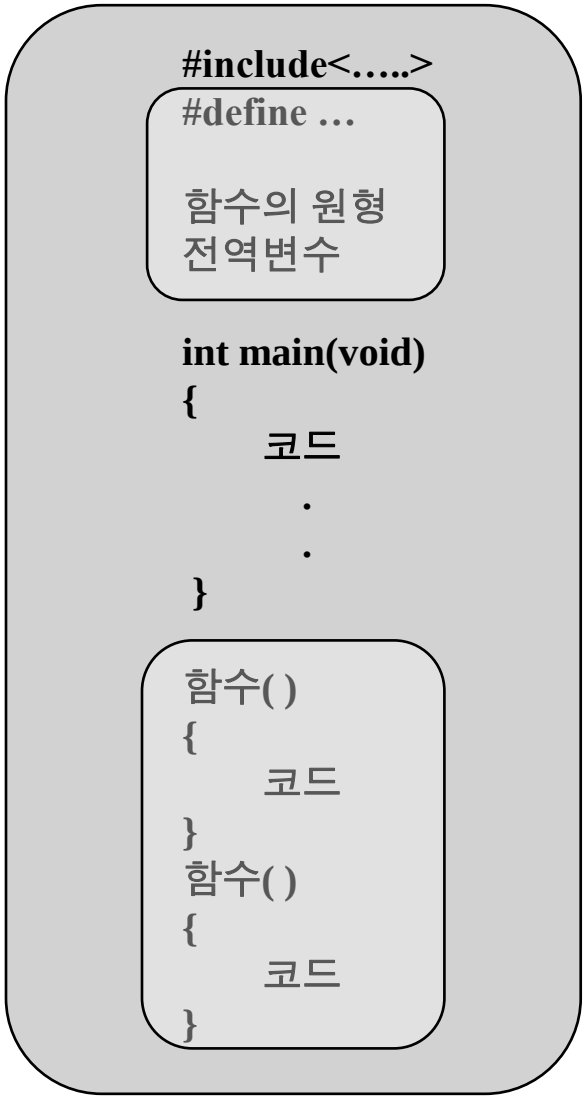


- #include <stdio.h> : printf 문을 사용하기 위해 삽입해야 하는 문장
  즉, printf와 scanf 함수는 이미 만들어져 있는 function들인데 이것들이 저장되어 있는 header file 이름이 stdio.h임. 컴파일러에게 이 헤더파일에 있는 function들을 쓰겠다고 선언함.
  ※ printf 에 대한 자세한 설명은 뒷 부분에서 하겠음

- 각 문장들의 끝은 반드시 세미콜론(;)으로 끝나야 함. 아니면 에러!!!

C 프로그램 구조

```
#include<stdio.h>

int main(void)
{
    float circ, area, radius;

    printf("\nPlease enter the value of the radius: ");
    scanf("%f", &radius);

    circ = 2  * 3.14 * radius;          // 원주
    area = 3.14 * radius * radius;      // 넓이

    printf("\nRadius is :          %10.2f", radius);
    printf("\nCircumference is : %10.2f", circ);
    printf("\nArea is :            %10.2f\n", area);

    return 0;
}
```



C 프로그램 구조

```
#include<stdio.h>

int main(void)
{
    float circ, area, radius;

    printf("\nPlease enter the value of the radius: ");
    scanf("%f", &radius);

    circ = 2  * 3.14     * radius;        // 원주
    area = 3.14 * radius * radius;      // 넓이

    printf("\nRadius is :              %10.2f", radius);
    printf("\nCircumference is : %10.2f", circ);
    printf("\nArea is :                %10.2f\n", area);

    return 0;
}
```

변수(Variables) 및 자료형(Types) 선언

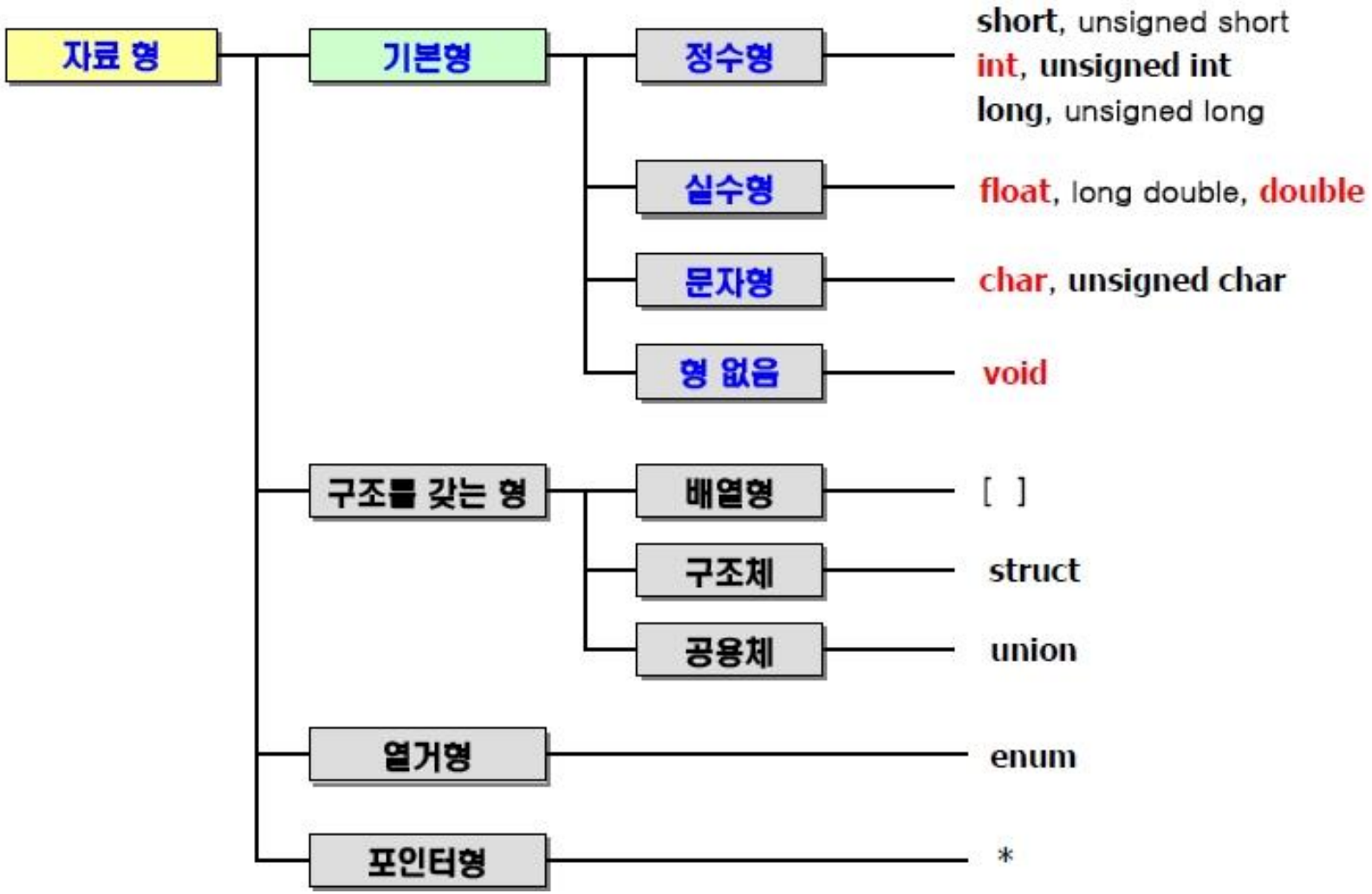
- 프로그램에서 사용하는 변수를 미리 선언해야만 함
- 선언된 변수는 메모리에 지정된 type에 맞는 크기로 배정
  - char 는 1byte, integer는 4bytes, …

Identifier (식별자)

- Identifier: 프로그래머가 임의로 만들어 사용하는 *변수, 상수, 함수 등 데이터나 객체의 이름*
- Identifier를 만들기 위한 규칙
  - 첫 번째 글자는 영문자 또는 '_'(언더스코어)여야 한다.
  - 영문자, 숫자 또는 언더스코어로만 구성되어야 한다
  - 첫 63자까지만 구별 가능 (더 이상 긴 이름을 사용할 수 없음)
  - Keyword는 사용할 수 없음 예) 등록된 키워드: int, double, if, else, 등
- Identifier의 예

Valid Names		Invalid Names	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	//System Defined Value	int	// Keyword

Data Types (자료형)



Data Types (자료형)

자료형		크기 (byte)	수의 표현 범위
char	char	1	$-2^7 \sim 2^7 - 1$ (-128 ~ 127)
	signed char		
	unsigned char		0 ~ $2^8 - 1$ (0~255)
int	short int	2	$-2^{15} \sim 2^{15}-1$(-32,768 ~ 32,767)
	unsigned short int		0 ~ 2^{16} (0 ~ 65,535)
	int	4	$-2^{31} \sim 2^{31}-1$ (-2,147,483,648 ~ 2,147,483,647)
	unsigned int		0 ~ $2^{32}-1$ (0 ~ 4,294,967,295)
long	long int	4	$-2^{31} \sim 2^{31} - 1$ (-2,147,483,648 ~ 2,147,483,647)
	unsigned long int		0 ~ $2^{32} - 1$ (0 ~ 4,294,967,295)
float	float	4	$-10^{128} \sim 10^{127}$: 소수 6자리 표현
double	double	8	$-10^{128} \sim 10^{127}$: 소수 15자리 표현
	long double	8 또는 그 이상	차이를 많이 보임 : double의 정밀도와 같거나 크다.

자료형의 크기는 컴퓨터에 따라 다름. 위 테이블은 보통 일반적인 인텔 프로세서 PC의 사양임
*** 컴퓨터의 종류마다 크기 사양이 다르므로 반드시 매뉴얼을 참조할 것!**

자료형: Character(문자형)

- **1byte로 저장되며 하나의 문자를 저장한다.**
- **저장할 수 있는 문자는 alphabet문자에 한정된 것이 아니라 숫자와 기타기호도 포함한다.**
- **컴퓨터는 0과 1만을 다루고 저장할 수 있으며 문자 자체를 저장할 수는 없기 때문에 각 문자에 해당하는 ASCII(American Standard Code for Information Interchange) code를 이용하여 저장한다.**

  예) 'A' => (65_{10}) = 01000001_2 'B' => (66_{10}) = 01000010_2

  'a' => (97_{10}) = 01100001_2 'b' => (98_{10}) = 01100010_2

  '0' => (48_{10}) = 00110000_2 '1' => (49_{10}) = 00110001_2

  '&' => (38_{10}) = 00100110_2 '*' => (42_{10}) = 00101010_2

- **char형은 0~255의 값을 갖는 작은 int형으로 해석될 수 있다. 그래서 C에서는 char형을 자주 int형과 같이 취급한다. (integral type variable)**

  예) char형 변수는 내부적으로는 정수형이므로 산술연산이 가능하다.

  - 다음의 경우 문자 'a'의 다음 문자인 'b'가 출력 된다.

```
char  ch = 'a';                 // ch = 97
printf("%c", ch+1);             // ch+1= 98 => 'b'
```

자료형: Boolean, int, float

- Boolean Type(부울형)
  - Boolean Type은 true와 false 오직 2가지 값으로 나타냄
  - Boolean Type이 규정되기 전에도 integer값을 이용해 표현
    - True ➔ 0 이외의 수 (positive or negative)
    - False ➔ 0 (zero)
- Integer Type(정수형)
  - int 형은 정수를 저장
  - short int, int, long int, long long int 로 구분됨
  - 각각의 integer size 들은 signed 혹은 unsigned 될 수 있음
- Real Type(실수형)
  - float 형은 실수를 저장
  - real type 은 float, double, long double로만 구분됨

Reserved words (예약어 또는 키워드)

- C언어에서 특별한 의미로 사용되는 단어
- 프로그램 내에서 재정의 되거나 다른 용도로 사용되어서는 안됨

auto	default	float	register	switch
break	do	for	return	typedef
cast	double	goto	signed	union
char	else	if	sizeof	unsigned
const	enum	int	static	void
continue	extern	long	struct	volatile
while				

Variables(변수) And Constants(상수)

- Variables(변수)
  - 값을 저장할 수 있는 기억장소에 이름을 붙인 것으로 선언된 후 쓰여야 함
  - 변수를 선언할 때는 변수의 type과 변수의 이름을 나란히 써줌
  - 변수의 선언은 함수의 첫 부분에서만 가능

```
bool      fact;
short     maxItems;        // Word separator: Capital
long      national_debt;   // Word separator: underscore
float     payrate;         // Word separator: Capital
double    tax;
char      code, kind;      // Poor style-see text
int       a, b;            // Poor style-see text
```

- variable initialization(변수의 초기화)
  - 변수를 초기화하지 않으면 어떤 값이 들어있는지 알 수 없음
    초기화 이전에 저장되어 있는 값을 쓰레기 값(garbage)이라 함

- Constants(상수)
  - 프로그램이 실행되는 동안에 변하지 않는 데이터 값

```
const float pi = 3.14159  // const 변수는 반드시 선언시 초기화 해야 함
```

ASCII 코드

- ASCII 코드에서 문자 상수와 대응되는 정수 값 (No particular rule)

문자 상수	'a'	'b'	'c'	...	'z'
대응하는 값	97	98	99	...	122
문자 상수	'A'	'B'	'C'	...	'Z'
대응하는 값	65	66	67	...	90
문자 상수	'0'	'1'	'2'	...	'9'
대응하는 값	48	49	50	...	57
문자 상수	'&'	'*'	'+'		
대응하는 값	38	42	43		

ASCII 코드

- 주의
  - 숫자를 표현하는 문자 상수의 코드 값과 실제 그 숫자 값은 특별한 관계가 없음
    - 즉, '2'의 값은 2가 아님!!
  - 'a', 'b', 'c' 등의 코드 값은 순서적임 (remember this)
    - 이것은 문자와 단어를 사전적 순서로 정렬할 때 편리함

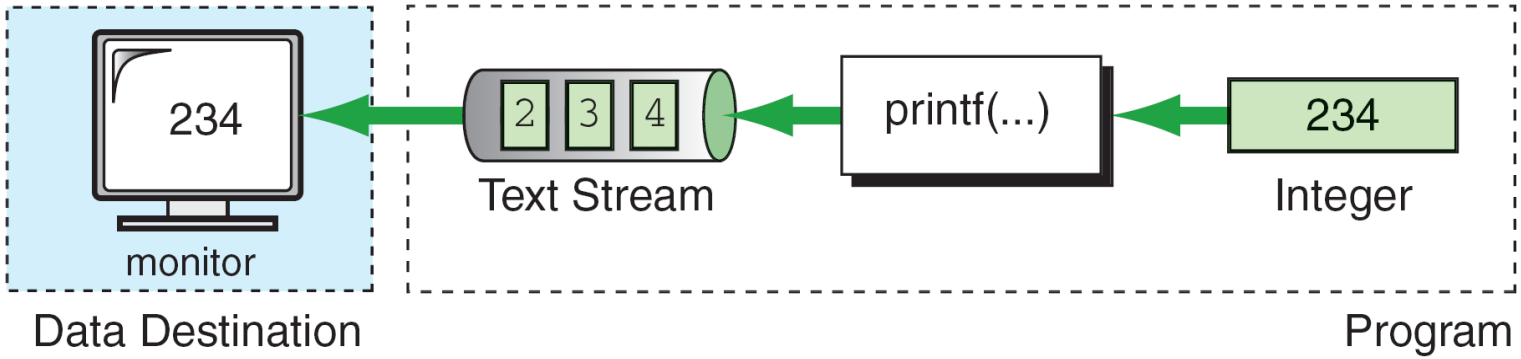
특수문자 표현을 위한 탈출 기법(escape character)

- **인쇄할 수 없는 특수문자는 탈출 기법을 사용하여 표현**
  - **예를 들어, 수평 탭(tab) 문자는 문자 상수와 문자열에서 '₩t' (backslash with t , 즉 ' \ t') 로 표현됨**
  - **'₩t'가 '₩'와 't' 두 문자로 기술되지만, 이것은 한 문자임**
  - **'₩' is called "escape character". (backslash)**
    - Backslash: 영문 키보드에는 backslash(' \ ') 기호로 되어 있지만, 한글키보드에는 Korean currency('₩') 기호로 대체되어 있어서 보이는 것과는 전혀 다른 이름으로 부름

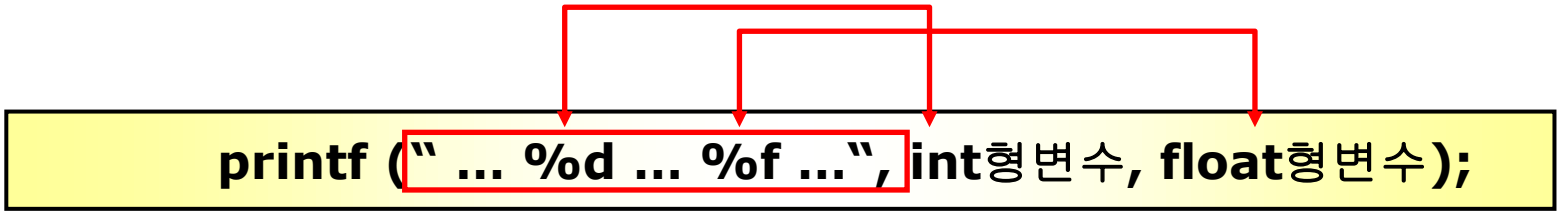
- **프로그램 내에서 특별한 의미를 갖는 문자들이 본래의 의미를 갖기 위해서도 탈출 기법을 사용해야 함**

출력함수(printf)

- C에서 출력을 하는 방법은 여러가지가 있지만 가장 쉽게 쓸 수 있는 것은 printf이다.

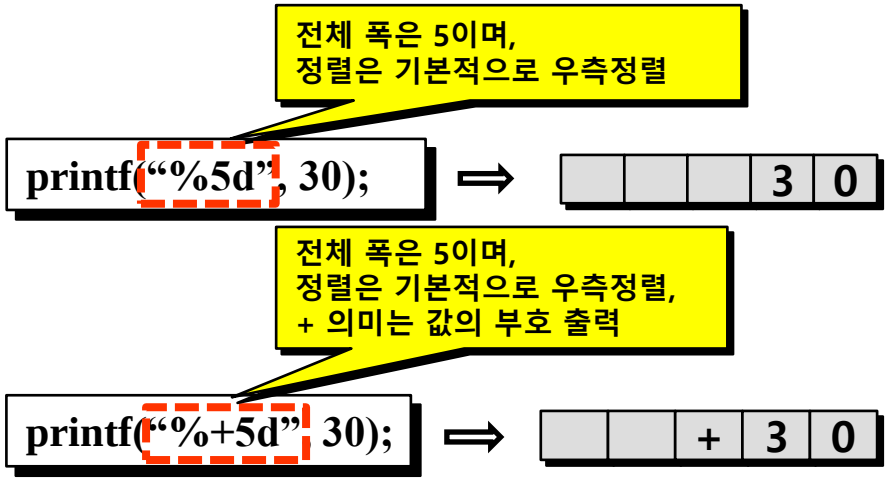


- " " 안에 출력할 내용을 적는다.
- 변수의 값을 출력할 때는 " " 안의 해당 위치에 형식문자(%d, %c, etc..)를 적고, " " 뒤에 해당하는 변수를 순서대로 적는다.



출력함수(printf)

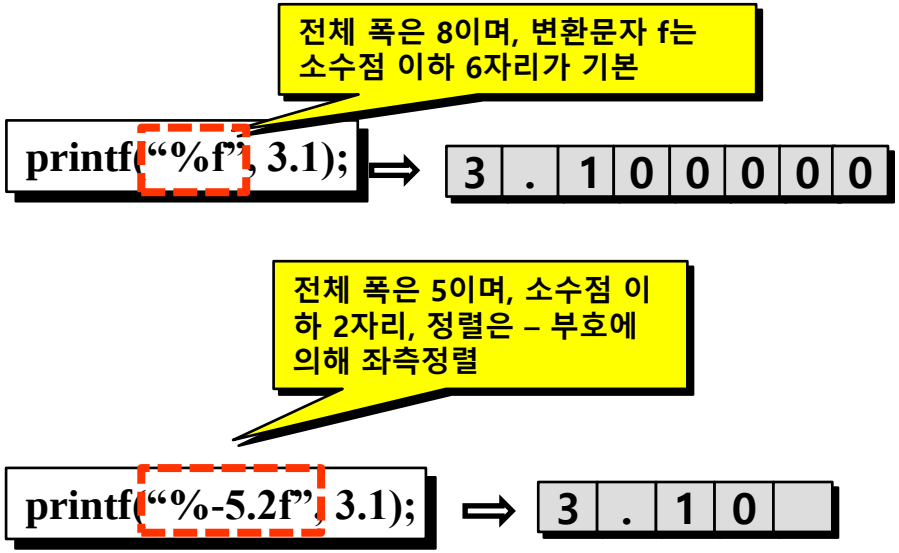
- int형 변수 값의 출력 형식
  - %d 사이에 폭을 나타내는 정수를 기술
  - 필드 폭을 지정하면 자동적으로 우측정렬이 된다. 좌측정렬을 위해서는 필드 폭 앞에 '-' 를 붙임



- float형 변수값의 출력 형식
  - 필드 폭을 지정하려면 %f 사이에 폭을 기술
  - %10f 라고 기술하면

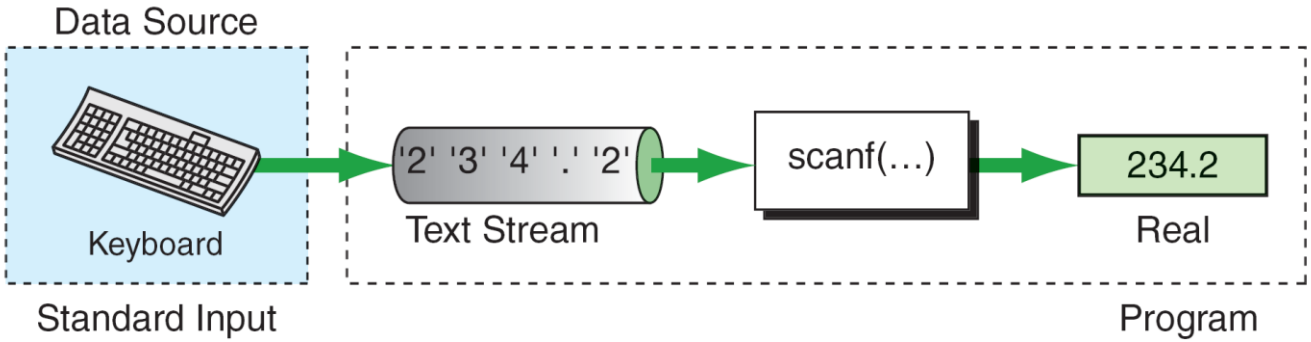
    전체 폭은 10, 소수점 이하 자리수 폭은 기본 6으로 지정
  - %10.5f 이면

    전체 폭은 10, 5는 소수점 이하 자리수의 폭을 의미
  - 지정한 전체 폭이 출력 값의 전체 폭보다 작으면, 지정된 작은 폭은 무시하고 원래의 출력 값의 폭으로 출력



출력함수(scanf)

- C에서 입력을 받는 방법 중가장 쉽게 쓸 수 있는 것은 scanf()이다.



  - scanf()는 표준 입력으로부터 여러 종류의 자료 값을 입력 받을 수 있음
  - 사용법은 printf()와 유사

  - 단, 입력 값을 저장할 변수의 이름 앞에 '&'를 반드시 붙여줘야 한다.
    (&은 해당 변수의 주소를 의미하는 연산자)

```
scanf("%d %f", &age, &weight);
```

입력함수(scanf) 실습

```
#include<stdio.h>

void main()
{
 int a;
 char b;

 scanf("%d %c",&a,&b);

 printf("입력(숫자)값:%d 입력(문자)값:%c",a,b);

}
```

- 1 k 를 입력한 후 엔터를 치면 변수 a에는 1이, 변수b에는 'k'의 값이 대입된다.
- printf 함수에 의해 변수a, b의 값이 화면에 출력되게 된다.

<결과>

```
1 k
입력(숫자)값:1 입력(문자)값:k
```

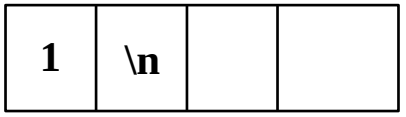
scanf사용시 주의할 점: 실습

```
#include<stdio.h>

int main(void)
{
   int code;
   char ch;
   printf("숫자를 입력하세요 ");
   scanf("%d", &code);
   printf("문자를입력하세요 ");
   scanf("%c", &ch);

   printf("결과:%d %c",code,ch);
   return 0;
}
```

결과

```
숫자를 입력하세요 1
문자를입력하세요 결과:1
_
```

- 첫 번째 scanf에서 1을 입력 후 enter를 치면 아래와 같이 버퍼상에 임시저장 됨, 1과 함께 enter값인 "\n"도 버퍼에 저장 됨
  * enter 값인 ₩n도 하나의 특수 문자로 취급됨.



버퍼

- 두 번째 scanf문은 버퍼의 enter값을 변수 ch에 받게 된다.

- 1은 int형 변수 code에 저장이 되고 버퍼에서 삭제된다.
- 두 번째 scanf의 %c는 버퍼에서 하나의 값만 읽음

버퍼

- %c는 버퍼에서 \n를 받고 scanf명령을 종료

- ch값에 enter값(\n)이 들어 있기 때문에 printf 출력시 enter(\n) 값이 출력되어 줄을 넘기게 된다.

scanf사용시 주의할 점-해결 방법 실습

```
#include<stdio.h>

int main(void)
{
   int code;
   char ch;
   printf("숫자를 입력하세요 ");
   scanf("%d", &code);
   getchar();
   printf("문자를입력하세요 ");
   scanf("%c", &ch);

   printf("결과:%d %c",code,ch);
   return 0;
}
```

결과

```
숫자를 입력하세요 1
문자를입력하세요 a
결과:1 a
```

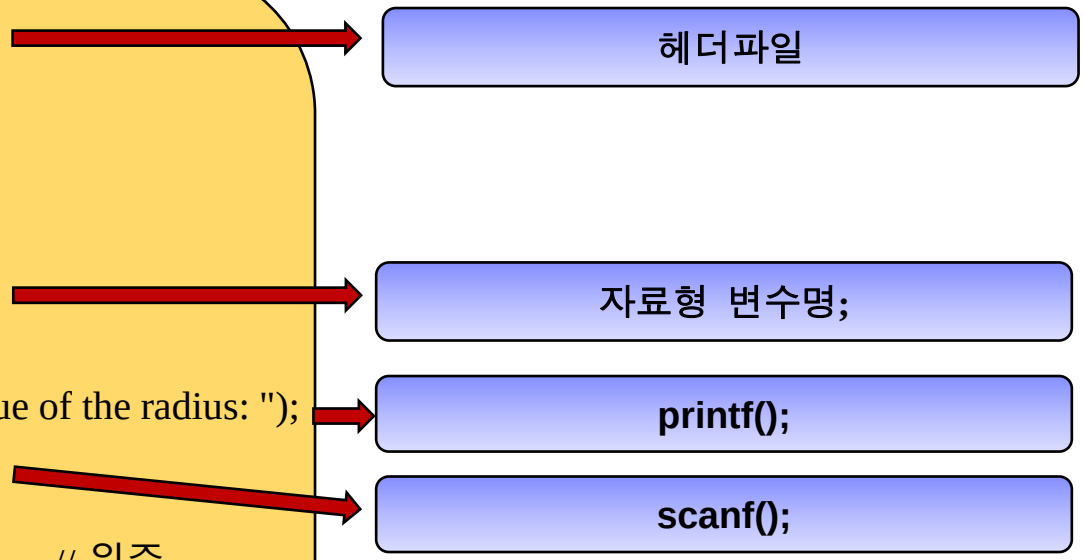
- 버퍼에 있던 enter값을 getchar()를 이용하여 비운다.

※getchar() : 버퍼에 있는 문자를 꺼내 읽어내고 꺼낸 문자를 버퍼에서 지움

- 두 번째 scanf는 버퍼에서 받아 올 값이 없으므로 다시 새로 값을 받는다.

Calculate a circle's area and circumference

```
#include<stdio.h>


int main(void)
{
    float circ, area, radius;

    printf("\nPlease enter the value of the radius: ");
    scanf("%f", &radius);

    circ = 2  * 3.14 * radius;          // 원주
    area = 3.14 * radius * radius;      // 넓이

    printf("\nRadius is :          %10.2f", radius);
    printf("\nCircumference is : %10.2f", circ);
    printf("\nArea is :            %10.2f\n", area);

    return 0;
}
```



Calculate a circle's area and circumference

```
#include<stdio.h>


int main(void)
{
    float circ, area, radius;

    printf("\nPlease enter the value of the radius: ");
    scanf("%f", &radius);

    circ = 2  * 3.14 * radius;          // 원주
    area = 3.14 * radius * radius;      // 넓이

    printf("\nRadius is :            %10.2f", radius);
    printf("\nCircumference is : %10.2f", circ);
    printf("\nArea is :              %10.2f\n", area);

    return 0;
}
```

- 이 프로그램은 입력 받은 값을 반지름으로 하는 원의 원주와 넓이를 구하는 프로그램이다.
- 각각의 값은 전체 폭 10자리, 그 중 소수점 아래 2자리로 맞추어 출력된다.

<실행결과>

```
Please enter the value of the radius: 3

Radius is :              3.00
Circumference is :       18.85
Area is :                28.27
```

C program Execution Procedure

Step1. Writing and Edit (hello.c)

```c
#include <stdio.h>

int main(void)
{
        printf("Hello C\n");

        return 0;
}
```

gcc는 C언어의 compile 명령어로서 compile시 옵션을 주지 않을 경우 default로 a.out이라는 실행 파일이 생성 된다.

Step2 & Step3. Compiling and Linking Programs

```
gr120090364@cspro:~$ vi hello.c
gr120090364@cspro:~$ gcc hello.c
```

Step4. Execution

```
gr120090364@cspro:~$ ./a.out
Hello C
```

C program Execution Procedure

Step1. Writing and Edit (hello.c)

```
#include <stdio.h>

int main(void)
{
        printf("Hello C\n");

        return 0;
}
```

gcc에서 –o 옵션을 사용하면 실행파일의 이름을 선언할수 있다. 여기서는 실행 파일의 이름을 hello.out 으로 지정하였다.

Step2 & Step3. Compiling and Linking Programs

```
gr120090364@cspro:~$ vi hello.c
gr120090364@cspro:~$ gcc -o hello.out  hello.c
```

Step4. Execution

```
gr120090364@cspro:~$ ./hello.out
Hello C
```

Example Program

```c
#include <stdio.h>

int main(void)
{
        char ch;
        int integer;
        float real;

        printf("Input one character, one integer, ont real number : ");
        scanf("%c %d %f", &ch, &integer, &real);
        printf("%c %d %f\n", ch, integer, real);

        return 0;
}
```

ch, integer, real 는 identifier 이고
ch는 문자형 변수, integer는 정수형 변수, real는 실수형 변수이다.

scanf() 함수를 통해 값을 입력 받는다.
입력받을 때는 변수이름 앞에 &를 붙여준다.

```
gr120090364@cspro:~$ vi example-1.c
gr120090364@cspro:~$ gcc -o example-1.out example-1.c
gr120090364@cspro:~$ ./example-1.out
Input one character, one integer, ont real number : d 7 4.23
d 7 4.230000
gr120090364@cspro:~$
```

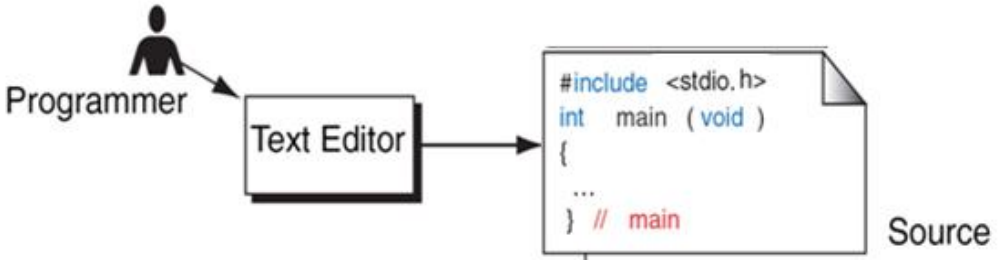
printf() 함수를 통해 값을 출력한다.
값을 입력받고 출력할 때 type에 따라 다른 형식문자(%c, %d, %f)를 사용한다.

하나의 scanf () 함수에서 여러 개의 입력 을 받을 때 각각의 입력 값은 스페이스로 구분을 하여 입력한다.

C program Execution Procedure

C Program 실행 과정

1) Edit (Write Code)
2) Compile
3) Link & Load
4) Execute

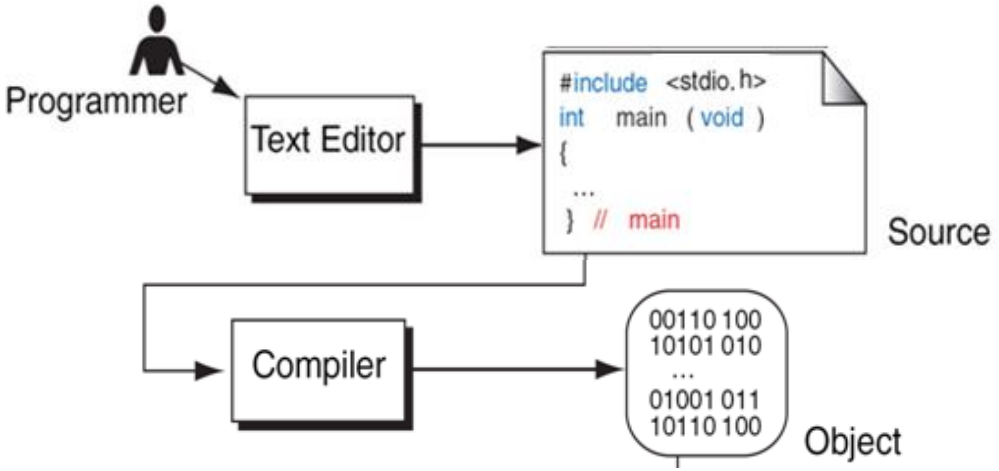


▫ 텍스트 에디터(Notepad 또는 vi)를 사용하여 프로그램 소스코드를 작성해서 저장

C program Execution Procedure

C Program 실행 과정

1) **Edit (Write Code)**
2) **Compile**
3) **Link & Load**
4) **Execute**

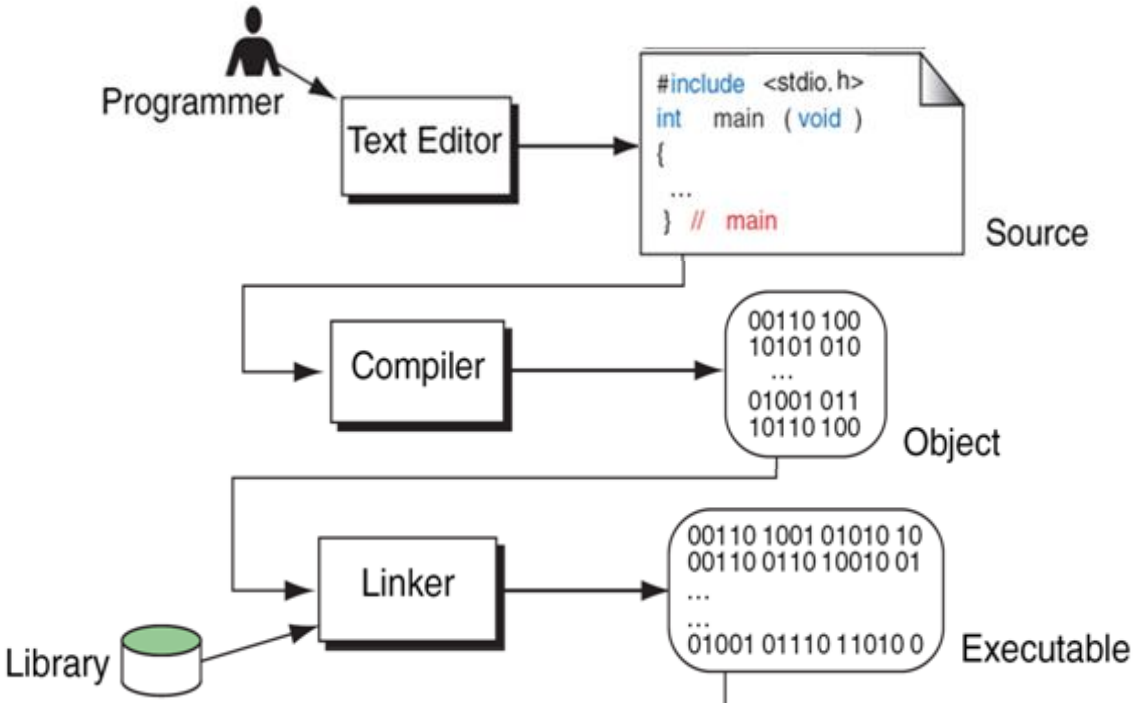


- 디스크에 저장된 소스파일은 기계어로 번역되어야 하는데 이 과정을 컴파일링 이라 부르며 C 컴파일러는 preprocessor 와 translator 두 가지 프로그램으로 나뉨
- Preprocessor : 기계어 번역을 위한 준비 수행
- Translator : 준비된 프로그램을 기계어로 번역하여 object 모듈(실행 가능한 완전한 프로그램은 아닌 기계어 모듈)을 만듬

C program Execution Procedure

C Program 실행 과정

1) **Edit (Write Code)**
2) **Compile**
3) **Link & Load**
4) **Execute**

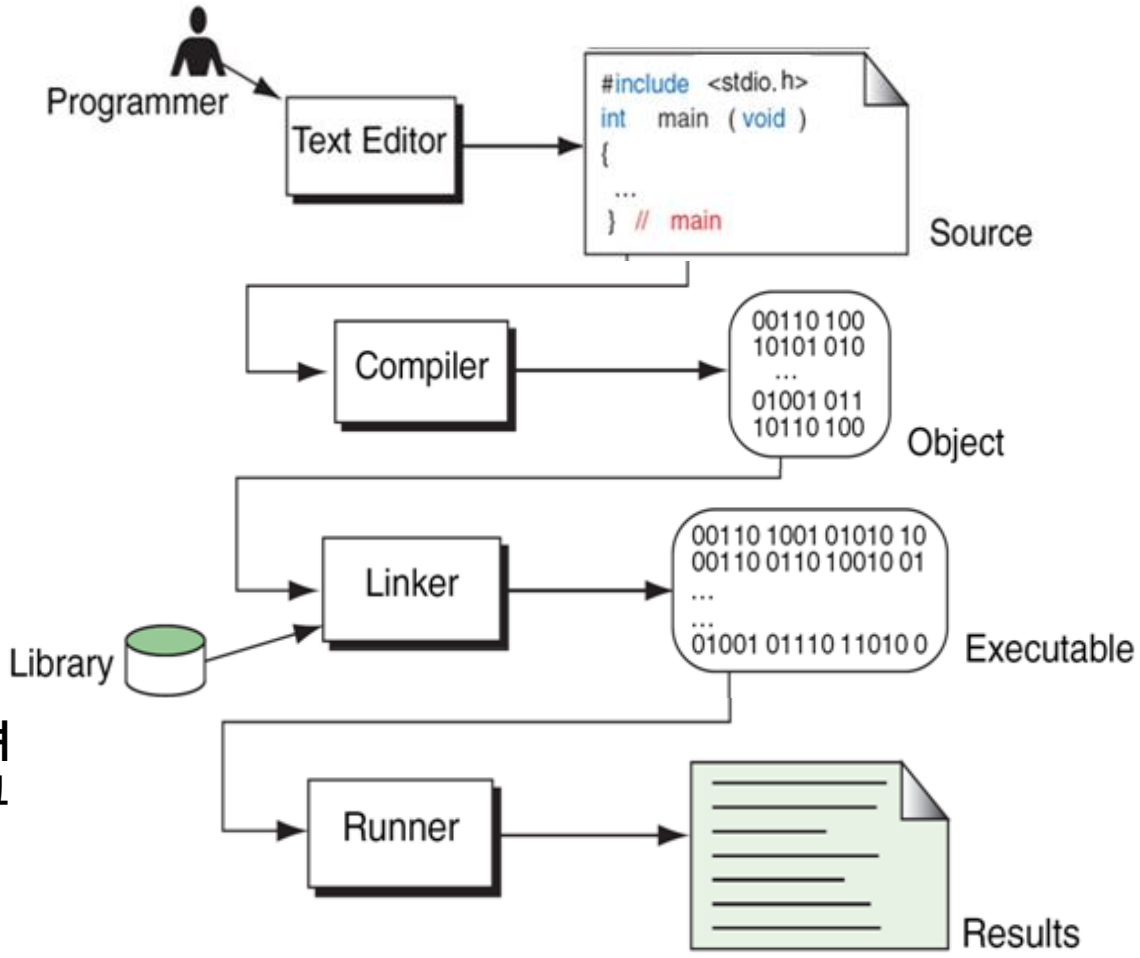


- **Linker는 번역된 object 모듈들과 library 함수들을 연결하여 실행 가능한 프로그램 (executable code)을 만듬**

C program Execution Procedure

C Program 실행 과정

1) **Edit (Write Code)**
2) **Compile**
3) **Link & Load**
4) **Execute**



- 보조기억장치에 저장 executable code를 실행하기 위해서는 운영체제 명령어가 필요 (ex. run)
- Loader라는 시스템 프로그램은 executable code를 메모리로 올려준다. 다 올라가면 OS가 그 프로그램의 실행을 시작 한다.
- 시작된 프로그램은 결과를 낸다.

정리

- **프로그램 내에서 다양한 Comment 작성 방법과 규칙을 잘 알고 있는가?**
- **유틸리티 함수 (예: scanf, printf) 와 라이브러리 (예: stdio.h)의 개념**
- **Data type의 종류와 선언 방법**
  - char 데이터를 숫자처럼 쓸 수 있는가?
  - 사용자 지정 변수(식별자)를 만드는 규정
- **printf, scanf 함수의 사용법**
  - 기본적인 사용 방법을 잘 숙지하고 있는가?
  - 각 데이터 타입에 따른 입출력 형식을 지정할 수 있는가?
  - 다양한 형태로 출력 위치를 지정할 수 있는가?
  - getchar() 함수의 필요성을 이해했는가?
- **간단한 프로그램을 만들고 수행시킬 수 있는가?**
- **C-program execution procedure의 기본적인 개념을 이해할 것**