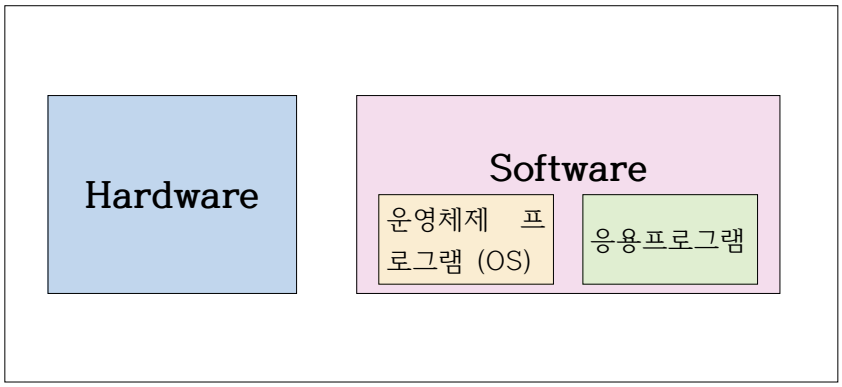
프로그래밍의 개념

1. Computer

●구성



●기능 : 명령어 list에 따라 데이터를 처리하는 기계

2. Program

●Computer에게 무엇을 어떻게 시킬지를 지시하는 명령(instruction)을 기록해 놓은 문서(List)

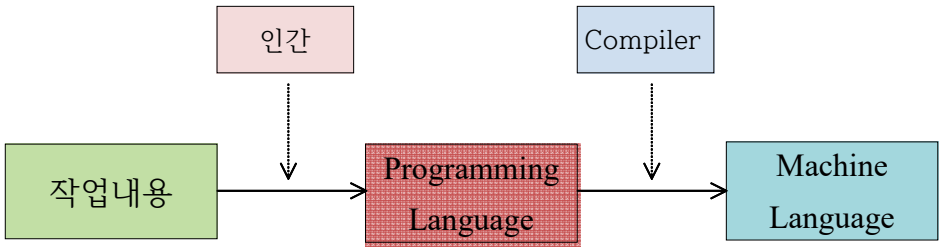
●동일 하드웨어 상에서 프로그램에 따라 다양한 작업이 가능(다양한 프로그램)

●역할 : 인간이 하고자 하는 작업을 컴퓨터에게 전달

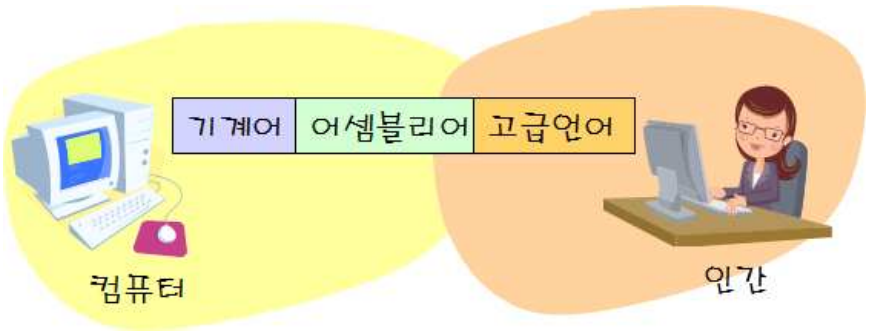
3. Programming Language

●Computer가 이해 할 수 있는 언어 ⇒ 기계어 (2진수)

●Programming Language : 인간과 기계어를 연결시켜주는 중간 언어



4. Programming Language 종류



●Machine Language : 컴퓨터가 이해하는 유일한 언어 (2진수 언어)

●Assembly Language : 2진수보다는 좀 더 인간 언어에 접근한 언어 (CPU에 의존)

●High-Level Language : 인간 언어에 거의 유사하게 접근한 언어(CPU에 무관)

Language	특 징
FORTRAN	1950년대에 가장 먼저 만들어진 언어. 수치계산과 과학기술 계산에 적합
COBOL	1959년에 만들어진 비즈니스 사무처리 언어. 구어체 문장형태로 설계
PASCAL	1970년대에 만들어진 교육용 언어
C	1970년대 초반 UNIX 운영체제(OS)를 위해 만들어진 언어
C++	1983년 C언어에 클래스개념을 도입한 객체 지향적 언어
Java	1995년에 만들어진 인터넷 웹페이지에서 실행이 가능한 언어

** C# , Python 등 기타 다양한 언어가 있음

5. C Language

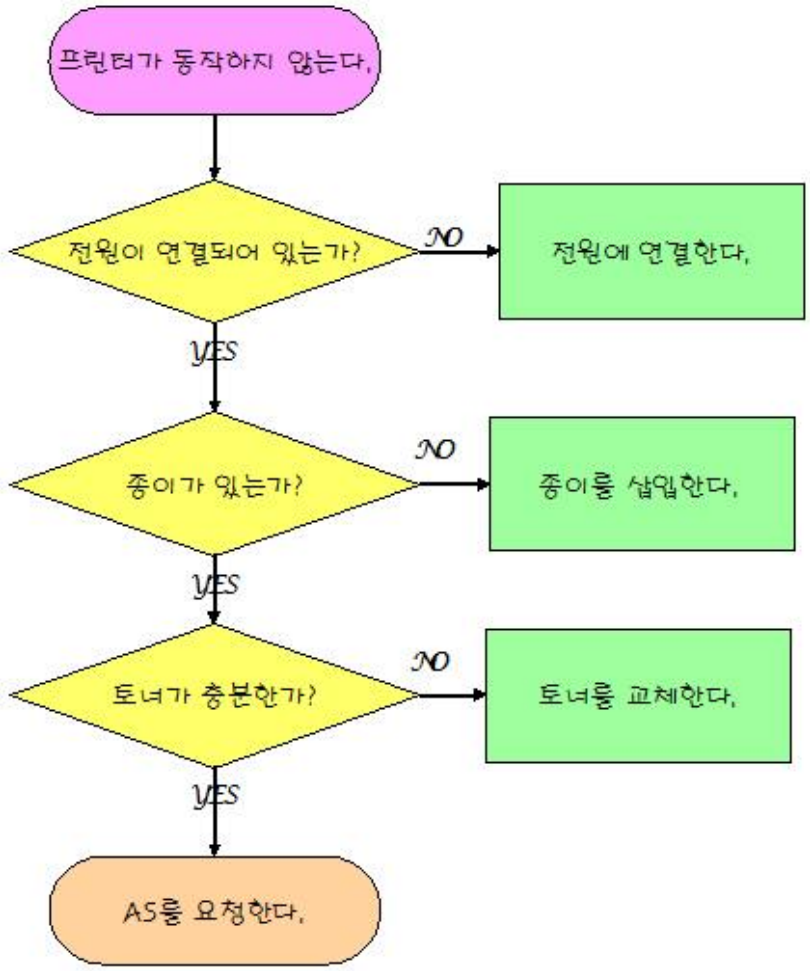
특 징	내 용
간결성	필요한 기능만 들어있고 표기법이 간결
효율성	작성된 프로그램의 크기가 작고 실행속도가 빠르며 메모리를 효과적으로 사용 가능 (Microprocessor, 자동화에 적합)
저급 프로그램 동시가능	저급 프로그램 : 하드웨어 직접제어 가능(스마트폰, 가전기기의 **embedded** 프로그램은 대부분 C 로 작성)
이식성이 좋음	다른 CPU를 가진 하드웨어에도 쉽게 이식가능
초보자에게 난해	교육용이 아니고 산업 현장에 적용되는 언어 (OS 프로그램도 가능)

6. Algorithms

●정의 : 문제를 해결하는 절차/방법

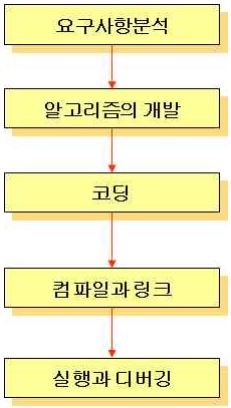
●Program Language에 무관 : 동일 알고리즘을 C나 Java 등 여러 언어로 구현가능

●Flow Chart : 알고리즘을 구현하기 위한 논리순서 또는 작업순서를 그림으로 표현한 것



<프린터 고장을 처리하는 알고리즘 Flow Chart>

7. Programming 순서

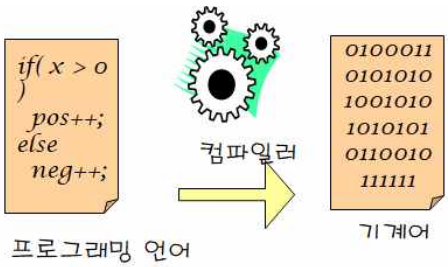


●요구사항 분석 : 프로그래머가 사용자의 요구사항을 정확하게 파악하는 과정

●알고리즘 개발 : 요구사항을 해결하기 위한 방법/절차 설계

●Coding (source code 작성) : program language를 사용하여 알고리즘을 구현. Editor를 이용하여 작업지시 문서를 형성하고 텍스트 파일 형태로 저장 (C 언어의 경우 000.c 형태)

●Compile : Source Code를 기계어로 변환. 변환 전에 source code의 문장들이 program language 문법에 맞는지를 체크하여 오류가 있으면 programmer에게 오류를 보고하고 컴파일 종료. 오류가 없으면 기계어로 변환하여 object file (000.obj) 생성.



●Link : object file과 연관된 라이브러리 파일들을 연결하여 executable file (실행파일, 000.exe) 생성.

Note Library(라이브러리) : 프로그래머들이 많이 사용하는 기능을 미리 작성하여 컴파일러에 내장

●실행 및 Debugging : 실행파일을 실행시켜 결과를 도출하여 원하는 문제해결 결과에

부합하는지를 평가. 실행도중 오류가 발생하거나 잘못된 결과가 나오면 다시 coding 단계로 돌아가 수정.

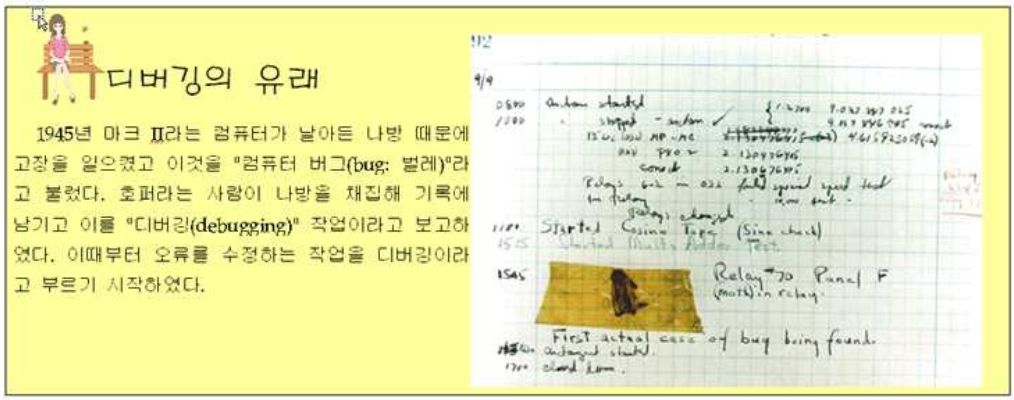
8. Debugger

●Debugging : 프로그램의 문법적 Error나 알고리즘의 논리적 Error를 수정하는 행위

●Debugger : Debugging하기 위한 도구

●문법적 Error : 프로그램 작성과정에서 언의의 문법에 맞지 않는 문장이 있는 경우 발생. 컴파일 단계에서 컴파일러가 error를 보고 해 줌.

● 논리적 오류 : 알고리즘을 구현하는 프로그램의 논리적 오류. 컴파일러가 error를 찾아내지 못함.



9. 통합개발환경 (IDE: Integrated Development Environment)

● 하나의 프로그램을 작성하여 실행하기 위해서는 Editor, Compiler, 그리고 Debugger가 필요.

● IDE : 프로그램 개발에 필요한 Editor, Compiler, Execution 및 Debugging 기능이 하나의 프로그램 안에 모두 넣은 도구.

● Visual C++ : 마이크로소프트사가 제공하는 C와 C++ 언어를 이용하여 콘솔 및 윈도우 형태의 응용 프로그램을 개발할 수 있는 통합개발환경

Note Consol 프로그램 : DOS환경에서 문자 입출력이 가능한 응용 프로그램.

Chap.1 Visual C 2013

1. Visual C++

●Visual C++ : 마이크로소프트사의 C언어 용 IDE 제품

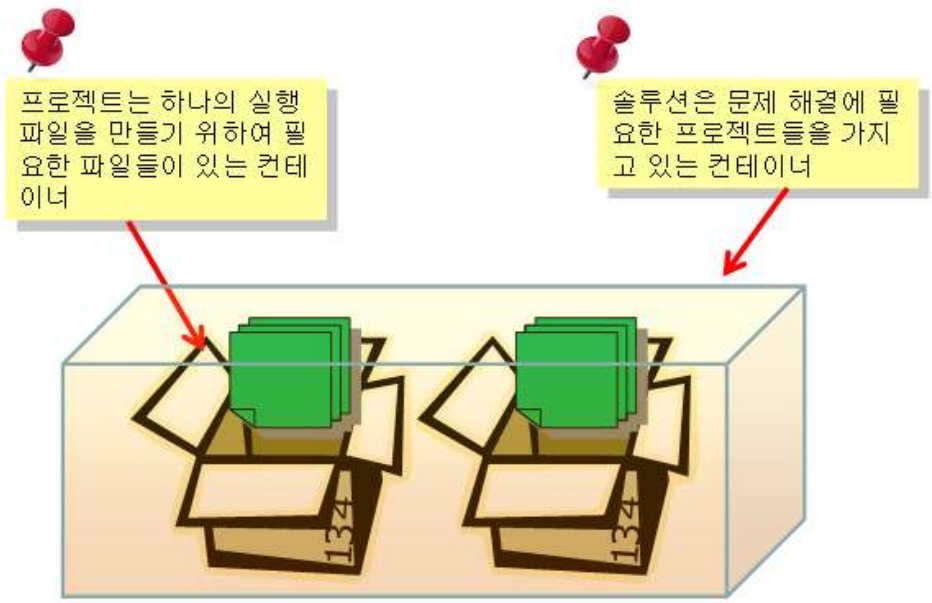
●윈도우 기반의 거의 모든 형태의 응용 프로그램 제작 가능

●사용 버전: 비주얼 C++ 2013

2. Project

●project : 하나의 실행파일을 만드는데 필요한 여러 가지 항목들이 들어 있는 컨테이너

●Solution : 문제해결에 필요한 여러 project들을 가지고 있는 컨테이너

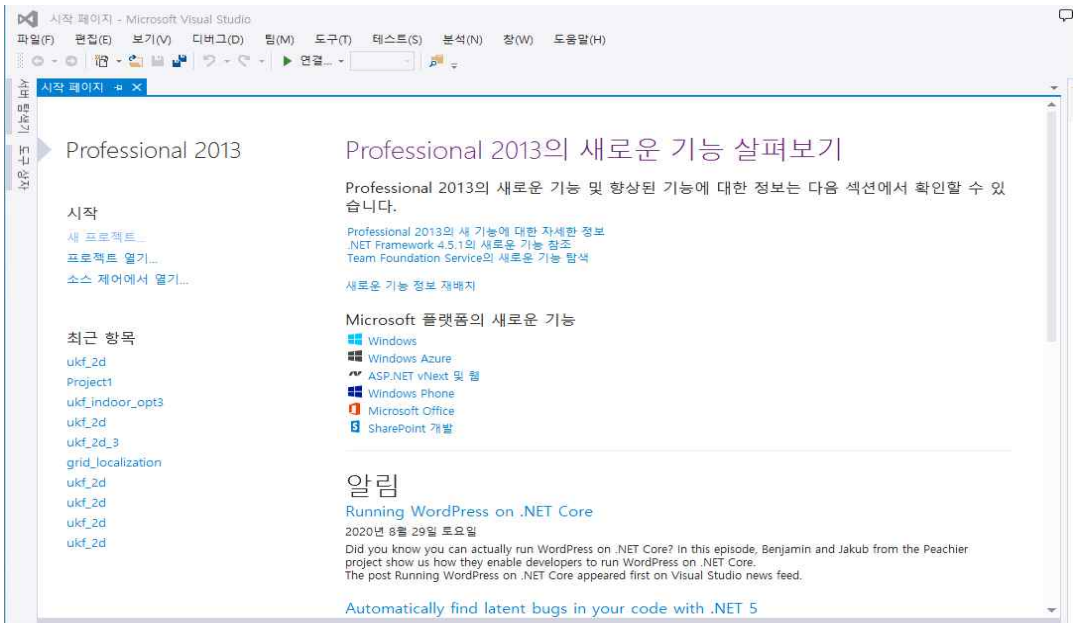


3. 초기 설정 및 프로젝트 생성

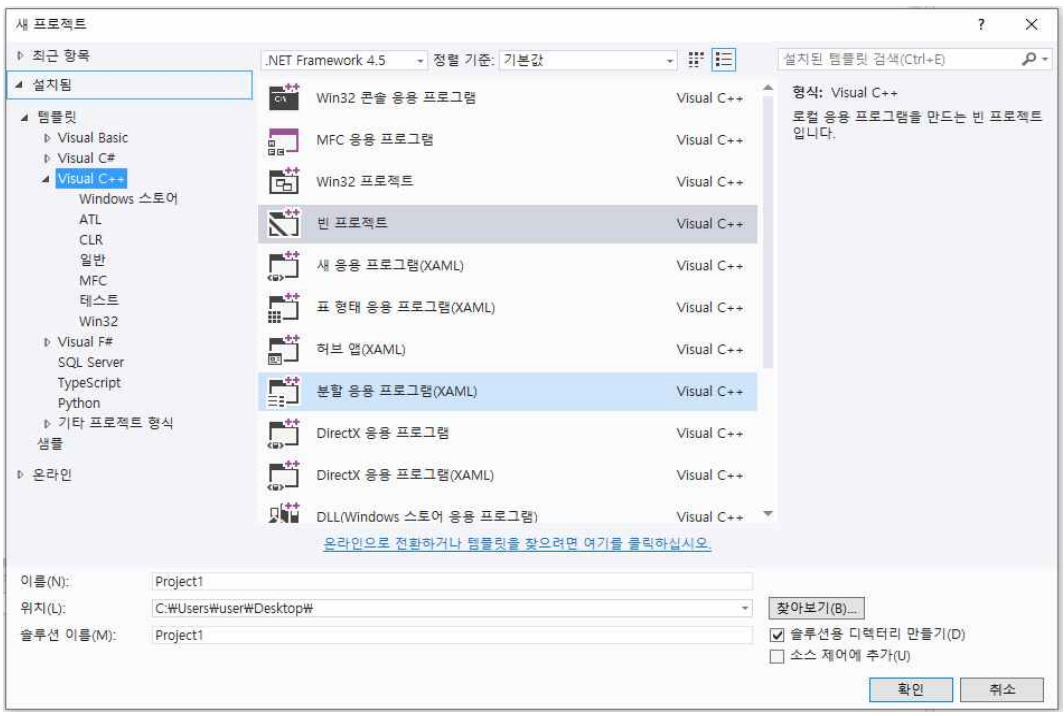
① 초기 설정 (visual studio를 설치 한 후)

- 창 색깔 선택

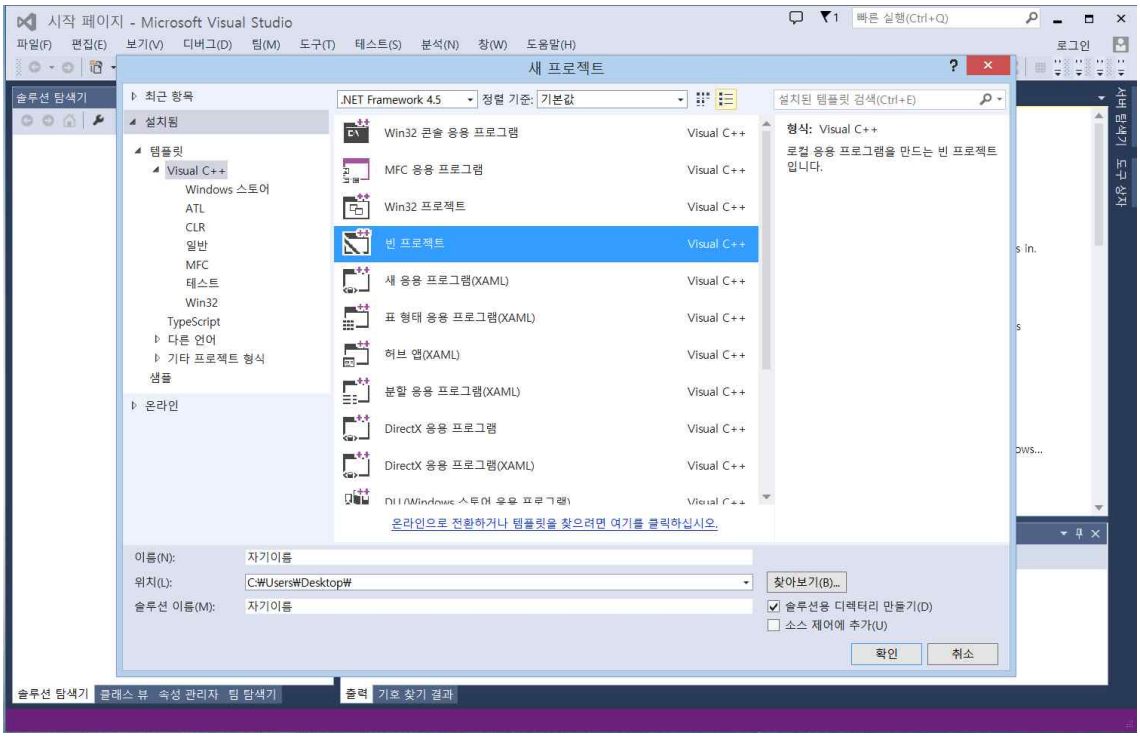
② Visual Studio 2013 실행



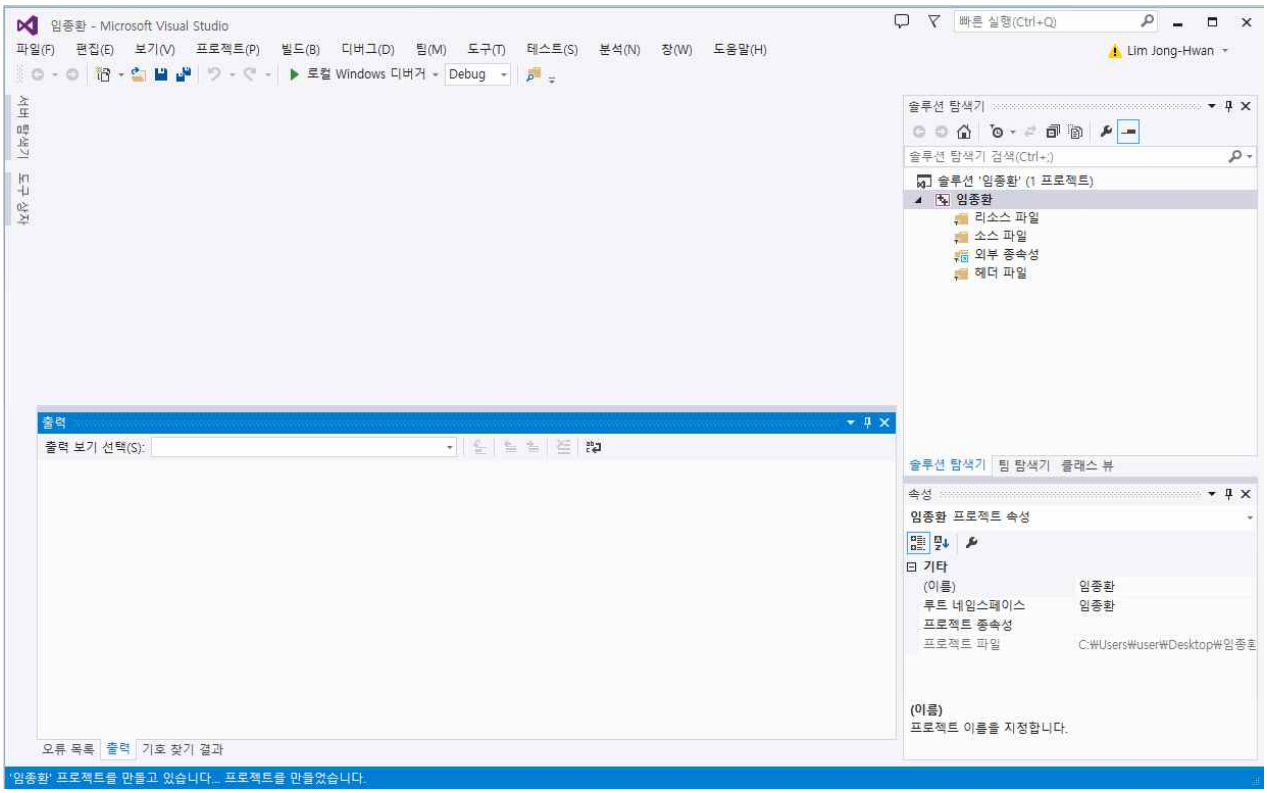
③ 새 project 만들기 : 시작 아래 '새 프로젝트...' 클릭 (또는 파일 메뉴의 '새로만들기' ⇒ '프로젝트' 선택)



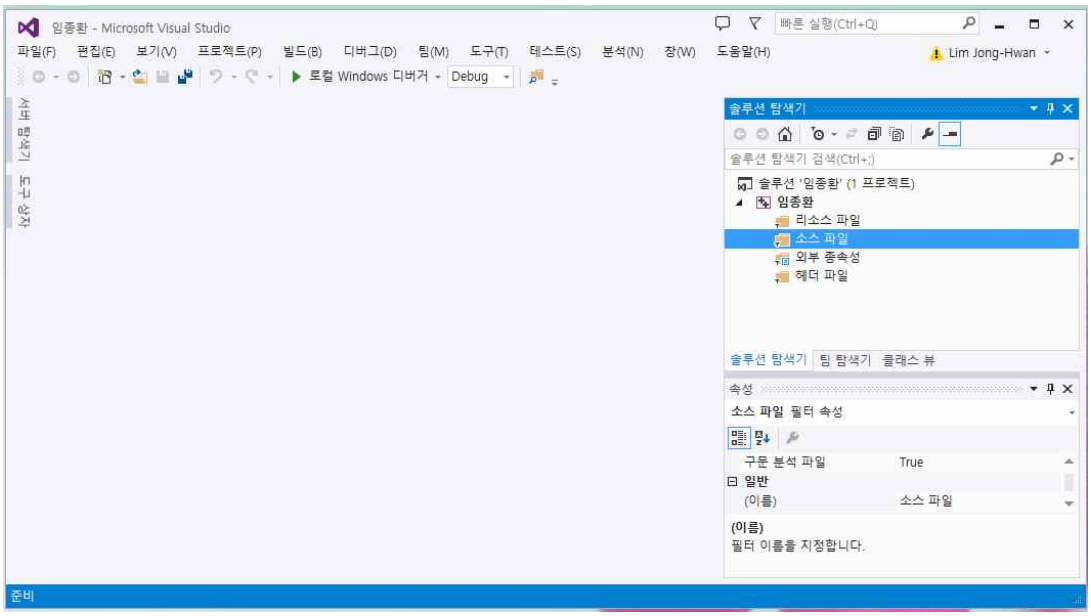
④ Project 선택 및 이름 만들기 : '빈프로젝트'를 선택하고 하단의 <이름>에 project 이름 입력 및 <위치>에 저장할 위치 지정 후 <확인> 클릭



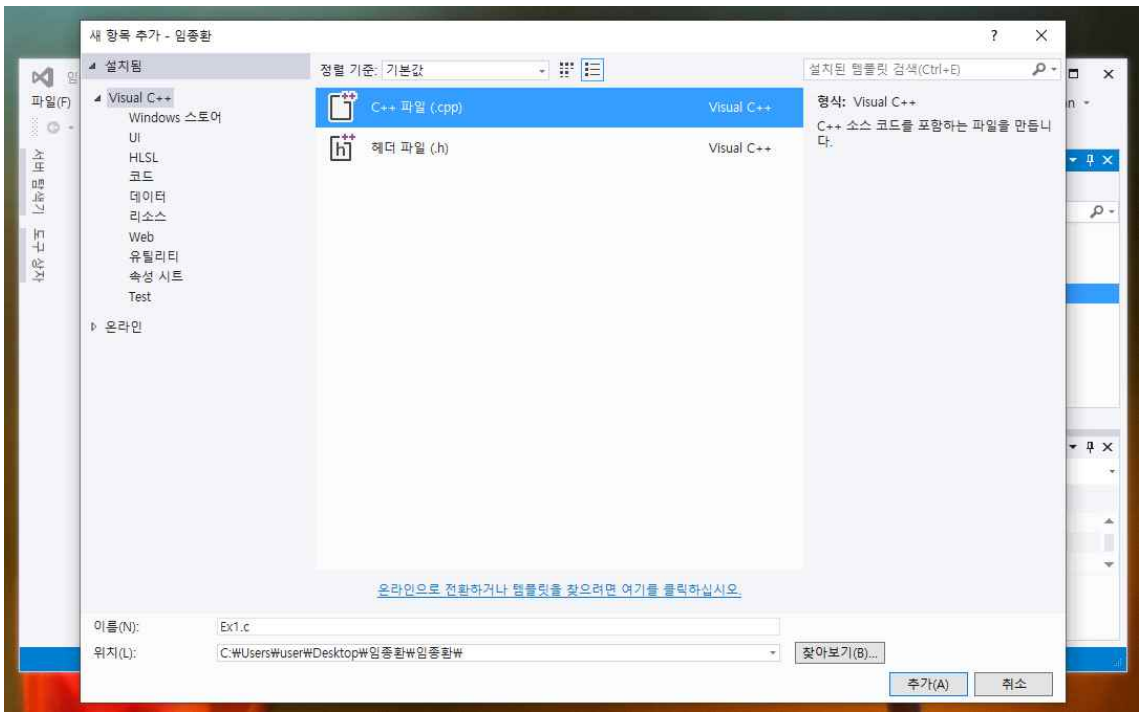
<결과화면>



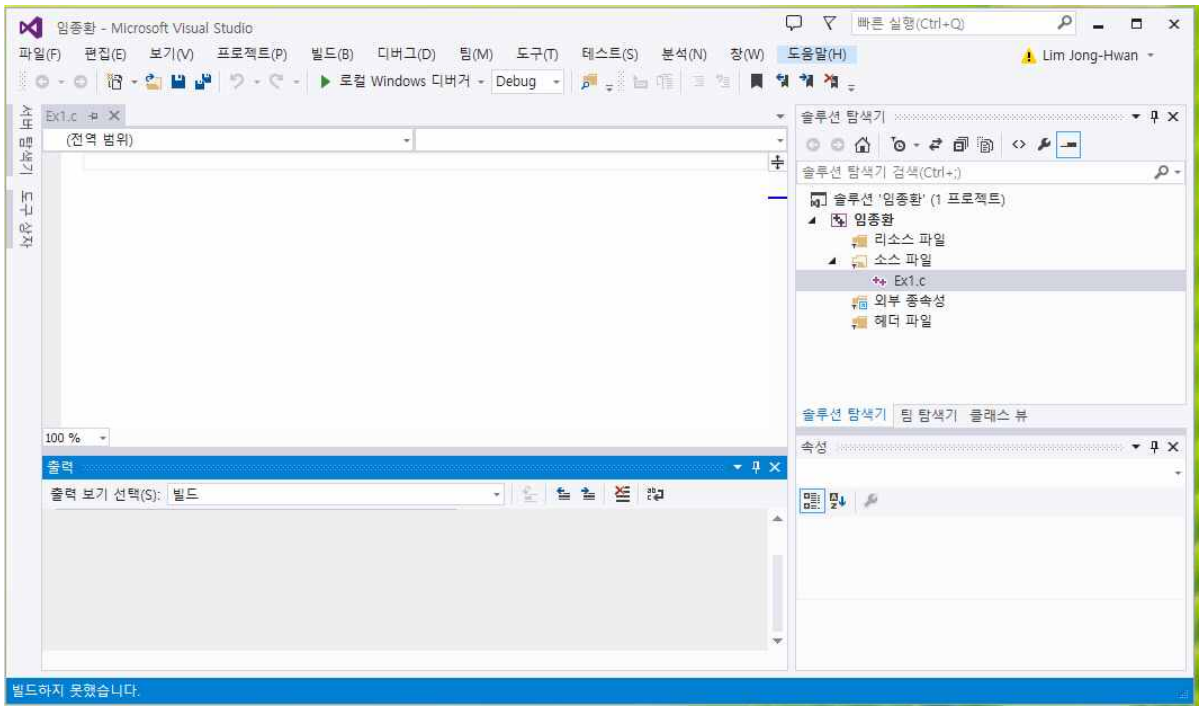
⑤ 새로운 프로그램 code 추가 : <소스파일>(마우스 오른쪽 버튼) ⇒ <추가> ⇒ <새항목> 클릭



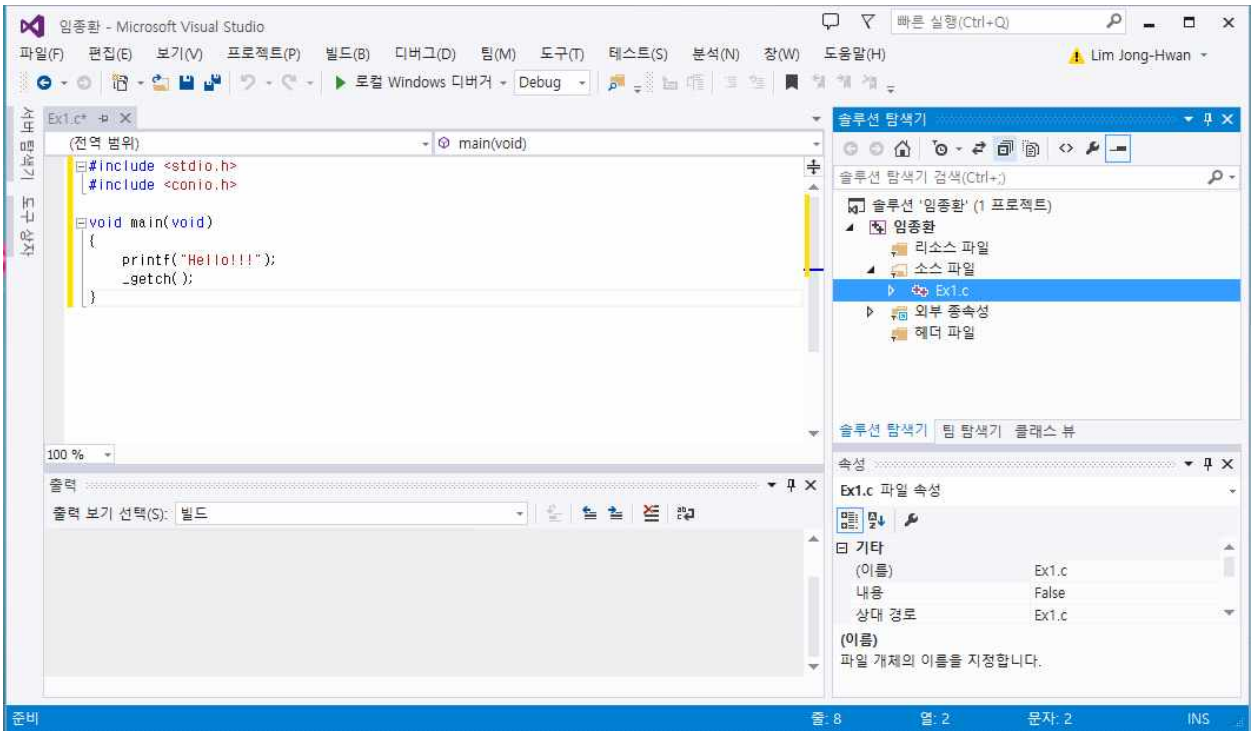
⑥ 'C++ 파일(.cpp)'을 선택하고 하단에 새 프로그램 이름(ex1.c) 및 저장위치 선택(위치는 자동으로 새로 생성된 프로젝트로 설정됨)하고 <추가>를 클릭



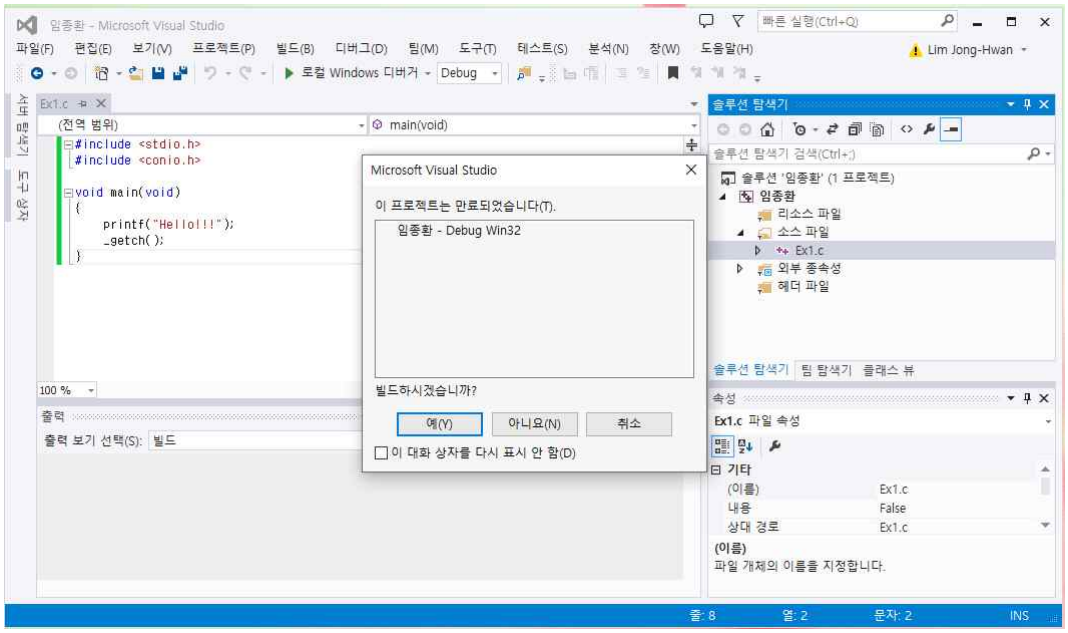
<결과화면>



⑦ 프로그램 Code 입력

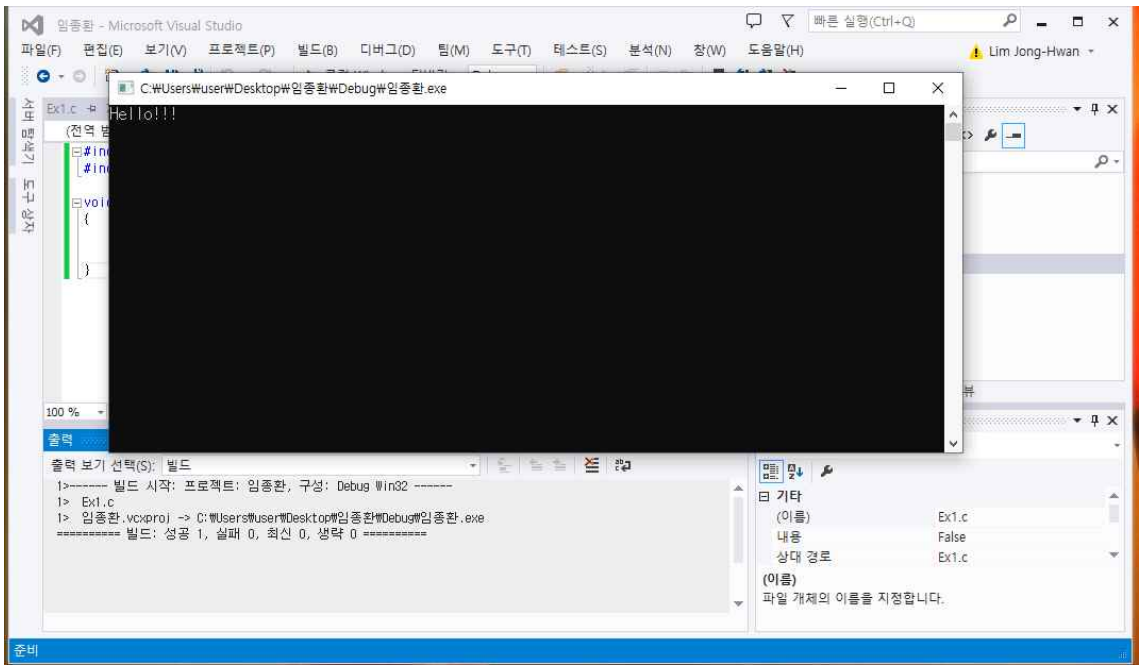


⑧ 프로그램 실행 : 상단 메뉴 <디버그>의 <디버깅하지 않고 시작> 선택



⇐ 이 화면이 나오면 '예' 클릭

⑨ 프로그램 실행화면



⇐ 아무 키나 누르면 실행화면 종료

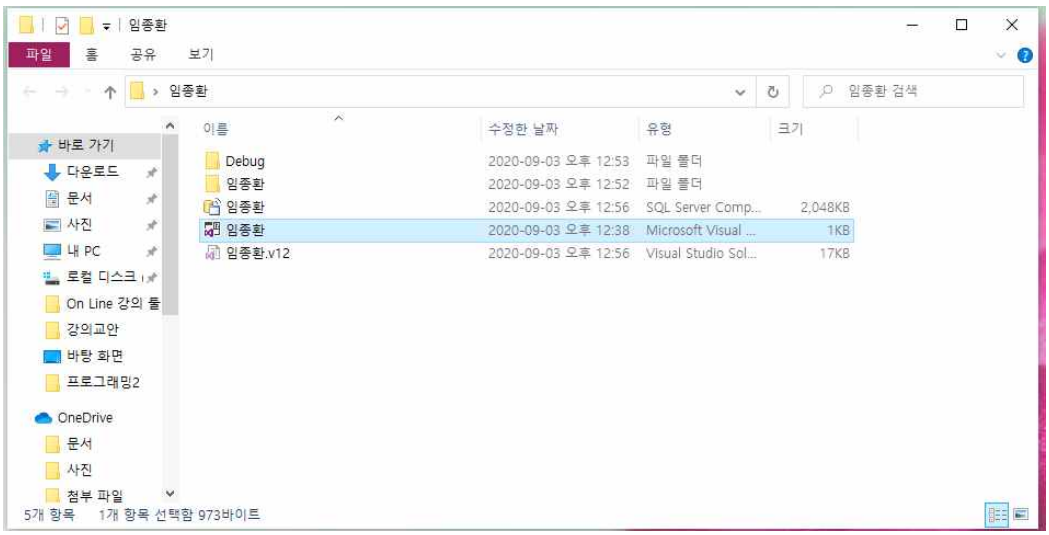
[주의] ① 실행 시 화면하단 빌드 메시지에 실패가 뜨면 실행이 안 됨 (버그 존재)

② 다음 프로그램 실행을 위해서는 반드시 이전 실행(콘솔)화면을 종료해야 함.

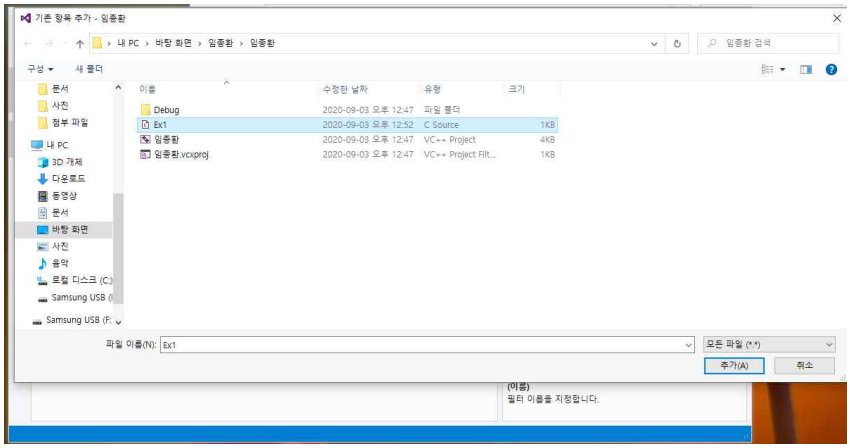
4. Project 불러오기

① 자신이 만들어 놓은 project가 저장된 경로로 이동 ⇒ 폴더 open

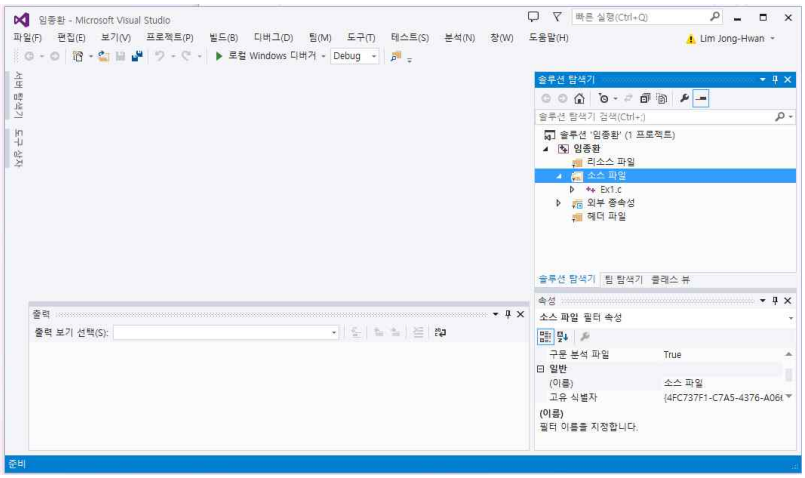
② 이전에 형성된 project를 더블 클릭 (프로젝트이름을 더블 클릭) ⇒ 이전 상태 화면 그대로 화면에 나타남



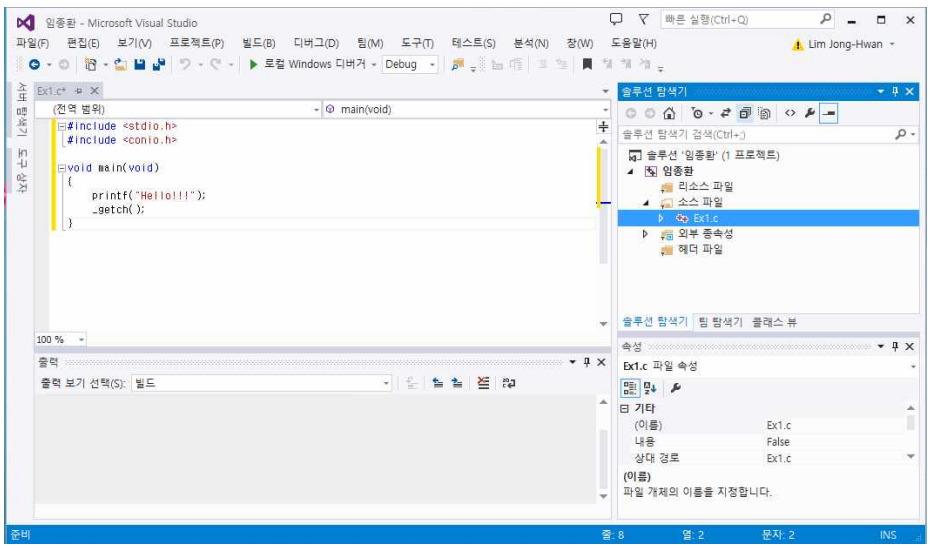
③ 프로젝트 내의 기존 프로그램 파일 프로젝트에 추가 (불러오기) : '소스파일' 선택 ⇒ 추가 ⇒ '기존항목' 선택



⇒ 불러올 프로그램 선택 후 더블 클릭 (또는 하단의 추가 클릭)



⇒ 불러온 프로그램을 열기 : 오른편 메뉴에서 추가한 파일명 더블 클릭



④ 프로젝트 내의 프로그램 제거 : 제거할 프로그램명에 마우스를 올리고 마우스 오른쪽 키 클릭 ⇒ 하부 메뉴에서 “프로젝트에서 제거” 선택

⑤ 기타 visual studio에 관한 내용은 학과 사무실에 문의 바람 (본 강의는 visual studio에 대한 것이 아니고 C언어에 관한 것임)

5. Visual 2013과 이 전 version의 다른 점

■ 보안성 강화를 위해 일부 2010에서 허용되던 library 함수가 더 이상 유효하지 않음

- getch() ⇒ _getch()
- scanf(“형식지정.. ”, &변수명, ...) ⇒ scanf_s(“형식지정.. ”, &변수명, ...sizeof(변수명)..)
- file_pointer=fopen(“filename”, “file_mode)⇒fopen_s(&file_pointer, ”filename“, file_mode”)
- fscanf(file_pointer,“filename”, “file_mode)⇒fscanf_s(file_pointer, ”filename“, file_mode”)

■ 2010에서 쓰던 함수를 2013에서도 유효하게 하는 방법

- 프로그램 헤더부분에 다음을 추가

```
#pragma warning(disable:4996)
```

■ 마이크로프로세서 프로그램은 아직도 2010이전 version의 C언어 기반이기 때문에 old version 문법도 알아 둘 필요가 있음

Chap. 2 C 프로그래밍 작성과정

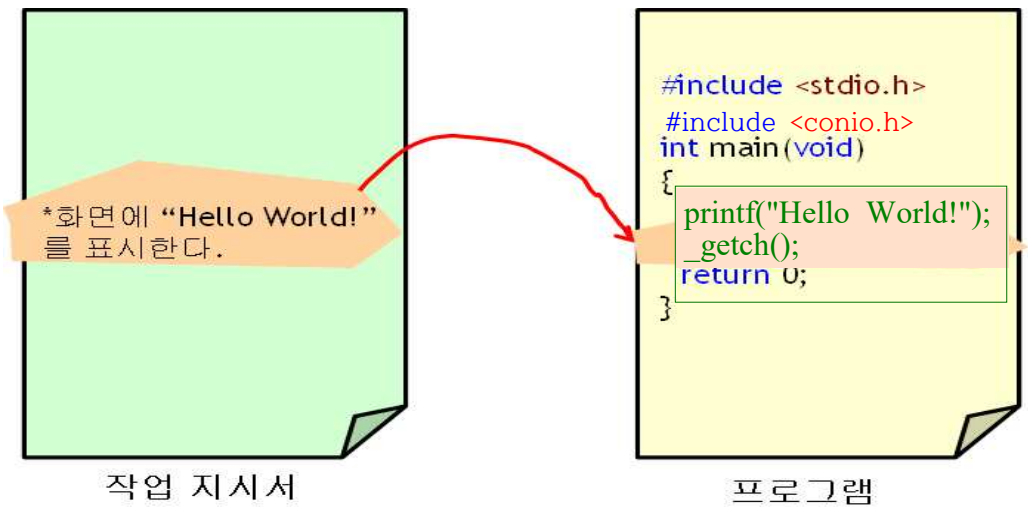
1. 프로그램 구조

■ ex1.c (확장자를 반드시 c로 할 것. cpp는 c++로서 문제 발생할 여지가 있음)

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

int main(void)
{
    printf("Hello World!");
    _getch(); // 출력화면을 정지시켜 놓기 위해 필요한 명령어
    return 0;
}
```

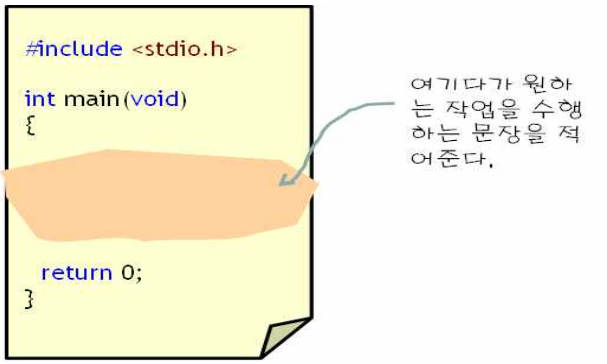
■ 프로그램 설명

· 전체적인 구조



* 프로그램 ⇒ 작업지시서

· 프로그램 내용 (작업지시서 위치)



프로그램

■ 프로그램 설명

```
#include <stdio.h>      <---  헤더파일
#include <conio.h>      <---  헤더파일
int main(void)          <---  main 함수
{
    printf("Hello World!");  <---  화면에 Hello World! 출력
    _getch();                <---  사용자가 키보드를 누를 때까지 프로그램 정지
    return 1;                <---  1 이라는 값을 return(반환)
}
```

① 헤더파일(Header File)

- 의미 : 자주 사용하는 함수나 정의 등의 정보를 가지고 있는 파일.
- 형식

- 필요한 header file은 반드시 include 시켜야 한다.
- printf() 함수는 stdio.h라는 헤더파일에 정의되어 있기 때문에 이 함수를 쓰기 위해서 stdio.h라는 헤더파일을 include 시킴.
- 또한 conio.h라는 헤더 파일은 _getch() 함수가 여기에 정의되어 있기 때문에 include 시킴

② main() 함수

- 의미 : C 언어에서는 반드시 있어야하는 주 함수. C 언어에서는 프로그램시작 시 반드시 main()이라는 함수를 먼저 찾아 이 함수부터 시작함.
- 따라서 하나의 C 프로그램에는 main()이 반드시 있어야 하며 하나의 main() 함수만이 있어야 한다.
- 이 main()함수 내에 필요한 작업내용을 프로그래밍 함

③ main() 함수 형식

```
int   main    (void)
```

함수형 함수이름 입력값 타입

- 함수 형 : 함수 형은 그 함수가 종료될 때 반환해 주는 값의 형식을 의미한다.

Type(함수 형)	return(반환) 값	예
int	정수형 값	1, 250
float	실수형 값	2.5, 376.46
void	반환값 없음	
char	문자형 값	a, !

이 예제에서는 함수의 형이 int 로 선언되어있으므로 정수(1)를 반환한다.

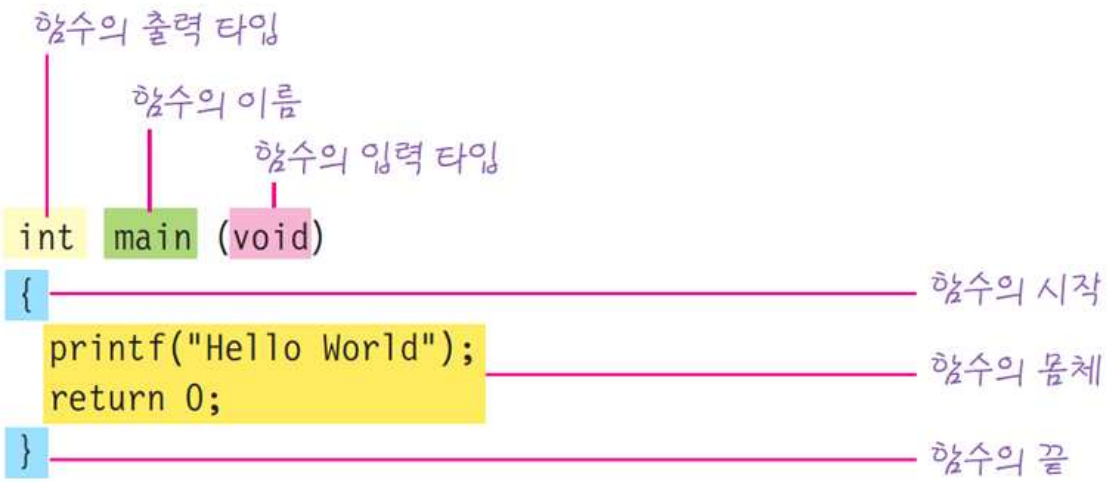
- 함수 이름 : main 은 함수 이름
- 함수 입력 값 : 함수가 처음 시작될 때 필요한 초기 값을 줄 수 있다. 최기 입력 값이 필요 없을 때는 void를 쓴다.

④함수 내용

```
int main(void)
{
    printf("Hello World!");  <--- 화면에 Hello World! 출력
    _getch();                <--- 사용자가 키보드를 누를 때까지 프로그램 정지
    return 1;                <--- 1 이라는 값을 return(반환)
}
```

- 함수 내용은 작업지시서의 내용을 수행하게 하는 여러 개의 문장으로 구성됨.
- 함수 내 문장들은 반드시 순차적으로 실행된다. 즉, printf()가 먼저 실행되고, 다음에 _getch()가 실행된 다음 마지막으로 1을 return하면서 함수가 종료된다.
- 각 함수 내 각 문장 끝에는 반드시 ;이 있어야 한다.
- 함수가 종료될 때는 함수 형과 동일한 타입의 값을 return(반환)해야 한다. 이 예제에서는 함수의 형이 int 이므로 정수형 값 1을 반환한다.

■ 일반 함수 형식



- 함수는 여러 개의 문장으로 구성
- 함수 내 문장들은 순차적으로 실행
- 각 문장 끝에는 반드시 ; 이 있어야 한다.
- 함수는 { 에서 시작하여 } 로 끝난다.

■ printf() 함수 호출

- printf()는 컴파일러가 제공하는 함수로서 헤더파일 stdio.h 내에 정의되어 있다.
- 함수 의미 : “ ” 내의 문자열을 화면(모니터)에 출력하는 함수
- 형식 : printf(“출력문자열”);

■ _getch();
- 컴파일러의 라이브러리 내의 conio.h 라는 헤더파일에 정의되어있는 함수
- 기능 : 키보드로부터 하나의 문자 입력
- 즉 아무 키보드가 입력될 때까지 프로세스 진행을 멈추는 역할 (이 함수가 없으면 순간적으로 프로그램이 실행되고 종료되어버리므로 화면의 출력 문자를 확인 할 수가 없음)

■ 함수 반환 값 return 1;
- 함수는 마지막에 필요한 경우 하나의 값을 그 함수를 호출한 쪽으로 반환(return) 할 수 있다.
- 형식

```
return 반환값;
```

- main()에서는 운영체제(OS)로 값을 반환

note] return값이 불필요한 경우는 함수 형(출력 타입)을 void로 해 준다.

2. printf() 함수 응용

1) 다음과 같은 문자열을 모니터에 출력하는 프로그램을 작성

- Hellow World를 출력한 다음 한 칸 내려가서 자기 이름은 영문 또는 한글로 출력되게 프로그램

■ 프로그램 순서

① 새로운 ex2.c 파일을 생성 (반드시 확장자는 .c 가 되어야 함)

② ex1.c의 내용을 ex2.c에 복사

③ ex1.c를 프로젝트에서 제거 (하나의 project에 두개의 main() 불가)

④ ex2 프로그램 내용을 수정하여 자기 이름이 출력되게 작성.

■ 프로그램 내용

① 첫 번째 시도

```
#include <stdio.h>
#include <conio.h>
int main(void)
{   printf("Hello World!");
    printf("Jong-Hwan Lim");
    _getch();
    return 1;
}
```

실제 프로그램 실행 결과

② 두 번째 시도 (파란색으로 된 // 이후는 설명문으로 프로그램 내용이 아님)

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    printf("Hello World!\n"); //'\n' 은 visual studio 에디터에서는 '₩n'으로 표시됨
    printf("Jong-Hwan Lim");
    _getch();
    return 1;
}
```

실제 프로그램 실행 결과

■ 고찰

① 프로그램 문장은 순차적으로 실행된다.

② printf()내의 '\n'은 화면의 커서를 다음 줄로 이동하게 한다.

2) 위 두 문장을 하나의 printf()문으로 출력 (ex3_1)

① '\n'은 printf()함수의 " " 내에 어디에도 위치할 수 있으며 복수로 위치할 수 있다.

② printf() 함수는 '\n'을 만나면 항상 커서의 줄 바꿈을 수행 한다.

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    printf("Hello World!\n Jong Hwan Lim");
    _getch();
    return 1;
}
```

3) return 이 없는 프로그램 : 위 예제에서 1을 return 하는 것은 아무 의미가 없음. 아무 값도 return 하지 아니하는 프로그램 (ex3_2)

- int 는 정수를 return 한다는 의미
- void 는 아무것도 return 하지 않는다는 의미

```
#include <stdio.h>
#include <conio.h>
void main(void)
{   printf("Hello World!\n Jong Hwan Lim");
    _getch();
}
```

4) printf()문 내부의 공백

■ 공백이 있는 printf()문 (ex4_1)

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    printf("    Hello World!\n");
 /* printf("Jong-Hwan Lim\n");*/
 // printf("Jong-Hwan Lim\n");
    _getch();
}
```

■ 고찰

① printf()문에서 공백은 문자열로 취급되어 모니터에 그대로 출력된다.

② /* ... */ 와 //는 주석문으로 그 사이의 문장이 무시된다.

NOTE] : /* ... */ 와 // 의 차이점

- // 는 시작부터 그 줄 끝까지의 문장만 무시
- /* ... */ 는 그 사이의 모든 문장이 무시됨 (ex4_2)

```
#include <stdio.h>
#include <conio.h>
void main(void)
{    printf("    Hello World!\n");
  // printf("Jong-Hwan Lim\n"); //또는 /*  */이용해서 두 문장 주석처리
  // printf("제주대학교\n");
     _getch();
}
```

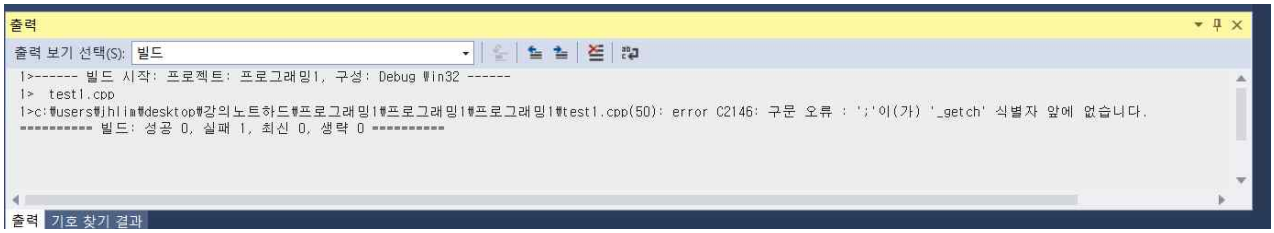
3. Debugging (오류 수정)

1) 에러와 경고

■ Error : 컴파일 또는 링크가 불가능한 심각한 오류. 프로그램 실행이 불가 (test5)

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
   printf("Hello World!\n")      // ';' 이 없음
   _getch();
}
```

*** 에러 메시지**

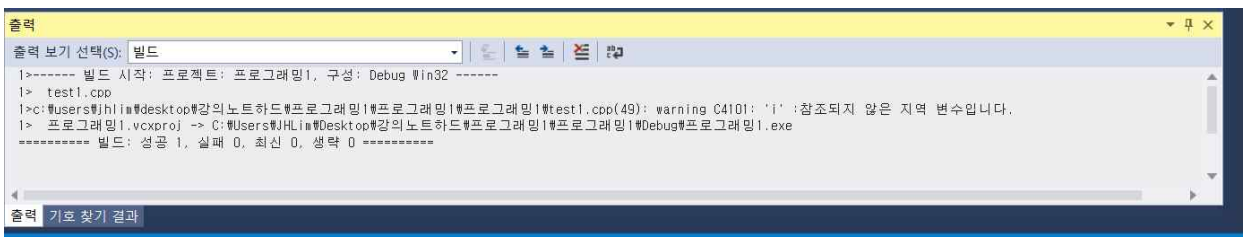


* 어떤 error가 발생?
- _getch 식별자 앞에 ; 이 없음

■ Warning (경고) : 컴파일, 링크 및 실행 가능하지만 문제를 일으킬 수도 있다는 메시지

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int i; // 변수정의
    printf("Hello World!\n");
    _getch();
}
```

*** 경고 메시지**



- 경고 메시지는 반드시 확인하여 문제 발생 여부를 확인 후 발생 소지가 있으면 수정

2) Error 종류

① 컴파일 에러 : 프로그램 코딩 시에 문법 오류에 의해 발생

② 실행 에러 : 컴파일 링크 및 실행이 되지만 실행도중 오류가 발생하면서 프로그램이 중단됨. 실행 중 프로그램 내부에서 0으로 나누는 경우가 발생하거나 잘못된 메

모리 참조 등에 의해 발생

③ 논리 에러

프로그램이 논리적으로 잘못이 있어 의도한 결과가 나오지 않는 에러

NOTE] ① ⇒ ② ⇒ ③ 순으로 error 수정이 어려움

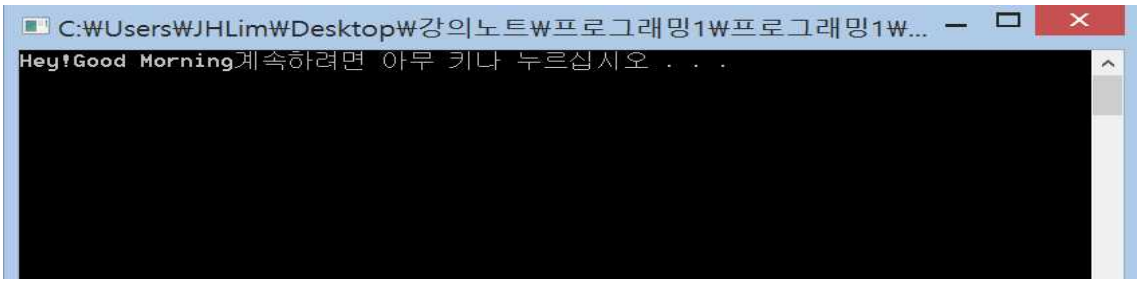
Ex] 논리 오류의 예

● 다음과 같은 모니터 출력 프로그램 작성

● 실제 coding

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
   printf("Hey!");
   printf("Good Morning!");
   _getch();
}
```

● 실행결과



3) Debugging : 실제 논리 error 디버깅 방법은 추후에 다시 설명

HW 1. 00번, 00번 (Due : 일주일 후)

Chap. 3 C 프로그래밍 구성요소

1. 프로그램 구조

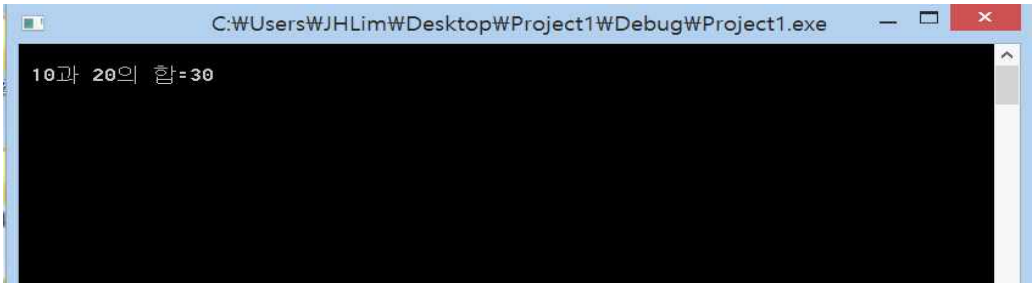
1) 예제 프로그램 1 (ex6)

■ Code

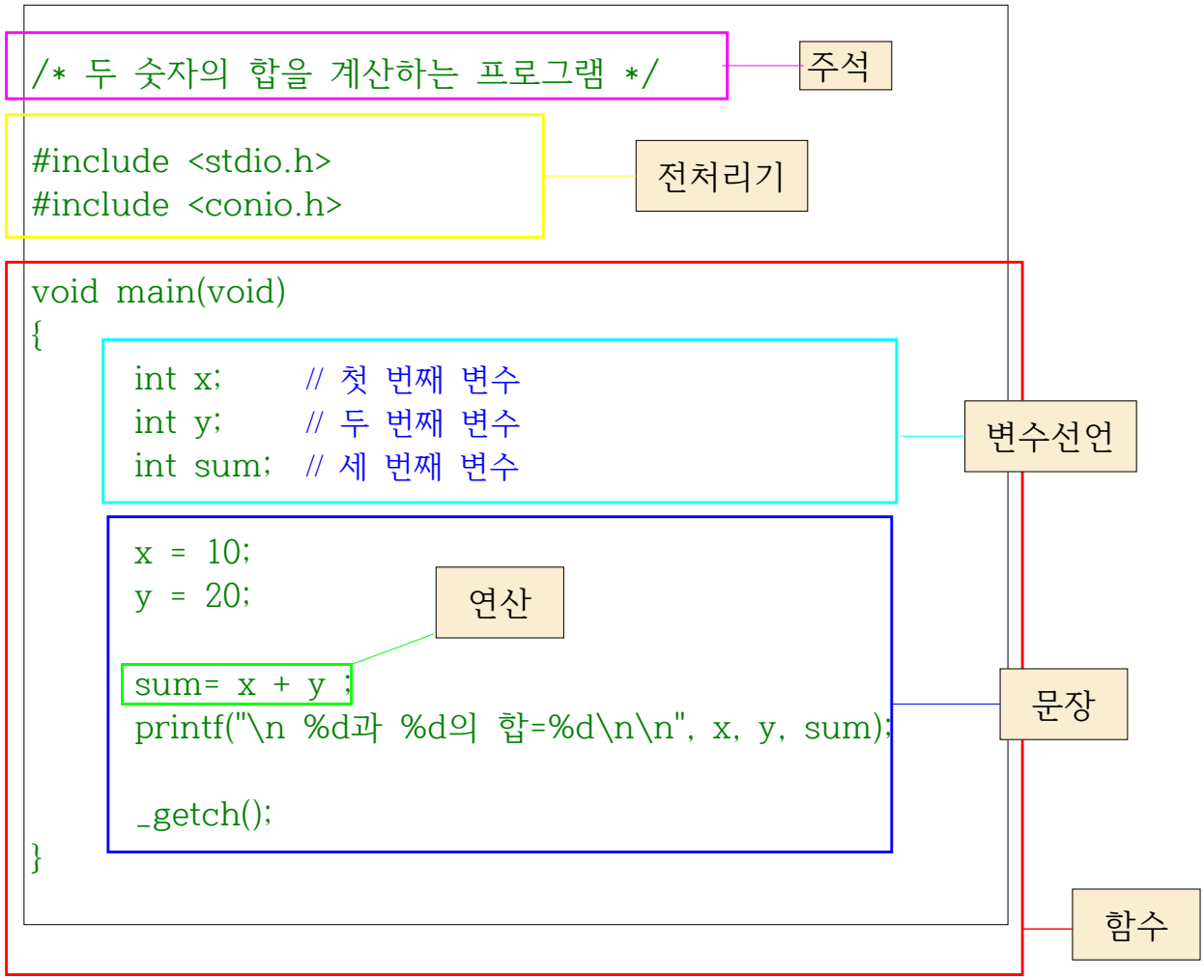
```
/* 두 숫자의 합을 계산하는 프로그램 */

    #include <stdio.h>
    #include <conio.h>

    void main(void)
    {       // 변수 선언
            int x;      // 첫 번째 정수변수 (정수를 저장 및 읽어 올 수 있는 변수)
            int y;      // 두 번째 정수변수
            int sum;    // 세 번째 정수변수

            x=10;
            y=20;
            sum = x + y;
            printf("\n %d과 %d의 합=%d\n\n", x, y, sum);

            _getch();
    }
```

■ 변수 : 프로그램 수행과정에서 어떤 값을 저장하거나 저장된 값을 읽어오기 위해 정의하는 메모리 공간 이름. 변수는 정수형, 실수형 그리고 문자형으로 각각 구변하여 선언하여야 함. (아래 자세한 설명 참조)

■ 실행 결과



■ 구조



■ 프로그램 구성 해석

① 주석 (comment) : 코드를 설명하는 글. 컴파일 시에 무시됨

- /* 한 줄로 된 주석 */
- 여러 줄로 된 주석

```
/* ----------------------
    프로그래머 : 홍길동
    날짜 : 2016.3.17.
    ---------------------- */
```

- // 여기서부터 이 줄 끝까지 영향

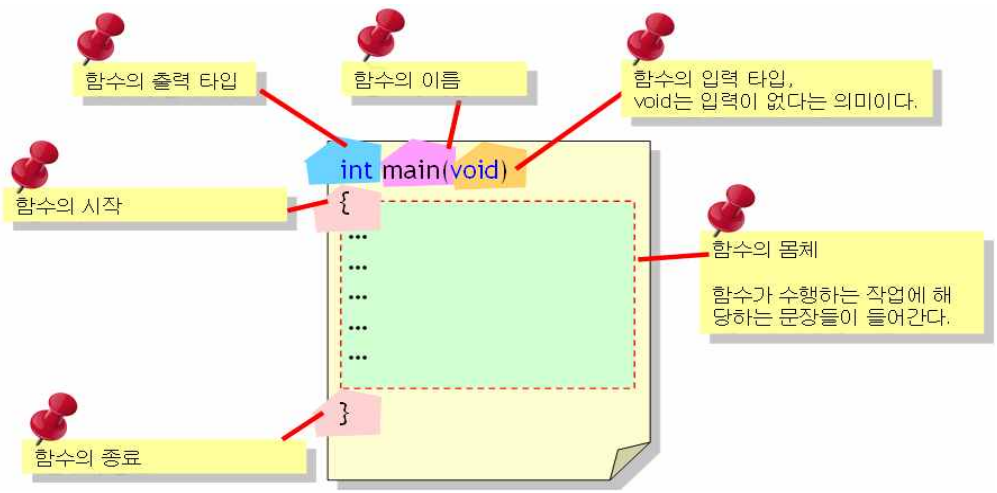
② 전처리기 : 전처리를 지시하는 지시어

내용	예	비고
헤더파일	#include <헤더파일>	라이브러리에 있는 헤더파일 포함
	#include "헤더파일 명"	프로그래머가 만든 헤더파일 포함(현재 project 내 파일)
	#include "파일 명"	프로그래머가 만든 다른 파일 포함
정의	#define ABS(x) (((double)(x)>=0.)? (x): -(x))	함수 정의
	#define PI 3.14159	수치 정의
	#define ESC 0x1b	입력키 정의

- 전처리기 끝에는 ';'을 붙이지 않는다.

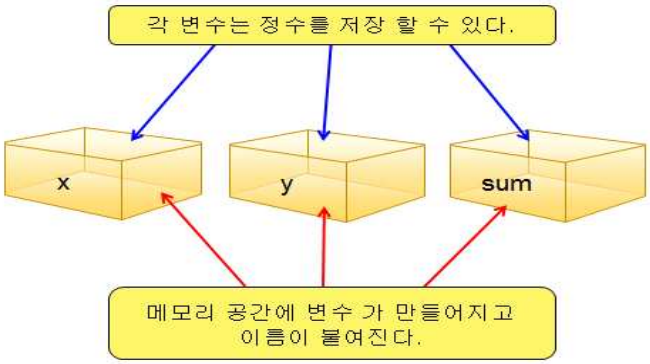
③ 함수

- C는 함수들로 구성된다.
- 함수는 그 함수가 호출 될 때 실행된다.
- 함수의 호출은 그 함수 이름을 적어주면 된다.
- main()함수가 맨 먼저 호출된다.
- main()함수는 윈도우 OS가 호출한다.
- 함수는 호출 시에 값을 입력할 수도 있고 종료 시에 값을 return 할 수도 있다.



③ 변수

- 프로그램에 사용하는 데이터(값)를 일시적으로 저장할 목적으로 사용하는 메모리 공간



- 변수는 반드시 함수의 맨 첫 부분에 정의되어야 한다.(C 기준)
- 변수는 반드시 형을 지정해야 한다.

  int x; (정수형) float r;(실수형) char k; (문자형)
- 변수명 : 영문자, 숫자, 밑줄문자 (_)로 구성한다.

  int num; int flag1; char sum_1;
- 변수명은 숫자로 시작 불가.

  int 1num; (x)
- 대문자와 소문자는 구별되며 C 언어 및 라이브러리에서 제공되는 변수명 함수명 등과 동일한 이름 사용 불가.
- 변수에 값을 대입하는 방법 : 대입연산자 ' = '을 이용

  x = 100; : 100 이라는 값을 x 변수가 지정된 메모리에 대입

  sum = x + y; : x와 y에 저장된 값을 더하여 sum에 대입 (sum 변수가 지정된 메모리에 저장)
- 더 자세한 변수에 대한 내용은 4장에서 설명

④ 수식

- 산술 연산자를 이용하여 수식 구성
- 산술 연산자 종류

연산	연산자	C 수식	수학에서의 기호
덧셈	+	x + y	x + y
뺄셈	-	x - y	x - y
곱셈	*	x * y	x×y
나눗셈	/	x / y	$\frac{x}{y}$
나머지	%	x % y	x mod y (x를 y로 나눈 나머지)

④ 모니터 출력 : printf()

- 모니터 표준 출력 함수
- 라이브러리에서 제공되며 stdio.h에 선언되어 있음
- 사용방법

printf(" 문자 열, 출력형식(변수 값 출력)... 캐리지 리턴", 출력변수...);

- 문자 열 : typing된 문자열을 그대로 출력

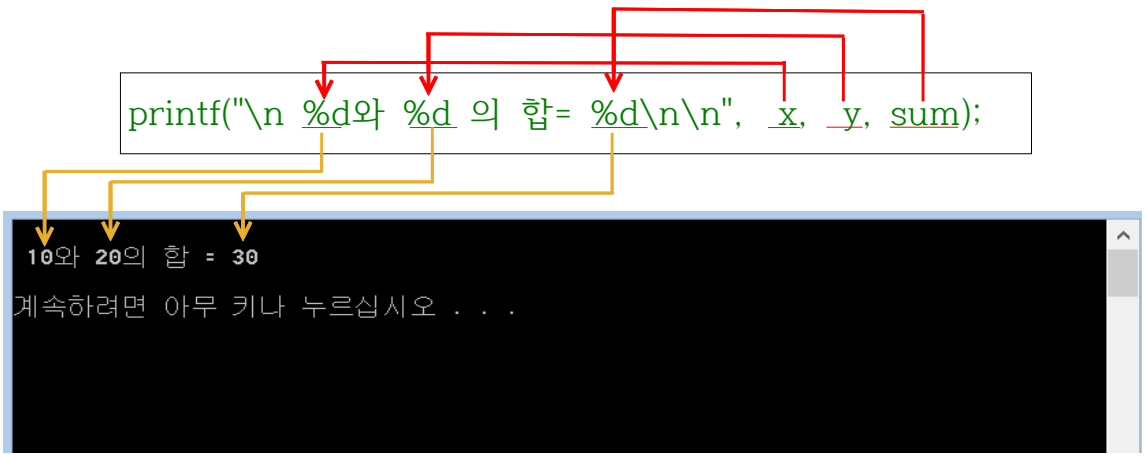
printf("제주대학교 메카트로닉스공학과");

- 출력형식 : 출력하려는 변수의 출력형식 지정 (복수개로 지정 가능)

형식 제어 문자열	의미	형태
%d	정수 형태로 출력	100
%f	실수 형태로 출력	3.141592
%c	문자 형태로 출력	A
%s	문자열 형태로 출력	Hello

printf("x=%d y=%d", x, y); : x는 정수형, y도 정수형으로 출력

- 캐리지 리턴 (\n) : 모니터의 커서를 다음 줄로 이동
- 출력변수 : 출력하려는 값을 가진 변수 (복수지정가능, 단 출력형식과 개수 및 형이 일치해야 함), 수식도 가능



예제] 다양한 형의 변수 출력 (ex7)

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int j = 2;                      // 정수형, 형 선언 시에 초기 값 대입가능
    float d = 3.2;                  // 실수 형
    char s = 'c';                   // 문자 형, ' '으로 대입
    char h[20] = "제주대학교"; // 문자열 형, " "로 초기화

    printf("j=%d d=%f s=%c h=%s\n", j, d, s, h);
    _getch();
}
```

고찰] printf("j=%f d=%f s=%c h=%s\n", j, d, s, h); 로 수정해서 실행해 볼 것 (형이 맞지 않으면 잘못된 출력)

예제) 정수 변수 x를 선언하고 x에 10을 대입한 다음 x값을 출력

■ printf() 함수의 변수 위치에 수식이 올 수 있다.

```
void main(void)
{   int j=2, k=3;      // 동일한 형의 변수는 여러 개 동시 선언가능
    printf("%d와 %d의 곱 : %d\n", j, k, j*k);
    _getch();
}
```

2. 키보드로부터 숫자나 문자를 입력하는 함수

■ scanf_s() : 키보드로부터 문자나 숫자를 입력받는 함수 (2010이하 version에서는 scanf())

- 표준 키보드 입력함수로서 stdio.h 에 정의
- 형식 : scanf_s("형식지정자 형식지정자 ... ", 변수주소, 변수주소, ...); - 형식 지정자의 종류 (printf()의 형식지정자와 동일)
- 형식 지정자 (printf()의 형식지정자와 동일)

형식 지정자	의미	예
%d	정수를 10진수로 입력한다	scanf("%d", &i);
%f	float 형의 실수로 입력한다.	scanf("%f", &f);
%lf	double 형의 실수로 입력한다.	scanf("%lf", &d);
%c	문자 형태로 입력한다.	scanf("%c", &ch);
%s	문자열 형태로 입력한다.	char s[10]; scanf("%s", &s);

- 변수의 주소 : 수명 앞에 &를 붙이면 그 변수가 지정된 주소를 나타낸다.
- 기능 : 키보드로부터 지정된 형식으로 수 또는 문자를 입력하여 변수주소 순서대로 대입

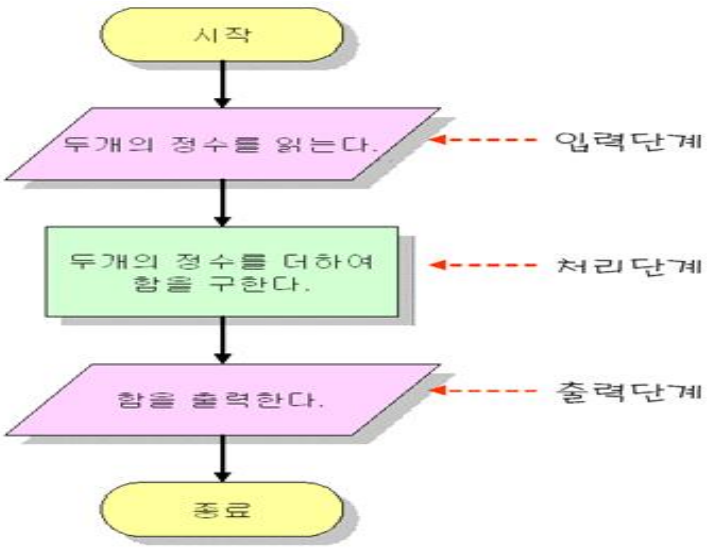
```
scanf_s("%d %f",&x, &y);
```

⇒ 첫 번째 키보드로 입력된 수를 정수형으로 x 변수에 대입하고 두 번째 입력된 수를 실수 형으로 y 변수에 대입

- 변수의 주소 : 변수 명 앞에 & 연산자를 붙이면 그 변수의 주소를 나타낸다.

■ 예제 : 키보드로 두 수를 입력 받아 그 수를 더하여 모니터에 출력하는 프로그램

① 알고리즘



② 프로그램 코드 (ex8)

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

void main(void)
{       int x, y, sum; //세 개의 정수형 변수 선언

        printf("첫 번째 숫자를 입력 하세요.\n");
        scanf_s("%d", &x);       // 입력한 수를 정수형으로 x 변수에 대입
        printf("두 번째 숫자를 입력 하세요.\n");
        scanf_s("%d", &y);       // 입력한 수를 정수형으로 y 변수에 대입
        sum= x+y;
        printf("%d와 %d의 합은 %d입니다.\n", x, y, sum);
        // printf("%d와 %d의 합은 %d입니다.\n", x, y, x+y);
        _getch();
}
```

*예비 test1) 키보드로부터 실수를 하나 입력받아 그 수를 출력하는 프로그램
*예비 test2) 키보드로부터 정수 두 개를 입력받아 두 수의 차을 모니터에 출력하는 프로그램 (출력형식 : 00와 00의 차는 00)

3. scanf_s 함수로 문자 또는 문자열을 입력하는 경우

- 2010 : scanf("형식지정자.. ", &변수명, ...)
- 2013 : scanf_s("형식지정자.. ", &변수명, ...sizeof(변수명)..)

■ 변수가 char 형일 때만 반드시 그 다음에 그 변수의 크기를 나타내는 sizeof (변수명) 추가 필요.

scanf_s()를 써서 문자나 문자열을 입력할 때는 그 다음에 sizeof(문자변수)를 추가해야 한다.

```
scanf_s("%d %c", &j, &c, sizeof(c));
```

■ 예제 : 숫자와 문자를 입력하여 출력하는 프로그램(ex8_2)

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

void main(void)
{
        int j;
        char s[20] = "";   // 문자열
        printf("정수 및 문자열 입력:");
        //scanf(""%d %s", &j, &s);     // 2010 version
        scanf_s("%d %s", &j, &s, sizeof(s)); //2013 version
        printf("정수:%d      문자열 : %s", j, s);
        _getch();
}
```

HW 2. 3장 연습문제 프로그래밍 0, 00번
*due : 1 주일 후

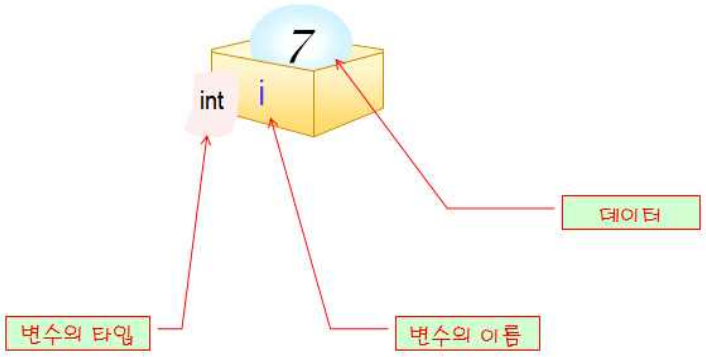
Chap. 4 변수와 자료형

1. 변수 (variables)

■ 프로그램에서 일시적으로 데이터를 임시 저장하는 공간 (컴퓨터 메모리에 위치)

■ 데이터는 반드시 어디엔가 저장되어야만 다음에 사용할 수 있다.

ex) int i = 7; → i를 정수형으로 선언하면서 동시에 7로 초기화



■ 데이터는 숫자, 문자 및 문자열이 될 수 있다. (실제는 이진수로 저장)

■ 변수에 저장된 값은 언제든지 다른 값으로 변경될 수 있다.

** 상수(constant) : 프로그램 내에서 항상 일정한 값을 유지

ex9_1)

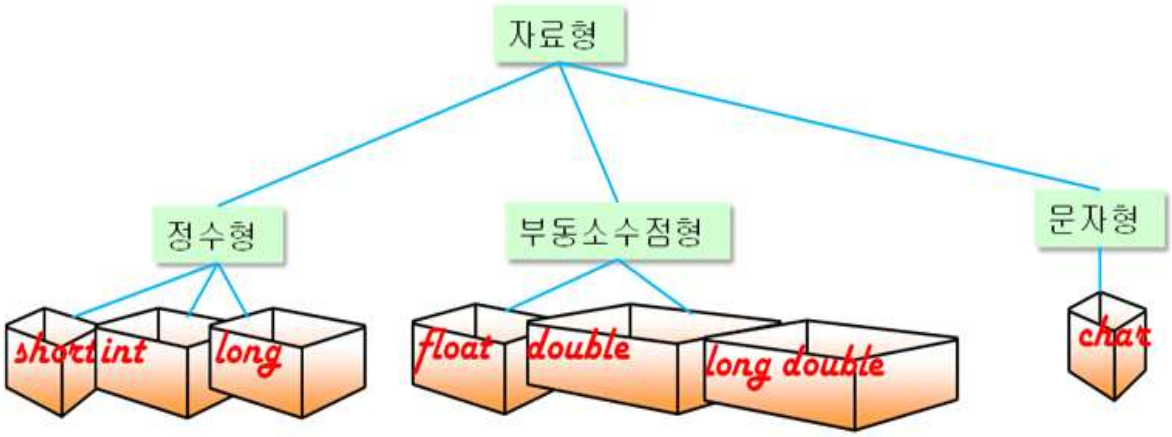
```
void main(void)
{       float radius;     // 변수
        float area;       // 변수

        printf("원의 반지름을 입력하시오:");
        scanf_s("%f",&radius);
        area=3.14159 * radius * radius;  // 3.14159는 컴파일 시 상수로 저장
        printf("원의 면적 : %f\n", area);
        _getch();
}
```

2. 자료형 (Data Type)

■ 데이터 종류에 따라 필요한 메모리 공간(크기)이 다르다.

■ C 언어는 효율적인 메모리 관리를 위해서 다양한 자료 형을 정의

■ 모든 변수는 반드시 자료 형을 선언해야 함



■ 변수의 자료 형에 따라 할당되는 메모리 크기가 다르다.

■ 컴퓨터는 모든 수 및 문자 등 데이터를 2진수 형태, 즉 1 또는 0으로 저장한다.

■ 데이터 저장 형태

- Bit : 2진수를 구성하는 한 단위
- Byte : 8 bit 가 1 byte가 됨. 예를 들어 x를 8bit 변수로 선언하고 x = 3이라 대입하면 실제 컴퓨터 메모리에는 다음과 같은 8 bit (1 byte) 형태로 저장 됨.

0	0	0	0	0	0	1	1

즉, 처음 6bit 는 0이 되고 나머지 두 bit 는 1이 됨.

- Bit 수에 따른 표현 가능한 수의 범위는 0 ~ ($2^{bit수}$ -1)이 됨. 예를 들어 4 bit인 경우는 2^3 -1 = 7 이므로 0 ~ 7 까지 8개의 숫자가 표현 가능함.

[NOTE] - Bit : 2진수를 구성하는 한 단위

- Byte : 8 bit = 1 byte

■ 자료 형 및 숫자 범위

자료형			설명	바이트수	범위
정수형	부호있음	short	short형 정수	2	-32768 ~ 32767
		int	정수	4	-2147483648 ~ 2147483647
		long	long형 정수	4	-2147483648 ~ 2147483647
	부호없음	unsigned short	부호없는 short형 정수	2	0 ~ 65535
		unsigned int	부호없는 정수	4	0 ~ 4294967295
		unsigned long	부호없는 long형 정수	4	0 ~ 4294967295
문자형	부호있음	char	문자 및 정수	1	-128 ~ 127
	부호없음	unsigned char	문자 및 부호없는 정수	1	0 ~ 255
부동소수점형		float	단일정밀도 부동소수점	4	1.2E-38 ~ 3.4E38
		double	두배정밀도 부동소수점	8	2.2E-308 ~ 1.8E308

- int a; 로 선언하는 경우 : int 형 자료는 크기가 4 byte로 메모리에 할당되므로 총 32bit 크기가 된다. 그러나 맨 처음 bit는 부호 bit 로 사용되므로 나머지 31bit 만 숫자로 사용됨. 따라서 2^{31} - 1= 2147483647이므로 0 ~ 2147483647까지 표현 가능하다. 반면에 첫 번째 bit가 1인 경우는 음수이며 이때는 0의 표현이 불필요하므로 -2^{31} = - 2147483648 까지 표현가능하다. 따라서 이 경우 -2147483648 ~ 2147483647 까지 표현 가능하다.

- unsigned int a; 로 선언하는 경우 : 부호 bit가 불필요하므로 0 ~ 2^{32} - 1= 4294967295 까지 표현이 가능.

- 32 bit 이상의 CPU를 갖는 컴퓨터에서는 int 나 long 형의 크기는 같다. (int는 CPU의 기본 비트를 기준으로 함)

3. 변수의 자료 형 크기를 return 하는 연산자 sizeof()

■ 문법 : sizeof(변수); 혹은 sizeof(자료형); : 자료형의 크기를 byte형태로 return 함

■ 각 변수형의 크기를 알아보는 프로그램 (ex9_2)

```
#include <stdio.h>
#include <conio.h>
void main(void)
{       int j;
        char c;
        float d;
        double m;

        printf("int의 크기 : %d 바이트\n", sizeof(j));
        printf("char의 크기 : %d 바이트\n", sizeof(c));
        printf("float의 크기 : %d 바이트\n", sizeof(d));
        printf("double의 크기 : %d 바이트\n", sizeof(m));
        printf("long 의 크기 : %d 바이트\n", sizeof(long));// 자료형
        _getch();
}
```

4. 변수명 규칙

■ **식별자 (identifier) : 식별하게 해주는 이름. 변수이름, 함수이름.**

■ **식별자 (변수명, 함수명) 규칙**

- 알파벳 문자와 숫자, 밑줄 문자 _로 구성 (Visual C 2010부터는 한글도 가능)
- 첫 번째 문자는 반드시 알파벳, 한글, 또는 밑줄 문자 _ (숫자는 불가)
  * 마이크로프로세서에서는 한글 불가능
- 대문자와 소문자를 구별
- C 언어의 키워드와 똑같은 이름은 허용되지 않는다. (Library 함수 명 포함)

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

■ **예제** (ex10)

```
void main(void)
{     int 정수 = 3;
      char 문자_c = 'A';
      float _d = 102.34;

      printf("정수 : %d \n", 정수);
      printf("문자  : %c\n", 문자_c);
      printf("실수  : %f\n", _d);
      _getch();
}
```

5. 변수 선언과 사용

■ 선언 형식 : 자료형 변수명;

```
int j;          // 하나의 변수 선언
float f, h, t; // 여러 개 변수 동시 선언
```

■ 지역변수와 전역변수로 나뉨

■ 지역변수 선언 위치 : 반드시 함수 첫 부분에 선언해야 함

■ 전역변수 (global variable)

- 전역변수는 함수 외부에 선언해야함
- 전역변수는 프로그램 전체에 다 유효함. (함수 내부에 선언되는 변수는 그 함수 내부에서만 유효함)

■ 변수 선언 예제

```
#include <stdio.h>
#include <conio.h>
int a, b;      // Global variable (전역변수)

int add(void) // 프로그래머가가 만든 함수
{
    return (a+b);
}
void main(void) // main 함수
{
    int sum; // Local variable (지역 변수)
    a=2;
    b=4;
    sum = add();  // 함수 add()가 return하는 값을 sum에 대입
    printf("수의 합은 %d\n", sum);
    _getch();
}
```

■ 변수에 값 대입방법

Type	변수 선언	값 대입 방법
정수	int j;	j = 34; (공백은 무관)
실수	float k;	k = 2.674;
문자	char a;	a = 'T';
문자열	char s[10];	- 변수 선언 시 대입 : char s[10]="JEJU"; - 변수 선언 후 문장 내에서는 s[0]='J'; s[1]='E'; s[2]='J'; s[3]='U'; (문자형으로 개별 입력해야 함)

- 실수변수는 정수로 입력해도 자동으로 실수형으로 저장됨
- 문자 변수에 문자를 대입할 때는 반드시 ' '를 써야함.

  예) c라는 문자형 변수에 문자 A를 대입할 때

  c = 'A';
- 문자열은 두 개 이상의 문자를 이미함. 따라서 문자열을 선언하는 경우 반드시 문자열크기는 변수명 뒤에 [ ]를 이용함. 문자열의 맨 마지막 byte에는 NULL 문자(\0)가 할당됨. 따라서 문자열 크기보다 하나 더 크게 선언해야 함.

  예) s라는 이름의 최대 문자 개수 10개인 문자열 선언하는 경우

  char s[11];
- 문자열 변수에 문자열을 대입 할 경우는 반드시 " "를 이용해야함.

  예) s라는 이름의 최대 문자 개수 10개인 문자열 변수에 Jeju라는 문자 대입

  char s[11] = "Jeju";

■ 문자 및 문자열 변수 예제

```
void main(void)
    {   char s[5];
        s[0] = 'L';
        s[1] = 'o';
        s[2] = 'v';
        s[3] = 'e';
        s[4] = '\0'; //NULL 필요
        printf("문자 : %s", s);
        _getch();
    }
```

```
void main (void)
{
        char s[5]="Love";
        printf("문자 : %s", s);
        _getch();
}
```

6. 변수의 초기화

■ 변수 선언 시에 초기화 가능

```
int j = 45;
float k = 2.;      // 실수 형은 소수점을 붙임
char c = 'A';      // 문자
char s[10] = "제주"; // 문자열 개수보다 하나 더 크게 잡아준다.
```

* 초기화 된 변수의 값도 언제든지 프로그램 내에서 변경가능

7. 정수형 변수

■ 종류 : short (2바이트), int (4바이트), long (4 바이트)

■ unsigned는 기존 형 보다 두 배 큰 수 표현가능 (음수는 사용 불가)

■ 오버플로어 (Overflow) : 그 변수가 나타낼 수 있는 최대 크기를 넘어가는 경우

■예제 [ex12]

```
#include <stdio.h>
#include <conio.h>
#include <limits.h> // 각 변수형의 최댓값이 정의되어 있는 헤더파일
void main(void)
{
    short s = SHRT_MAX; // 32767, limit.h에 정의 되어 있는 수
    unsigned short u = USHRT_MAX; //65535
    s = s+1;
    u = u+1;
    printf("s = %d u = %d\n", s, u);// s=1000 0000 0000 0000 : -32768
                                    // u=0000 0000 0000 0000 : 0
    _getch();
}
```

** 고찰

- short : - 32768 ~ +32767
  s=s+1 ⇒ 최상위 비트만 1 나머지 비트 전부 0 ⇒-32768
- unsigned short : 0 ~ 65535
  u=u+1 ⇒ 16 비트 전부 0 ⇒ 0

■ 정수 상수

- 프로그램 내에 정수상수를 표현하면 컴파일러는 기본적으로 int 형으로 인식
- 만약 상수를 다른 형으로 하려면 숫자 뒤에 접미사를 붙인다.

EX] 123L; // 123은 int 형이 아닌 long 형으로 메모리에 저장됨

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

ex) c = 25L * b * 15u; // 25는 long 형, 15는 unsigned 형

■ 기호상수 : 직접 숫자를 쓰지 않고 그 숫자를 나타내는 대표 기호로 표현

- 기호상수 정의 법

1. #define 기호상수 값
2. const 자료형 변수명 = 값;

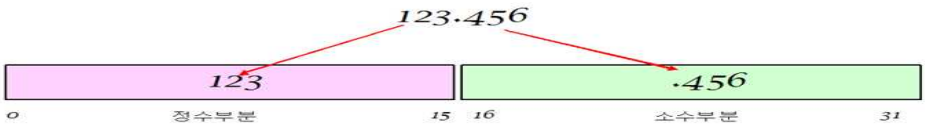
EX] #define PI 3.14159, const int N = 2; (const는 상수를 정의)

- #define 방법의 기호 상수는 반드시 전처리부에 위치해야 한다.
- const 형은 함수 내부에 선언 가능하다.
- ex13]

```
#include <stdio.h>
#include <conio.h>
#define NUM 5

void main(void)
{
    int mul;
    const int N = 3; // N을 상수형으로 정의
    //N=N+1;    // error
    mul=NUM*NUM;
    printf("%d의 제곱은 %d\n", NUM, mul);
    printf(" %d의 제곱은 %d\n", N, N*N);
    _getch();
}
```

8. 부동소수점형 (floating point)

1) 고정 소수점 방식 : 정수부와 소수부에 일정한 비트를 할당하여 표현

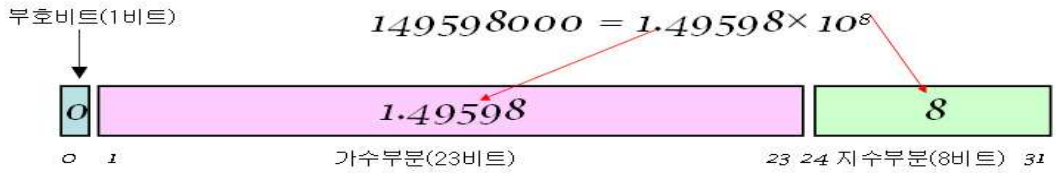


■ 32bit 중 첫 번째 bit 는 부호로, 그 다음 15비트는 정수부분, 나머지 16 bit 는 소수점 이하 부분으로 쓰는 경우. 이 경우는 최대로 큰 수가 32767.65535 임. 따라서 숫자 범위가 너무 작음.

2) 부동소수점(floating point) 방식 : 가수 부분과 지수부분으로 나누어 표현

■ 가수부분 : 실수의 유효자리 부분

■ 지수 부분 : 실수 값의 범위 결정



- 예를 들어 65435.456은 32 bit 고정 소수점 방식으로는 표현 불가능. 그러나 이것을 6.5435456×10^4 형식으로 표현하면 가능함. 따라서 앞의 23 bit를 유효숫자 65435456을 표현하고 나머지 8 bit는 10의 승수인 4를 나타내는 bit로 쓰면 고정 소수점 방식의 문제가 해결 됨. 이와 같이 소수점이 일정하게 정해지지 아니하고 숫자에 따라 바뀐다고 해서 부종소수점(floating point) 방식이라 함.

■ 부동 소수점 자료형 (시스템 마다 다를 수 있음)

자료형	명칭	크기	범위
float	단일정밀도(single-precision) 부동소수점	32비트	$\pm 1.17549\times10^{-38} \sim \pm 3.40282\times10^{+38}$
double long double	두배정밀도(double-precision) 부동소수점	64비트	$\pm 2.22507\times10^{-308} \sim \pm 1.79769\times10^{+308}$

■ 지수 표기법

실수	과학적 표기법	지수 표기법
123.45	1.2345×10^2	1.2345e2
12345.0	1.2345×10^5	1.2345e5
0.000023	2.3×10^{-5}	2.3e-5
2,000,000,000	2.0×10^9	2.0e9

*예를 들어 0.000023은 2.3e-5 또는 2.3E-5로 표기 할 수 있음.

■ 부동소수점 출력 법

- 기본적으로 소수점 6자리만 출력, 그 이하는 반올림
- %f : float 형 실수 출력 (유효 자리수 : 소수점 6자리)
- %lf : double 형 실수 출력 (유효 자리수 : 소수점 6자리)
- %10.8f : 전체 10자리로 출력. 소수점 이하는 8자리로 출력하라는 의미
- %e : 지수형으로 출력

■ 예제 (ex14)

```
void main(void)
{ float    x=1.234567891234567890;
  double  y=1.234567891234567890;
  printf("x=%f\n",x);        // 기본: 소수점 6자리 출력 (반올림)
  printf("y=%lf\n",y);       // 기본: 소수점 6자리 출력(반올림)
  printf("자리수지정 : x=%30.25f\n",x); //전체 자리수 30, 소수점 자리수 25
                                        //float : 소수점 7자리까지만 유효
  printf("자리수지정 : y=%30.25lf\n",y); //double :소수점 16자리까지만 유효
  y= 0.123456789123456789;  // double형은 유효숫자가 최대 16자리
  printf("지수 출력 : y=%e\n",y);      // 자리수 미지정 : 소수점 6자리 출력
  printf("자리수지정 지수출력 : x=%.10e\n",y);  // 소수점 10자리 출력,
                                                // 최대 16자리,y는 double 이므로
  _getch();
}
```

```
C:₩Users₩SAMSUNG₩Desktop₩test₩Debug₩test.exe
x=1.234568
y=1.234568
자리수지정 : x=   1.2345678806304932000000000
자리수지정 : x=1.2345678912345679000000000
지수 출력 : y=1.234568e-001
자리수지정 지수출력 : x=1.2345678912e-001
```

■ 실수 표기법

- x=1.2345;
- x=2.; // 소수점 이하가 0인 경우 ‘ . ’만 표가 해도 됨
- x = .23; // 소수점 앞부분이 0인 경우
- x= 1.2e10; // 1.2 x 10^{10}
- x= 1.2e-8; // 1.2 x 10^{-8}

■ 실수 형에도 overflow(범위 보다 큰 수)와 underflow(너무 작은 수)가 발생한다.

예제) ex15

```
void main(void)
{
    float x= 1E39;          // float 형 범위 초과
    float y=1.23456E-46;    // 0에 가까움
    printf("x=%e\n",x);     // overflow : 무한대 출력
    printf("y=%e\n",y);     // underflow : 0 출력
    _getch();
}
```

9 문자형

1) 문자와 아스키 코드

■ 컴퓨터는 문자도 숫자로 표현 (2진 숫자만 표현가능하기 때문)

■ 따라서 일정한 숫자범위를 정해서 각 숫자별 대응되는 문자를 지정하고 있음

ex) 65 ⇒ A, 66 ⇒ B

■ 아스키(ASCII) 코드 : 숫자 0 ~ 127 까지 128개의 숫자를 일정한 문자에 대응시킨 국제적 규격 코드 (p.155)

Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자	Dec	Hex	문자
40	28	(	60	3C	<	80	50	P	100	64	d	120	78	x
41	29	)	61	3D	=	81	51	Q	101	65	e	121	79	y
42	2A	*	62	3E	>	82	52	R	102	66	f	122	7A	z
43	2B	+	63	3F	?	83	53	S	103	67	g	123	7B	{
44	2C	,	64	40	@	84	54	T	104	68	h	124	7C	|
45	2D	-	65	41	A	85	55	U	105	69	i	125	7D	}
46	2E	.	66	42	B	86	56	V	106	6A	j	126	7E	~
47	2F	/	67	43	C	87	57	W	107	6B	k	127	7F	DEL
48	30	0	68	44	D	88	58	X	108	6C	l			
49	31	1	69	45	E	89	59	Y	109	6D	m			
50	32	2	70	46	F	90	5A	Z	110	6E	n			
51	33	3	71	47	G	91	5B	[	111	6F	o			
52	34	4	72	48	H	92	5C	\	112	70	p			
53	35	5	73	49	I	93	5D	]	113	71	q			
54	36	6	74	4A	J	94	5E	^	114	72	r			
55	37	7	75	4B	K	95	5F	_	115	73	s			
56	38	8	76	4C	L	96	60	‘	116	74	t			
57	39	9	77	4D	M	97	61	a	117	75	u			
58	3A	:	78	4E	N	98	62	b	118	76	v			
59	3B	;	79	4F	O	99	63	c	119	77	w			

* 숫자 49는 문자 ‘1’에 대응

■ 예제 (ex16)

```
void main(void)
{    char c1;
     int  c2;  // char로 선언해도 동일함
     c1 = 'A'; // ASCII코드 65
     c2 = 65;
     printf("문자 출력 c1=%c\n", c1); // 문자 출력
     printf("숫자출력 c1 = %d\n", c1); // 문자로 초기화된 것을 숫자로 출력
     printf("문자 출력 c2=%c\n", c2);  // 숫자로 초기화된 것을 문자로 출력
     c1 = 49;
     printf("문자 출력 c1=%c\n", c1); // 숫자 49를 문자로 출력 (문자 1)
     printf("숫자 출력 c1 = %d\n", c1); // 숫자로 출력
     _getch();
}
```

■ 예제)키보드 자판 문자들의 아스키 코드 값 알아보는 프로그램 (ex17)

```
void main(void)
{
        char c;
        while (1)  // while 블록을 무한 반복
        {
                printf("아무 키나 누르세요. (e는 종료):");
                scanf_s("%c", &c, sizeof(c));
                if (c == 'e') break;  // 입력키가 e이면 while 블록 탈출(종료)
                printf("\n입력한 %c의 ASCII코드 값은 %d입니다.\n\n", c, c);
                fflush(stdin);    // 버퍼에 존재하는 입력(엔터키) 값 제거
        }
}
```

2) 문자형 변수

■ 형식 : char 문자명;

ex) char d; // 8 bit 크기로 메모리에 잡힘

■ 문자 변수 대입

```
char k;
k = 'd';      // d라는 문자를 변수 k에 대입
```

■ 문자 출력 : %c

```
char k = 'A';
printf("%c", k);
```

■ 문자 입력

```
char k;
scanf("%c", &k);     // 2010 version
scanf_s("%c", &k, sizeof(k)); // 2013 version
```

3) 제어 문자

■ 화면에 출력할 수는 없고 제어 목적으로 이용되는 문자

ex) 아스키 7에 해당하는 BEL은 소리는 내는 제어 문자

```
void main(void)
{
        char b = 7;  //bell 소리 아스키 코드번호
        printf("%c", b);
        //printf("%c",7);  // 같은 기능 (ASCII 7= b)
        _getch();
}
```

■ 이스케이프 시퀀스 : 특별한 기능의 제어문자를 \로 규정한 것. 반드시 printf()문의 " " 사이에 들어가야 한다.

제어 문자 이름	제어 문자 표기	값	의미
널문자	\0	0	문자열의 끝을 표시
경고(bell)	\a	7	"삐"하는 경고 벨소리 발생
백스페이스(backspace)	\b	8	커서를 현재의 위치에서 한 글자 뒤로 옮긴다.
수평탭(horizontal tab)	\t	9	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
줄바꿈(newline)	\n	10	커서를 다음 라인의 시작 위치로 옮긴다.
수직탭(vertical tab)	\v	11	설정되어 있는 다음 수직 탭 위치로 커서를 이동
폼피드(form feed)	\f	12	주로 프린터에서 강제적으로 다음 페이지로 넘길 때 사용된다.
캐 리 지 리 턴 (carriage return)	\r	13	커서를 현재 라인의 시작 위치로 옮긴다.
큰따옴표	\"	34	원래의 큰따옴표 자체
작은따옴표	\'	39	원래의 작은따옴표 자체
역슬래시(back slash)	\\	92	원래의 역슬래시 자체

■ 예제) (ex18_1)

```
void main(void)
{
        printf("\a"); // bell 소리, printf("%c",7); 과 동일
        _getch();
        printf("\tHello!"); // 출력이 한 tap 진행하여 실행
        _getch();
        printf("\r");    // 커서를 현재 라인의 맨 처음으로 이동
        _getch();
}
```

■ 이스케이프 시퀀스 이용 예제 (ex18_2)

```
void main(void)
{
  int id, pass;

  printf("아이디와 패스워드를 각각 4개의 숫자로 입력하세요.\n");
  printf("ID:____\b\b\b\b"); // _ 4개이므로 4번 커서 뒤로 이동
  scanf_s ("%d", &id);
  printf("PASS:____\b\b\b\b");
  scanf_s ("%d", &pass);
  printf("입력한 아이디 : %d\t패스워드 : %d", id, pass );
  getch();
}
```

■ char 형은 별다른 형 변환 없이 정수형으로 계산이나 출력 가능 (ex19)

```
void main(void)
{
  char code = 'A'; // A -> 65

  printf("%d %d %d\n", code, code + 1, code + 2);
  printf("%c %c %c\n", code, code + 1, code + 2);
  printf("%c \n", '6' - '1');    // 54 - 49 = 5
  _getch();
}
```

H.W 2 : 4장 연습문제 (programming)00, 00번

Chap. 5 수식과 연산자

1. 수식과 연산자

1) 개념

■ 수식

```
i = 5 * 8;  // 수식
```

- 5, 8 : 피연산자 (operand)
- * : 연산자 (operator)

■ 연산자의 분류

연산자의 분류	연산자	의미
대입	=	오른쪽을 왼쪽에 대입
산술	+ - * / %	사칙연산과 나머지 연산
부호	+ -	
증감	++ --	증가, 감소 연산
관계	> < == != >= <=	오른쪽과 왼쪽을 비교
논리	&& || !	논리적인 AND, OR
조건	?	조건에 따라 선택
콤마	,	피연산자들을 순차적으로 실행
비트 단위 연산자	& | ^ ~ << >>	비트별 AND, OR, XOR, 이동, 반전
sizeof 연산자	sizeof	자료형이나 변수의 크기를 바이트 단위로 반환
형변환	(type)	변수나 상수의 자료형을 변환
포인터 연산자	* & []	주소계산, 포인터가 가리키는 곳의 내용 추출
구조체 연산자	. ->	구조체의 멤버 참조

2) 산술 연산자

■ 종류

연산자	기호	의미
덧셈	x+y	x와 y를 더한다
뺄셈	x-y	x에서 y를 뺀다.
곱셈	x*y	x와 y를 곱한다.
나눗셈	x/y	x를 y로 나눈다.
나머지	x%y	x를 y로 나눌 때의 나머지값

■ 예제) ex20.c

```
void main(void)
{
        int x, y;

        printf("두개의 정수 입력 : ");
        scanf_s("%d %d", &x, &y);   // x=3, y=2 입력
        //printf("%d + %d = %d\n", x, y, x + y);
        //printf("%d - %d = %d\n", x, y, x - y);
        //printf("%d * %d = %d\n", x, y, x*y);
        printf("%d / %d = %d\n", x, y, x / y); // 결과 값 주의
        //printf("%d %% %d = %d\n", x, y, x%y); //%% : % 자체 출력
        _getch();
}
```

* NOTE : 정수형 끼리 나눗셈은 소수점 이하는 버려짐(int 형으로 바뀜)

■ 수식 연산자 사용 시 주의사항

$y = mx + b$ → `y = m*x + b`

$y = ax^2 + bx + c$ → `y = a*x*x + b*x +c`

$m = \frac{x + y + x}{3}$ → `m = (x + y + z) / 3`

■ 예제) 실수 형 입력 및 계산 (ex21.c)

```
void main(void)
{       float x, y;
        printf("두개의 실수 입력 :");
        scanf_s("%f %f", &x, &y);
        //printf("%f + %f = %f\n", x, y, x + y);
        //printf("%f - %f = %f\n", x, y, x - y);
        //printf("%f * %f = %f\n", x, y, x*y);
        printf("%f / %f = %f\n", x, y, x / y);
        _getch();
}
```

* Note : 실수 입력 시에 정수로 입력해도 실수로 받아들인다.

■ 나머지 연산자 (%)

- 첫 번째 피연산자를 두 번째 피연산자로 나누었을 경우의 나머지 계산

  10 % 6 = 4

- 짝/홀수 구분에 이용

  (x % 2) 가 0이면 짝수, 1이면 홀수

- 어떤 수의 배수 판단

  (x % 7)이 0이면 7의 배수

■ 예제) 초를 입력하면 그것을 분 및 초로 환산 (ex22.c)

```
#define S_P_M 60
void main(void)
{       int input, time, minute, second;
        printf("초단위의 시간을 입력하시오.(32억초 이하) : ");
        scanf_s("%d", &input);                  // 초단위의 시간을 읽는다.
        minute = input / S_P_M; //분 계산 (몫 : 정수계산)
        second = input % S_P_M; // 초 계산(나머지)
        printf("%d초는 %d분 %d초입니다. \n", input, minute, second);
        _getch();
}
```

■ 증감 연산자 : ++, --

연산자	C 언어 수식	수학	비고
++	a++;	a=a+1	a 값을 사용한 다음 a를 1 증가
	++a;	a=a+1	a를 먼저 1 증가시킨 다음 a값을 이용
--	a--;	a=a-1	a 값을 사용한 다음 a를 1 감소
	--a;	a=a-1	a를 먼저 1 감소시킨 다음 a값을 이용

■ 예제) ex23.c

```
void main(void)
{
    int i=0;
    printf("i의 값은 %d 입니다.\n", i++);     //0
    printf("i = %d 입니다.\n", i);             //1
    _getch();
    i=0;
    printf("i의 값은 %d 입니다.\n", ++i);     //1
    printf("i = %d 입니다.\n", i);             //1
    _getch();
}
```

■ 대입연산

- x = 1; // x에 1을 대입

  x = x + 10; // x =11이 새롭게 대입됨

- y = 10 + (x = 2 + 7); // x에는 9가 대입되고 y에는 19가 대입됨.

- x = y = 3; // x, y 에 각각 3이 대입됨

◒ 예제) ex24.c

```
void main(void)
{
        int x, y;

        x = 1;
        printf("수식 y=x+1 : y = %d, x=%d \n", y=x+1, x);
        printf("수식 y=10+(x=2+7) : y=%d, x = %d \n", y=10+(x=2+7), x);
        printf("수식 y=x=3 : y=%d, x=%d \n", y=x=3, x);
        _getch();
}
```

■ 복합 대입연산

연산자	C 언어 수식	수학적 의미
+=	a +=10	a = a+10
-=	a -=8	a = a-8
*=	a *=45	a = a*45
/=	x /=7	x = x/7
%=	x %=11	x = x%11

◒ 예제) ex25.c

```
void main(void)
{  int x = 10, y = 10, z = 33;
   x += 1;            // x = x +1
   y *= 2;            // y = y*2
   z %= x + y;        // z = z % (x+y), (33 % 31)
   printf("x = %d  y = %d  z = %d \n", x, y, z);
   _getch();

}
```

2. 형 변환

1) 자동적인 형 변환

■ 컴파일러에 의해 자동적으로 형이 변환

■ 대입연산 시 자동 변환 : 대입 시에 오른 쪽 수는 자동적으로 왼쪽 변수의 형으로 변환

- 올림변환

```
double  f = 10; // 10은 double 형으로 자동 변환
```

- 내림변환

```
int k = 3.14159;  // k에는 3이 저장됨
```

■ char형이나 short형 값이 수식에 쓰이면 자동적으로 int형으로 승격

■ 수식 연산 시의 자동 형 변환 : 두 형 중 등급이 높은 형으로 변환

```
int j = 10;
float k = 1.2;
printf("%f", j + k);  // j는 float 형으로 바뀌어 계산됨
```

◒ 예제) ex26.c

```
void main(void)
{       unsigned char c;
        int i;
        float f;
        c = 256;              // char 형으로 내림 변환 (0)
        i = 1.23456 + 10;     // int 형으로 내림 변환
        f = 10 + 20;          // float형으로 올림 변환
        printf("c = %d, i = %d, f = %f \n", c, i, f);
        i=10;
        f=1.2;
        printf("%f", i + f);  // 등급이 높은 float 형으로 변환
        _getch();
}
```

2) 명시적 형 변환

■ 프로그래머가 강제로 형을 변환

■ 형 변환을 하고자 하는 변수나 수 앞에 (형)을 붙임.
단 그 문장에서만 형이 변환되며 그 이 후는 다시 원래 형이 됨

ex) (float)i

◒ 예제) ex27.c

```
void main(void)
{   int j=5;
    double f;
    f = j/4;  // 우변이 먼저 계산 (정수형으로 계산)
    printf("%f\n", f); //1.0
    f = (float)j/4;     // 명시적 형 변환
    printf("%f\n", f); //1.25
    j =1.2 + 1.81;    // 왼쪽 변수 형으로 변환
    printf("%d\n", j); //3
    j =(int)1.2 + (int)1.8;  // 우변이 명시적 변환되어 먼저 계산
    printf("%d\n", j); //2
    f =1.2 + (int)1.8;
    printf("%f\n", f); //2.2
    _getch();
}
```

3. 기타 연산자

1) 관계연산자

- 두 개의 피 연산자를 비교한 연산자
- 결과는 참(=1) 아니면 거짓(=0)이 됨 (0 또는 1의 값만 가짐)
- 종류

연산자	의미
x == y	x와 y가 같은가?
x != y	x와 y가 다른가?
x > y	x가 y보다 큰가?
x < y	x가 y보다 작은가?
x >= y	x가 y보다 크거나 같은가?
x <= y	x가 y보다 작거나 같은가?

- 예제: ex28.c

```
void main(void)
{    int x, y;
     printf("두개의 정수를 입력하시오: ");
     scanf_s("%d%d", &x, &y); // x=3, y=4 대입
      printf("x == y의 결과 값: %d\n", x == y);        // 거짓
     printf("x != y의 결과 값: %d\n", x != y);  // 참
     printf("x > y의 결과 값: %d\n", x > y);          // 거짓
     printf("x < y의 결과 값: %d\n", x < y);          // 참
     printf("x >= y의 결과 값: %d\n", x >= y); // 거짓
     printf("x <= y의 결과 값: %d\n", x <= y); // 참
     _getch();

}
```

*** 주의
1. '=' 은 대입, '=='은 비교연산자임
2. x=y 는 y의 값을 x에 대입, x==y의 값은 0 아니면 1
3. 2 < x < 5와 같이 쓰면 오류. (2 < x) && (x <5)와 같이 써야 함

- 예제) ex29.c

```
void main(void)
{    int x=1, y=2, z=4;

     printf("x<z<y의 결과 값 : %d\n", x<z<y); //거짓이지만 1이 출력
     printf("x<z<y의 결과 값 : %d\n", (x<z) && (z<y));//&& 논리 AND

     _getch();

}
```

2) 논리 연산자

■ 여러 개의 조건을 논리적으로 따지는 연산자 (0 또는 1의 값만 return)

연산자	논리	값(의미)
x && y	논리 AND 연산	x와 y가 모두 참일 때만 1 (참), 그 이외는 0 (거짓)
x || y	논리 OR 연산	x와 y중 하나만 참이어도 1 (참), 그 이외는 0 (거짓)
!x	논리 NOT 연산	x가 참이면 0 (거짓), x가 거짓 이면 1 (참)

NOTE] C언어에서는 0 이외의 값은 모두 참으로 간주 (음수도 참)

■ 다양한 경우의 논리연산

- (2<3) && (2<5)의 값 : 1 (참)
- (3<2) || (2<5)의 값 : 1 (참)
- (100 && 1)의 값 : 1 (참)
- (0 || 23)의 값 : 1 (참)
- (-2 && 2)의 값 : 1 (참)

■ 논리 식 표현방법

- x = 1, 2, 3, 중의 하나인가? : (x==1) || (x==2) || (x==3)
- x가 30이상 70 미만 : (x >30) && (x<70)
- x가 1도 아니고 2도 아니다. : (x != 1) && (x !=2)

NOTE] 논리연산자의 우선순위 : 괄호 안>NOT>AND>OR

- x=2; y=3;
  z= !x || y && 4 ; // !x(=0)를 먼저 계산, 그리고 y && 4 (=1)를 계산, 최후로 0 || 1을 계산 함

■ 예제) ex30.c

```
void main(void)
{ int a, b, c, d;
  a=0;    b=3;
  printf("%d && %d 의 연산값은 %d\n", a, b, a && b); //0
  printf("%d || %d 의 연산값은 %d\n", a, b, a || b); //1
  printf("!%d 의 연산값은 %d\n", a, !a); // 1
  _getch();
  a=2; b=3; c=4; d=5;
  printf("둘 다 참인 AND 연산 결과는 %d\n", (a<b) && (c<d));  //1
  _getch();
  printf("하나가 거짓인 OR 연산 결과는 %d\n", (a>b) || (c<d));
  _getch();
```

```
    a=2; b=3;
    c = !a || b && 4;
    printf(" (!a || b && 4)의 결과는 %d\n", c); //1
    _getch();
}
```

■ 예제 :어떤 숫자가 3과 5의 공통 배수인지 판단하는 프로그램(ex31.c)

```
void main(void)
{       int a, res;
        printf("정수 입력 :");
        scanf_s("%d", &a);
        res = ((a % 3)==0) && ((a%5)==0); // 0:거짓 1 : 참
        printf(" 결과는 %d", res);
        _getch();
}
```

■ 예제 :어떤 숫자가 4의 배수이지만 10의 배수는 아님을 판단 (ex32.c)

```
void main(void)
{ int a, res;
  printf("정수 입력 :");
  scanf_s("%d", &a);
  res = ((a % 4)==0) && ((a % 10) !=0);
  printf(" 결과는 %d", res);
  _getch();
}
```

3) 조건 연산자

■ 3항 연산자로서 다음과 같다. (x>y)? x : y;

- k = (x>y)? x : y;
- x가 y보다 크면 x를 그렇지 않으면 y를 k에 대입

■ 사용 예 (ex33)

```
void main(void)
{   int a, x, y;
    x = -1;
    a = (x>0)? x : -x;   // x의 절대 값 계산
    printf("x의 절대 값은 %d\n", a);
    x = 1, y =2;
    a = (x>=y) ? x : y; // 둘 중 큰 수 찾기
    printf("%d과 %d 중 큰 수는 %d\n", x, y, a);
    a = -3;
    (a>=20)? printf("%d은 양수", a) : printf("%d은 음수", a);   //a가 0보다 크면 양
                                                                수, 그렇지 않으면 음수(0포함)를 출력
    _getch();
}
```

4) 콤마 연산자

■ 콤마로 연결된 문장이나 수식은 순차적으로 수행된다.

```
int i, j, k;
i=10, j=20, k=30 ;
```

4. Bit 연산자

1) 진수와 bit

■ 10진수, 16진수와 2진수 관계 (대소문자 구별 없음)

10진수	2진수(4 bit)	16진수
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

■ 2 진수와 16진수 상호변환

ex) 00110100_2 를 16 진수로 변환

- 0011 0100 ⇒ 2진수를 4 bit 단위로 분리
- 3 4 ⇒ 4 bit 2진수를 각각 16진수로 표현
- $00110100_2 \Rightarrow 34_{16}$: 표현된 16진수를 조합

ex) $3f8_{16}$ 을 2진수로 변환

- 3 f 8
- 0011 1111 1000 ⇒ 16진수의 각 자리를 4 bit의 2진수로 변환
- $3f8_{16} \Rightarrow 001111111000$: 변환된 4 bit의 2진수를 조합

2) bit 연산자

■ bit 논리 연산

논리연산	AND	OR	NOT
첫 번째 수	00110101	00110101	00110101
두 번째 수	10101100	10101100	
결과	00100100	10111101	11001010

0	0	1	1	0	1	0	1

1	0	1	0	1	1	0	0

0	0	1	0	0	1	0	0

* NOTE

논리	2진수	전압	프로그래밍
참	1	5V	0제외 모든 수
거짓	0	0V	0

■ bit 연산자 종류

연산	연산자	C 언어 수식	의미
비트연산	<<	a << b	a를 b bit 만큼 왼쪽으로 shift
	>>	a >> b	a를 b 비트만큼 오른쪽으로 shift
	&	a & b	a와 b를 비트 단위로 AND
	|	a | b	a와 b를 비트 단위로 OR
	~	~a	a의 각 비트 값을 NOT

■ 16진수 대입법 및 출력형식

- 16진수 대입

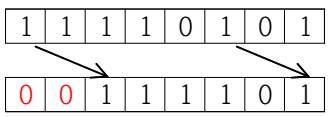
int a = 0x3f8; ⇒ a라는 정수형 변수에 16진수로 3f8을 대입

- 16진수로 출력

printf(" %x ", a); ⇒ a에 대입된 수를 16진수로 모니터에 출력

■ shift 연산자

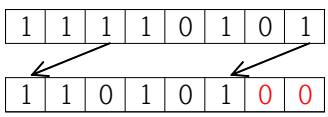
ex) a=0xf5; ⇒ a>>2;



⇐ 각 bit 값을 2 bit 만큼 오른쪽으로 이동, 빈자리는 0으로 채워짐

(1 bit 오른쪽 shift는 2로 나누는 효과)

ex) a=0xf5; ⇒ a<<2;



⇐ 각 bit 값을 2 bit 만큼 오른쪽으로 이동, 빈자리는 0으로 채워짐

(1 bit 왼쪽 shift는 2로 곱하는 효과)

■ 예제10) ex34

```
void main(void)
{
 int x=0x7, y=0x3;  // 16진수로 대입
 printf("%x>>1의 결과=%x\n", x, x>>1); // 0111 >> 1 = 0011 (3)
 printf("%x>>2의 결과=%x\n\n", x, x>>2); // 0111 >>2 = 0001 (1)
 _getch();
 x=0x1;
 printf("%x<<1의 결과=%x\n",x, x<<1); // 0001 <<1 = 0010 (2)
 printf("%x<<2의 결과=%x\n",x, x<<2); // 0001 << 2 =0100 (4)
 printf("~%x의 결과=%x\n\n", x, ~x);    // 0000...0000 0001 -> 1111...1111
                                        // 1110 (fffffffe)
 _getch();
 x=0x5;
 printf("%x & %x의 결과=%x\n", x, y, x&y); //0101 & 0011=0001 (1)
 printf("%x | %x의 결과=%x\n\n", x, y, x|y); // 0101 | 0011 =0111 (7)
 _getch();
}
```

■ 응용 예

- 8 bit의 입력 register에서 3번째 채널 입력 값이 ON 인지를 check
- 3번째 채널입력이 ON⇒입력register 변수의 3번째 bit가 1인지를 check

0	1	0	1	0	1	1	0	⇐ register (8 bit)
0	0	0	0	0	1	0	0	⇐ 0x04 (3번째 bit만 1)
								AND
0	0	0	0	0	1	0	0	⇐ 결과 : 0x04(1) 또는 0x00(0)

- 즉 3번째 bit만 1인 0x04를 AND 해서 그 결과가 0x04와 같으면 그 채널 값은 1 그렇지 않으면 0

■ 예제) ex35

```
#define CHECK 0x04
void main(void)
{  unsigned char reg, result;
   reg=0xff;      // 0xff = 11111111, 0xfb = 11111011
   result = reg & CHECK; // bit and
    (result == CHECK)? printf("3번째 입력채널은 ON\n") : printf("3번째 입력
                        채널은 OFF.\n");
   _getch();
}
```

H.W : Ch.5 프로그램 연습문제 00번 및 00 번 프로그램

* 타 학과나 선배가 한 프로그램은 적발하여 0점 처리

Chap. 6 조 건 문

1. 제어문이란?

■ 기본적으로 C언어는 프로그램 list의 코드순서에 따라 차례대로 수행됨

■ 그러나 조건에 따라 실행을 다르게 하거나 또는 반복할 필요가 있음

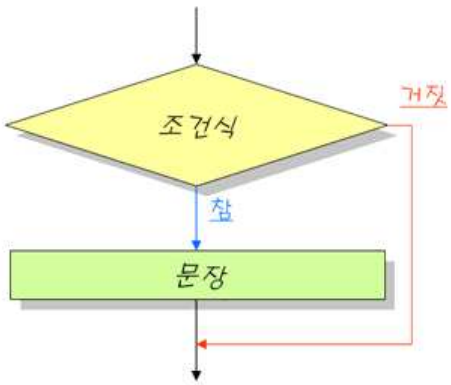
■ 제어문 : 조건문 + 반복문

* 조건문 : 점수가 60점 이상이면 합격, 그렇지 않으면 불합격
* 반복문 : 1에서 100까지 정수를 더하는 경우

2. if 문

■ 정의

```
if (조건식)          // 조건이 참(값이 0이 아닐 때)일 때만
        문장;        // 문장을 실행
```



■ 예

- 단일문장

```
if(n>0)
 printf("양수입니다.");   // 이 문장은 조건이 참(n>0)일 때만 실행
다음문장 ;                 // if와 관계없는 문장
```

- 복합문장 : 조건에 따라 수행해야할 문장이 여러 개 일 때

```
if (score>=80)
{    // 조건이 참이면 { } 내의 모든 문장을 실행
     printf("합격입니다.\n");
     printf("장학금도 받을 수 있습니다.");
}
다음문장 ;        // {}밖의 문장은 if와 무관한 문장
```

- 조건문 대신 숫자 (또는 변수)가 if ()의 조건식에 들어 갈 수 있음

```
if (n)  // n이 0이 아닌 수이면 참
     printf("0이 아닙니다.);
```

- 복잡한 조건식도 가능

```
     if((score>=80) && (score<90))
          printf("B 학점입니다.");
```

■ 예제36)

```
void main(viod)
{  int number;
   printf("정수입력(음수가능) :");
   scanf_s("%d",&number);
   if(number<0)
   {
          printf("음수입니다.");
          number = -number;
   }
   printf("입력한 수의 절대 값은 : %d", number);

   _getch();
}
```

3. If – else 문

■ 정의 : if 와 else 중 하나를 실행

```
if (조건식)
{       // 조건이 참 (또는 0이 아닌 수)이면 이 블록 내 문장 실행,
        //  단일 문장이면 { } 불필요
        문장들;
}
```

```
else     // if의 조건이 거짓 (또는 0)이면 이 블록실행,
{        // 단일 문장이면 { } 불필요
        문장들;
}
```

■ 예제37)

```
void main(void)
{
      int score;
      printf("점수 :");
      scanf_s("%d",&score);
      if (score>=80)      // score 가 80 이상이면 if 내 문장 수행
      {
            printf("합격입니다.\n");
            printf("장학금도 받을 수 있습니다.\n");
      }
      else      // score 가 80 미만이면 else 내 문장 수행
      {
            printf("불합격입니다.\n");
            printf("열심히 해서 다시 도전하세요.\n");
      }
      _getch();
}
```

■ 예제 38 : 문자 조건문

```
void main(void)
{
      char c;
      char sch[30]="제주대학교";
      char name[20]="홍길동";

      printf("문자 입력(s 또는 n) : ");
      scanf_s("%c", &c, sizeof(c));

      if( c=='s')
            printf("학교는 %s입니다.\n", sch);

      else    // 그렇지 않으면 c=='n'
            printf("이름은 %s 입니다.\n", name);
      _getch();
}
```

◒ 문제점 : s도 n도 아닌 문자 입력 시 무조건 else 실행

4. 다중 if-else 문

■ 중첩 if (다중 if 가능)

```
if (score<80)
        printf("C 학점입니다.");
if ((score>=80) && (score<90))
        printf("B 학점입니다.");
-------------------------------------------------
if (score>=80) // 바로 위 문장과 동일
        if (score<90)
                printf("B 학점입니다.");
```

■ {}로 구분되지 않는 한 else는 가장 가까운 if와 매치된다.

```
if( score >= 80 )
        if( score >= 90 )
                printf("당신의 학점은 A입니다. \n");
        else
                printf("당신의 학점은 B입니다. \n");
-------------------------------------------------------------
if( score >= 80 )
        if( score >= 90 )
                printf("당신의 학점은 A입니다. \n");

else  printf("당신의 학점은 C입니다. \n"); // 80이상일때만 유효
```

◒ 그러나 혼동을 피하기 위해 {}를 사용하는 것이 좋다.

■ 예제 39

```
void main(void)
{       int score;

        printf("성적 :");
        scanf_s("%d", &score);

        if( score >= 80 )
        {
                if( score >= 90 )
                        printf("당신의 학점은 A입니다. \n");
                else
                        printf("당신의 학점은 B입니다. \n");
        }
```

```
        else
        {
                if( score >= 70 )
                        printf("당신의 학점은 C입니다. \n");
                else
                        printf("당신의 학점은 D 또는 F입니다. \n");
        }
        _getch();
    }
```

■ if - else if - else 문

```
if(조건1)
{       문장1;
}
else if(조건2)
{       문장2;
}
else if(조건3)
{       문장3;
}
.....
else // 위 조건들이 전부 거짓이면
{       문장n;
}
```

⇐ 조건이 참이 되는 문장(들)이 수행되며 아무 조건에도 맞지 않으면 else 내의 문장 n이 수행된다.

■ 예제 40) 문자 분류예제 (알파벳과 숫자 또는 기타문자로 분류)

```
void main(void)
{   char c;
    printf("문자를 입력하시오: ");
    scanf_s("%c", &c, sizeof(c));
    if( c >= 'A' && c <= 'Z' ) //아스키코드 값 범위조건
            printf("%c는 대문자입니다. \n", c);
    else if( c >= 'a' && c <= 'z' )
            printf("%c는 소문자입니다. \n", c);
```

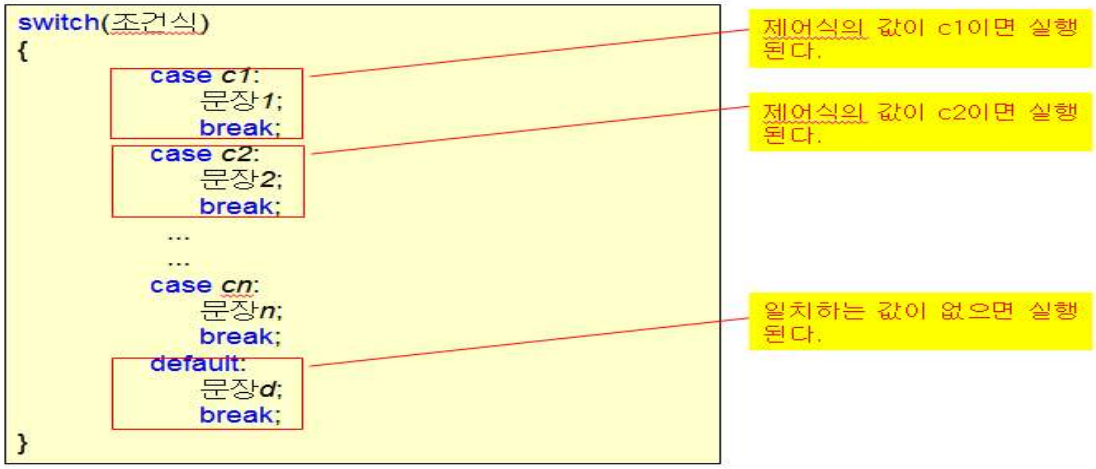
```
    else if( c >= '0' && c <= '9' )
            printf("%c는 숫자입니다. \n", c);
    else printf("%c는 기타문자입니다. \n", c);
    _getch();
}
```

■ 예제 41) 계산기 예제

```
void main(void)
{   char op;
    int x, y;

    printf("수식 입력(예: 2 + 5) :");
    scanf_s("%d %c %d", &x, &op, sizeof(op), &y);
    if( op == '+' )
            printf("결과 : %d", x+y);
    else if( op == '-' )
            printf("결과 :  %d", x-y);
    else if( op == '*' )
            printf("결과 :  %d", x*y);
    else if( op == '/' )
            printf("결과 :  %.3f", (float)x/y); //실수로 변환
    else if( op == '%' )
            printf("결과 :  %d", x%y);
    else
            printf("지원되지 않는 연산자입니다. ");
    getch();
}
```

5. switch 문

■ 정의 : 여러 가지 경우 중 하나를 선택하는 문

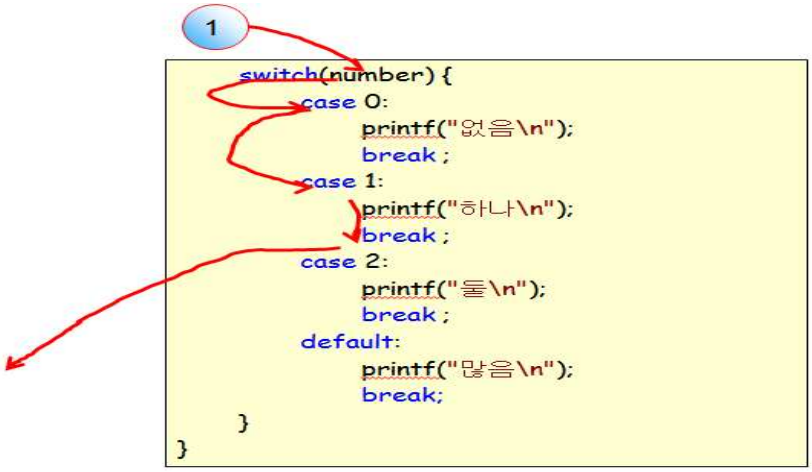


- 조건식의 값과 일치하는 case를 실행. 만약 조건식과 일치하는 case가 없으면 default를 실행
- 각 case 는 break로 끝나야 한다. ⇒ case는 반드시 break를 만날 때만 탈출

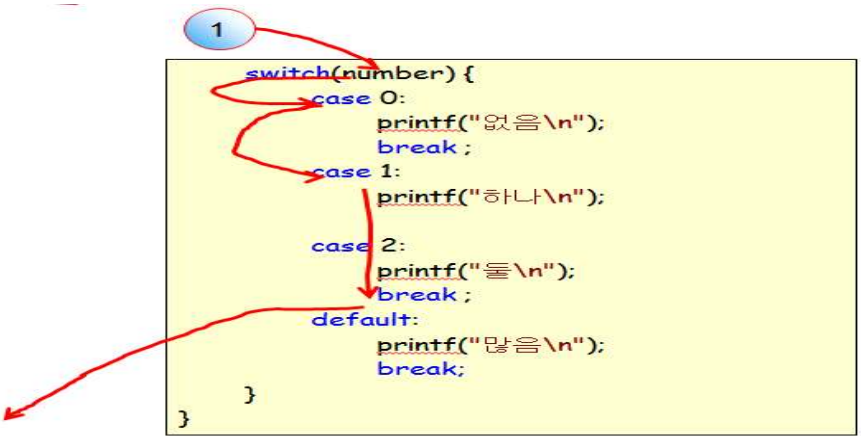
■ 사용 예 (ex42)

```
void main (void)
{       int number;
        printf("정수를 입력하시오(음수 제외):");
        scanf_s("%d", &number);

        switch(number) {
                case 0:
                        printf("없음\n");
                        break ;
                case 1:
                        printf("하나\n");
                        break ;
                case 2:
                        printf("둘\n");
                        break ;
                default:
                        printf("셋 이상\n");
                        break;
         }
        _getch();
}
```

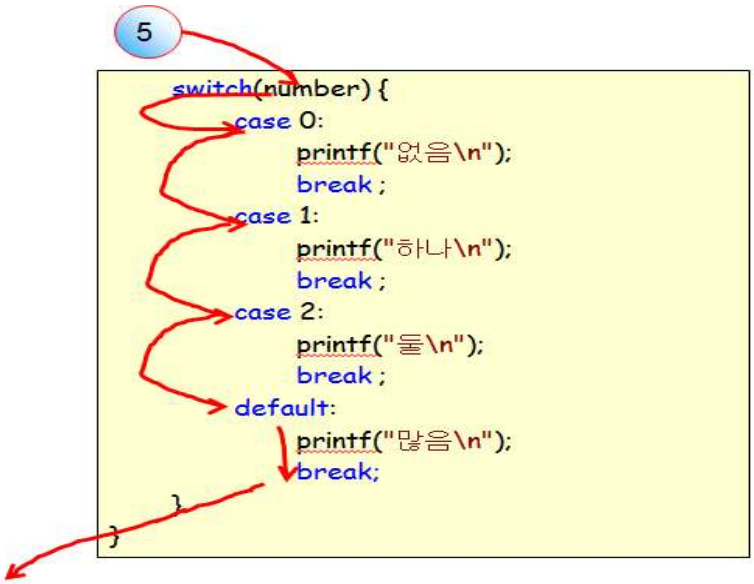
- 1을 입력한 경우



- break가 생략된 경우



- default 역할

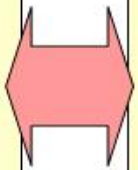


■ 예제 44) 키보드로 문자하나를 입력받아 s 이면 학교를, n 이면 이름을, a 이면 주소를, 그 이외 문자이면 "잘못 입력하였습니다."를 출력하는 프로그램

```
void main(void)
{       char c;
        char sch[30]="제주대학교"; // 문자열 초기화
        char name[20]="홍길동";
        char add[30]="아라동 1번지";
        printf("문자 입력(s, n 또는 a) : ");
        scanf_s("%c", &c, sizeof(c));  // 문자하나를 입력
        switch(c){                    // c가 조건에 해당
                case 's' :            // c가 문자 s 인 경우, 문자는 ' '가 필요
                        printf("학교는 %s", sch);
                        break;
                case 'n' :                  // c가 n 인 경우
                        printf("이름은 %s", name);
                        break;
                case 'a' :                  // c가 a 인 경우
                        printf("주소는 %s", add);
                        break;
                default :                   // 그 외의  경우
                        printf("잘못 입력하였습니다.");
                        break;
        }
          _getch();
}
```

■ switch 문과 if – else 문

```
int main(void)
{
  int number;
  scanf("%d",&number);

  if( number == 0 )
     printf("없음\n");
  else if( number == 1 )
     printf("하나\n");
  else if( number == 2 )
     printf("둘\n");
  else
     printf("많음\n");
}
```



```
switch(number)
     {
          case 0:
               printf("없음\n");
               break;
          case 1:
               printf("하나\n");
               break;
          case 2:
               printf("둘\n");
               break;
          default:
               printf("많음\n");
               break;
     }
```

■ switch 문에서 주의할 점 (if -else 문과의 차이 점)

1)case에는 반드시 정수나 문자만 가능(변수나 실수, 문자열은 불가능)

2)수식은 가능하나 변수가 포함되거나 결과가 정수가 아니면 불가.

```
switch(number)
{
        case x:                         // 변수는 사용할 수 없다.
                printf("x와 일치합니다. ");
                break;
        case (x+2):                     // 변수가 들어간 수식은 사용할 수 없다.
                printf("수식과 일치합니다. ");
                break;
        case 0.001:                     // 실수는 사용할 수 없다.
                printf("실수");
                break;
        case "001":                     // 문자열은 사용할 수 없다.
                printf("문자열");
                break;
}
```

■ 예제 43) 각 달의 일수 출력

```
void main(void)
{
    int month, day;
    printf("달을 입력 : ");
    scanf_s("%d", &month)

    switch(month)
    {
        case 2:
            days = 28; // 윤년이 아닌 해를 고려
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            days = 30;
            break;
        default:
            days = 31;
            break;
    }
    printf("%d월의 일수는 %d입니다. \n", month, days);
    _getch();
}
```

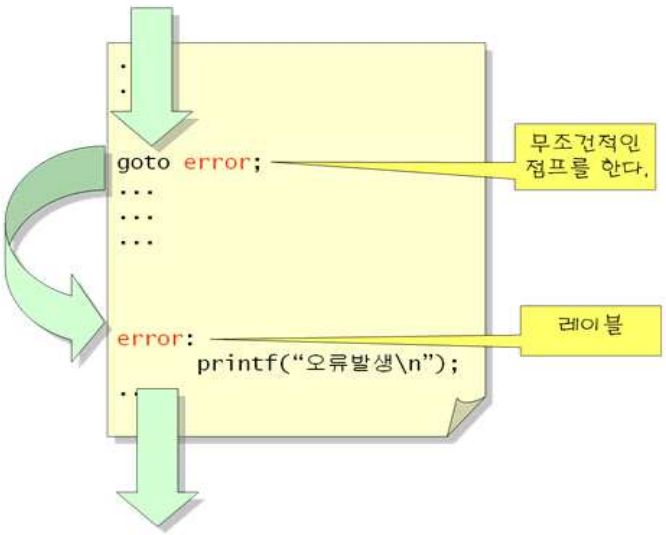
예비 test) 년도를 입력하면 윤년인지 아닌지를 판단하는 프로그램을 조건문으로 작성

1) 윤년 조건 :
   - 연도가 4로 나누어서 떨어진다.
   - 100으로 나누어서 떨어지는 해는 윤년이 아니다.
   - 400으로 나누어서 떨어지면 무조건 윤년이다.
2) 출력 형식 : "000년은 윤년입니다." 또는 "000년은 윤년이 아닙니다."로 출력

6. goto 문

■ 정의

```
goto 레이블;      // 무조건 레이블 위치로 강제 점프
  문장들....;
레이블:
```



■ 예제 : 절대 값 프로그램

```
void main(viod)
{  int number;
   printf("정수입력 :");
   scanf_s("%d",&number);
   if(number>=0)        // 양수이면 바로 next로 점프
         goto next;
   number = -number;
next:                   // 레이블, 레이블 뒤에는 반드시 : 이 옴
   printf("입력 수의 절대 값은 : %d", number);
   _getch();
}
```

■ 예제 44) 구구단 프로그램(2단)

```
void main(void)
{      int i = 1;
loop: // goto에 의해서만 작동, 그 이외에는 의미가 없음
       printf("%d x %d = %d \n", 2, i, 2 * i);
       i++;
       if( i == 10 ) goto end;
       goto loop;
end: // goto에 의해서만 작동, 그 이외에는 의미가 없음
       _getch();
}
```

■ 예제 47) 1+2+3+... +99+100을 구하는 문제

```
void main(void)
{      int i = 1, sum=0;
next:  sum += i;
       i++;
       if (i>100)
             goto end;
       goto next;
end:   printf("1부터 100까지 합 =%d", sum);
       _getch();
}
```

* 참고 (p.231)
scanf_s()를 이용하여 키보드로 문자를 연속으로 입력할 때는 반드시 fflush(stdin);을 중간에 넣어야 함 (숫자는 상관없음)

```
void main(void)
{       char a;
        int b;
        printf("문자입력:");
        scanf_s("%c", &a);
        printf("%c 가 입력되었음\n", a);
        //fflush(stdin);  //buffer에 있는 문자를 모두 삭제
        printf("문자입력:");
        scanf_s("%c", &a);
        printf("%c 가 입력되었음\n", a);
        _getch();
}
```

■ 예제) 제곱근 구하는 방법 (5의 제곱근을 구하여 출력)

```
#include <math.h> //각종 수학식이 정의된 헤더파일
void main(void)
{       float a=5, b;
        b = sqrt(a);  //sqrt(수 또는 변수) : 제곱근 구하는 함수
        printf("%f의 제곱근은 %f", a, b);
        _getch();
}
```

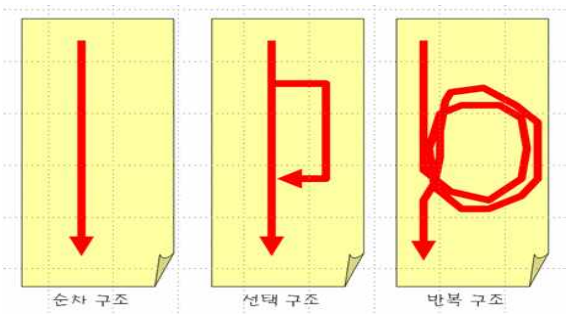
[H.W3]
- 연습문제 00, 00번 (p,259)

Chap. 7 반 복 문

1. 반복문이란?

■ 같은 처리과정을 되풀이 하게 하는 문

■ 프로그램 제어방법



■ 필요성

```
#include <stdio.h>

int main(void)
{
      int meter;

      meter = 0 * 1609;
      printf("0 마일은 %d미터입니다\n", meter);

      meter = 1 * 1609;
      printf("1 마일은 %d미터입니다\n", meter);

      meter = 2 * 1609;
      printf("2 마일은 %d미터입니다\n", meter);
      return 0;
}
```

동일과정

■ 반복문 종류 : for, while

2. while 루프

■ 형식 : 조건이 참(0이외의 수)인 경우 계속해서 while내의 문장을 반복하고 거짓이 되면 탈출하여 다음문장을 수행

```
while(조건식 또는 숫자, 변수)
{
        while 내 문장들;
}
다음 문장;
```

■ 예

```
void main(void)
{       int meter;
        int i = 1;
        while(i < 4) // i가 4보다 작으면 while문 수행
        {
                meter = i * 1609;
                printf("%d 마일은 %d 미터입니다\n", i, meter);
                i++;
        }
        _getch();
}
```

■ 예제 45) 구구단

```
void main(void)
{       int n, j=1;
        printf("원하는 구구단의 단 :");
        scanf_s("%d", &n);
        while(j<=9)
        {       printf("%d x %d = %d\n", n, j, n*j);
                j++;
        }
        _getch();
}
```

■ 예제46) 입력 개수가 정해지지 않은 점수평균 (음수가 입력되면 중단)

```
void main(void)
{
        int grade = 0, sum = 0, n = 0; //grade의 초기화 이유는?

        while (grade >= 0)              // 음수이면 while 탈출
        {
                printf("점수 :");
                scanf_s("%d", &grade); // 먼저 점수를 입력 받고
                sum += grade;          // 점수를 합산
                n++;                   // 입력된 점수 개수 증가
        }
        n--; // 음수 점수입력으로 증가된  점수 개수 하나 감소
        sum -= grade; // 점수 합에서 음수 점수 뺌

       if (n>=1) // n=0이면 점수가 입력되지 않았음
                printf("점수 평균 : %.2f\n", (float)(sum) / n);
       else
                printf("점수가 입력되지 않았습니다.");
        _getch();
}
```

■ 최댓값 찾기 (입력 중단은 Ctrl+z를 사용하는 방법)

```
void main(void)
{       int num, max = 0;
        printf("양의 정수를 입력하시오(종료는 Ctrl+z\n).");

        while (scanf_s("%d", &num) != EOF) // Ctrl+z 는 EOF(-1)
        {               // 정상적인 입력이 이루어지면 scanf_s의 값은 1
                if (num > max)
                        max = num;
        }
        printf("최댓값은 %d", max);
        _getch();
}
```

*scanf_s()는 입력된 데이터 개수를 return한다.

■ 다중 while() 문

```
while(조건식 또는 숫자, 변수)
{
        문장들;
        while (조건식 또는 숫자, 변수)
        {
                문장들;
        }
}
다음 문장;
```

■ 예제 46_3) *로 정사각형 모양 만들기

```
void main(void)
{
  int i=0, j;

  while (i <10)
  {
        j = 0; //j를 0으로 초기화
        while (j < 10)
        {
              printf("* ");
              j++;
        }
        printf("\n");
        i++;
  }
  _getch();
}
```

■ 무한반복 while 문

① 무한 반복

```
int i = 1;
while(i < 10)
{
      printf("반복중입니다\n");
      i--;
}
```

② 무한반복

```
int i = 0;
while(i < 3)
        printf("반복중입니다\n");
        i++;
```

③ 무한반복

```
while (1)
{
        문장;
        문장;
}
```

④ 숫자가 건너뛰는 경우 (자주 저지르는 실수)

```
int j=0;
while (j !=7) //2씩 증가하므로 7이 되지 않음
        j +=2;
```

■ 예제 : n번 제곱 값 계산프로그램

```
void main(void)
{       int j, x. n, mul=1;

        printf("거듭제곱할 수 입력:");
        scanf_s("%d", &x);
        printf("거듭제곱 횟수 입력 :");
        scanf_s("%d", &n);
        j=1;
        while (j++ <= n)
                mul *=x;
        printf("%d의 %d거듭제곱 값은 %d\n",x, n, mul);
        _getch();
}
```

3. do - while 루프

■ 형식

```
do
{
        do~while 문장들;
}
while(조건);   // while 문이 루프의 맨 마지막에 옴 ( ; 유의)
```

- do~while문장이 하나 이상 일 때는 반드시 {}가 필요

■ 내용

- 조건이 참이면 루프를 반복수행 (while과 동일)
- 그러나 do – while 은 루프를 적어도 한번은 수행한다.

① while

```
int j=5;
while (j<4)
{       printf("j=%d",j);      // 수행되지 않음
        j++;                    // 수행되지 않음
}
다음 문장;
```

② do - while

```
int j=5;
do
{       printf("j=%d", j);      // 한번은 수행
        j++;                    // 한번은 수행
}while (j<4);
다음 문장;
```

■ 예제 47) 메뉴 선택

```
void main(void)
{     int j;
      do
      {     printf("다음 메뉴 중 하나를 선택\n");
            printf("1 : 새로만들기\n");
            printf("2 : 파일열기\n");
            printf("3 : 파일닫기\n");
            scanf_s("%d", &j);
      }while(j<1 || j>3);  // 1~3 사이의 숫자가 아니면 계속 반복
      printf("선택한 메뉴는 %d", j);
      _getch();
}
```

*고찰 : while을 쓰면 어떻게? ->처음 한번은 무조건 실행되게.

■ 예제 48) 숫자 알아맞히기 게임1

```
void main(void)
{   int answer =37;        // 정답
    int guess;             // 추측 값
    int tries = 0;         //시도 횟수

    do  {
          printf("정답 추측: ");
          scanf_s("%d", &guess);  // 추측한 숫자 입력
          tries++;      // 시도 횟수 증가
          if (guess >answer)  // 정답보다 큰 경우
                printf("추측한 수가 정답보다 큽니다.\n");
          if (guess <answer)  // 정답보다 작은 경우
                printf("추측한 수가 정답보다 작습니다.\n");
    } while (guess !=answer);
    printf("축하합니다. %d번 만에 맞추었습니다.", tries);
    _getch();
  }
```

■ 보충 : 난수 생성 (random number generation)

```
#include <time.h>
#include <stdlib.h> //rand(); 가 정의되어 있는 헤더파일
void main(void)
{       int j, n=0;
        srand((int)time(NULL)); // 랜덤 함수 초기화
        while(n++<10)
        {
                j = rand() % 11; // 0에서 10사이의 랜덤 수를 발생시켜 변
                        수 j에 대입. rand()는 0~32767의 랜덤 수 발생
                printf("%d ", j);
        }
        _getch();
}
```

■예제 8-9) 숫자 알아맞히기 게임 2

```
void main(void)
{
  int j=0, ans, guess;

  srand((int)time(NULL)); // random 함수 초기화
  ans = rand()%100;  // 0~99 사이의 랜덤 수 발생

  while (1) // 무한 반복,
  {
        printf("\n숫자를 맞춰보세요(0에서 99사이) :");
        scanf_s("%d",&guess);
        j++;  // 시도 횟수 증가

        if(guess<ans)
                printf("추측한 수<정답");
        else if(guess>ans)
                printf("추측한 수>정답");
        else
        {       printf("\n축하합니다!!! 정답은 %d입니다.", ans);
                printf("\n당신은 %d회 만에 맞추셨습니다.\n", j);
                _getch();
                return;  // 프로세스를 윈도우로 돌려줌 ->종료
        }
  }
}
```

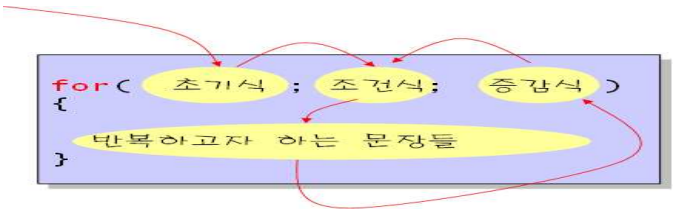
4. for 루프

■ 형식

```
for (초기식; 조건식; 증가식)
{
        문장들 ;
}
```

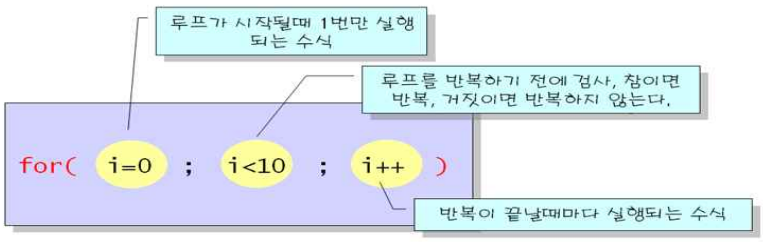
■ 루프 순환 구조

① 초기식을 실행한다.
② 반복 조건을 나타내는 조건식을 계산한다.
③ 수식의 값이 거짓이면 **for** 문의 실행이 종료된다.
④ 수식의 값이 참이면 문장이 실행된다.
⑤ 증감식을 실행하고 ②로 돌아간다.



■ 순환 예

```
int i;
for(i = 0; i < 10; i++)
        printf("Hello World!\n");
```



⇒ i=0부터 i=9까지 10번 반복하면서 “Hello World!”를 출력

1번째 반복	0	참	반복
2번째 반복	1	참	반복
3번째 반복	2	참	반복
4번째 반복	3	참	반복
5번째 반복	4	참	반복
6번째 반복	5	참	반복
7번째 반복	6	참	반복
8번째 반복	7	참	반복
9번째 반복	8	참	반복
10번째 반복	9	참	반복
11번째 반복	10	거짓	중지

■ 예제 49) 1에서 10까지 정수 합

```
void main(void)
{
        int i, sum;
        sum = 0;
        for(i = 1; i <= 10; i++)
                sum += i;    // sum = sum + i;와 같음
        printf("1부터 10까지의 정수의 합= %d \n",sum);
        _getch();
}
```

*while()문으로 바꾸면? (test?)

■ 예제 50) 네모그리기

```
void main(void)
{
        int i;
        printf("************\n");  // * 12개
        for(i = 0; i < 5; i++)
                printf("*          *\n");  // space 10개
        printf("************\n");
        _getch();
}
```

■ for 루프의 다양한 형태

```
for (i = 10; i > 0; i-- )
    printf("Hello World!\n");
```

```
for (i = 0; i < 10; i += 2 )
    printf("Hello World!\n");
```

```
for (i = 1; i < 10; i *= 2 )
    printf("Hello World!\n");
```

```
for (i = 0; i < 100; i = (i * i) + 2 )
    printf("Hello World!\n");
```

```
for ( ; ; )
    printf("Hello World!\n");
```

```
for ( ; i<100; i++ )
    printf("Hello World!\n");
```

```
for (i = 0, k = 0; i < 100; i++ )
    printf("Hello World!\n");
```

```
for (printf("반복시작"), i = 0; i < 100; i++ )
    printf("Hello World!\n");
```

■ 다중 for 문

```
for( 초기식; 조건식; 증감식)
{       문장들;
        for(초기식 ; 조건식; 증감식)
        {
                문장들;
        }
        문장들;
}
```

■ 예제51) 구구단 출력

```
void main(void)
{  int x, y;
   for(y=2; y<10; y++)
   {       for(x=1; x<10; x++)
                printf("%dx%d = %d \n", y, x, y*x);
           printf(" \n");
```

```
            _getch();
        }
        _getch();
    }
```

5. break와 continue문

1) break 문

■ 반복 루프(for() 또는 while() 문)를 벗어나고자 할 때 사용

```
void main(void)
{       int j;
        for(j=1; j<10 ; j++)
        {       // if(j>5) break;   // j=5까지 실행 후 강제 탈출
                printf(" 2 x %d = %d\n", j, j*2);
        }
        _getch()
}
```

■ 예제 53) 원금 100만원을 연 30%이자로 몇 년을 적금하면 원금의 10배가 되는가. (복리로 계산)

```
#define SEED_MONEY 1000000   // 원금
void main(void)
{       int year=0, money=SEED_MONEY;
        while(1)   // 무한 반복문
        {       year++;
                money += money*0.3; // 1년에 30%씩 이자가 늘어남
                if( money >= 10*SEED_MONEY ) //초기 돈의 10배
                        break;
        }
        printf("10배가 되는 기간은 %d년", year);
        _getch();
}
```

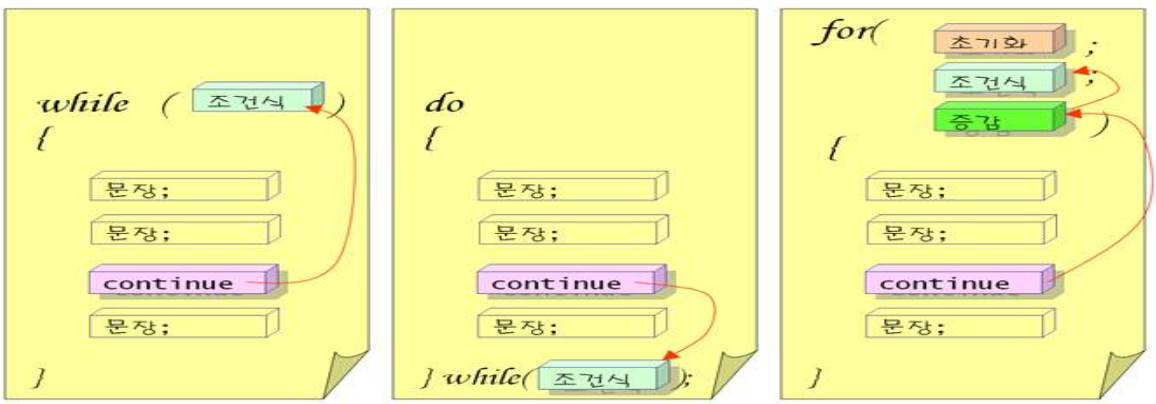
■ break는 하나의 반복 루프만 탈출가능. 여러 개의 반복루프를 한 번에 탈출하기 위해서는 goto문을 사용

예제 54)

```
void main(void)
{   int x, y;
    for(y = 1; y < 10000; y++)
    {
        for(x = 1; x < 50; x++)
        {
            if(_kbhit())        // stdlib.h 에 정의
            {       goto out;
                //break;        // break;를 쓸 경우는?
            }
            printf("%d ", x);
        }
        printf(" y=%d\n",y);
    }
    out:
    _getch()
}
```

2) continue 문

■ 현재 반복을 현 시점에서 중단하고 다음 반복 시작



■ 예(ex55_1)

```
void main(void)
{   int i;
    for(i=0 ; i<10 ; i++)
    {
        if( i%3 == 0 ) // 3의 배수이면 continue
            continue;
        printf("%d ", i); // 항상 실행되나?
    }
    _getch();
}
```

■ 예제 55) 키보드로 입력한 알파벳 소문자를 대문자로 바꾸는 프로그램(ex55_2)

```
void main(void)
{   char letter;
   while(1) // 무한 반복
   {     printf("소문자를 입력하시오(종료는 Q): ");
         scanf_s("%c", &letter, sizeof(letter));
         fflush(stdin);  // 버퍼에 남아있는 엔터키 문자 없앰.

         if( letter == 'Q' )
                  break;    // while문 탈출, 즉 종료
         if( letter < 'a' || letter > 'z' )  // 입력 문자가 소문자가 아닌 경우
         continue ; // while 첫머리로 돌아감
         letter -= 32; // 소문자를 대문자로 변환, 아스키코드 값 참조
         printf("변환된 대문자는 %c입니다. \n", letter);
   }
}
```

■ 예제 56) 달력 출력

```
#define START_DAY 2 // 첫 번째 날의 요일, 0: 일요일 ~ 6:토요일
#define DAY_MONTH 31 //한 달 일수
void main(void)
{
int i, day;   //i:날짜, day :요일

printf("=====================\n");
printf("일 월 화 수 목 금 토 \n");  // 공백 한 칸
printf("=====================\n");

for (day = 0; day < START_DAY; day++) //시작요일 앞은 공백을 만듬.
         printf("    "); //공백 세 칸

for (i = 1; i <= DAY_MONTH; i++)
{
         if (day == 7)  //한 주가 끝나면
         {
                  day = 0;        // 요일은 일요일로 다시 초기화
                  printf("\n");     // 줄 바꿈
         }
         printf("%2d ", i);      //날짜 출력
         day++;                  //요일 증가
}
printf("\n=====================\n");
 _getch();
}
```

■ 예제 57) n번 제곱 값을 구하는 프로그램

```
void main(void)
{     int x = 3;   // 수
      int n = 8;   //거듭제곱 횟수
      int i, val = 1; //초기 값 주의
      for (i = 0; i < n; i++)
            val *= x;
      printf("%d의 %d 승은 %d", x, n, val);
      _getch();
}
```

6. 응용 프로그램

■ 라스베이거스 슬롯머신 게임 조건

- 초기 자금은 50달러
- 한 번에 1달러를 걸 수 있음 (즉 한번에 1달러를 따거나 잃을 수 있음)
- 돈을 딸 확률 50%
- 게임 종료 : 가진 돈을 다 잃거나 목표 금액인 250달러에 도달

Q : 게임조건이 위와 같을 때 어떤 사람이 라스베이거스에 100번 갔었다면 몇 번이나 250달러를 따서 돌아 올 수 있을까?

```
#include <stdlib.h>
#include <time.h>
void main(void)
{       int initial_money = 50;         //초기 금액
        int goal_money = 250;           // 목표 금액
        int num_wins = 0;               //250달러를 딴 횟수
        double prob;                    //random으로 발생된 0.0 ~1.0 사이 값
        int j, cash;
        srand(time(NULL));              //rand() 함수 초기화
        for ( j = 0; j < 100; j++)
        {
            cash = initial_money;       //금액을 초기화
            while (cash>0 && cash<goal_money) //돈이 있고 목표금액보다 적은 경우
            {
                prob = (double)(rand())/RAND_MAX; // 0.0~ 1.0실수 ,32767
                if (prob <= 0.5)                // 이길 확률이 50%
                    cash++;                     // 돈이 1달러 추가 됨
                 else                           //그 외는 진 경우. 돈이 1달러 차감 됨
                    cash--;
            }                                   // 1회 게임 종료
            if (cash == goal_money)             // 목표치에 도달한 경우
                  num_wins++;                   // 이긴 횟수 증가
        }
        printf("\n%d번 중 %d번 성공", j, num_wins);
        _getch();
}
```

All of you will be good programmers!!!

(See you next semester.)