

**Robotics**



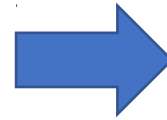
대경혁신인재양성프로젝트  
**HuStar**

# C프로그래밍

# C언어

- 프로그래밍 언어
  - 컴퓨터와 사람과의 사이에서 대화를 위한 공통의 대화 수단
  - C언어, C++, Java, python, ...
- C언어
  - 작고 빠른 프로그램을 개발하기 위해 설계
  - 대부분의 다른 프로그래밍 언어 보다 더 낮은 수준
  - Embedded 환경에 적합

```
C hello.c > ...  
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      puts("C Rocks!");  
6      return 0;  
7  }
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
  
PS C:\Users\sori\Documents\Hustar> .\hello.exe  
C Rocks!
```

C언어로 작성된 프로그램

Hustar로봇아카데미

프로그램 실행결과

# C언어- 작동 방식

- 한국어 ≠ 영어
  - 번역(통역) 하는 사람을 통해 의사 소통
  - 직접 영어를 배워 의사소통

안녕,



hello

- 사람의 언어 ≠ 기계어
  - 기계어를 직접 학습하기에는 어려움이 있음
  - 번역(통역) 하는 프로그램(컴파일러, complier)을 통해 의사 소통
  - 컴파일(compile)이라는 과정을 통해 기계어로 번역

hello



Compiler



```
Entry Point
00401010 40 00 A1 0F A1 40 00 C1 E0 02 A3 13 A1 40 00 52 @.??취??R
00401020 6A 00 E8 D5 88 00 00 8B D0 E8 8A 10 00 00 5A E8 j.端?.땡?...Z?
00401030 20 04 00 00 E8 83 10 00 00 6A 00 E8 94 1C 00 00 ...?..j.?..
00401040 59 68 B8 A0 40 00 6A 00 E8 AF 88 00 00 A3 17 A1 Yh를@j.괘??
00401050 40 00 6A 00 E9 C3 6A 00 00 E9 C2 1C 00 00 33 C0 @j.端j.端...3?
00401060 A0 01 A1 40 00 C3 A1 17 A1 40 00 C3 60 BB 00 50 ??찾?.??P
00401070 B0 BC 53 68 AD 0B 00 00 C3 B9 9C 00 00 0B C9 가Sh?...첫?...?
00401080 74 4D 83 3D 0F A1 40 00 00 73 0A B8 FE 00 00 00 tM??..s.뵈...
00401090 E8 D7 FF FF FF B9 9C 00 00 00 51 6A 08 E8 6C 88 頑□□□뵈...Q??
```

# Compiler(컴파일러)

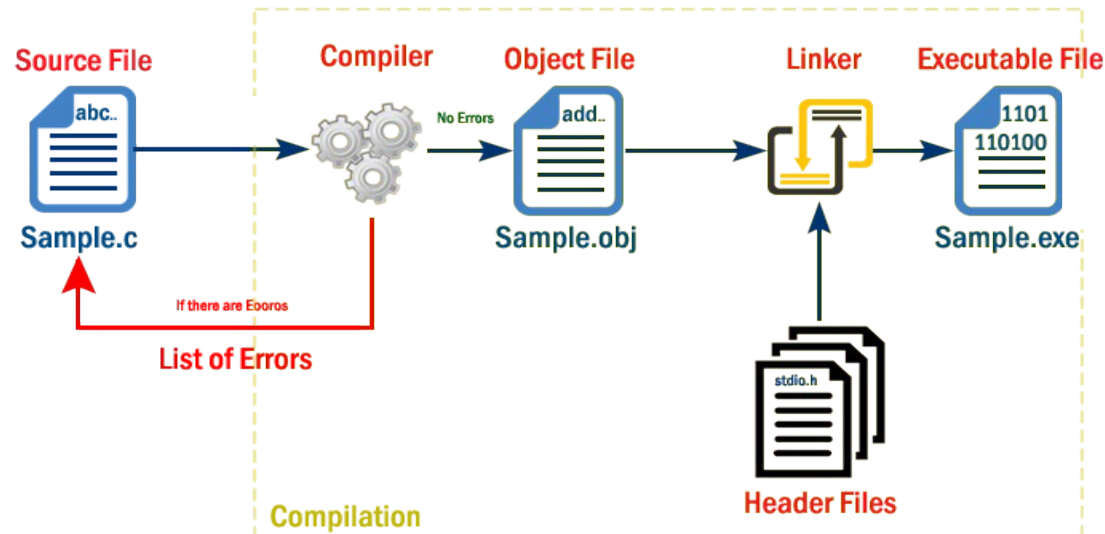
- 컴파일러

- 사람의 언어를 기계어로 번역 하는 역할
- 대표적인 예) GCC, Clang



- 컴파일

- 고급언어(C언어)로 작성된 컴퓨터 프로그램을 기계어로 번역하는 과정



# C언어의 특징

- C언어

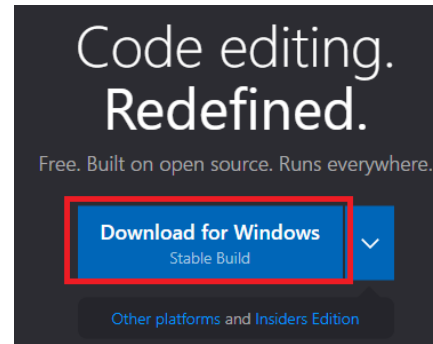
- 1971년 UNIX 개발을 위해 Dennis Richie와 Ken Thompson이 개발
- 절차지향적 특성을 가짐
  - 정해진 순서의 실행흐름을 중시
- C언어로 작성된 프로그램은 이식성이 좋음
  - CPU의 종류에 상관없이 실행이 가능
- C언어로 구현된 프로그램은 좋은 성능을 보임
  - 상대적으로 요구하는 메모리의 양이 적고, 속도를 저하시키는 요소를 최소화함

# C언어 개발환경

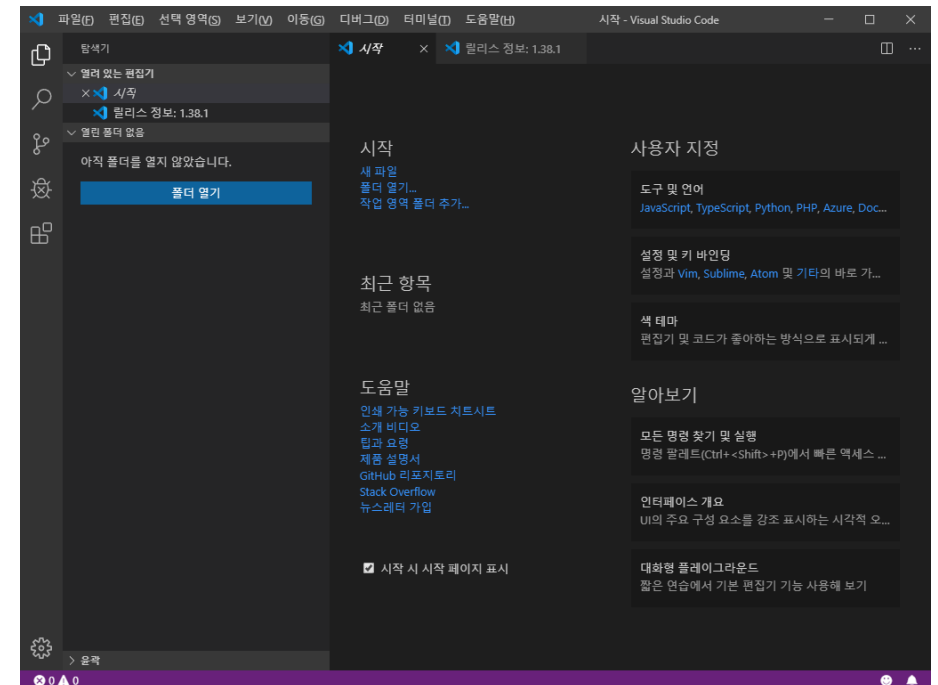
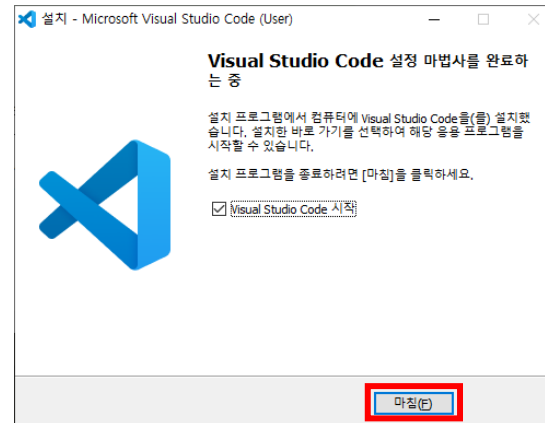
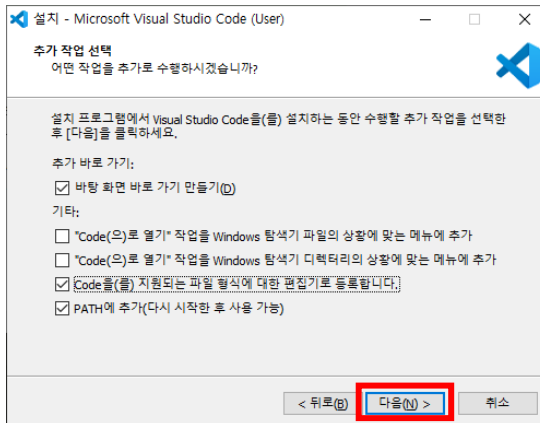
- Editor: Visual Studio Code(VS Code)
  - <https://code.visualstudio.com/>
- OS: windows 10
- Compiler: mingw-w64 GCC
  - <https://sourceforge.net/projects/mingw-w64/>

# Visual Studio Code install

- <https://code.visualstudio.com/> 접속
- 프로그램 다운



## 설치파일 실행



실행 화면

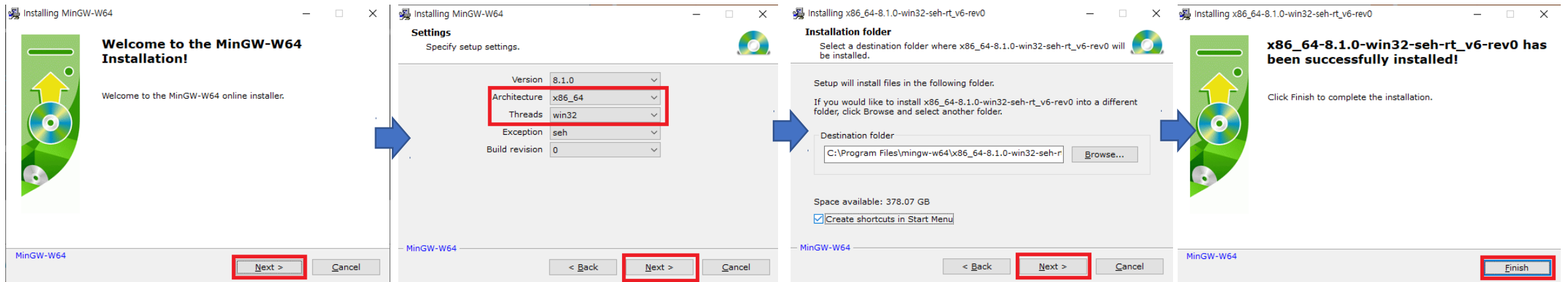
# GCC(ming64) install

- <https://sourceforge.net/projects/mingw-w64/> 접속

- GCC 다운



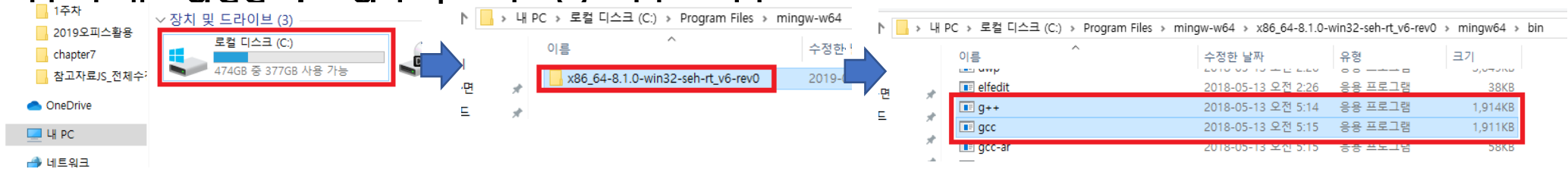
- 설치파일 실행



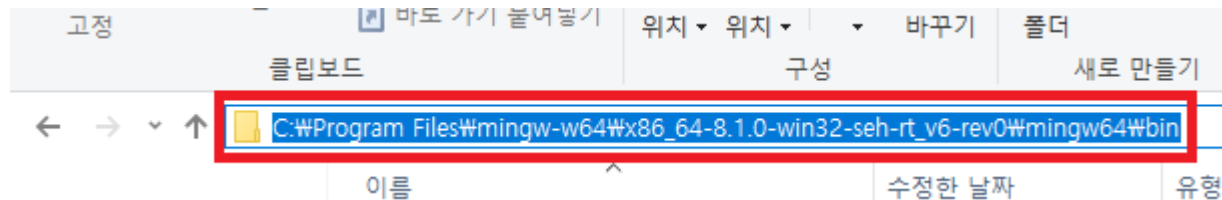


# GCC 설치 경로 확인

- GCC 경로를 설정하기 위해 GCC 설치 위치 파악
  - “C:\Program Files\mingw-w64\x86\_64-8.1.0-win32-seh-rt\_v6-rev0\mingw64\bin”  
위치에 “gcc”. “a++” 실행 파일 확인

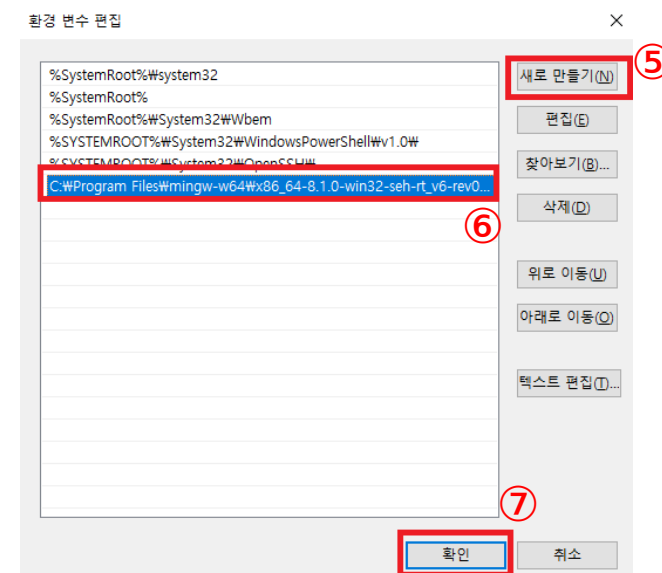
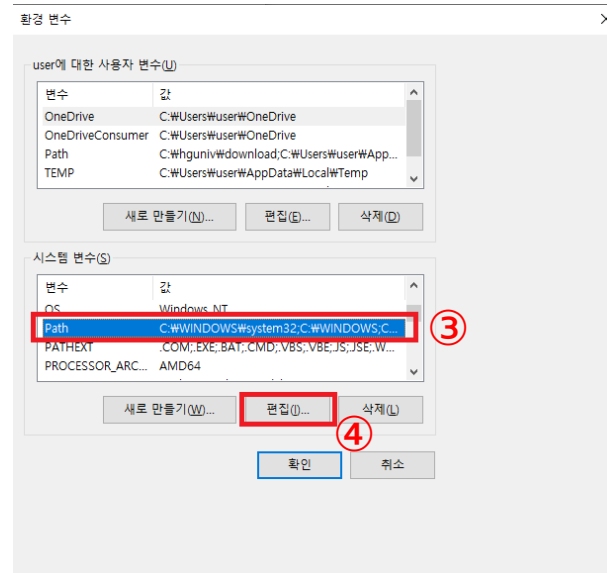
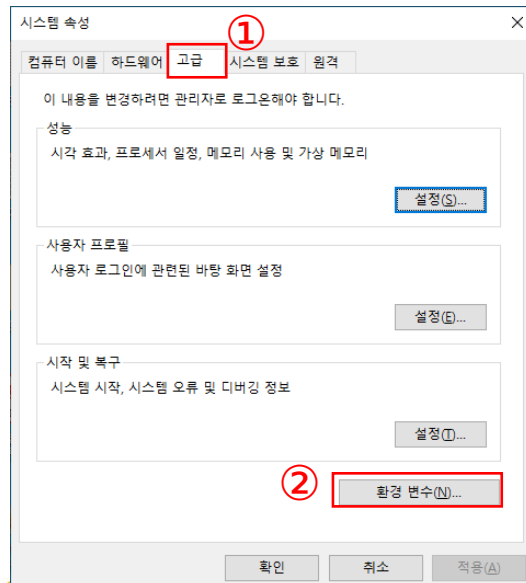
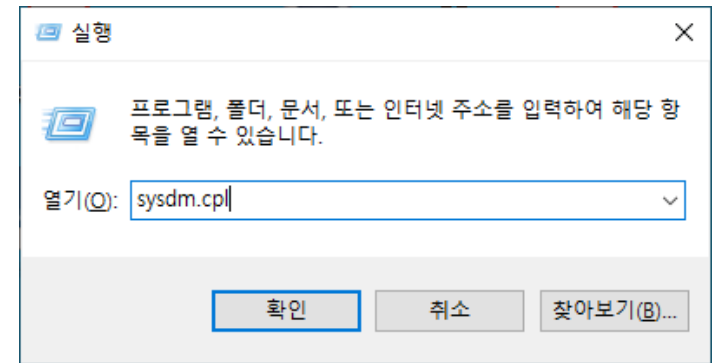


- 탐색기



# GCC 경로 설정

- 환경변수 설정 화면 열기
  - 윈도우키+R을 눌러 실행창에서 논으.cpl 입력
- 시스템 속성 > 고급 탭 > 환경 변수
- 시스템 변수 > path 선택 > 편집 > 새로 만들기 > GCC설치 경로 입력 > 확인



# GCC 경로 설정 test

- 윈도우+R 실행창에서 cmd 입력
- Command 에 gcc -v 를 실행 시켜 아래와 같은 결과 확인

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.295]
(c) 2019 Microsoft Corporation. All rights reserved.

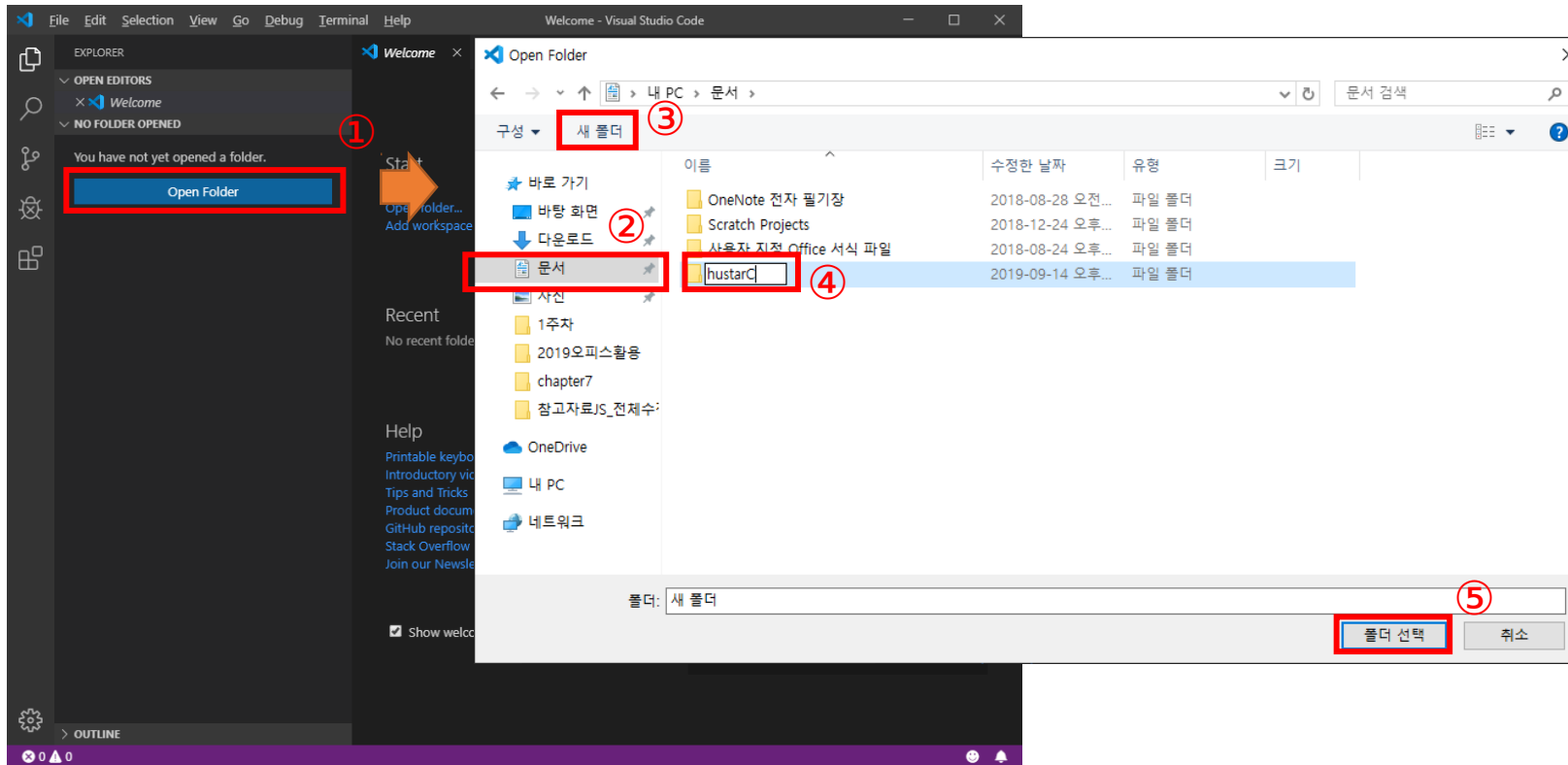
C:\Users\User>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=C:/Program Files/mingw-w64/x86_64-8.1.0-win32-seh-rt_v6-rev0/mingw64/bin/./libexec/gcc/x86_64-w64-
mingw32/8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w
64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-win32-seh-rt_v6-rev0/mingw64 --enable-shared --enable
-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=win32 --ena
ble-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --
enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpat
h --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=noco
na --with-tune=core2 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --
with-mpfr=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-s
tatic --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-win32-seh-rev0, Built by
MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x
86_64-810-win32-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/
prerequisites/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-win32-seh-rt_v6
-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-
mingw32-static/include' CPPFLAGS='-I/c/mingw810/x86_64-810-win32-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prere
quisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' LDFLAGS='-pipe -fno-i
dent -L/c/mingw810/x86_64-810-win32-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L
/c/mingw810/prerequisites/x86_64-w64-mingw32-static/lib '
Thread model: win32
gcc version 8.1.0 (x86_64-win32-seh-rev0, Built by MinGW-W64 project)

C:\Users\User>
```

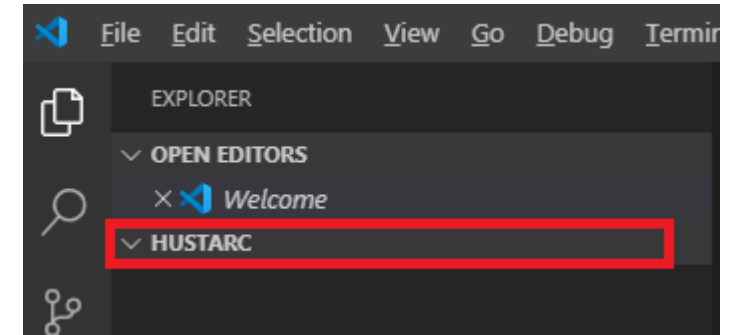
주의: cmd 창 실행전 환경  
변수 설정과 관련된 모든  
창을 종료해야 함!

# Visual Studio Code 프로젝트 폴더 설정

- Visual Studio Code에서 code를 작성하기 전 작업 폴더 (workspace) 설정 필요



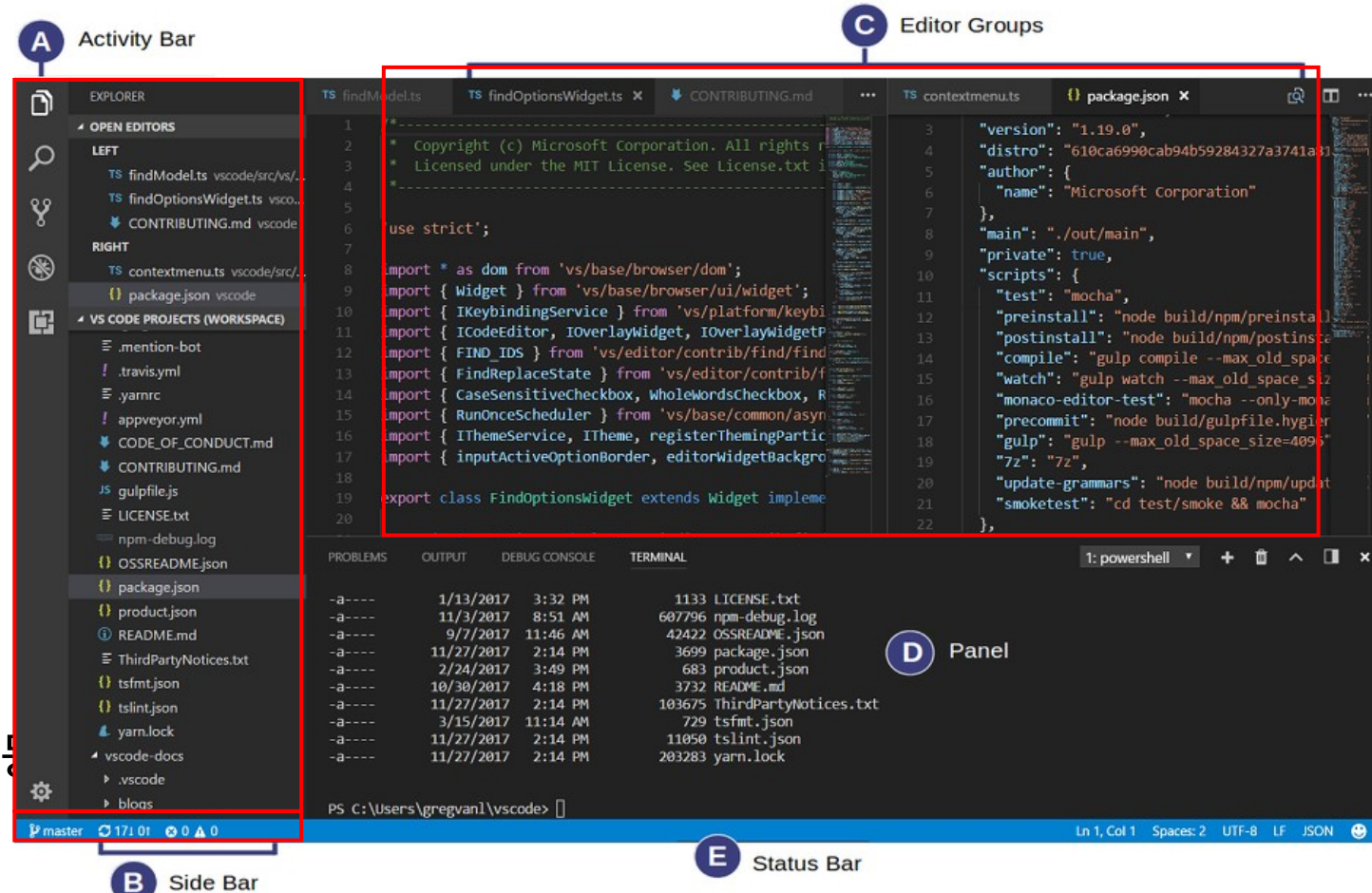
Open Folder > 문서 > 새 폴더  
> 새 폴더 이름을 hustarC로  
변경 > hustarC 선택 > 폴더  
선택



설정된 작업 폴더 확인

# VS Code 인터페이스

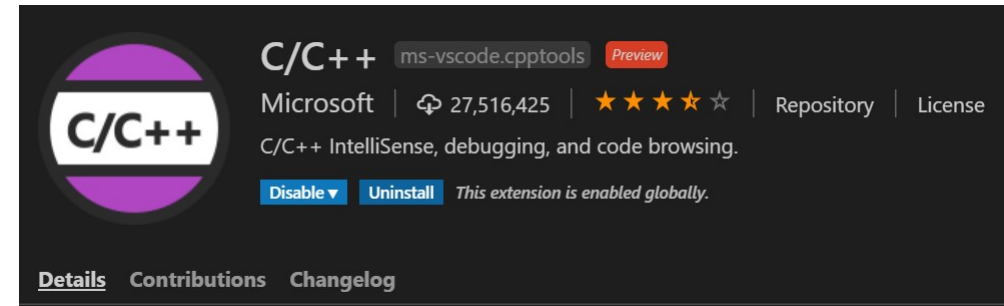
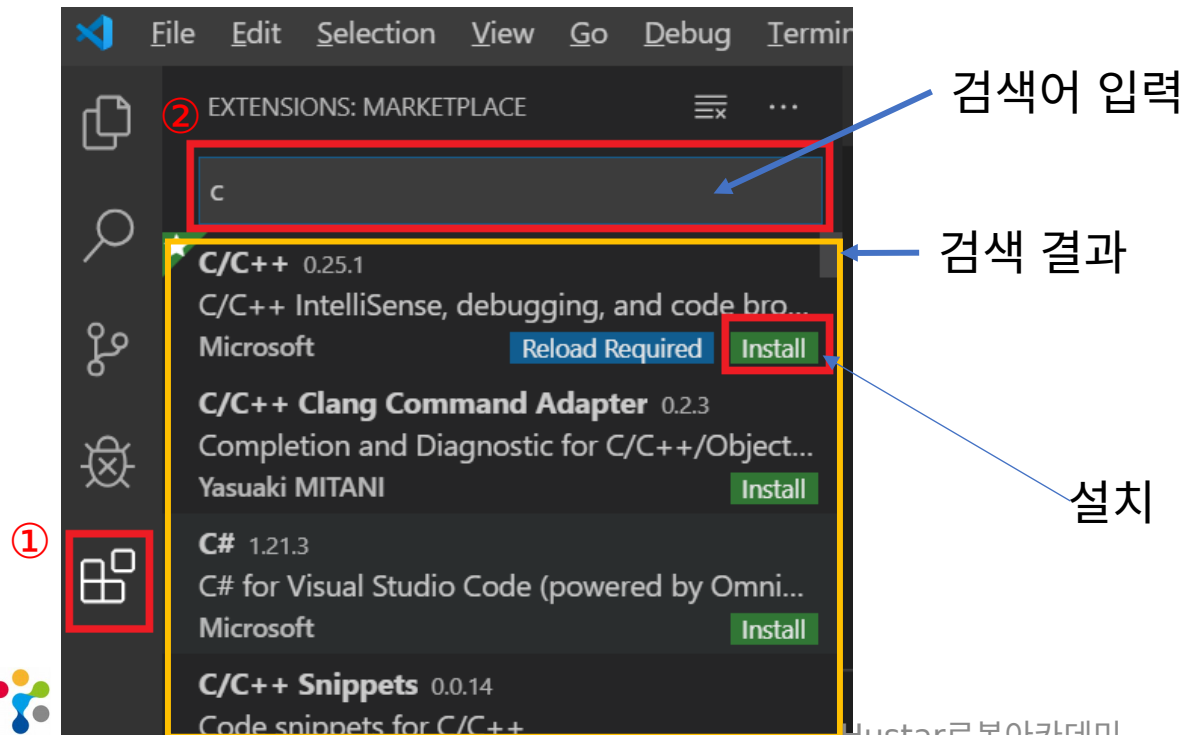
- C영역: Editor
  - 코드 작성 영역
- A영역: Active bar
  - 탐색기, 검색, git, debug, extension을 제공
- B영역: Side bar
  - Active bar에서 선택한 기능의 설정 또는 사용
- D영역: Panel
  - Output, debugging info, error 등 보거나 terminal을 사용할 수 있는 영역
- E영역: Status Bar
  - 프로젝트나 파일에 대한 정보를 확인(언어, 인코딩 등)



# VS Code Extension

- Extension

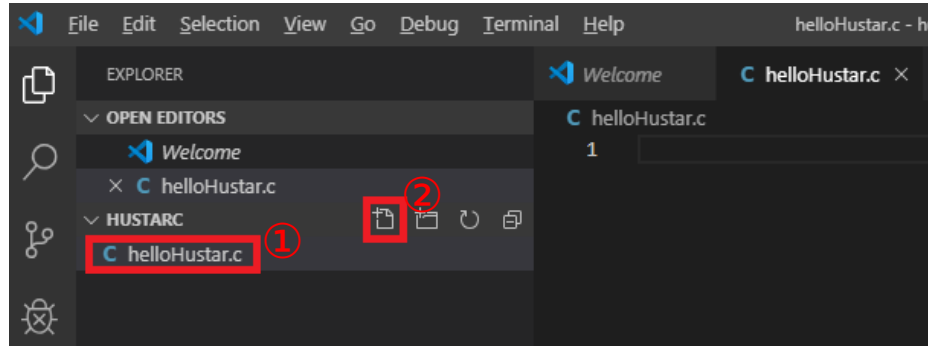
- VS code에서 프로그램을 작성에 도움을 주는 기능
- 사용자가 원하는 Extension(확장기능)을 설치할 수 있음
- Extension 대한 내용을 확인할 수 있음



Extension에 대한 자세한  
설명

# VS Code에서 프로그램 작성

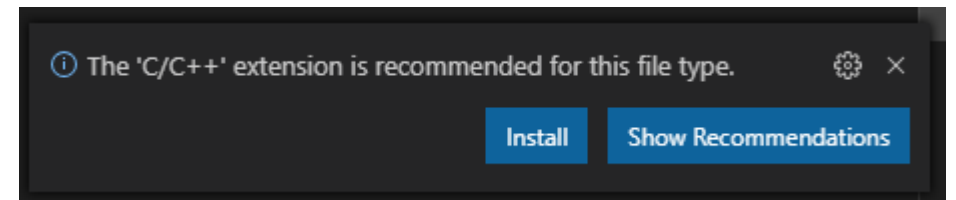
- 파일을 추가하기 위해 작업 폴더 오른쪽에 나타나는 파일 추가 버튼 클릭 > 파일명을 “helloHustar.c”로 지정



- helloHustar.c 파일 작성

```
helloHustar.c
1  #include <stdio.h>
2
3  int main(){
4      printf("hello Hustar");
5      return 0;
6  }
```

**TIP.** 오른쪽 아래에 나타나는 추천 Extension을 설치하면 소스코드 작성에 도움을 받을 수 있음

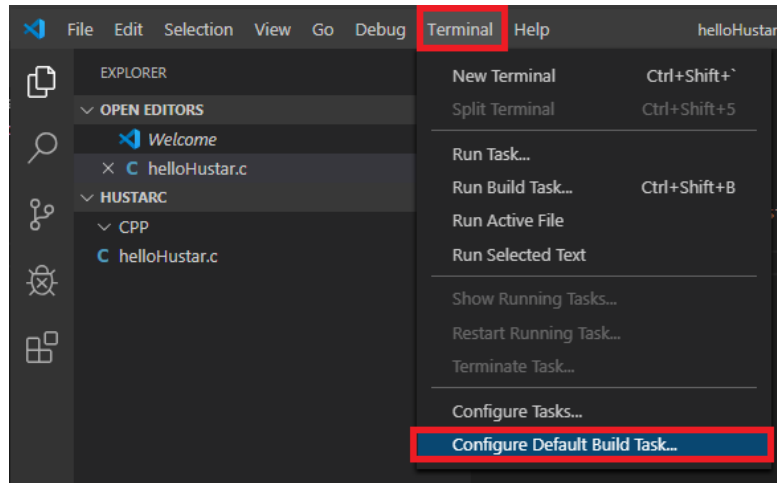


소스코드 작성 후 파일 저장  
(ctrl+s)

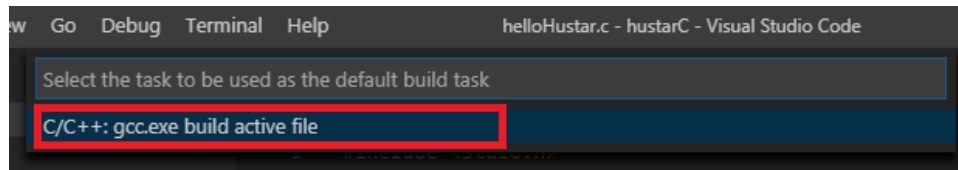


# VS Code C 컴파일러 설정

- VS Code 메뉴 Terminal > Configure default build task 선택



- C/C++: gcc.exe build active file 선택





# VS Code에서 C 컴파일러 설정

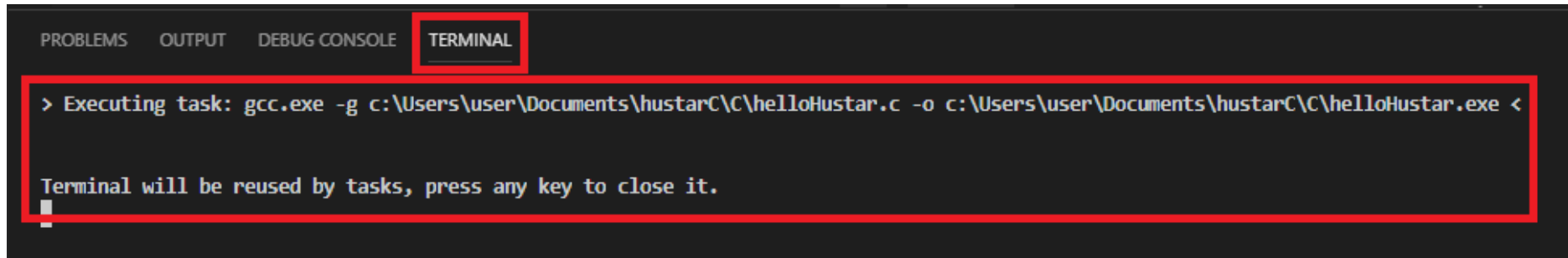
- GCC 설정 파일 확인

```
.vscode > {} tasks.json > [ ] tasks > {} 1 > [ ] problemMatcher
1  {
2    // See https://go.microsoft.com/fwlink/?LinkId=733558
3    // for the documentation about the tasks.json format
4    "version": "2.0.0",
5    "tasks": [
6      {
7        "type": "shell",
8        "label": "gcc.exe build active file",
9        "command": "gcc.exe",
10       "args": [
11         "-g",
12         "${file}",
13         "-o",
14         "${fileDirname}\\${fileBasenameNoExtension}.exe"
15       ],
16       "options": {
17         "cwd": "C:\\\\Program Files\\mingw-w64\\x86_64-8.1.0-win32-seh-rt_v6-rev0\\mingw64\\bin"
18       },
19       "problemMatcher": [
20         "$gcc"
21       ],
22       "group": {
23         "kind": "build",
24         "isDefault": true
25       }
26     ],
27   },
28 }
```

자신의 gcc설치 경로

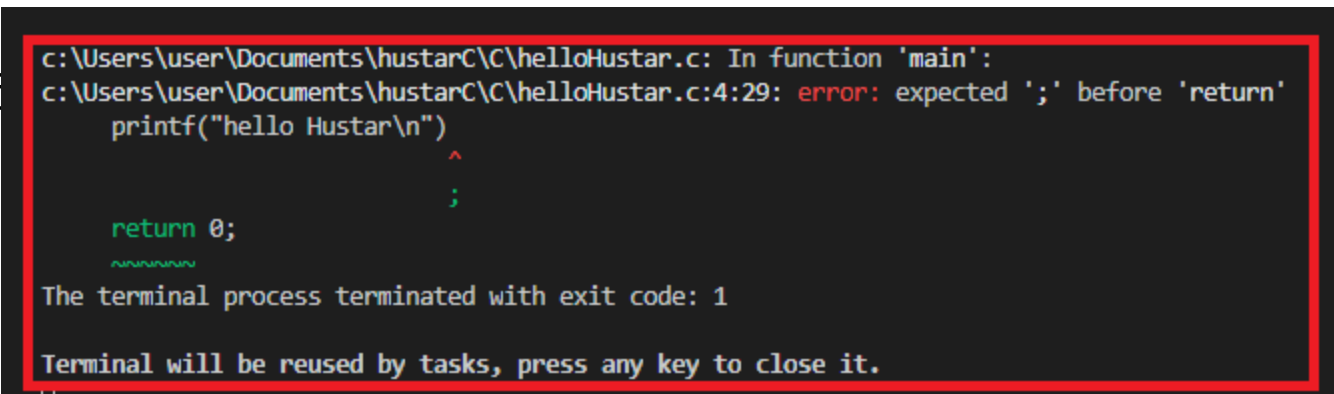
# VS Code Compile

- GCC 컴파일러 설정이 잘 되어 있는 경우
  - Ctrl + shift + b 를 눌러 컴파일
  - 오류 없이 컴파일 완료



A screenshot of the VS Code terminal interface. The 'TERMINAL' tab is selected and highlighted with a red box. The terminal shows the command: `> Executing task: gcc.exe -g c:\Users\user\Documents\hustarC\C\helloHustar.c -o c:\Users\user\Documents\hustarC\C\helloHustar.exe <`. Below the command, it says: `Terminal will be reused by tasks, press any key to close it.`

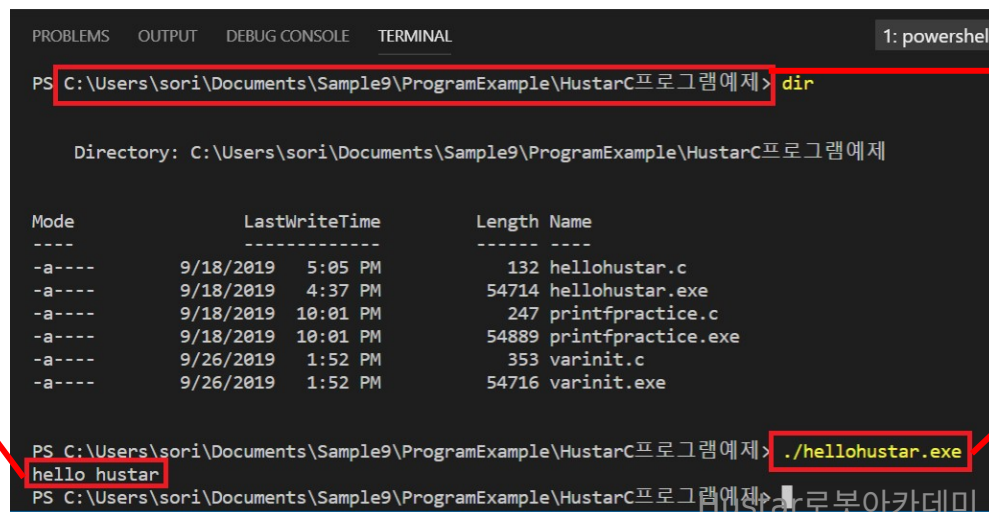
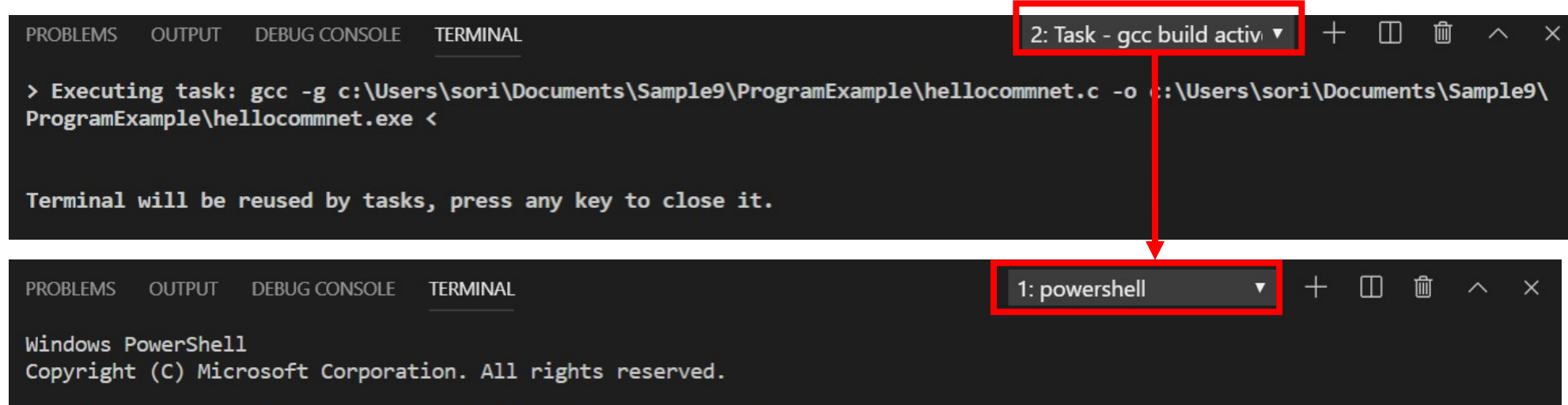
- 오류



A screenshot of the VS Code terminal interface showing a compilation error. The terminal text is: `c:\Users\user\Documents\hustarC\C\helloHustar.c: In function 'main':  
c:\Users\user\Documents\hustarC\C\helloHustar.c:4:29: error: expected ';' before 'return'  
printf("hello Hustar\n")  
^  
;  
  
return 0;  
~~~~~  
The terminal process terminated with exit code: 1  
Terminal will be reused by tasks, press any key to close it.`

# VS Code에서 exe파일 실행

- 컴파일이 완료화면에서 enter를 눌러 터미널 변경



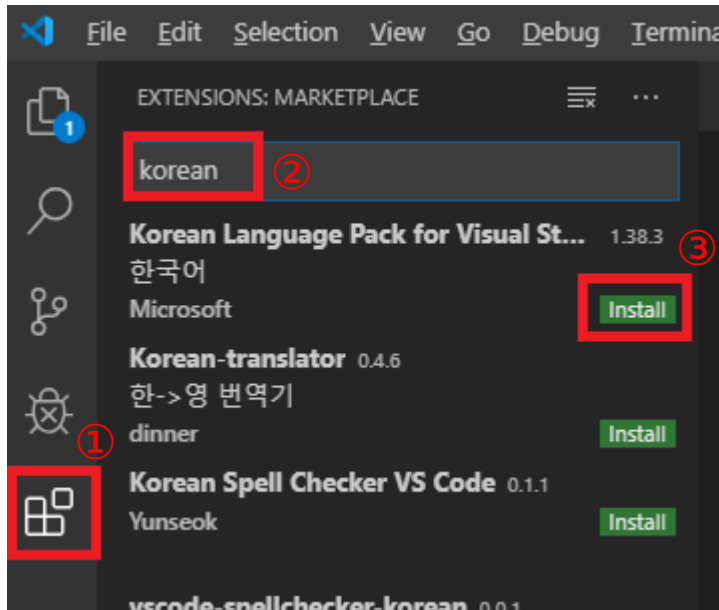
Cd명령으로 소스코드가 있는 경로로 이동  
Ex) `cd HustarC프로그램예제`

컴파일된 실행파일

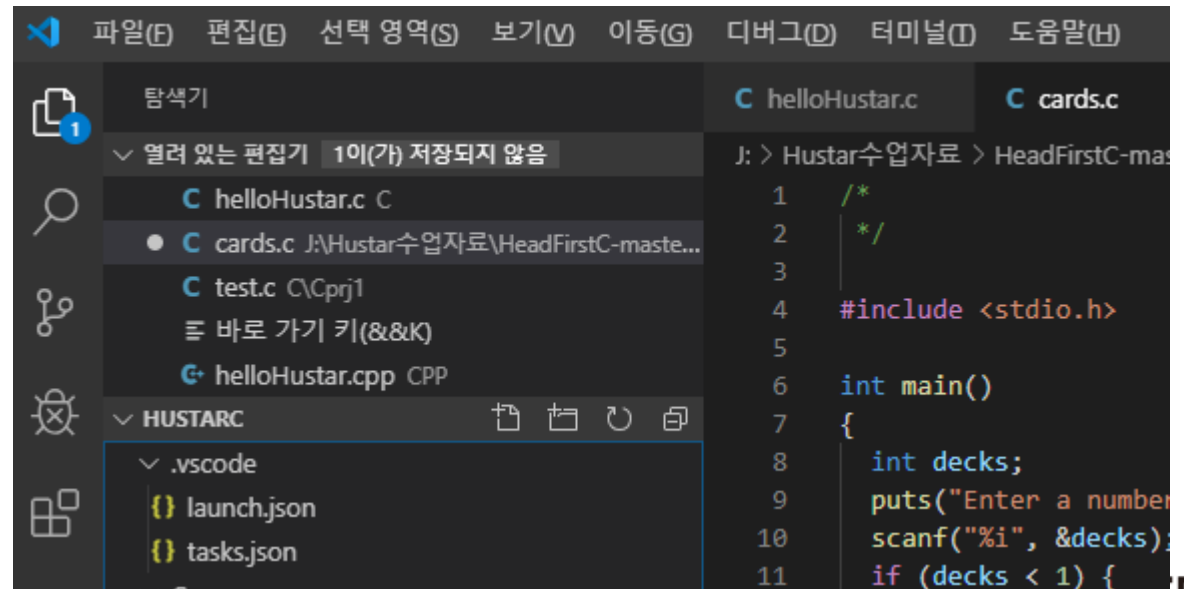
실행결과

# VS Code 한글팩 설치

- Extension 버튼 선택 > 검색에 korean 입력 > Korean language pack for visual studio code > install



- VS Code 재실행
  - 인터페이스 한국어로 변환됨



# 가장 먼저 실행하는 프로그램 hellohustar

실습하기

```
/*처음 시작하는 c 프로그램*/  
  
#include <stdio.h>  
  
int main(void){  
  
    printf("hello hustar\n");  
  
    return 0;  
}
```

실행결과

```
PS C:\Users\sori\Documents\Sample9\ProgramExample\HustarC프로그램예제> .\hellohustar.exe  
hello hustar  
PS C:\Users\sori\Documents\Sample9\ProgramExample\HustarC프로그램예제> □
```

# C프로그램의 구조

`/*처음 시작하는 c 프로그램*/`

`#include <stdio.h>`

```
int main(void){  
  
    printf("hello hustar\n");  
  
    return 0;  
}
```

```
/*  
 * 신발 안에 들어 있는 카드의 숫자를 계산하는 프로그램  
 * (c) 2014, Histar  
 */  
  
#include <stdio.h>  
  
int main()  
{  
    int decks;  
    puts("Enter a number of decks");  
    scanf("%i", &decks);  
    if (decks < 1) {  
        puts("That is not a valid number of decks");  
        return 1;  
    }  
    printf("There are %i cards\n", (decks * 52));  
    return 0;  
}
```

주석부분:

일반적으로 C프로그램은 프로그램을 설명하는 주석으로 시작

Include:

전처리 헤더 파일을 포함 시켜  
컴파일러에게 어떤 외부 코드를 사용할지  
안내

main함수:

C프로그램에서 가장 중요한 함수, main  
함수는 프로그램 코드가 실행되는  
시작지점

# 주석(comment)

- 주석은 프로그램 내에 삽입된 메모
- 컴파일의 대상에서 제외
  - 주석의 유무는 프로그램의 실행 결과에 영향을 주지 않음
- 주석 입력 방법
  - 블록 단위 주석
    - `/* 주석 내용 */`
    - 두 줄 이상의 주석을 처리
  - 행 단위 주석
    - `// 주석 내용`
    - 한 줄의 주석을 처리

```
/*
   제목: Hello world 출력
   기능: 문자열 출력
   파일명: HelloComment.c
   수정날짜: 2019.09.16
   작성자: Hustar
*/

#include <stdio.h> //헤더파일 선언

int main(void){ //main함수 시작
    /*
     * 이 함수 내에서는 하나의 문자열을 출력.
     * 결과는 Terminal monitor로 출력됨
     */
    printf("hello world! \n"); //문자열 출력
    return 0; // 0 반환
} //main 함수 종료
```

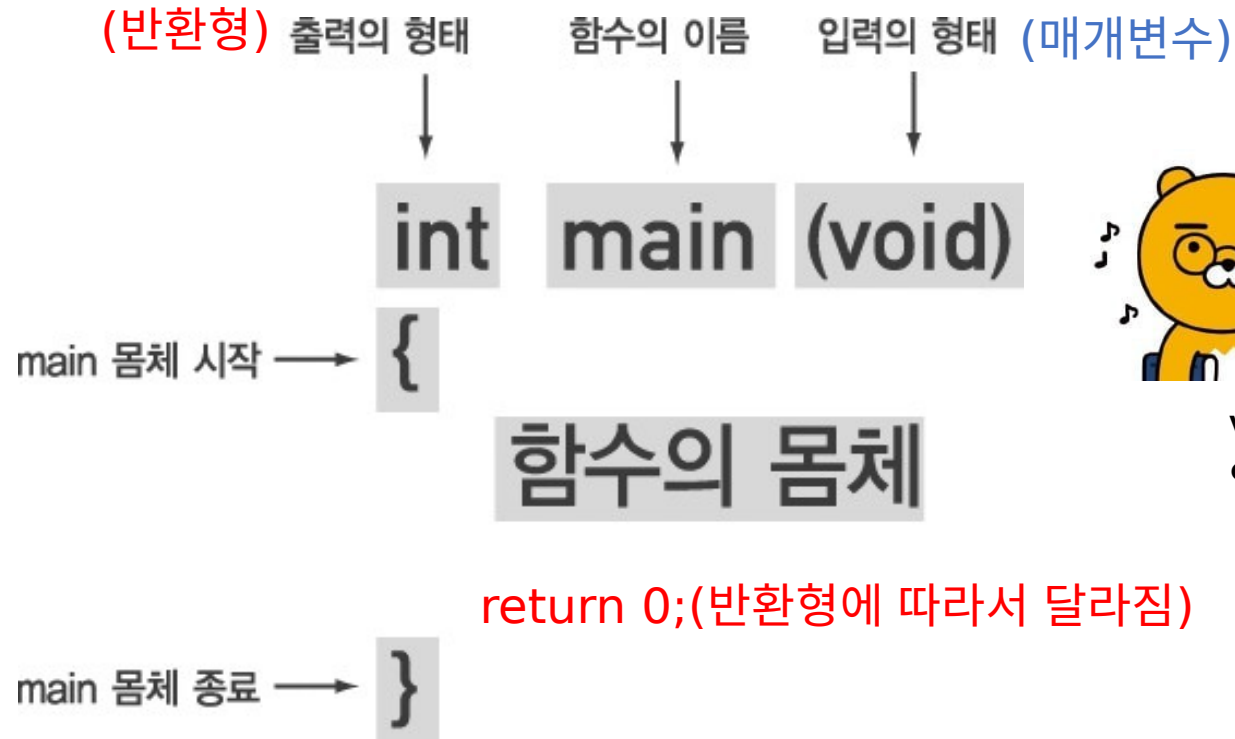
# 전처리 문장(#include)

- **#include** <파일명> or **#include** “경로 및 파일명”
  - #include: 지정된 파일의 내용을 소스코드에 포함하라는 의미
  - 컴파일러에게 프로그램에서 사용될 라이브러리를 알려주는 역할
  - <>
    - 컴파일러가 미리 지정된 위치에서 해당 파일을 찾음
  - “”
    - 사용자가 지정한 위치에서 해당 파일을 찾음
- **stdio.h**
  - Standard input out을 위한 헤더파일
    - 표준 입출력을 위한 함수를 제공하는 역할(printf, scanf,... etc)
    - 참고: [https://en.wikibooks.org/wiki/C\\_Programming/stdio.h](https://en.wikibooks.org/wiki/C_Programming/stdio.h)



# main()함수

- main()함수의 구조



생각하기!

void 형식의 함수를 선언할 수 있으나,  
일반적으로 main()의 반환형을 int로 선언하는 이유?

시스템(운영체제)에서 함수의 반환 값을  
보고 프로그램이 정확히 실행되었는지  
판단

# main()함수 내용 printf()

- printf() 모니터에 출력하는 역할
  - (“ ”)안에 있는 내용을 모니터로 출력
  - \n(new line):
    - 줄을 바꾸는 문자(개행문자)
- ;(semicolon) :
  - C언어에서 문장의 끝을 알려주는 역할
  - 모든 문장은 반드시 ‘;’으로 끝나야 함

```
int main(void){  
    printf("hello hustar\n");  
    return 0;  
}
```

Parameter(인자):  
함수에서 전달되는 변수 or 데이터

# printf() 실습

- 실습

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

```
int main(){  
    printf("Hello Everbody\n");  
    printf("%d\n", 1234);  
    printf("%d %d\n", 10, 20);  
    return 0;  
}
```

printf()는 형식을 맞추어서 모니터에 출력  
출력할 내용을 항상 “”로 묶어야 함

정수를 출력하기 위해서는 %d의 서식 문자  
(conversion)를 사용해야함

10이 첫번째 %d에 대응 되어 출력  
20이 두번째 %d에 대응 되어 출력

# printf()

- 실습을 참고하여 아래와 같이 출력하는 프로그램을 작성

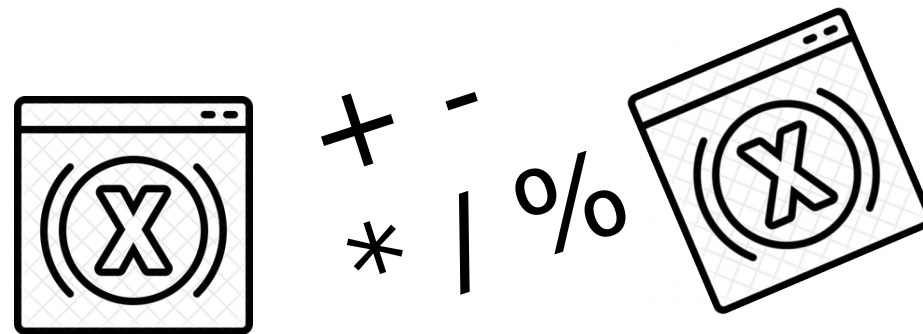
```
Hello Hustar  
이름: 휴동이  
나이:25 출석은 총5일 하였습니다.
```

출력결과가 아래와 같이 나올 경우 인코딩 방식을  
EUC-KR로 변경해야 함

```
Hello Hustar  
?대짖: ?대륙??  
?생싯:25 異싯 짖? 珒????생??듬뺏??
```

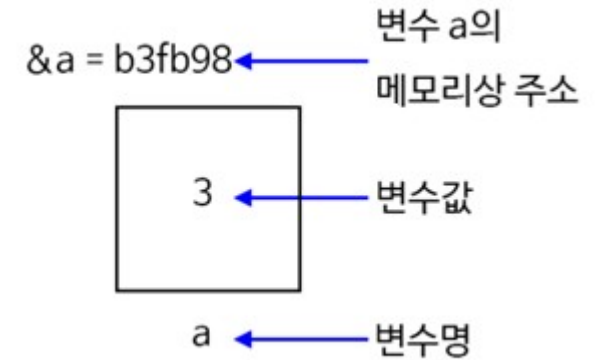


## 변수 와 연산자



# 변수(variable)

- 변수(variable)
  - 수학: 임의의 값을 대입할 수 있는 문자
  - C언어: 값을 저장할 수 있는 **메모리 공간에 붙은 이름**



## • 변수 선언 방법

| 데이터형식  |                                                               |
|--------|---------------------------------------------------------------|
| 데이터 형식 | 키워드                                                           |
| 정수형    | <b>int</b> , short int, unsigned int, long int, long long int |
| 실수형    | <b>float</b> , <b>double</b> , long double                    |
| 문자형    | <b>char</b>                                                   |

## 변수명;

### 변수명 선언 방법

- ✓ 변수의 이름은 알파벳, 숫자, 언더바(\_)로 구성
- ✓ 대소문자 구분
- ✓ 변수의 시작은 알파벳, 언더바(\_)로 할 것
- ✓ 이름 사이에 공백이 삽입될 수 없음
- ✓ 키워드는 사용할 수 없음(int, float, char ...)

### TIP.

1999년에 발표된 C99표준에서는 변수의 선언위치에 아무런 제한을 두지 않고 있다.

# 변수(variable)

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

```
int main(void){
```

```
    int num;
```

```
    num=20;
```

```
    return 0;
```

```
}
```

변수 선언 문장

**int** 정수의 저장이 가능한 메모리 공간이 필요  
**num** 메모리 공간의 이름을 num이라고 함

num이라는 변수에 20 저장

C89 버전: 변수 선언은 반드시  
함수의 시작 부분 이어야 함

C99 버전: 변수의 선언을 반드시  
함수의 시작에서 할 필요 없음

- 변수 선언: 변수를 만드는 것을 의미
- 변수는 선언하고 반드시 초기화를 해야함  
초기화하지 않은 변수는 알 수 없는 쓰레기 값이 저장됨

```
for(int a=0;a<10;a++){  
    a= a+1;  
}
```

# 변수(variable)- 실습

```
/*  
    변수 선언 연습  
    작성자:  
    작성일:  
*/  
  
#include <stdio.h>  
  
int main(void){  
    int num1, num2;  
    int num3=30, num4=40;  
  
    printf("num1: %d, num2: %d \n", num1, num2);  
    num1=10;  
    num2=20;  
  
    printf("num1: %d, num2: %d \n", num1, num2);  
    printf("num3: %d, num4: %d \n", num3, num4);  
  
    return 0;  
}
```



# C의 변수

- 다음 중 C언어 변수로 사용 가능한 것은?

- int 7ThVal;
- int phone#;
- int your name;
- int \_int;

변수에 계산 결과 값을 저장하고  
출력하는 예제 작성

```
/* example of variable and operator  
   file name: simpleadd.c */  
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int result;                //변수 선언
```

```
    result=3+4;                //덧셈 결과 저장
```

```
    printf("덧셈 결과 : %d \n", result);
```

```
    printf("%d 더하기 %d는 %d 입니다. \n", 3, 4, result);
```

```
    printf("변수 result에 저장된 값 : %d \n", result);
```

```
    return 0;
```

```
} Hustar로봇아카데미
```

# 연산자(Operator)

- result=3+4;
  - result 변수에 3과 4를 더한 결과를 저장
  - '+' 더하기 연산자와 '=' 대입연산자가 사용되었음
- C언어 연산자의 종류

| 연산자 기호                         |                                |                                        |
|--------------------------------|--------------------------------|----------------------------------------|
| 산술연산자<br>(arithmetic operator) | 단항(unary)                      | -, +, ++, --                           |
|                                | 이항(binary)                     | +, -, *, /, %                          |
|                                | 대입<br>(assignment)<br>Compound | =<br>+=, -=, *=, /=, % =               |
| 관계연산자(relational operator)     |                                | < , <= , == , != , >= , >              |
| 논리연산자(logical operator)        |                                | && ,    , !                            |
| 삼항연산자(ternary operator)        |                                | ? :                                    |
| 비트연산자(bitwise operator)        |                                | & ,   , ^ , ~ , << , >>                |
| 기타연산자                          |                                | (콤마) , sizeof , pointer ( * , & ) , -> |

# 산술연산자(arithmetic operators)

- 단항연산자(unary)

- 부호형(singed): +1, -1과 같이 operand(피연산자의)의 부호를 설정
- 증감연산자
  - 전위형(prefix): ++a(a= a+1), --a(a= a-1)와 같이 사용
  - 후위형(postfix): a++(a=a+1), a--(a=a-1)와 같이 사용
  - prefix(선증가(선감소), 후연산)
  - postfix(선연산, 후증가(후감소))

| 연산자 | 연산의 예    | 의미  | 결합성 |
|-----|----------|-----|-----|
| +   | +i       | +   | ☒   |
| -   | -i       | -   | ☒   |
| ++  | a++, ++a | 1증가 | ☒   |
| --  | a--, --a | 1감소 | ☒   |

# 산술연산자(arithmetic operators)

- 전위, 후위 연산자 예시

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

```
int main(void){  
    int var=0, a=1;
```

```
    var = ++a;    //변수 var의 값은 2, 변수 a의 값은 2 (변수 a의 값을 1 증가 후 var에 대입)  
    printf("%d:%d\n", var,a);  
    var = a++;    // 변수 var의 값은 2, 변수 a의 값은 3 (변수 a의 값을 var에 더한 후 1 증가)  
    printf("%d:%d\n",var,a);  
    var = --a;    // 변수 var의 값은 2, 변수 a의 값은 2 (변수 a의 값을 1감소 후 var에 대입)  
    printf("%d:%d\n",var,a);  
    var = a--;    // 변수 var의 값은 2, 변수 a의 값은 1(변수 a의 값을 var에 대입 후 1 감소)  
    printf("%d:%d\n",var,a);
```

```
    return 0;
```

```
}
```

```
./main  
2:2  
2:3  
2:2  
2:1
```

# 산술연산자(arithmetic operators)

- 이항연산자 예시

| 연산자 | 연산의 예 | 의미  | 결합성   |
|-----|-------|-----|-------|
| =   | a=20  | 대입  | Right |
| +   | a=4+3 | 덧셈  | Left  |
| -   | a=4-3 | 뺄셈  | Left  |
| *   | a=4*3 | 곱셈  | Left  |
| /   | a=4/3 | 나눗셈 | Left  |
| %   | a=4%3 | 나머지 | Left  |

산술 연산 예시

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

```
int main(void) {  
    int num1=9, num2=2;
```

```
    printf("%d+%d=%d \n", num1, num2,  
           num1+num2);
```

```
    printf("%d-%d=%d \n", num1, num2, num1-  
           num2);
```

```
    printf("%d*%d=%d \n", num1, num2,  
           num1*num2);
```

```
    printf("%d/%d=%d \n", num1, num2,  
           num1/num2);
```

```
    printf("%d%%%d=%d \n", num1, num2,  
           num1%num2);
```

```
    return 0;  
}
```

결과

```
./main  
9+2=11  
9-2=7  
9*2=18  
9/2=4  
9%2=1
```

# 산술연산자(arithmetic operators)

- 대입연산자: **=**
  - 오른쪽에 있는 피연산자(r-value, operand)를 왼쪽(l-value)의 변수에 대입하는 의미
  - l-value에는 반드시 값의 대입이 가능한 변수를 사용
  - 예시) 대입 연산자를 사용한 대입문  
변수명 = 값;  
변수명 = 변수;    `int x = 2;`  
변수명 = 수식;    `x = x * 3 + 2;`

# 산술연산자(arithmetic operators)

- 복합대입연산자(compound operator):
  - 우선 순위가 낮아 사용시 유의

| 복합 대입 연산자 | 의미        | 결합성   |
|-----------|-----------|-------|
| a += b    | a = a + b | right |
| a -= b    | a = a - b |       |
| a *= b    | a = a * b |       |
| a /= b    | a = a / b |       |
| a %= b    | a = a % b |       |

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

```
int main(void){  
    int num1=2, num2=4, num3=6;  
    num1 += 3;  
    num2 *= 4;  
    num3 %= 5;  
  
    printf("result: %d, %d, %d \n", num1, num2, num3);  
    return 0;  
}
```

결과 result: 5, 16, 1

# 관계연산자(relational operators)

- 관계연산자:
  - 데이터의 대소와 동등의 관계를 비교하는 연산자
  - 연산결과: True(1) 또는 False(0)
  - 산술연산자보다 우선순위가 낮음

| 연산자 | 연산의 예 | 의미             | 결합성  |
|-----|-------|----------------|------|
| <   | a<b   | a가 b보다 작은가     | Left |
| >   | a>b   | a가 b보다 큰가      |      |
| ==  | a==b  | a와 b가 같은가      |      |
| !=  | a!=b  | a와 b가 같지 않은가   |      |
| <=  | a<=b  | a가 b보다 작거나 같은가 |      |
| >=  | a>=b  | a가 b보다 크거나 같은가 |      |

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

```
int main(){  
    int a=2, b=1;  
  
    printf("%d",a<b);  
    printf("%d",a>b);  
    printf("%d",a==b);  
    printf("%d",a!=b);  
    printf("%d",a<=b);  
    printf("%d",a>=b);  
  
    return 0;  
}
```

실행결과  
0  
1  
0  
1  
0  
1  
1



# 논리연산자(logical operators)

- 논리연산자:
  - AND, OR, NOT을 계산하는 연산자
  - 연산 결과: True(1) 또는 False(0)

| 연산자 | 연산예<br>시 | 의미                              | 결합성  |
|-----|----------|---------------------------------|------|
| &&  | a&&b     | a, b 모두 true면 true 리턴           | left |
|     | a  b     | a, b 중 하나라도<br>true면 true 리턴    | left |
| !   | !a       | true면 false를,<br>false면 true 리턴 | left |

```
#include <stdio.h>

int main(){
    int num1=10, num2=12;

    printf("result of &&: %d\n", (num1==10 && num2==12));
    printf("result of ||: %d\n", (num1<12 || num2>12));
    printf("result of !: %d", !num1);

    return 0;
}
```

결과  
result of &&: 1  
result of ||: 1  
result of !: 0

C언어는 0이 아닌 모든 값을 참(true)로 간주

# 논리연산- Short-circuit

- Short-circuit evaluation

- 연산이 필요 없는 경우 연산을 생략

`expr1 && expr2`

만약 expr1의 연산 결과가 false일 경우 `expr2`의 결과에 상관 없이 전체 수식의 결과는 false로 판단됨,

따라서 expr2의 연산을 생략함

`expr1 || expr2`

만약 expr1의 연산 결과가 true일 경우 `expr2`의 결과에 상관 없이 전체 수식의 결과는 `true`로 판단됨,

따라서 expr2의 연산을 생략함

```
int main(){
    int i, j, result ;

    result = ( i = 1 ) && ( j = 2 ) ;
    printf("result = %d, i = %d, j = %d\n", result, i, j) ;
    result = ( i = 0 ) && ( j = 3 ) ;
    printf("result = %d, i = %d, j = %d\n", result, i, j) ;

    result = ( i = 0 ) || ( j = 0 ) ;
    printf("result = %d, i = %d, j = %d\n", result, i, j) ;
    result = ( i = 2 ) || ( j = 5 ) ;
    printf("result = %d, i = %d, j = %d\n", result, i, j) ;
}
```

실행결과

```
result = 1, i = 1, j = 2
result = 0, i = 0, j = 2
result = 0, i = 0, j = 0
result = 1, i = 2, j = 0
```

# 연산자 우선순위(expression evaluation)

- 연산자 우선순위
  - 연산순서를 결정짓는 순위
- 연산자 결합법칙(Associativity, 결합성)
  - 우선순위가 동일한 두 연산자가 하나의 수식에 있는 경우, 어떤 연산을 우선하여 실행할 것인지 결정해 놓은 것

| Precedence                                  | Associativity |
|---------------------------------------------|---------------|
| ( ) ++ (postfix) -- (postfix)               | left □        |
| + (unary) - (unary) ++ (prefix) -- (prefix) | right □       |
| * / %                                       | left □        |
| + -                                         | left □        |
| < <= > >=                                   | left □        |
| == !=                                       | left □        |
| &&                                          | left □        |
|                                             | left □        |
| ? :                                         | right □       |
| = += -= *= /= etc                           | right □       |

# 연산자 우선순위(expression evaluation)

|   |                             |                                         |
|---|-----------------------------|-----------------------------------------|
|   |                             | $a = b += c++ - d + e / -f$             |
| 1 | Postfix                     | $a = b += (c++) - d + e / -f$           |
| 2 | Prefix and unary from right | $a = b += (c++) - d + e / (-f)$         |
| 4 | multiplicative              | $a = b += (c++) - d + (e / (-f))$       |
| 5 | Additive from left          | $a = b += ((c++) - d) + (e / (-f))$     |
| 6 | Additive                    | $a = b += (((c++) - d) + (e / (-f)))$   |
| 7 | Assignment from right       | $a = (b += (((c++) - d) + (e / (-f))))$ |
| 8 | Assignment                  | $a = (b += (((c++) - d) + (e / (-f))))$ |

Robotics



input



# 입출력과 데이터형식(자료형)

# 데이터의 출력

- 표준 출력

- Terminal에서 text를 출력하는 것
- printf()를 이용하여 출력 하는 것이 가장 대표적
  - %d는 10진수의 정수의 입출력을 의미하는 “서식지정자(conversion specifier)”

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\sori>_
```

printf( “%d” ,  
val );

↓                      ↓

val변수가 가지고 있는      정수를 저장하고  
정수를 출력하는            있는 val변수  
서식지정자

# 데이터 입력

PS C:\Users\sori\Documents\Sample9\ProgramExamp1  
24시간 입력: 1100

- 표준 입력

- Terminal에서 키보드로 데이터를 입력
- 예시) scanf 함수를 이용한 정수의 입력
  - & 연산자는 메모리에서 변수의 주소를 의미
  - scanf의 경우 형식을 지정하여 데이터를 입력할 때 사용
  - scanf외에도 다양한 표준입력 함수가 있음(gets(), getchar(),...

변수 val에 저장하라.

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

10진수 정수형으로 입력 받아서

앰퍼센드  
(ampersand)

Hustar로봇아카데미

# scanf()예시

```
#include <stdio.h>
int main(){
    int result;
    int num1, num2;

    printf("first integer :");
    scanf("%d",&num1);
    printf("second integer :");
    scanf("%d", &num2);

    result = num1+num2;
    printf("%d + %d = %d\n", num1, num2, result);

    return 0;
}
```

실행결과

```
first integer :1
second integer :2
1 + 2 = 3
```



# 서식지정자(Conversion specifier)

- 데이터를 출력하기 위해서는 scanf/printf함수를 사용
- 서식지정자
  - scanf()/printf() 함수에서 서식화 된 입/출력을 지원하기 위해 만들어진 특수 문자
  - 많이 사용되는 서식지정자

- %d(정수)
- %f(실수)
- %c(?)



%c는 어떤 자료형의 지정자일까?

%c: 문자

데이터형식에 따라서  
서식지정자가 달라짐

```
scanf("%d-%f",&num1, &fnum1);
```

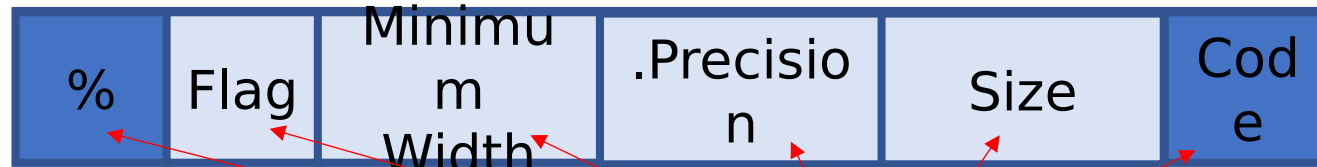
```
printf("integer: %d / float: %f", num1, fnum1);
```

“ ”에서 서식을 만들고 해당 서식지정자에 값을 대입

# 서식지정자

## • 서식지정자의 구성

- `printf(“%+5.3ld”, num);` Minimum width: 최소 출력 넓이



`%+5.3ld`

### Precision:

- 실수의 출력에서는 소수점 이하의 몇 번째 자리까지 출력할 것인지 지정
- 정수(문자열)의 출력에서는 화면에 최대 출력할 문자나 숫자의 개수를 의미

| Flag | 의미                                           |
|------|----------------------------------------------|
| -    | 왼쪽으로 정렬하여 출력                                 |
| +    | 부호를 포함하여 출력                                  |
| 0    | 오른쪽 정렬로 출력 시 Minimum width의 빈 공간을 0으로 채워서 출력 |
| 공백   | 양수일 때 부호 생략, 음수일 때 - 표시                      |
| #    | 지정된 진수 형식에 따라 C언어에서 사용되는 진수표현 형태로 값을 나타냄     |

| Size | 의미                                                               |
|------|------------------------------------------------------------------|
| h    | “%d, %i, %u, %o, %x” 와 함께 Short int자료형 출력                        |
| l    | “%d, %i, %u, %o, %x” 와 함께 Long 자료형 출력<br>%f 와 double 형식 출력 “%lf” |
| ll   | Long long int 자료형 출력<br>“%lld”                                   |
| L    | Long double 자료 형식 출력                                             |

# 다양한 서식지정자

| Data Type              | Format Specifier |
|------------------------|------------------|
| int                    | %d, %i           |
| char                   | %c               |
| float                  | %f               |
| double                 | %lf              |
| short int              | %hd              |
| unsigned int           | %u               |
| long int               | %li              |
| long long int          | %lli             |
| unsigned long int      | %lu              |
| unsigned long long int | %llu             |
| signed char            | %c               |
| unsigned char          | %c               |
| long double            | %Lf              |

# 서식지정자 활용 예시

```
#include <stdio.h>
int main(){
    int num;
    float fnum;
    double dnum;

    printf("integer-float-double: ");
    scanf("%d-%f-%lf",&num,&fnum,&dnum);

    printf("integer:%d, float:%f, double:
%lf",num, fnum, dnum);

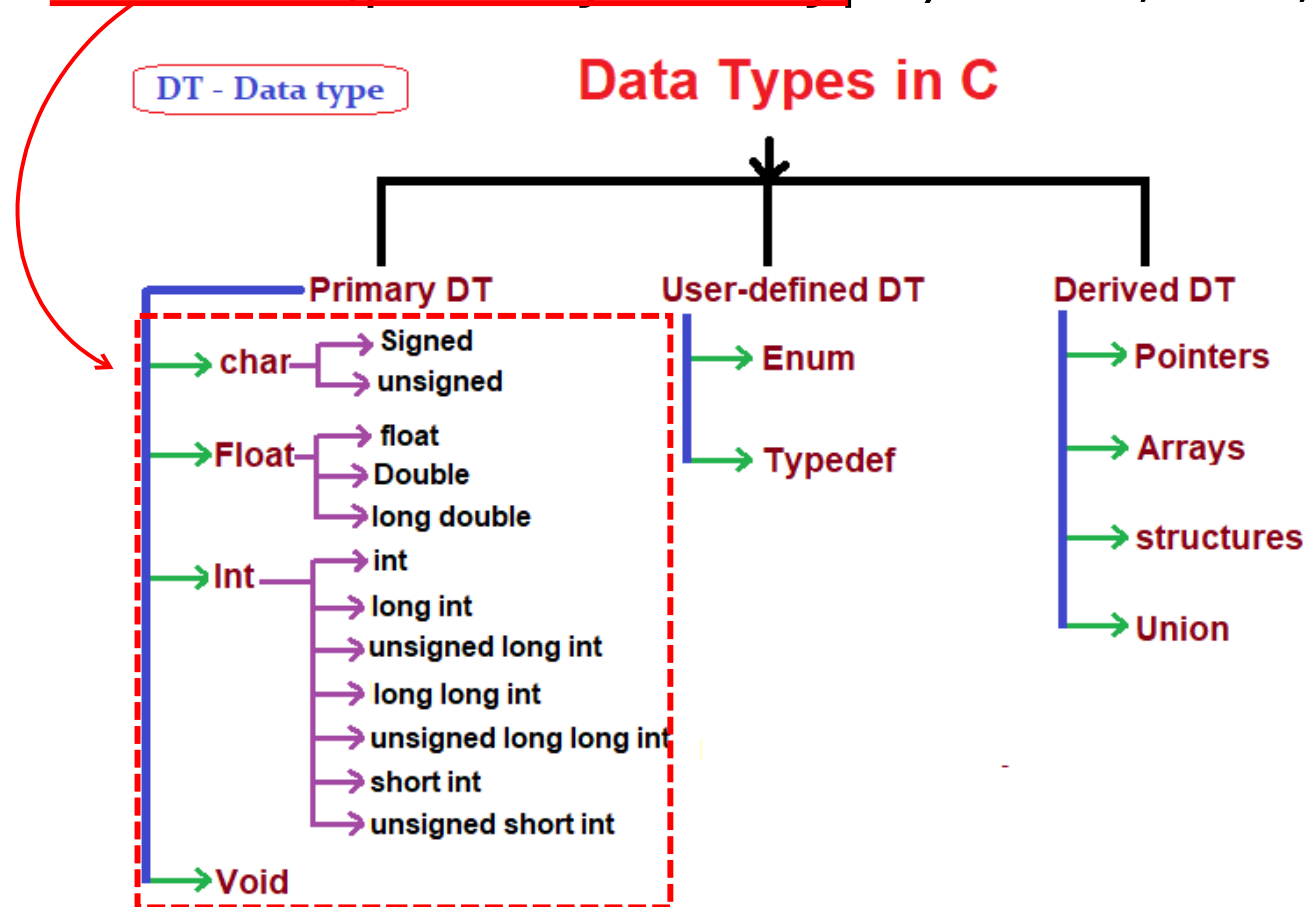
    return 0;
}
```

실행결과

```
integer-float-double: 12-1.2-2.5
integer:12, float:1.200000, double:2.500000
```

# 데이터 형식

- 데이터 형식(자료형) → 컴퓨터에게 어떤 자료를 사용할 것인지 알려주는 역할
  - 컴퓨터에서 데이터를 표현/저장 하기위한 형식
  - C언어의 기본 자료형(primary datatype)은 정수, 실수, 문자로 구분



# 데이터형식

- 정수(Integer)
  - long int(unsigned long int)
  - short int(unsigned short int)
  - int(unsigned int)
- 실수(real number)
  - float
  - double
  - long double
- 문자(character)
  - char
- 논리(logic)
  - \_Bool (C99)

# 정수(integer)

- 정수

- 정수는 signed(부호형)와 unsigned(비부호형)로 나누어짐
- Signed integer는 부호비트로 +(positive), -(negative)로
- Unsigned integer는 0~(positive)의 숫자만 표현
- C에서 정수의 기본 형식은 signed int(4byte)

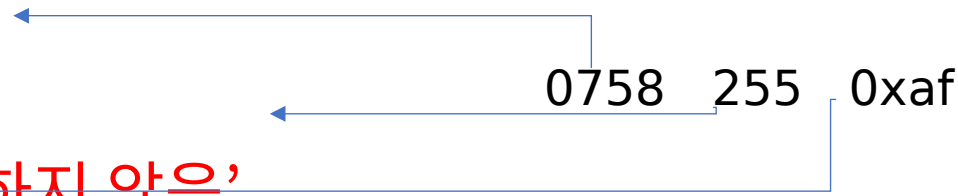
**Signed Integer**  
+/- | number |  
1 0 0 0 0 0 0 1

| Type                 | Smallest Value                              | Largest Value                               |
|----------------------|---------------------------------------------|---------------------------------------------|
| short int (2byte)    | -32,768( $-2^{15}$ )                        | 32,767( $2^{15}-1$ )                        |
| unsigned short int   | 0                                           | 65,535( $2^{16}-1$ )                        |
| <b>int (4byte)</b>   | <b>-2,147,483,648(<math>-2^{31}</math>)</b> | <b>2,147,483,647(<math>2^{31}-1</math>)</b> |
| unsigned int         | 0                                           | 4,294,967,295( $2^{32}-1$ )                 |
| long int (4byte)     | -2,147,483,648( $-2^{31}$ )                 | 2,147,483,647( $2^{31}-1$ )                 |
| unsigned long int    | 0                                           | 4,294,967,295( $2^{32}-1$ )                 |
| long long int(8Byte) | $\sim (-2^{63})$                            | $\sim (2^{63}-1)$                           |
| unsigned long long   |                                             |                                             |

# 정수(integer)

- 정수는 8진수(octal), 10진수(decimal), 16진수(Hexadecimal)로 표현

- 8진수는 '0으로 시작'
- 10진수는 '0으로 시작하지 않음'
- 16진수는 '0x로 시작'



| 출력예시<br>Data type | Conv. specification | Example                                    |
|-------------------|---------------------|--------------------------------------------|
| short int         | %hd                 | short int s = 5;<br>printf("%hd\t", s);    |
| long int          | %ld                 | long int l = 5;<br>printf("%ld\t", l);     |
| long long int     | %lld                | long int l = 5;<br>printf("%lld\t", l);    |
| unsigned int      | %u                  | unsigned int u = 12;<br>printf("%u\t", u); |

[output]  
5 5 5 12



# 실수(float)

- 실수

- C에서 실수를 표현하는 데이터형식 float, double, long double로 형식을 구분
  - float: 소수의 정밀도가 중요하지 않은 데이터를 다룰 때(Single-precision floating-point)
  - double: 소수의 정밀도가 중요한 데이터를 다룰 때(Double-precision floating-point)
  - long double: 거의 사용되지 않음(Extended-precision floating-point)

| 형식          | 최솟값                      | 최댓값                    | Precision |
|-------------|--------------------------|------------------------|-----------|
| float       | $3.40 \times 10^{-38}$   | $3.40 \times 10^{38}$  | 6 digits  |
| double      | $1.79 \times 10^{-308}$  | $1.79 \times 10^{308}$ | 15 digits |
| long double | $3.40 \times 10^{-4932}$ | $1.7 \times 10^{4932}$ | 19 digits |

long double의 경우 시스템(CPU, 운영체제)에 따라 차이가 많이 있음

# 실수(float)

- 출력예시

| Data Type   | Conv. specifiers | Example                                          |
|-------------|------------------|--------------------------------------------------|
| float       | %f               | float s = 5.12345;    printf("%f\t", s) ;        |
|             | %e               | float s = 0.000005;    printf("%e\t", s) ;       |
|             | %g               | float s = 0.000005;    printf("%g\t", s) ;       |
| double      | %lf              | double s = 5.12345;    printf("%lf\t", s) ;      |
| long double | %Lf              | long double s = 5.12345;<br>printf("%Lf\n", s) ; |

[output]

```
5.123450    5.000000e-006    5e-006    5.123450  
5.123450
```

# 문자(char)

- 문자(char):

- 문자 하나만 있는 경우, 1byte의 크기를 가짐
- 알파벳, 숫자, 연산자, 특수문자를 표현
- 문자를 표현할 때 가장 유명한 코드

ASCII(American Standard Code for Information Interchange)코드

- ASCII코드는 8bit로 256개의 문자를 표현 가능
- 문자를 나타낼 때는 반드시 “(작은 따옴표)로 묶어야 함

```
char ch;
```

```
ch = 'a';    /* lower-case a */
ch = 'A';    /* upper-case A */
ch = '0';    /* zero          */
ch = ' ';    /* space           */
```

# 문자(char)

- ASCII코드
  - 주요코드

| 종류         | 문자상수                       | 아스키 코드값   |
|------------|----------------------------|-----------|
| 숫자 문자(10개) | '0'~'9'                    | 48~57     |
| 대문자(26개)   | 'A'~'Z'                    | 65~90     |
| 소문자(26개)   | 'a'~'z'                    | 97~122    |
| 특수 문자(33개) | ' '(공백), '@', '#', ...     | 32~       |
| 제어문자(33개)  | '\0', '\t', '\n', '\r' ... | 0,9,10,13 |

| Dec | Hx | Oct | Char                               | Dec | Hx | Oct | Html  | Chr   | Dec | Hx | Oct | Html  | Chr | Dec | Hx | Oct | Html   | Chr |
|-----|----|-----|------------------------------------|-----|----|-----|-------|-------|-----|----|-----|-------|-----|-----|----|-----|--------|-----|
| 0   | 0  | 000 | <b>NUL</b> (null)                  | 32  | 20 | 040 | &#32; | Space | 64  | 40 | 100 | &#64; | @   | 96  | 60 | 140 | &#96;  | `   |
| 1   | 1  | 001 | <b>SOH</b> (start of heading)      | 33  | 21 | 041 | &#33; | !     | 65  | 41 | 101 | &#65; | A   | 97  | 61 | 141 | &#97;  | a   |
| 2   | 2  | 002 | <b>STX</b> (start of text)         | 34  | 22 | 042 | &#34; | "     | 66  | 42 | 102 | &#66; | B   | 98  | 62 | 142 | &#98;  | b   |
| 3   | 3  | 003 | <b>ETX</b> (end of text)           | 35  | 23 | 043 | &#35; | #     | 67  | 43 | 103 | &#67; | C   | 99  | 63 | 143 | &#99;  | c   |
| 4   | 4  | 004 | <b>EOT</b> (end of transmission)   | 36  | 24 | 044 | &#36; | \$    | 68  | 44 | 104 | &#68; | D   | 100 | 64 | 144 | &#100; | d   |
| 5   | 5  | 005 | <b>ENQ</b> (enquiry)               | 37  | 25 | 045 | &#37; | %     | 69  | 45 | 105 | &#69; | E   | 101 | 65 | 145 | &#101; | e   |
| 6   | 6  | 006 | <b>ACK</b> (acknowledge)           | 38  | 26 | 046 | &#38; | &     | 70  | 46 | 106 | &#70; | F   | 102 | 66 | 146 | &#102; | f   |
| 7   | 7  | 007 | <b>BEL</b> (bell)                  | 39  | 27 | 047 | &#39; | '     | 71  | 47 | 107 | &#71; | G   | 103 | 67 | 147 | &#103; | g   |
| 8   | 8  | 010 | <b>BS</b> (backspace)              | 40  | 28 | 050 | &#40; | (     | 72  | 48 | 110 | &#72; | H   | 104 | 68 | 150 | &#104; | h   |
| 9   | 9  | 011 | <b>TAB</b> (horizontal tab)        | 41  | 29 | 051 | &#41; | )     | 73  | 49 | 111 | &#73; | I   | 105 | 69 | 151 | &#105; | i   |
| 10  | A  | 012 | <b>LF</b> (NL line feed, new line) | 42  | 2A | 052 | &#42; | *     | 74  | 4A | 112 | &#74; | J   | 106 | 6A | 152 | &#106; | j   |
| 11  | B  | 013 | <b>VT</b> (vertical tab)           | 43  | 2B | 053 | &#43; | +     | 75  | 4B | 113 | &#75; | K   | 107 | 6B | 153 | &#107; | k   |
| 12  | C  | 014 | <b>FF</b> (NP form feed, new page) | 44  | 2C | 054 | &#44; | ,     | 76  | 4C | 114 | &#76; | L   | 108 | 6C | 154 | &#108; | l   |
| 13  | D  | 015 | <b>CR</b> (carriage return)        | 45  | 2D | 055 | &#45; | -     | 77  | 4D | 115 | &#77; | M   | 109 | 6D | 155 | &#109; | m   |
| 14  | E  | 016 | <b>SO</b> (shift out)              | 46  | 2E | 056 | &#46; | .     | 78  | 4E | 116 | &#78; | N   | 110 | 6E | 156 | &#110; | n   |
| 15  | F  | 017 | <b>SI</b> (shift in)               | 47  | 2F | 057 | &#47; | /     | 79  | 4F | 117 | &#79; | O   | 111 | 6F | 157 | &#111; | o   |
| 16  | 10 | 020 | <b>DLE</b> (data link escape)      | 48  | 30 | 060 | &#48; | 0     | 80  | 50 | 120 | &#80; | P   | 112 | 70 | 160 | &#112; | p   |
| 17  | 11 | 021 | <b>DC1</b> (device control 1)      | 49  | 31 | 061 | &#49; | 1     | 81  | 51 | 121 | &#81; | Q   | 113 | 71 | 161 | &#113; | q   |
| 18  | 12 | 022 | <b>DC2</b> (device control 2)      | 50  | 32 | 062 | &#50; | 2     | 82  | 52 | 122 | &#82; | R   | 114 | 72 | 162 | &#114; | r   |
| 19  | 13 | 023 | <b>DC3</b> (device control 3)      | 51  | 33 | 063 | &#51; | 3     | 83  | 53 | 123 | &#83; | S   | 115 | 73 | 163 | &#115; | s   |
| 20  | 14 | 024 | <b>DC4</b> (device control 4)      | 52  | 34 | 064 | &#52; | 4     | 84  | 54 | 124 | &#84; | T   | 116 | 74 | 164 | &#116; | t   |
| 21  | 15 | 025 | <b>NAK</b> (negative acknowledge)  | 53  | 35 | 065 | &#53; | 5     | 85  | 55 | 125 | &#85; | U   | 117 | 75 | 165 | &#117; | u   |
| 22  | 16 | 026 | <b>SYN</b> (synchronous idle)      | 54  | 36 | 066 | &#54; | 6     | 86  | 56 | 126 | &#86; | V   | 118 | 76 | 166 | &#118; | v   |
| 23  | 17 | 027 | <b>ETB</b> (end of trans. block)   | 55  | 37 | 067 | &#55; | 7     | 87  | 57 | 127 | &#87; | W   | 119 | 77 | 167 | &#119; | w   |
| 24  | 18 | 030 | <b>CAN</b> (cancel)                | 56  | 38 | 070 | &#56; | 8     | 88  | 58 | 130 | &#88; | X   | 120 | 78 | 170 | &#120; | x   |
| 25  | 19 | 031 | <b>EM</b> (end of medium)          | 57  | 39 | 071 | &#57; | 9     | 89  | 59 | 131 | &#89; | Y   | 121 | 79 | 171 | &#121; | y   |
| 26  | 1A | 032 | <b>SUB</b> (substitute)            | 58  | 3A | 072 | &#58; | :     | 90  | 5A | 132 | &#90; | Z   | 122 | 7A | 172 | &#122; | z   |
| 27  | 1B | 033 | <b>ESC</b> (escape)                | 59  | 3B | 073 | &#59; | :     | 91  | 5B | 133 | &#91; | [   | 123 | 7B | 173 | &#123; | {   |
| 28  | 1C | 034 | <b>FS</b> (file separator)         | 60  | 3C | 074 | &#60; | <     | 92  | 5C | 134 | &#92; | \   | 124 | 7C | 174 | &#124; |     |
| 29  | 1D | 035 | <b>GS</b> (group separator)        | 61  | 3D | 075 | &#61; | =     | 93  | 5D | 135 | &#93; | ]   | 125 | 7D | 175 | &#125; | }   |
| 30  | 1E | 036 | <b>RS</b> (record separator)       | 62  | 3E | 076 | &#62; | >     | 94  | 5E | 136 | &#94; | ^   | 126 | 7E | 176 | &#126; | ~   |
| 31  | 1F | 037 | <b>US</b> (unit separator)         | 63  | 3F | 077 | &#63; | ?     | 95  | 5F | 137 | &#95; | _   | 127 | 7F | 177 | &#127; | DEL |

Source: [www.asciitable.com](http://www.asciitable.com)

# 문자(char)

- 문자와 정수는 교환하여 표현 가능
  - 문자 'a' => 97 로 표현
  - 문자 'A' => 65로 표현
  - 문자 '5' => 53으로 표현

```
#include <stdio.h>

int main(){
    char ch ;
    int i ;

    i = 'a' ;
    printf("%d = %c \n", i, i) ;

    ch = 65 ;
    printf("%d = %c \n", ch, ch) ;

    ch = ch + 1 ;
    printf("%d = %c \n", ch, ch) ;

    ch++ ;
    printf("%d = %c \n", ch, ch) ;

    return 0;
}
```

결과  
97 = a  
65 = A  
66 = B  
67 = C

# 상수(constant)

- 프로그램에서 text로 작성되어 있는 데이터(정수, 실수, 문자 등)

```
i = 'a';          문자 상수
printf("%d = %c \n", i, 23);  정수 상수
```

- 정수 상수

- C에서 정수는 8진수, 10진수, 16진수로 표현 가능      0377      255      0x7fff
- 정수 상수의 기본 데이터 형식은 int
- L을 접미사로 사용하여 long형식으로 변경
- U을 접미사로 사용하여 unsigned형식으로 변경

```
long n=2468L;      // 상수의 데이터형식을 long으로 변경
unsigned int a=1025U; // 상수의 데이터형식을 unsigned int로 변경
unsigned long int uli=12345UL; // 상수의 데이터형식을 unsigned long int로 변경
```

# 상수(constant)

- 실수 상수

- 실수 상수의 기본 데이터형식은 double      57.      57.0      57.0e-02      .57E0
- E or e를 이용하여 지수로 표현 가능
- F를 이용하여 데이터형식을 float으로 변경

```
// F를 이용하여 데이터형식을 float으로 변경  
float fnum=35.1F;  
double dnum=35.2F;
```

- 문자 상수

- 문자 상수는 항상 “(작은따옴표)로 묶어줘야 함

‘A’ ‘C’ ‘\n’ ‘\t’ ‘+’

# 상수(constant)

- 상수의 선언
  - const 키워드를 이용하여 선언할 수 있음

```
const int MAX=100;    // MAX는 상수로 값 변경 불가  
const double PI=3.1415; // PI는 상수로 값 변경 불가  
  
MAX=100; //상수를 변경하는 코드로 컴파일 오류
```



# 변수와 상수 크기

- sizeof(*object*) operator
  - Object에 할당되어 있는 메모리의 크기를 byte단위로 반환

```
printf("char = %d\n", sizeof(char)) ;  
printf("integer = %d\n", sizeof(int)) ;  
printf("float = %d\n", sizeof(float)) ;  
printf("double = %d\n", sizeof(double)) ;  
printf("integer = %d\n", sizeof(4)) ;  
printf("double = %d\n", sizeof(4.0)) ;
```

실행결과

```
char = 1  
integer = 4  
float = 4  
double = 8  
integer = 4  
double = 8
```

# 자료형 변환

- 자료형 변환

- 데이터의 표현 방식을 변경하는 것

- 자동 형 변환 (묵시적 형 변환)

```
float fa= 13.0 + 15;
```

- Implicit type conversion

- 자동으로 데이터 형식을 변환

- 어떤 연산자가 두 형식의 데이터를 계산 한다면, 두 형식 중에서 더 큰 자료형으로 데이터의 형식을 변환 하는 것

- 강제 형 변환(명시적 형 변환)

```
int ia= (int)13.5 + 15;
```

- Explicit type conversion

- 프로그래머가 강제로 데이터 형식을 변환

(자료형)변수 or 상수

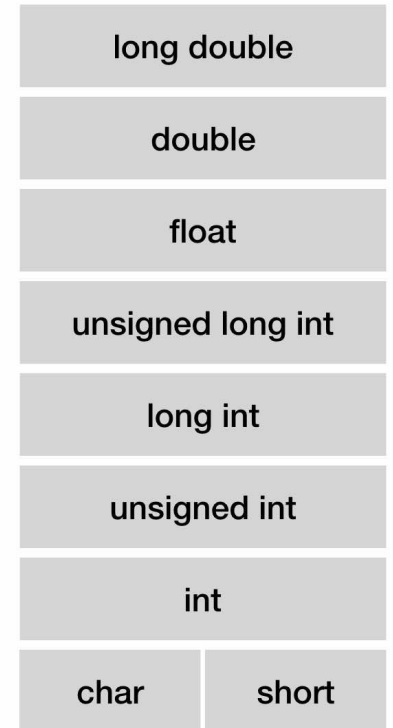
# 자동 형 변환

- 데이터의 손실이 일어나지 않는 방향으로 변환
  - Promotion(형 확장)

```
char c ;
short int s ;
int i ;
unsigned int u ;
long int l ;
float f ;
double d ;

i = i + c ;      /* c is converted to int */
i = i + s ;      /* s is converted to int */
u = u + i ;      /* i is converted to unsigned int */
l = l + u ;      /* u is converted to long int */
d = d + f ;      /* f is converted to double */
```

conversion  
hierarchy



# 강제 형 변환

- 프로그래머가 지정한 형식으로 데이터를 변환하는 것
  - 데이터 손실이 있어도 데이터 형식의 변환 가능 => cast
  - **()** => 형 변환 연산자(cast operator)

```
float f=83.56, frac_part, quotient ;
int i, dividend=5, divisor=2 ;

frac_part = f - (int) f ;           //
Assign 0.56 into frac_part
i = (int) f ;                       //Assign 83 into i

quotient = (float) dividend / divisor ;    /* 1 */
quotient = ((float) dividend) / divisor ;  /* 2 */
quotient = dividend / (float) divisor ;    /* 3 */
quotient = (float)(dividend / divisor) ;   /* 4 */
```

quotient에  
저장되는 값을  
생각해보자!



실행결과  
1~3: 2.5  
4: 2

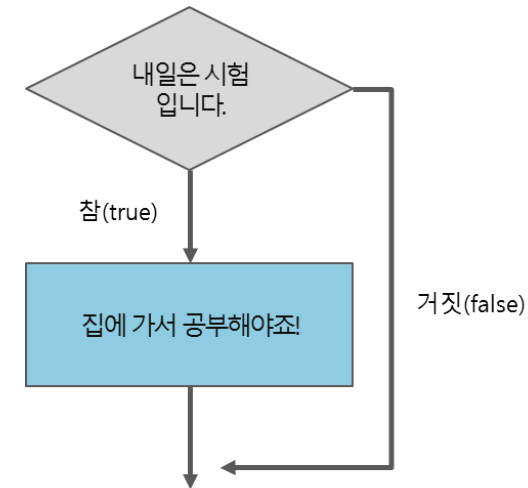
# Robotics



대경혁신인재양성프로젝트



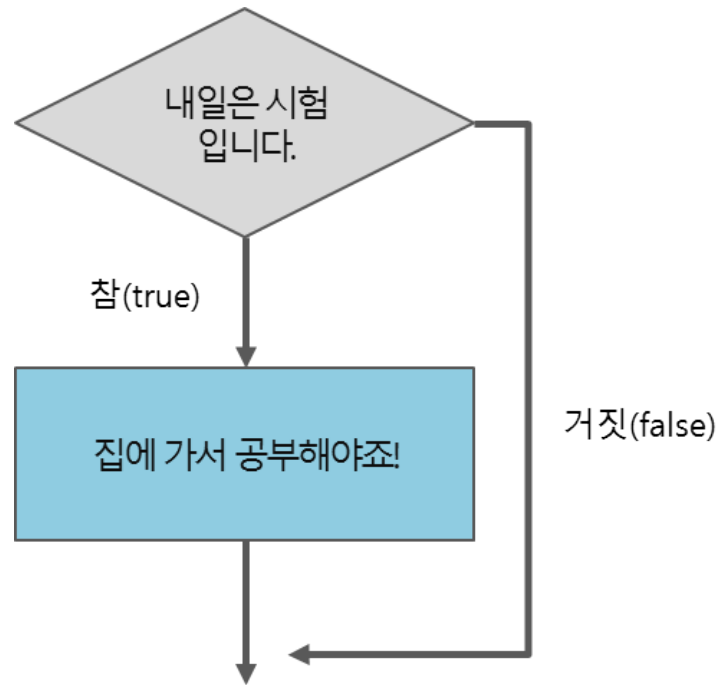
## 조건문(if, switch) Select statement



# 조건문(selection statement)

- 조건문

- 조건에 따라서 실행하는 구문(statement)이 달라지는 것
- if와 switch 구문, 삼항연산자



# if조건문

- if조건문
  - 조건식이 True/False에 따라서 구문의 실행 여부를 판단
  - if / if - else / if - else if - else 3가지의 형태가 있음
  - 조건식의 결과는 항상 True(0이외의 다른 값) or False(0)

```
if (조건식) {  
  
    실행구문  
}
```

```
if (조건식) {  
    실행구문1  
} else {  
    실행구문2  
}
```

```
if (조건식1) {  
    실행구문1  
} else if (조건식2) {  
    실행구문2  
} else {  
    실행구문3  
}
```

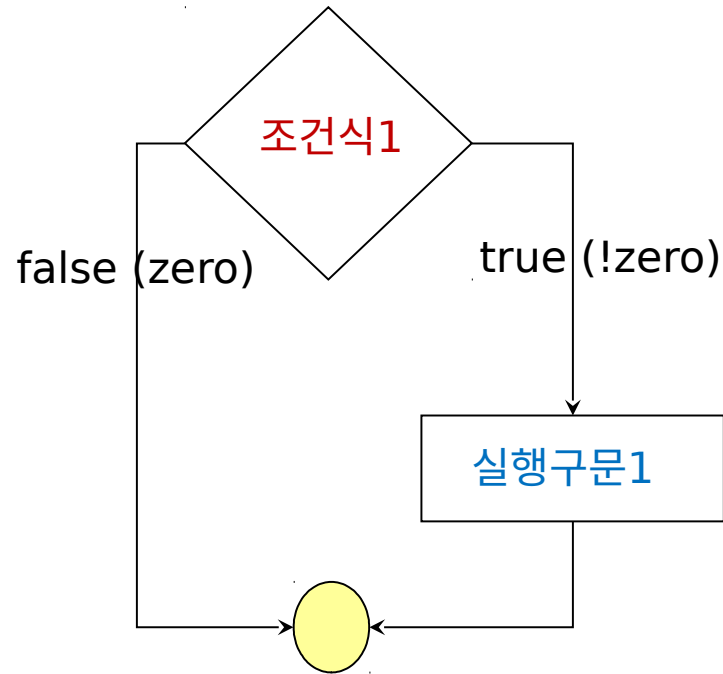
# if문

- if문

- 조건식 1이 true인 경우 실행구문1을 수행
- 실행구문이 여러 문장일 경우 반드시 {}으로 묶어야 함  
그렇지 않다면 {}생략 가능

```
if(조건식1){  
    실행구문1  
}
```

```
if(조건식1)  
    실행구문1
```



```
if(a%10){  
    printf("a is not a multiple of 10\  
n");  
    printf("a is %d.\n", a);  
}
```

```
if(a>10)  
    printf("a is greater than 10\  
n");
```

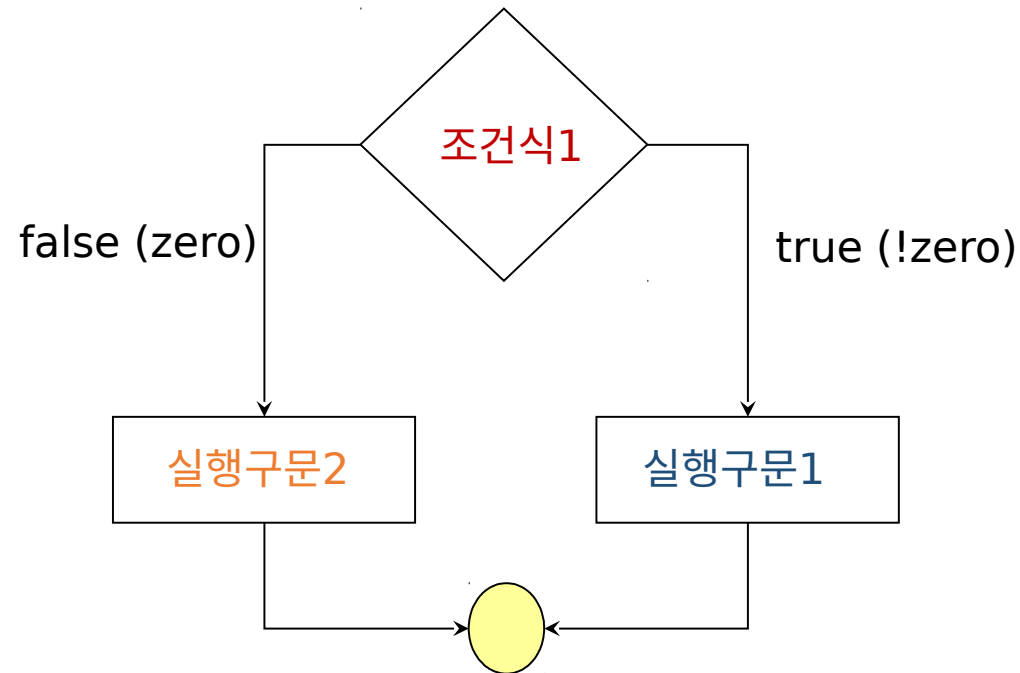


# if-else 문

- If-else

- 조건식1이 true인 경우 실행구문1을 수행, 그렇지 않다면 실행구문2를 수행

```
if(조건식1){  
    실행구문1  
}else{  
    실행구문2  
}
```



# if-else 문

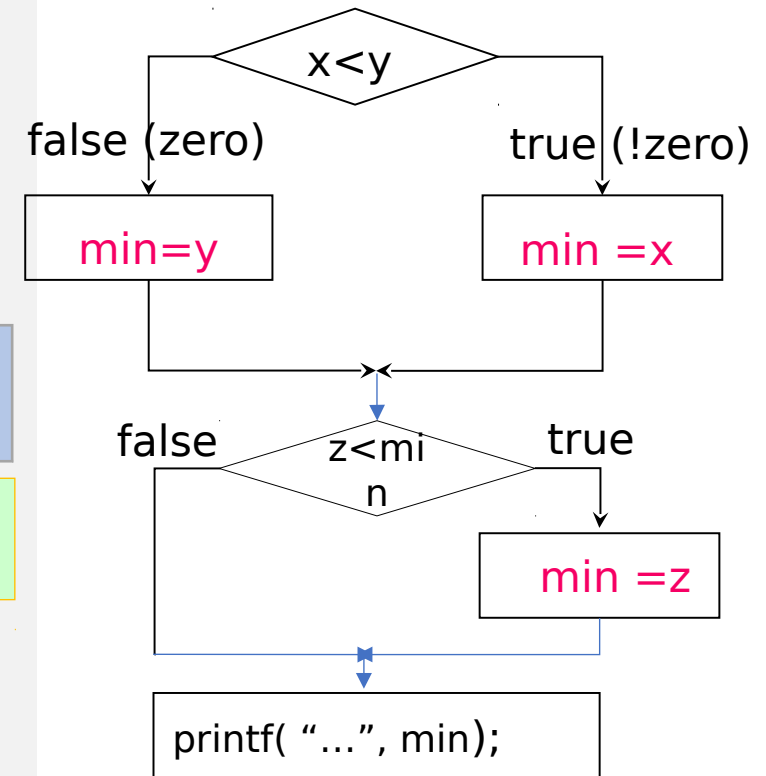
```
#include <stdio.h>    //  
find the minimum of three values
```

```
int main(){  
    int x, y, z, min;  
  
    printf("Input three integers : ");  
    scanf("%d%d%d", &x, &y, &z);  
  
    if (x < y)    min = x;  
    else        min = y;  
  
    if (z < min)    min = z;  
  
    printf("The minimum value is %d\n", min );  
  
    return 0;  
}
```

[input]  
Input three integers : 3 7

2

[output]  
The minimum value is 2

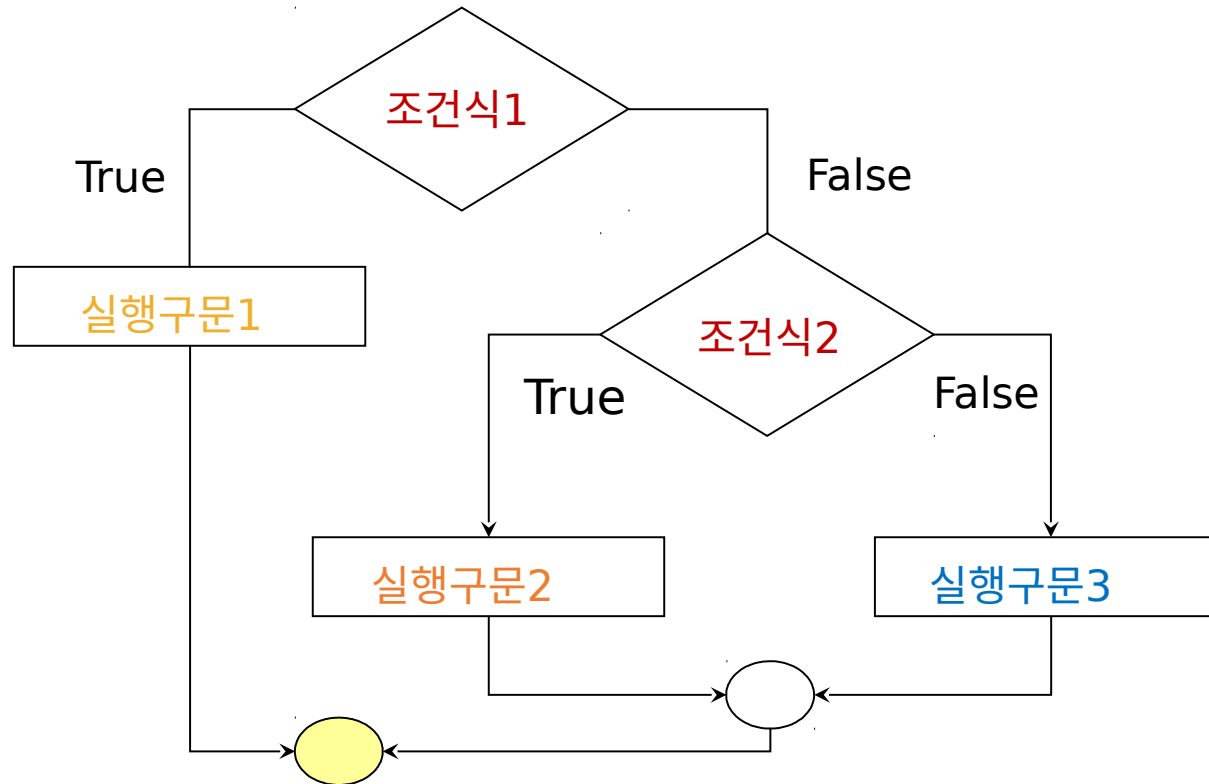


# if-else if-else 문

- If-else if-else

- 조건식1이 true인 경우 실행구문1을 수행, 그렇지 않다면 조건식2가 true인 경우 실행구문2를 수행, 그렇지 않다면 실행구문3을 수행

```
if (조건식1)
    실행구문1
else if (조건식2)
    실행구문2
else
    실행구문3
```



# if-else if-else 문

```
int c;  
scanf("%d", &c); // 정수 입력  
  
if (c == 0)          // 0인 경우  
    printf("Zero");  
else if (c < 10)      // 10 이하인 경우  
    printf("Single digit");  
else if (c < 100)     // 100 이하인 경우  
    printf("Two digit");  
else printf("Other"); // 그 외의 경우
```



프로그램의 문제점은?

-100이 입력의 경우에도  
single digit이라고 화면에 출력됨

문제점을 수정하는 프로그램 작성

# if-else if-else 문 실습

- 두 실수와 연산자를 입력 받아 연산 결과 출력 프로그램 작성

```
input two real numbers(float float): 10 20  
choose operator +, -, *, /, :-  
10.000000 - 20.000000=-10.000000
```

# 삼항연산자(ternary operator)

- 삼항연산자

조건식 ? 실행구문1 : 실행구문2

if (조건식) 실행구문1  
else 실행구문2

- ?(물음표)와 :(콜론)을 구성
- If-else문과 같은 역학
- 조건식이 true일 경우 실행구문1 수행, 그렇지 않고 조건식이 false인 경우 실행구문2 수행
- Return문과 함께 사용되는 경우도 있음

```
return i > j ? i : j;
```

```
if (i > j)
    printf("%d\n", i);
else
    printf("%d\n", j);
```

printf("%d\n", i > j ? i : j);

# 삼항연산자(ternary operator)

- 삼항 연산자를 이용하여 절대값을 구하는 프로그램

```
#include <stdio.h>

int main(){
    int num, abs;
    printf("input integer:");
    scanf("%d",&num);

    abs = num>0 ? num : num*(-1);
    printf("abs: %d\n",abs);

    return 0;
}
```

# switch문

- Switch문

```
switch(조건식){  
    case label1: 실행구문1; break;  
  
    case label2: 실행구문2; break;  
  
    default: 실행구문3;  
}
```

If-else if-else 문과 유사한  
기능 그렇지만 실행이 빠르고  
가독성이 좋음

- 여러 조건에서 해당하는 조건의 문장을 실행
- Switch - case 로 구성

- 조건식

- 반드시 괄호로 묶어야 함
    - 결과는 정수와 문자로 한정(실수형 오류)

- Label

- 정수 또는 문자의 상수만 사용할 수 있음

- Break

- switch 문을 종료하고 다음 코드를 실행하는 역할

- Default

- If문에서 else와 같은 역할, 모든 case가 아닌 경우 실행, 생략가능



Switch문 조건식에 정수와  
문자만 사용할 수 있는 이유는?

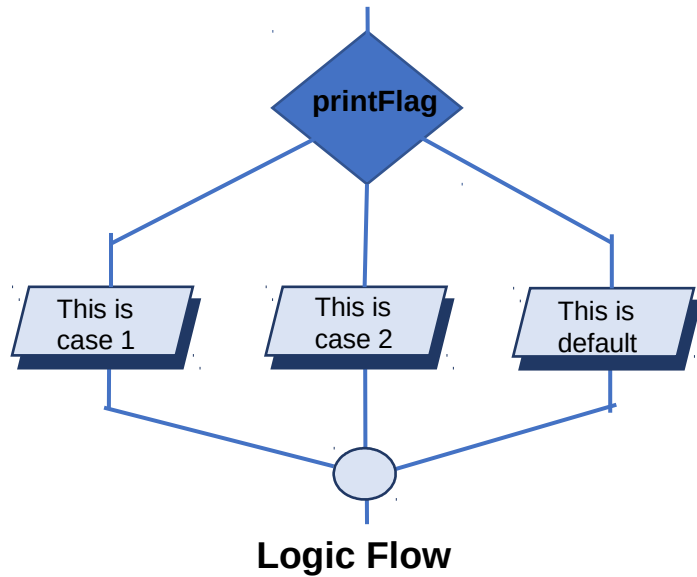
Switch문 조건식에는 원칙으로는  
정수형식만 사용할 수 있기 때문  
C에서 문자는 정수(ASCII코드)로 다루어  
지기 때문



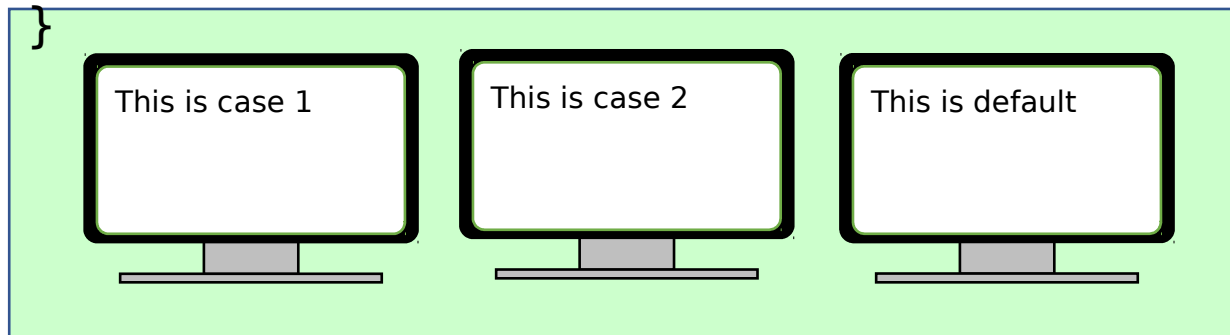
# switch문

- Switch와 break문의 관계
  - Break문이 있는 경우 break문을 실행하여 switch문을 종료

Break문이 있는 switch  
예시



```
switch (printFlag){  
    case 1 : printf("This is case  
1");  
                break;  
    case 2 : printf("This is case  
2");  
                break;  
    default : printf("This is  
default");  
                break;  
}
```



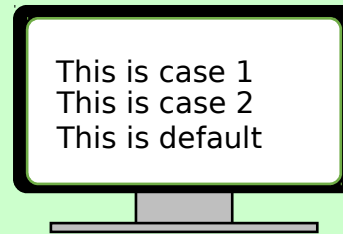
# switch문

- Switch와 break문의 관계2

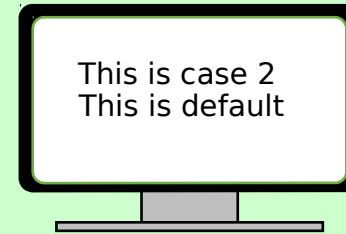
- Break문이 없는 경우 해당조건 이후의 모든 case의 실행구문을 동작

//without break

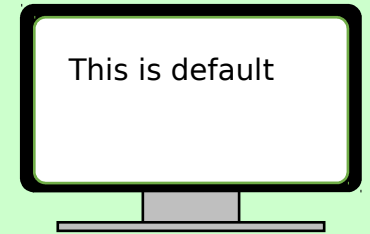
```
switch(printFlag)
{
    case 1 : printf("This is case 1\n");
    case 2 : printf("This is case 2\n");
    default : printf("This is default\n");
} //switch
```



**printFlag is 1**



**printFlag is 2**



**printFlag is not 1  
or 2**

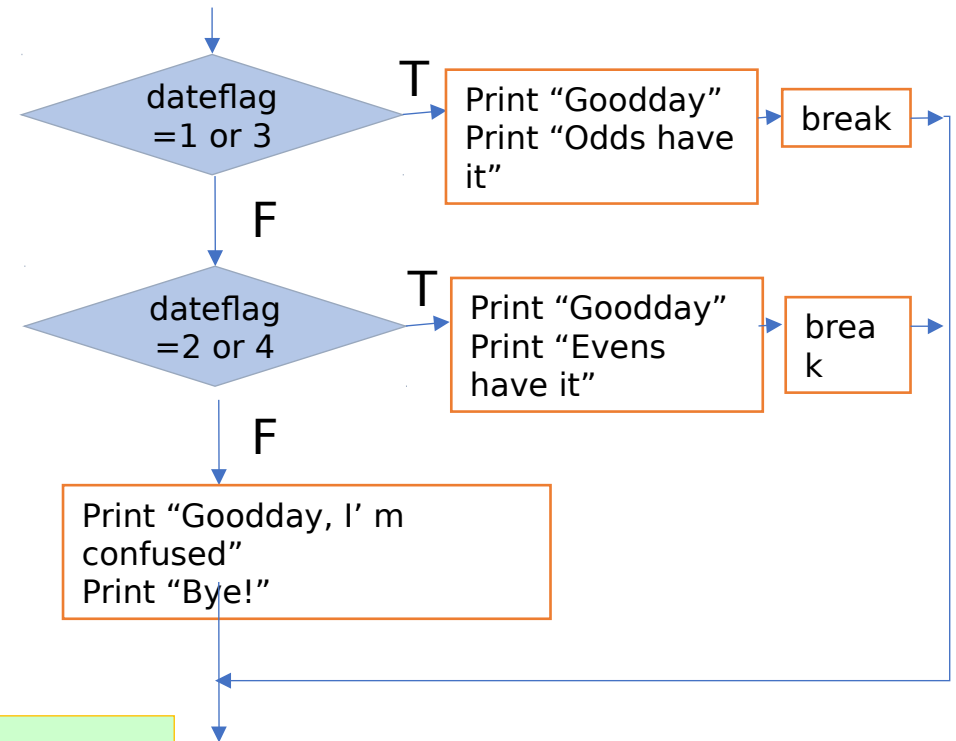
# Switch문 예시

```
/* Program fragment that demonstrates multiple cases  
for one set of statements */
```

```
int dateflag = 2;
```

```
switch (dateflag){  
    case 1 :  
    case 3 : printf("Good Day \n") ;  
             printf("Odds have it!\n") ;  
             break;  
    case 2 :  
    case 4 : printf("Good Day \n") ;  
             printf("Evens have it!\n") ;  
             break;  
    default : printf("Good Day, I'm confused \n") ;  
             printf("Bye! \n") ;  
             break;} //switch
```

[outputs]  
Good Day  
Evens have it



# Switch문 예시

If the value of grade is 4

```
switch (grade)
{
    case 4 : printf("A") ; break;
    case 3 : printf("B") ; break;
    case 2 : printf("C") ; break;
    case 1 : printf("D") ; break;
    default : printf("Illegal grade");
}
```

[output]  
A

```
switch (grade)
{
    case 4 : printf("A") ;
    case 3 : printf("B") ;
    case 2 : printf("C") ;
    case 1 : printf("D") ;
    default : printf("Illegal grade")
}
```

[outputs]  
ABCDIllegal  
grade

```
switch (grade)
{
    case 5 : case 4 : case 3 : printf("Passing"); break;
    case 2 : case 1 :      printf("Failing"); break;
    default :              printf("Illegal grade");
}
```

[outputs]  
Passing

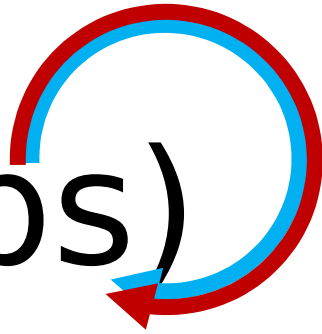
# Switch문-실습

- 11~99의 정수를 읽어주는 프로그램

11~99사이의 값을 입력하시오:14  
십사

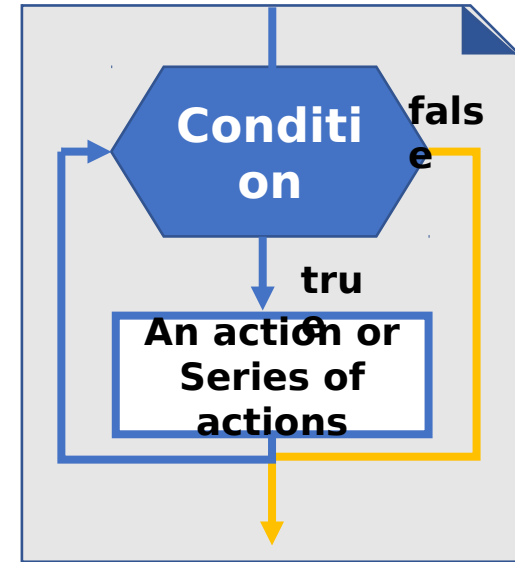


# 반복문(loops)



# 반복(loops)

- 반복문(iteration statement)
  - 같은 문장을 반복하여 실행하는 구문
  - 반복문의 종류
    - While
    - For
    - Do-while
  - 반복을 판단하는 조건식을 설정할 수 있음
    - While, for의 경우 조건식을 판단하고 반복을 수행(pretest)
    - Do-while의 경우 반복을 우선(1회) 수행하고 조건식을 확인하여 반복 판단(posttest)

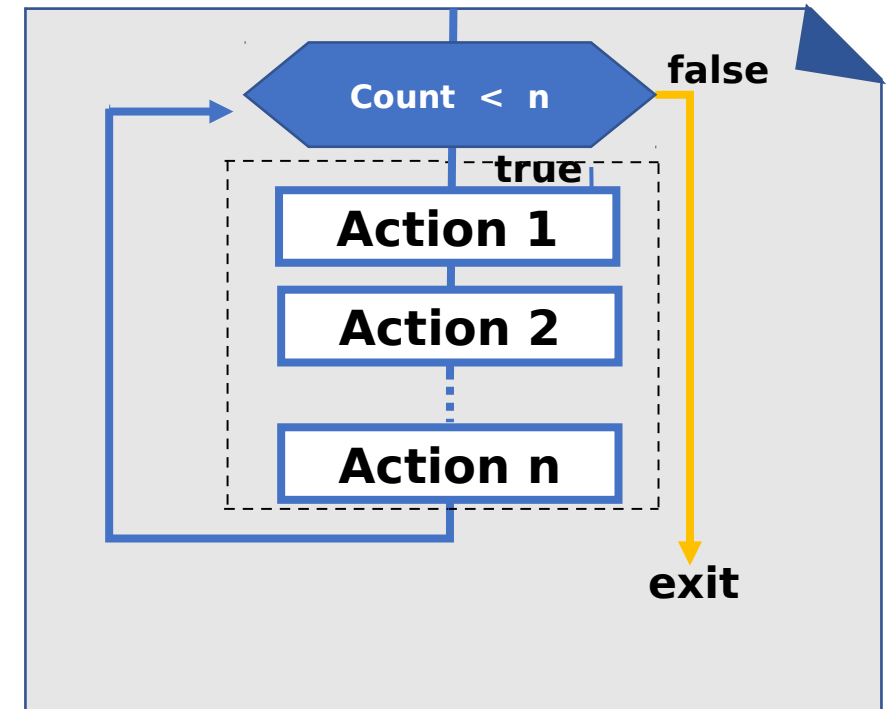


# While문

- While

```
while(조건식) {  
    실행구문;  
}
```

- 조건식이 true인 경우 실행구문을 반복
  - 조건식이 false일 때 반복을 중단





# While문 예시

```
int i, n = 10;  
;  
i = 1;  
while ( i < n)  
    i = i * 2;
```

i가 n(10)보다 작은가? 판단  
i에 i\*2의 결과 값을 저장하는 문장을 반복

i(1) < n(10)  
1 \* 2      i=2      < 10  
2 \* 2      i=4      < 10  
2 \* 4      i=8      < 10  
2 \* 8      i=16 수행

i가 16의 값을 가지면 10보다 크기 때문에  
반복을 중단

# While문 예시

- 1부터 n까지의 합

```
#include <stdio.h>

int main(){
    int i, last, sum = 0 ;

    i = 1 ;
    printf("input lastnumber n: ");
    scanf("%d", &last);

    while ( i <= last){
        sum = sum + i ;
        i = i +1;
    }

    printf("1 to %d sum is %d\n", last, sum);

    return 0;
}
```

# While문 예시

- 0이 입력될 때까지 입력되는 값을 더하는 프로그램

```
#include <stdio.h>

int main(){

    int n, sum = 0 ;

    printf("sums a series of numbers.\n") ;
    printf("Enter integers (0 to terminate) :
") ;

    scanf("%d", &n);
    while ( n != 0 ) {
        sum += n ;
        scanf("%d", &n) ;
    }
    printf("The sum is : %d\n", sum ) ;

    return 0;
```

```
sums a series of numbers.
Enter integers (0 to terminate) : 12
1
3
4
0
The sum is : 20
```

# While문 실습

- 구구단을 출력하는 프로그램. 단, 단수는 입력

몇 단? 5

$$5 * 1 = 5$$

$$5 * 2 = 10$$

$$5 * 3 = 15$$

$$5 * 4 = 20$$

$$5 * 5 = 25$$

$$5 * 6 = 30$$

$$5 * 7 = 35$$

$$5 * 8 = 40$$

$$5 * 9 = 45$$

# For문

- For문은 counter를 제어하면서 동작하는 반복문

```
for(초기화; 조건식; 증감식){  
    수행구문;  
}
```

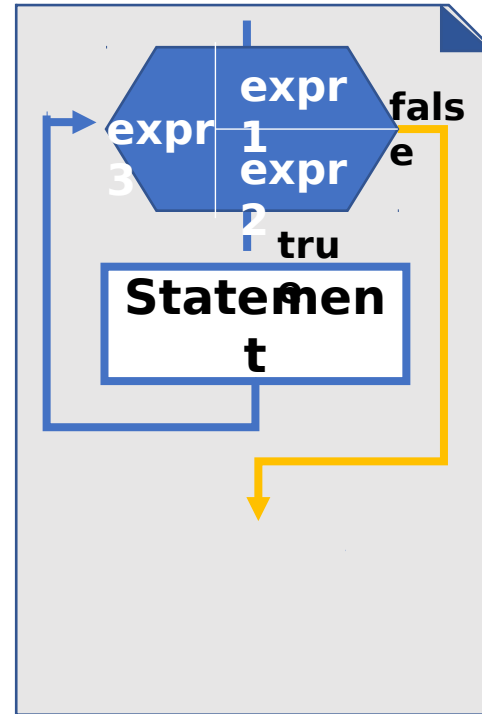
- 조건을 먼저 확인하고 true일 경우 반복(pretest)
- For문은 ()에 초기화, 조건식, 증감식 3개의 수식이 들어있는 형태
  - 각 수식을 ;(세미콜론)으로 구분
  - 초기화: counter의 초기화
  - 조건식: 반복여부를 판단
  - 증감식: counter의 증가 또는 감소
  - 수행구문: 반복실행 대상 문장

```
for(int i=0;i<3;i++){  
    printf("Hustar robot!\n");  
}
```

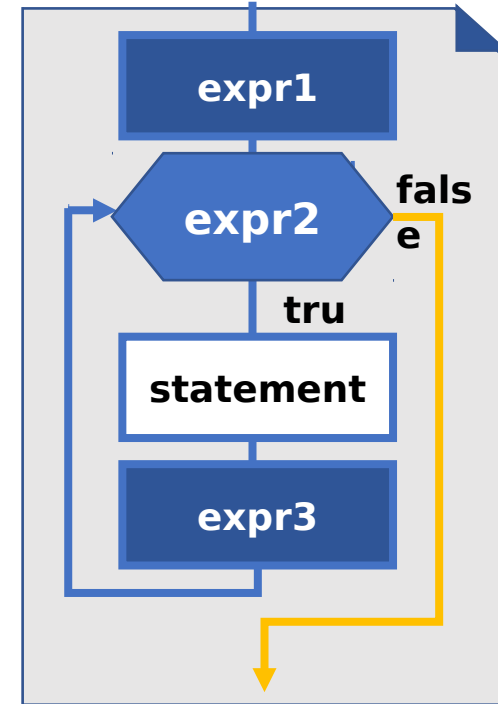
# For문

- For & while 문 비교
  - For문은 초기화(expr1)이 반복문 안에 포함
  - while문은 초기화(expr1)이 반복문 외부에 있음

```
for(int i=0; i<3; i++){  
    printf("Hustar robot!\n");  
}  
  
int i=0;  
while(i<3){  
    printf("Hustar robot!\n");  
    i++;  
}
```



for()



while()

# For문 예제

- 1부터 n까지의 합계 구하기, 단 n은 사용자 입력

```
#include <stdio.h>

int main(){
    int n, sum=0;

    printf("n까지의 합\n n을 입력하세요:");
    scanf("%d",&n);

    for(int i=1;i<=n;i++)
        sum+=i;

    printf("1부터 %d까지의 합은 %d입니다.",n,sum);

    return 0;
}
```

n까지의 합  
n을 입력하세요:5  
1부터 5까지의 합은 15입니다.

# For문 생략

- For문에서 초기화, 조건식, 증감식을 생략 가능

for ( ; i > 0 ; --i ) ...// need an initialization before for statement

for ( i = 5 ; i > 0 ; ) ...//updating statement should be in loop body

for ( ; i > 0 ; ) ...

for ( ; ; ) ...//same as for(;1;)



# For문 실습

- 입력 받은 실수의 평균을 구하는 프로그램. 단, 입력은 -(음수)가 입력 되면 중단되고 평균이 출력된다.

```
input real number(minus to quit):1
input real number(minus to quit):2
input real number(minus to quit):3
input real number(minus to quit):4
input real number(minus to quit):5
input real number(minus to quit):-1
average: 3.0
```

# do - while 문

- do - while 문

- 수행문장을 한번 수행하고 조건식을 판단하는 post-test loop

```
do{  
    수행문장;  
}while(조건식);
```

- do - while문은 언제 사용?

- 일반적으로 while문을 사용

- 조건식이 앞 부분에 위치해서 코드를 작성/이해하기 좋기 때문
- 그렇지만 “반복영역이 무조건 한 번 이상 실행 되어야 한다”는 경우 사용

```
do{  
    printf("hello hustar\  
n");  
}while(0);
```



hello hustar

```
while(0){  
    printf("hello hustar\  
n");  
}
```



출력 결과 없음

# do - while 예시

- Do - while을 while문으로 변환
  - 2단 출력

```
int i, n = 16 ;  
i = 1 ;  
  
do{  
    printf("%d\  
n", i);  
    i = i * 2 ;  
} while ( i <= n) ;
```

1  
2  
4  
8  
16

```
int i, n = 16 ;  
i = 1 ;  
  
while ( i <= n){  
    printf("%d\  
n", i);  
    i = i * 2 ;  
}
```

1  
2  
4  
8  
16

# do - while문 예시

- 자릿수 구하는 프로그램

```
/
* Calculates the number of digits in an integer */
#include <stdio.h>

int main(){
    int digits = 0, n ;

    printf("Enter a nonnegative integer : ") ;
    scanf("%d", &n) ;

    do {
        n /=10;
        digits++;
    } while ( n > 0 ) ;

    printf("The number has %d digit(s)\n", digits ) ;

    return 0;
```

Enter a nonnegative integer : 23421  
The number has 5 digit(s)

# 무한루프와 break

- 무한루프

- 반복문에서 조건식이 항상 true(1)이어서 반복문을 빠져나오지 못하는 상태

```
while(1){  
    printf("hello hustar\  
n");  
}
```

```
for(;;){  
    printf("hello hustar\  
n");  
}
```

```
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar  
hello hustar
```

- 무한루프에서 빠져나오는 방법

- Terminal에서 **ctrl+c**를 눌러서 프로그램 중단

```
int num=0;  
while(1){  
    printf("hello hustar\n");  
    if(num>10) break;  
    num++;  
}
```

```
int num=0;  
for(;;){  
    printf("hello hustar\  
n");  
    if(num>10) break;  
    num++;  
}
```

# Continue

- Continue

- 실행중인 반복문에서 continue 이후의 문장을 실행하지 않고 다음 반복을 수행

- Break

- 실행중인 반복문에서 탈출하여 다음 문장을 실행

```
for ( ; ; ) {  
    printf("Enter a number(enter 0 to stop) :  
    ");  
    scanf("%d", &n);  
    if (n == 0) break;  
    printf("%d cubed is %d \n", n, n * n * n);  
}
```

For문을 빠져나와 다음 문장을 수행

```
while (n < 10) {  
    scanf("%d", &i);  
    if (i == 0) continue;  
    sum += i;  
    n++;  
    /* continue jump to here */  
}
```

sum +=i; n++; 수행 없이 다음 반복 수행

# Continue와 break 비교

```
#include <stdio.h>

int main(){
    int n = 0, i = 1;
    int sum = 0;

    for ( ; i <= 10; i++ ) {
        if ( i == 5 ) break;
        sum += i;
        printf("sum=%d\n", sum);
    }

    return 0;
}
```

sum=1  
sum=3  
sum=6  
sum=10

```
#include <stdio.h>

int main(){
    int n = 0, i = 1;
    int sum = 0;

    for ( ; i <= 10; i++ ) {
        if ( i == 5 ) continue;
        sum += i;
        printf("sum=%d\n", sum);
    }

    return 0;
}
```

sum=1  
sum=3  
sum=6  
sum=10  
sum=16  
sum=23  
sum=31  
sum=40  
sum=50

# 중첩 반복문

- For, while을 중첩하여 사용할 수 있음

```
for (초기화1; 조건식1; 증감식1 ) {  
    for (초기화2; 조건식2; 증감식2 ) {  
        수행구문1;  
    }  
}
```

```
while(조건식1){  
    while(조건식2){  
        수행구문;  
        증감식2;  
    }  
    증감식1  
}
```



# 중첩반복문

- $i+k$ 가 2의 배수인 경우 count 증가

```
#include <stdio.h>

int main(){
    int i , k , count = 0 ;

    for ( i=1 ; i < 4 ; i++)
        for ( k=1 ; k <= 3 ; k++)
        {
            if ((i + k) % 2 == 0 )
                count++ ;
        }
    printf("Count = %d \n", count);

    return 0;
}
```

Count = 5

| i | k | count |
|---|---|-------|
| 1 | 1 | 1     |
|   | 2 | 1     |
|   | 3 | 2     |
| 2 | 1 | 2     |
|   | 2 | 3     |
|   | 3 | 3     |
| 3 | 1 | 4     |
|   | 2 | 4     |
|   | 3 | 5     |

# 중첩반복문-실습

- 구구단 프로그램, 2단부터 n단까지 출력, n은 사용자 입력

```
input n:3
```

2단

2\*1= 2   2\*2= 4   2\*3= 6   2\*4= 8   2\*5= 10

2\*6= 12   2\*7= 14   2\*8= 16   2\*9= 18

3단

3\*1= 3   3\*2= 6   3\*3= 9   3\*4= 12   3\*5= 15

3\*6= 18   3\*7= 21   3\*8= 24   3\*9= 27