- 학습교재: https://algo.datahub.pe.kr/
- 소스코드: https://algo.datahub.pe.kr/src/
- Smart OJ: https://soj.datahub.pe.kr/

충북교육연구정보원 정보영재교육원 | SW·AI교실 | 정보아카데미 강사

흥덕고등학교 교사 박정진

2026 정보올림피아드 첫걸음

< C언어 >

충북학생정보올림피아드

내 용		학교대회(예선)	도대회(본선)
신청		5. 15.(금) 10:00~ 5. 22.(금) 18:00까지 교육연구정보원 대회 플랫폼 및 대회문항 활용 신청	6. 29.(월) 10:00~ 7. 6.(월) 18:00까지 도대회 참가 신청
안내		5. 26.(화) 예정	7. 14.(화) 예정
대회 실시	초	6. 17.(수)	9. 2.(수)
	중	6. 18.(목)	9. 3.(목)
	고	6. 19.(금)	9. 4.(금)
결과발표		6. 26.(금) 예정	9. 11.(금) 예정
비고		– 학교자체계획에 의하여 **자율 실시** – 위 학교대회 날짜에 교육연구정보원 대회 플랫폼 및 대회문항 활용 신청시 **대회문제는 교육연구정보원에서 제공** **(희망학교는 교육행정데이터통합관리로 신청)**	

충북학생정보올림피아드

- 참가 방법
  - 학교대회는 참가학생의 소속학고, 도대회는 충북교육연구정보원에서 실시
  - https://cberi.goorm.io/ 에 감독교사로 부터 받은 대회 전용계정으로 접속 (개인계정 사용 불가)
  - [문제목록] 메뉴에서 대회 문제를 확인하고, 원하는 프로그래밍 언어(C, C++, Python)을 선택 후 코드 작성
  - 코드 작성 후 [코드 제출하기 버튼 클릭
  - 채점결과 확인

충북학생정보올림피아드

- 대회 개요
  - 수학적 지식과 논리적 사고능력을 필요로 하는 **알고리즘**과 **자료구조**를 적절히 사용하여 **프로그램 작성 능력**을 평가
  - 프로그램 작성을 통한 문제해결 능력 평가
  - 프로그래밍 실기평가로만 진행
  - 초 • 중 • 고등부 각 5문항 내외 출제
  - C, C++, Python 프로그래밍 언어 사용가능
  - 동점처리 기준:
    ① 바른코드 제출개수
    ② 점수
    ③ 문제풀이 시간
    ④ 답안 제출횟수

2026. 충북학생정보올림피아드 대비 콘텐츠 안내

구분	사이트 주소
정보올림피아드 기출 문제	https://coding.cberi.go.kr 문제 > 문제집 > 기출문제집
정보올림피아드 기출 해설강의	https://cmooc.cberi.go.kr 강좌보기 > 학생대회 > 2022, 2023, 2024, 2025 정보올림피아드 해설 강의
C언어	https://cmooc.cberi.go.kr 강좌보기 > 프로그래밍언어 > 차근 차근 C언어 강의
C++언어	https://cmooc.cberi.go.kr 강좌보기 > 프로그래밍언어 > C++ 입문
파이썬	https://cmooc.cberi.go.kr 강좌보기 > 프로그래밍언어 > 파이썬 입문
정보올림피아드 관련 학습 사이트	codeup.kr, koistudy.net, codingfun.net
bitCode 기초 100제	https://coding.cberi.go.kr 문제 > 기초 100제

프로그래밍 언어

- 프로그래밍 언어란?
  - 컴퓨터를 이용하기 위한 언어
  - 컴퓨터에게 명령이나 연산을 시킬 목적으로 설계되어 기계와 의사소통을 할 수 있게 해주는 언어
  - 사람이 원하는 작업을 컴퓨터가 수행할 수 있도록 프로그래밍 언어로 일련의 과정을 작성하여 일을 시킨다.
- 프로그래밍 언어의 종류
  - C / C++
  - C#, Java
  - Fortran, ALGOL
  - LISP
  - Python
  - Kotlin
  - Dart

C

- Structured programming
- High Speed
- Less Library functions
- Memory Management done by programmer
- Pointers available
- Function Renaming not possible
- Harder syntax
- Architecture language

VS

Python

- Interpreted language
- Slow speed
- Rich Library
- Automatic garbage collection
- Pointers not Available
- Function Renaming can be done.
- Easy syntax
- General purpose language

Start here - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

Projects Files FSymbols

Workspace

Start here

Code::Blocks

The open source, cross-platform IDE

Release 20.03 rev 11983 (2020-03-12 18:24:30) gcc 8.1.0 Windows/unicode - 64 bit

Create a new project

Open an existing project

Tip of the Day

Visit the Code::Blocks forums

Report a bug or request a new feature

Recent projects

D:₩MyProjects₩Algorithm₩ShortestPathAll₩ShortestPathAll₩ShortestPathAll.cbp

D:₩MyProjects₩Test₩Test₩Test.cbp

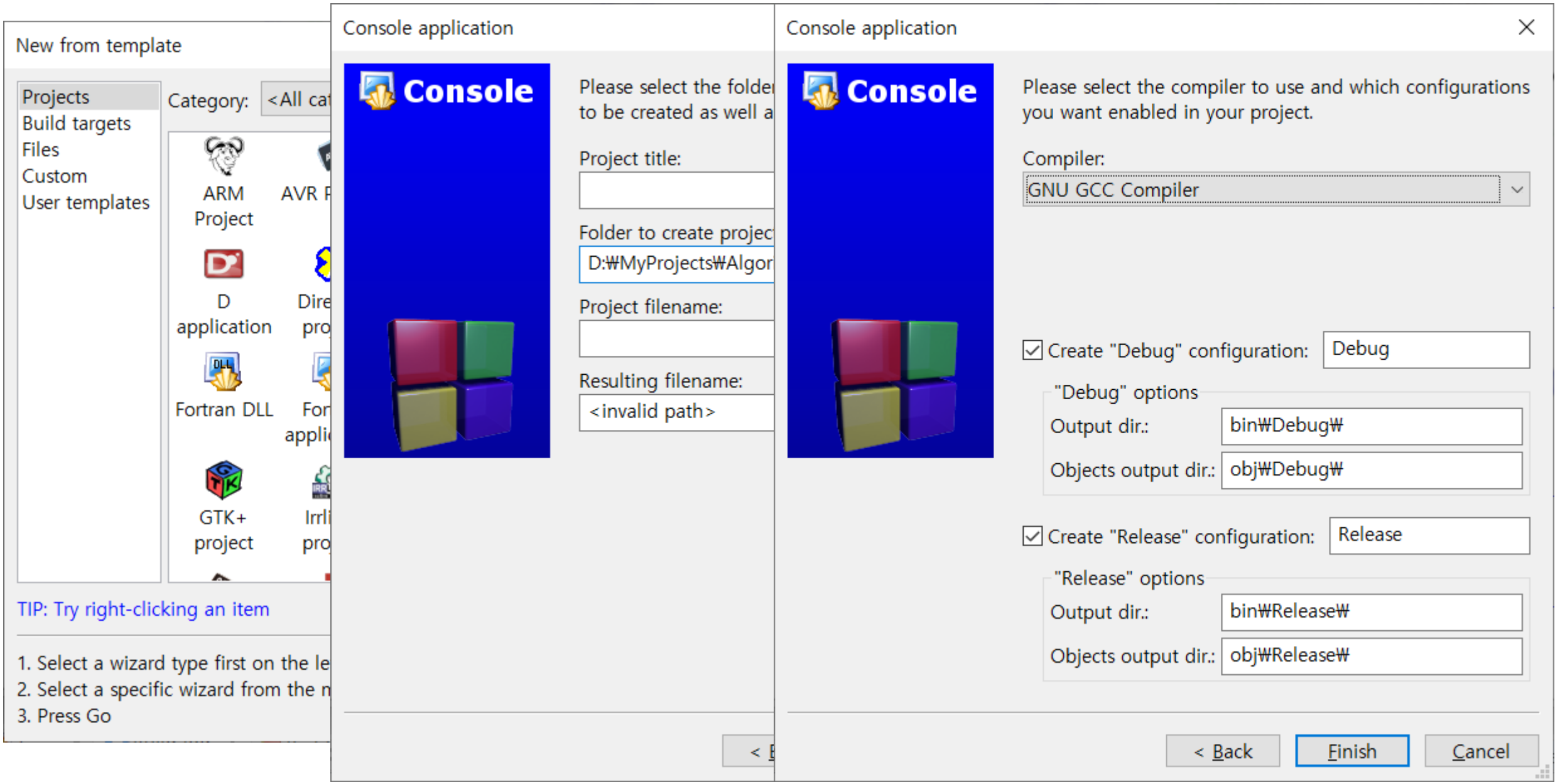
Recent files

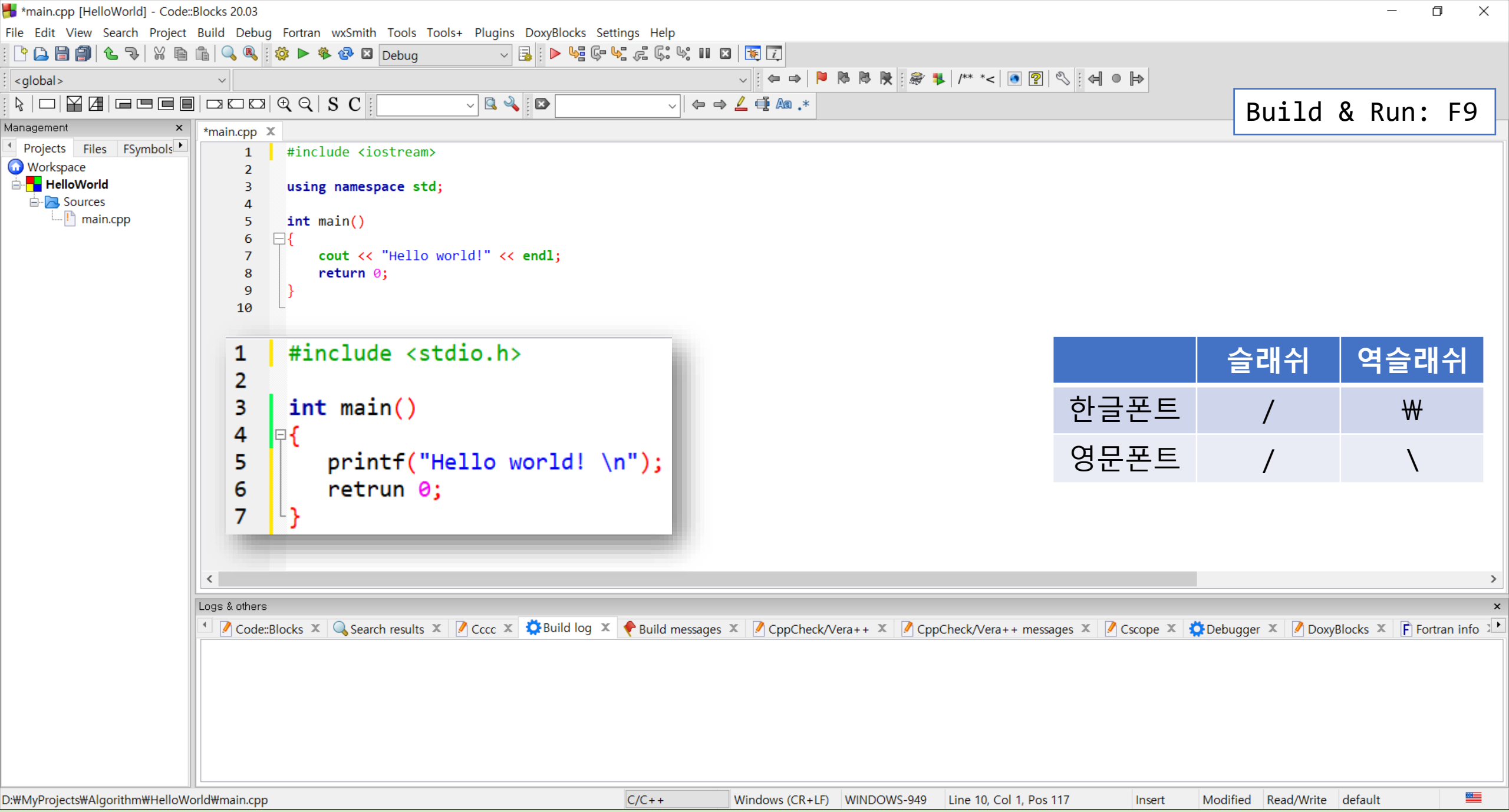
Logs & others

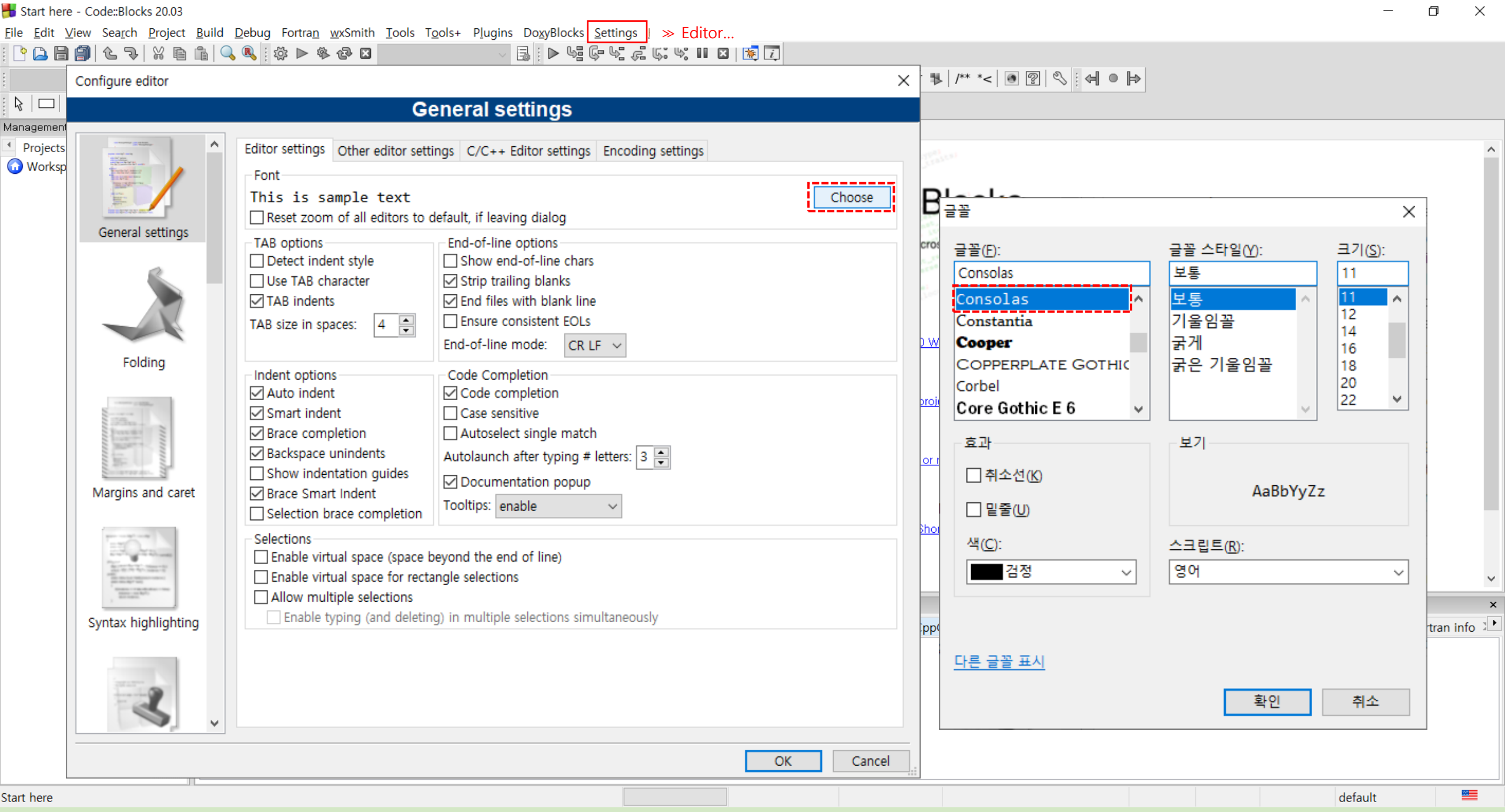
Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info

Start here

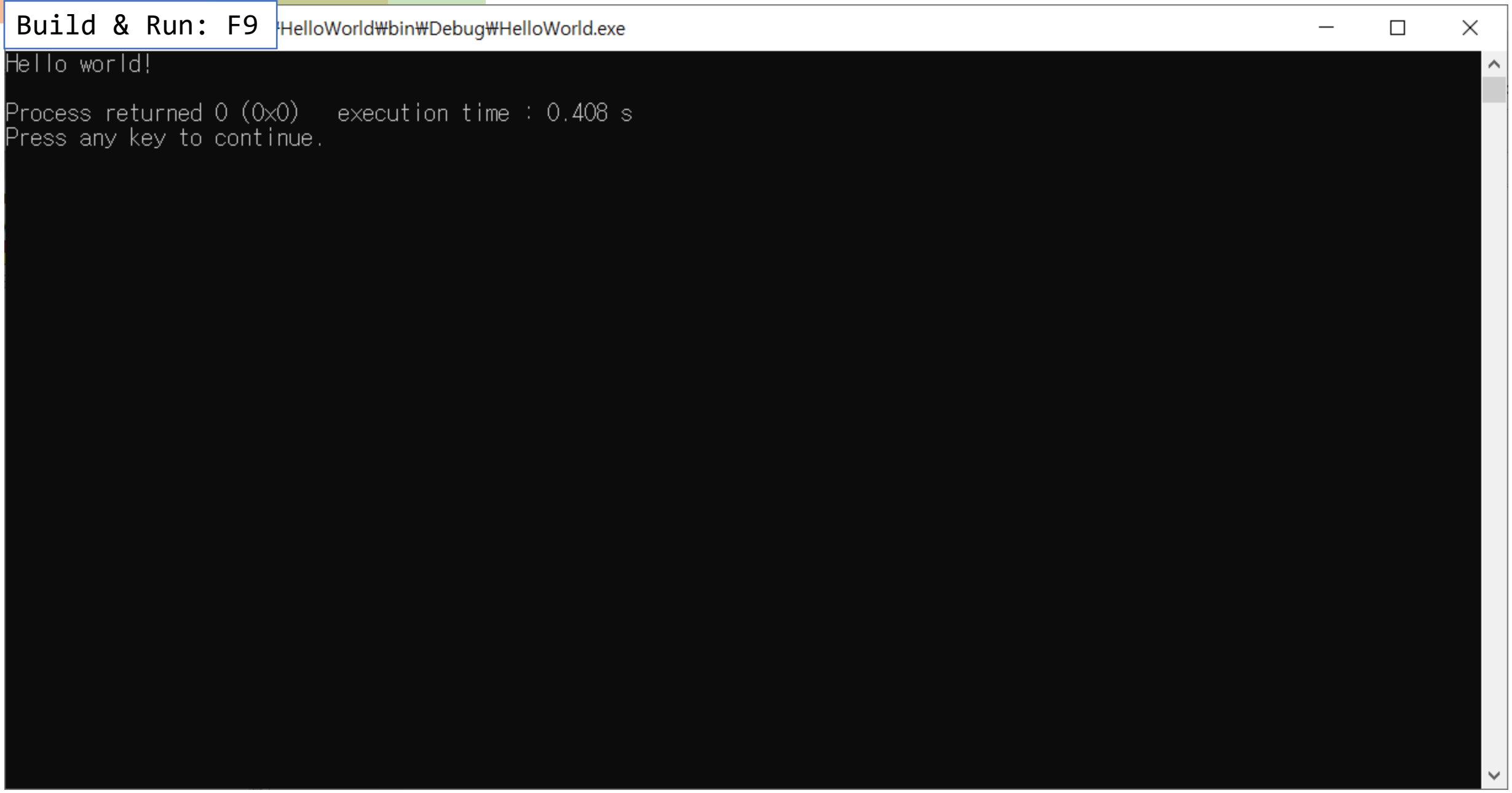
default


New from template
Projects
Build targets
Files
Custom
User templates
Category:
<All cat
ARM Project
AVR F
D application
Dire pro
Fortran DLL
For appli
GTK+ project
Irrli pro
TIP: Try right-clicking an item
1. Select a wizard type first on the le
2. Select a specific wizard from the n
3. Press Go
Console application
Console
Please select the folde
to be created as well a
Project title:
Folder to create projec
D:₩MyProjects₩Algor
Project filename:
Resulting filename:
<invalid path>
< B
Console application
Console
Please select the compiler to use and which configurations you want enabled in your project.
Compiler:
GNU GCC Compiler
Create "Debug" configuration:
Debug
"Debug" options
Output dir.:
bin₩Debug₩
Objects output dir.:
obj₩Debug₩
Create "Release" configuration:
Release
"Release" options
Output dir.:
bin₩Release₩
Objects output dir.:
obj₩Release₩
< Back
Finish
Cancel


*main.cpp [HelloWorld] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
Build & Run: F9
Management
Projects Files FSymbols
Workspace
HelloWorld
Sources
main.cpp
*main.cpp
#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" << endl;
return 0;
}
#include <stdio.h>
int main()
{
printf("Hello world! \n");
retrun 0;
}
슬래쉬 역슬래쉬
한글폰트 / ₩
영문폰트 / \
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info
D:₩MyProjects₩Algorithm₩HelloWorld₩main.cpp
C/C++
Windows (CR+LF)
WINDOWS-949
Line 10, Col 1, Pos 117
Insert
Modified
Read/Write
default

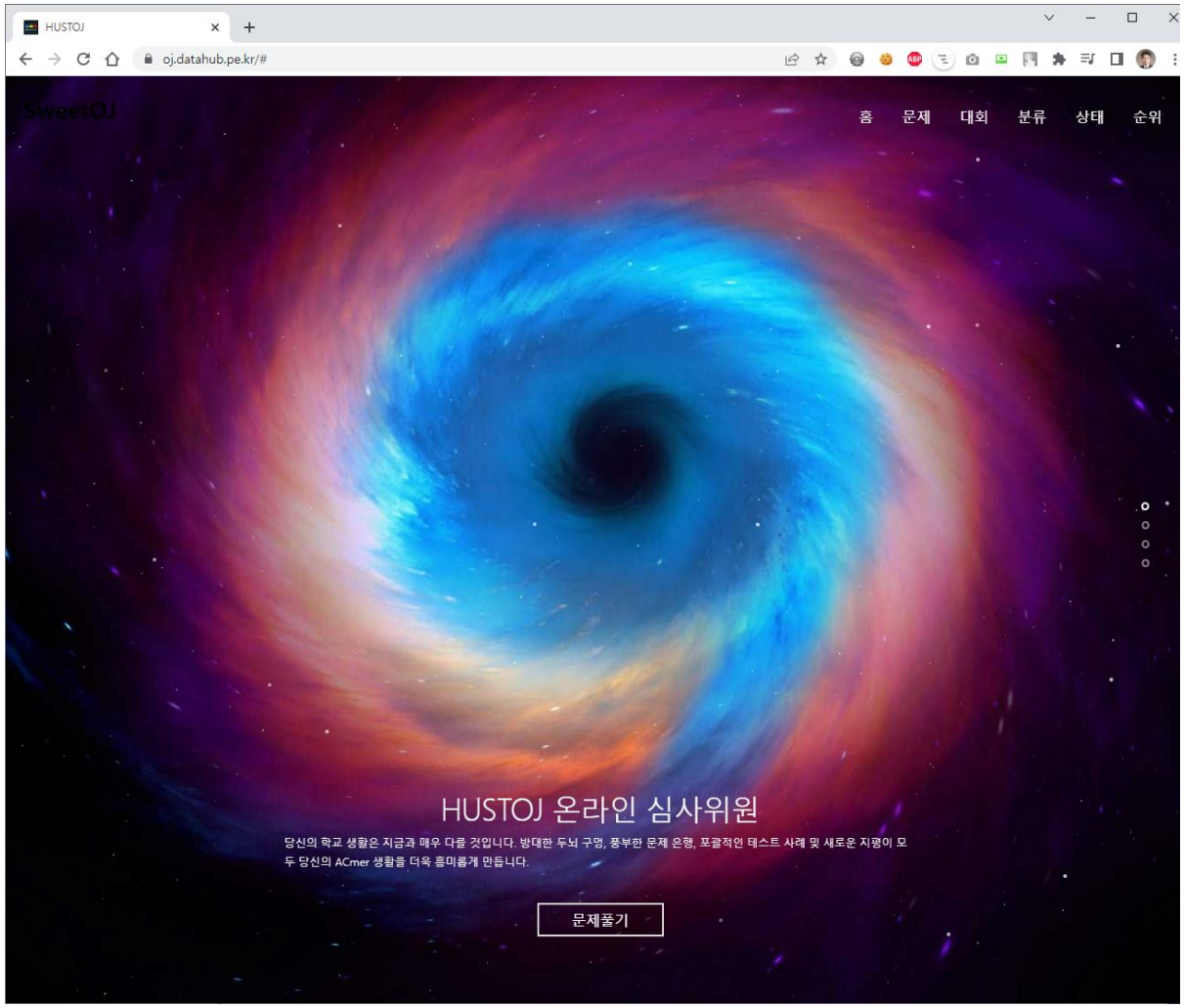

Start here - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings » Editor...
Configure editor
General settings
General settings
Folding
Margins and caret
Syntax highlighting
Editor settings
Other editor settings
C/C++ Editor settings
Encoding settings
Font
This is sample text
Choose
Reset zoom of all editors to default, if leaving dialog
TAB options
Detect indent style
Use TAB character
TAB indents
TAB size in spaces: 4
End-of-line options
Show end-of-line chars
Strip trailing blanks
End files with blank line
Ensure consistent EOLs
End-of-line mode: CR LF
Indent options
Auto indent
Smart indent
Brace completion
Backspace unindents
Show indentation guides
Brace Smart Indent
Selection brace completion
Code Completion
Code completion
Case sensitive
Autoselect single match
Autolaunch after typing # letters: 3
Documentation popup
Tooltips: enable
Selections
Enable virtual space (space beyond the end of line)
Enable virtual space for rectangle selections
Allow multiple selections
Enable typing (and deleting) in multiple selections simultaneously
OK
Cancel
글꼴
글꼴(F):
Consolas
Constantia
Cooper
COPPERPLATE GOTHIC
Corbel
Core Gothic E 6
글꼴 스타일(Y):
보통
기울임꼴
굵게
굵은 기울임꼴
크기(S):
11
12
14
16
18
20
22
효과
취소선(K)
밑줄(U)
색(C):
검정
보기
AaBbYyZz
스크립트(R):
영어
다른 글꼴 표시
확인
취소
Start here
default

Build & Run: F9


HelloWorld¥bin¥Debug¥HelloWorld.exe
Hello world!
Process returned 0 (0x0)   execution time : 0.408 s
Press any key to continue.

OJ 사용법

- OJ(온라인 저지)란?
  - 프로그래밍 대회에서 프로그램들을 시험할 목적으로 만들어진 온라인 시스템이다. 대회 연습용으로 사용되기도 한다.
  - 본 수업용 OJ
    - https://soj.datahub.pe.kr



Online Judge 프로그램 접속 계정 정보

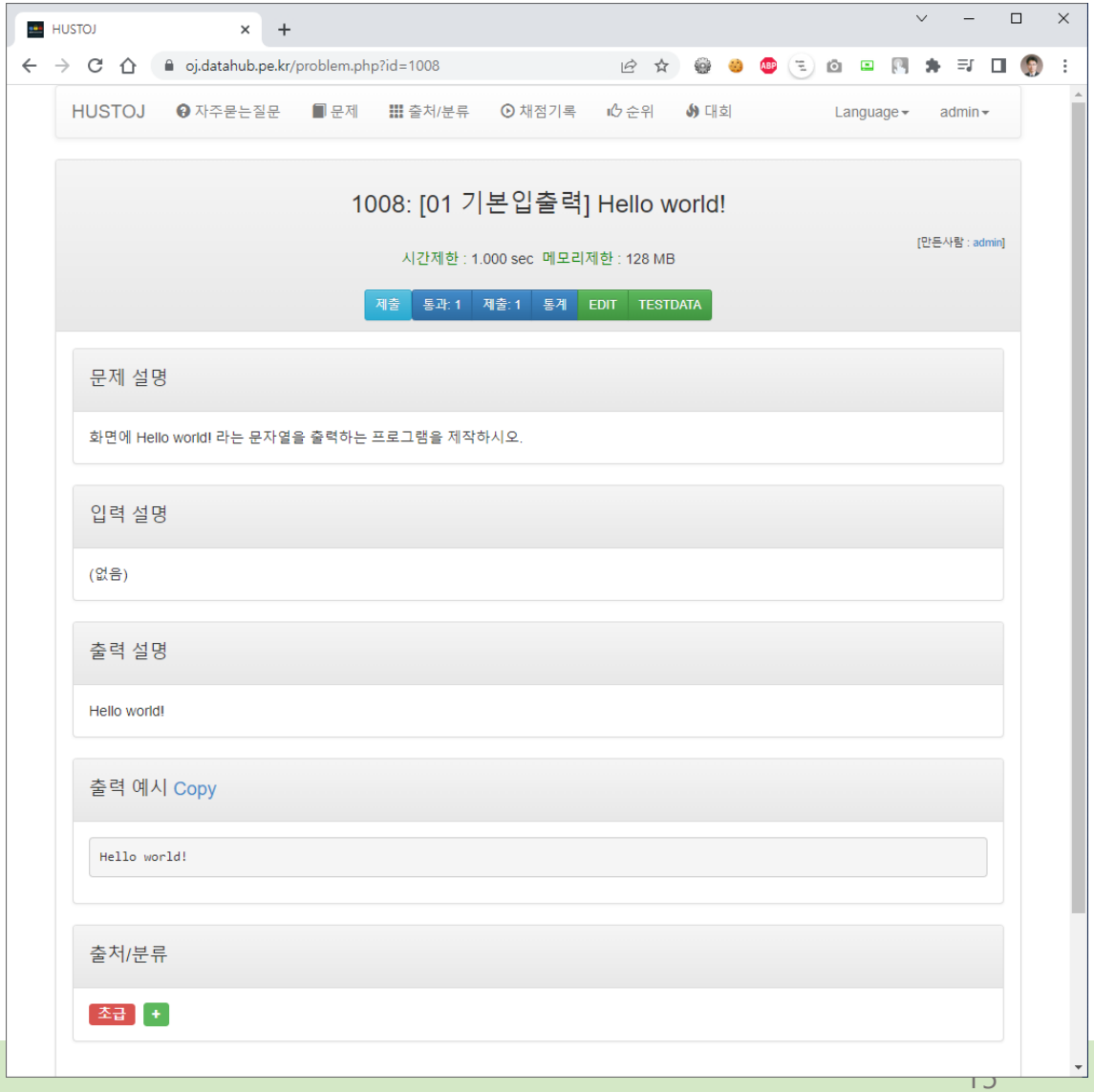
순번	소속학교	학년	이름	ID	PW
1	금천중	중 1	윤예준	cio2601	iopw01
2	솔밭중	중 1	안유진	cio2602	iopw02
3	세광중	중 2	문종현	cio2603	iopw03
4	원봉중	중 2	고재민	cio2604	iopw04
5	충주북여중	중 2	허서윤	cio2605	iopw05
6	산남중	중 3	박수현	cio2606	iopw06
7	솔밭중	중 3	문성완	cio2607	iopw07
8	청주대성고	고 1	안지우	cio2608	iopw08

순번	소속학교	학년	이름	ID	PW
9	금천고	고 2	이선재	cio2609	iopw09
10	금천고	고 2	함지우	cio2610	iopw10
11	금천고	고 2	음정현	cio2611	iopw11
12	금천고	고 2	이준석	cio2612	iopw12
13	금천고	고 2	최민준	cio2613	iopw13
14				cio2614	iopw14
15				cio2615	iopw15
16					

OJ 사용법

- 파일제출
  - CodeBlock 등 IDE에서 프로그램 작성
  - 완성된 프로그램을 OJ에 업로드 후
  - 채점 결과 확인

  - 채점 결과 종류
    - 모두 맞음
    - 틀림 | 정확도: __%
    - 실행중 에러 | 정확도: __%
    - 컴파일 에러



Hello World 예제

- 기본 예제

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world! \n");
6  }
```

```
// 주석문
/*
   여러줄 주석
*/
```

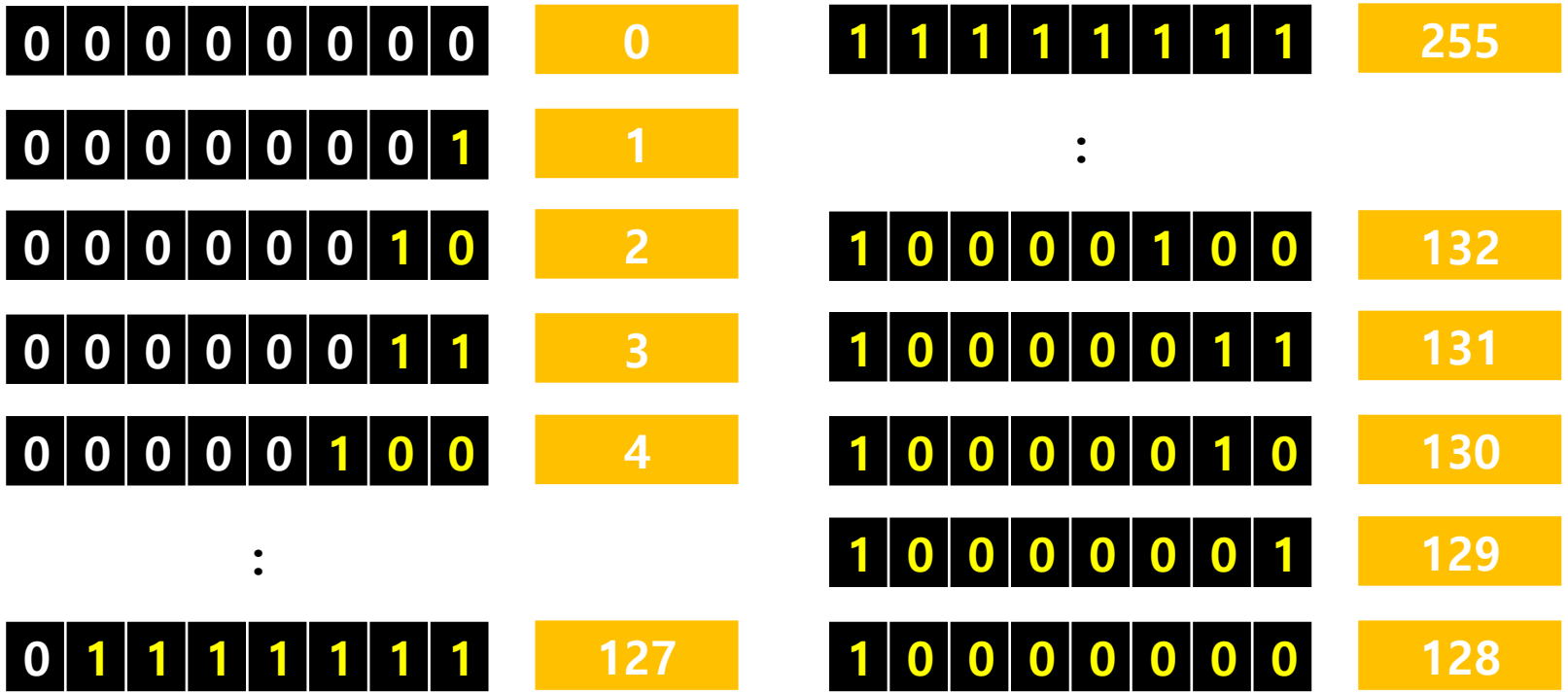
- main() 함수
  - 프로그램의 시작 지점
  - { } 시작과 끝을 의미
- printf("Hello world! \n");
  - printf() 함수: 표준 출력 장치에 출력하는 기능을 하는 함수
  - 인수: "Hello world! \n" 를 printf 라는 함수에 전달
- #include <stdio.h>
  - 표준 입출력 함수들에 대한 정보를 가지고 있는 stdio.h 라는 파일을 불러온다.
- 문장의 끝
  - 함수 내에 존재하는 문장의 끝에는 세미콜론 문자 ; 를 붙여준다.

C언어의 자료형(data type)

- **정수형**
  - char
    (문자형) - 숫자도 저장하지만 주로 문자를 저장함
    character
  - short
    short int
  - **int**
    integer
  - long
    long integer
  - long long
    64bit long

수의 표현 방식

- 정수(양수)의 표현 - unsigned



수의 표현 방식

- 정수 (양수, 음수)의 표현 - signed


```
1111 1000 (-?)
0000 0111 2의보수
```

수의 표현 방식

- 큰 수의 표현

0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0	
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1	1	
0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 0	2	
0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 1	3	
0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0	4	
:			
0 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	32767	$2^{15}-1$
1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	-32768	-2^{15}

수의 표현 방식

- 더 큰 수의 표현

00000000	00000000	00000000	00000000	0
00000000	00000000	00000000	00000001	1
00000000	00000000	00000000	00000010	2
00000000	00000000	00000000	00000011	3
00000000	00000000	00000000	00000100	4
00000000	00000000	00000000	00000101	5
01111111	11111111	11111111	11111111	$2^{31}-1$
10000000	00000000	00000000	00000000	-2^{31}

C언어의 자료형(data type)

용도	타입	크기		signed(부호있음)	unsigned(부호없음)
정수형 (문자형)	**char**	1byte	8bit	-2^{7} ~ $2^{7}-1$ (127)	0 ~ $2^{8}-1$ (256)
정수형	short	2byte	16bit	-2^{15} ~ $2^{15}-1$ (≒3.2만)	0 ~ $2^{16}-1$ (≒6.5만)
	int	4byte	32bit	-2^{31} ~ $2^{31}-1$ (≒21억)	0 ~ $2^{32}-1$ (≒42억)
	long	4byte	32bit	-2^{31} ~ $2^{31}-1$ (≒21억)	0 ~ $2^{32}-1$ (≒42억)
	long long	8byte	64bit	-2^{63} ~ $2^{63}-1$ (≒922경)	0 ~ $2^{64}-1$ (≒1844경)
실수형	float	4byte	32bit	3.4×10^{-38} ~ 3.4×10^{38}	
	double	8byte	64bit	1.7×10^{-308} ~ 1.7×10^{308}	

변수 (variable)

- 변수의 의미
  - '변하는 수' 라는 의미
  - 무언가를 기억해야 할 때 사용
  - 자료를 저장하는 공간에 이름은 붙인 것
  - 프로그래머가 이름(변수명)을 결정

- 변수의 사용
  - 선언을 한 뒤부터 사용 가능
  - = 연산자로 값을 할당
  - 선언과 동시에 할당 가능(초기화)
  - 초기화 하지 않으면 쓰레기 값

```c
#include <stdio.h>

int main() {
    // 변수 선언(초기화 없음)
    int age;
    // 변수에 값 할당
    age = 20;

    // 변수 선언(초기화 없음)
    int height;

    // 변수 선언과 동시에 할당
    char blood = 'A';
    // 변수 선언과 동시에 할당
    double pi = 3.14159;
}
```

주소	내용(값)	이름
1001	20	age
1002		
1003		
1004		
1005		height
1006		
1007		
1008		
1009	'A'	blood
1010	3.14159	pi
1011		
1012		
1013		
1014		
1015		
1016		
1017		

서식문자

```c
#include <stdio.h>

int main() {
    int age;    // 변수 선언
    age = 20;   // 변수에 할당
    int height, weight;
    char blood = 'A';   // 선언과 할당
    double pi = 3.141592;

    printf("age: %d \n", age);
    printf("height: %d \n", height);
    printf("blood: %c \n", blood);
    printf("pi: %lf \n", pi);
    printf("pi: %.2lf \n", pi);
}
```

- 서식문자

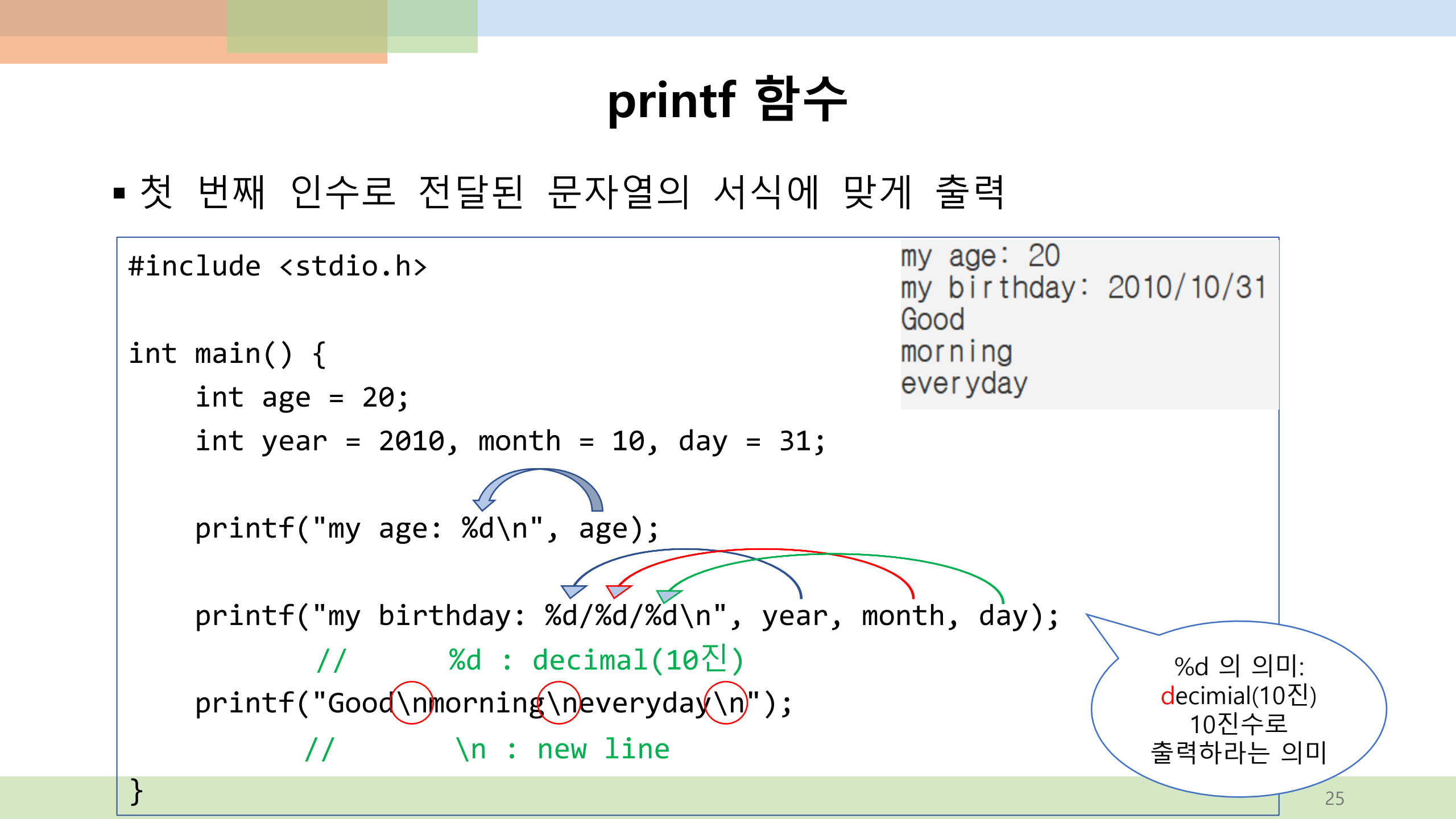
서식문자	출력 형태	사용 타입
%c	단일 문자	char
%d	부호 있는 10진 정수	char, short, int
%i	부호 있는 10진 정수, %d와 같음	
%f, %lf	부호 있는 10진 실수	float, double
%s	문자열	char[]
%o	부호 없는 8진 정수	char, short, int,
%u	부호 없는 10진 정수	
%x	부호 없는 16진 정수, 소문자 사용	
%X	부호 없는 16진 정수, 대문자 사용	
%e	e 표기법에 의한 실수	float, double
%lld, %llu	부호 있는, 부호 없는 long long 정수	long long
%g	값에 따라서 %f, %e 둘 중 하나를 선택	float, double
%G	값에 따라서 %f, %G 둘 중 하나를 선택	
%%	% 기호 출력	

printf 함수

- 첫 번째 인수로 전달된 문자열의 서식에 맞게 출력

```
#include <stdio.h>

int main() {
    int age = 20;
    int year = 2010, month = 10, day = 31;

    printf("my age: %d\n", age);

    printf("my birthday: %d/%d/%d\n", year, month, day);
        //      %d : decimal(10진)
    printf("Good\nmorning\neveryday\n");
        //      \n : new line
}
```

```
my age: 20
my birthday: 2010/10/31
Good
morning
everyday
```



상수 (constant)

1. 매크로 상수

- #define 문 사용

```c
#include <stdio.h>
#define PI 3.141592

int main() {
    int r = 5;
    double s;
    // 원면적 구하기 코딩방식①
    s = 3.141592*r*r;

    // 원면적 구하기 코딩방식②
    s = PI*r*r;
    printf("%lf\n", s);
}
```

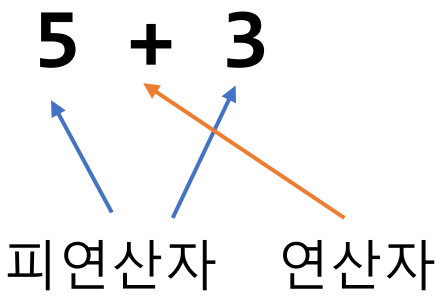
2. 변수의 고정

- const 키워드 사용

```c
#include <stdio.h>

int main() {
    const double PI=3.141592;
    int r = 5;
    double s;
    // 원면적 구하기 코딩방식①
    s = 3.141592*r*r;

    // 원면적 구하기 코딩방식②
    s = PI*r*r;
    printf("%lf\n", s);
}
```

연산자(operator)

- 연산자와 피연산자



- 이항연산자
  - 피연산자가 두 개인 연산자
  - +, -, x, ÷
- 단항연산자
  - 피연산자가 한 개인 연산자
  - -, !, ~

- C언의 연산자 표기

의미	수학	C언어	비고
덧셈	+	+	
뺄셈	-	-	
곱셈	×	*	v
나눗셈	÷	/	v
나머지		%	
같다	=	==	v
다르다		!=	v
크다	>, ≥	>, >=	
작다	<, ≤	<, <=	
그리고	And	&&	
또는	Or	||	

연산자(operator)

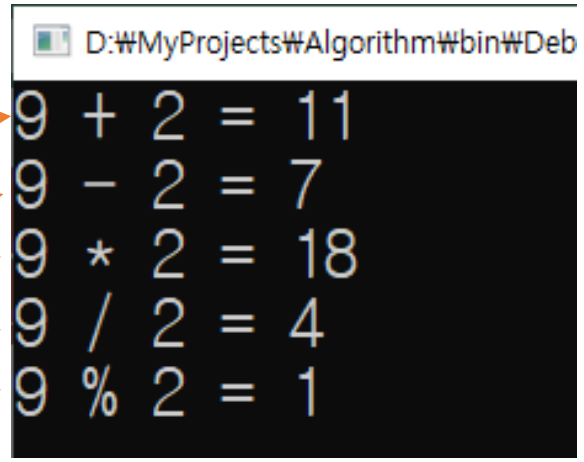
- 대입 연산자와 산술 연산자

연산자	설명	결합방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) num = 20;	←
+	두 피연산자의 값을 더한다. 예) num = 4+3;	→
-	왼쪽 피연산자의 값에서 오른쪽 피연산자 값을 뺀다. 예) num = 4-3;	→
*	두 피연산자의 값을 곱한다. 예) num = 4*3;	→
/	왼쪽의 피연산자 값을 오른쪽 피연산자 값으로 나눈 몫을 구한다. 예) num = 7/3;	→
%	왼쪽의 피연산자 값을 오른쪽 피연산자 값으로 나눈 나머지를 구한다. 예) num = 7%3;	→

연산자(operator)

- 대입 연산자와 산술 연산자 실습

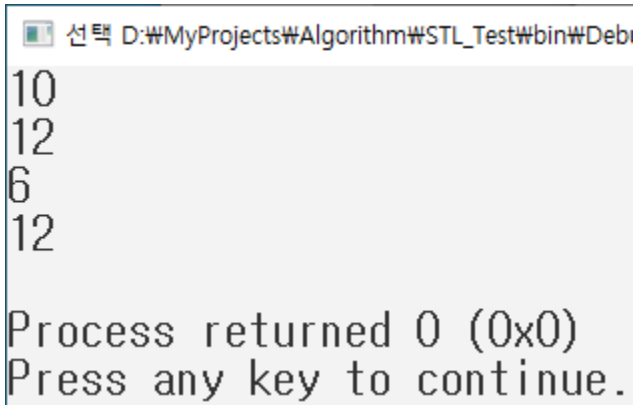
```
#include <stdio.h>

int main() {
    int n1 = 9, n2 = 2, res;
    res = n1 + n2;
    printf("%d + %d = %d \n", n1, n2, res);
    res = n1 - n2;
    printf("%d - %d = %d \n", n1, n2, res);

    printf("%d * %d = %d \n", n1, n2, n1*n2);
    printf("%d / %d = %d \n", n1, n2, n1/n2);
    printf("%d %% %d = %d\n", n1, n2, n1%n2);
}
```



연산자(operator)

- 증감 연산자

연산자	연산의 예	의미	결합성
++a	printf("%d", ++a)	선 증가, 후 연산	←
a++	printf("%d", a++)	선 연산, 후 증가	←
--b	printf("%d", --b)	선 감소, 후 연산	←
b--	printf("%d", b--)	선 연산, 후 감소	←




```
#include <stdio.h>

int main() {
    int a=10;
    printf("%d\n", a++);
    printf("%d\n", ++a);

    int x=2, y=3;
    printf("%d\n", (x++)*(y++));
    printf("%d\n", (x*y));
}
```

연산자(operator)

- 연산자의 우선순위와 결합방향

우선순위	연산자유형		연산자종류	결합방향
높음	식, 구조체, 공용체 연산자		( ) [ ] -> .	좌 → 우
	단항 연산자		! ~ ++ -- - (형명칭) * & sizeof	좌 ← 우
↑	이항 연산자	승, 제	* / %	좌 → 우
		가, 감	+ -	좌 → 우
		비트 이동	<< >>	좌 → 우
		대소 비교	< <= > >=	좌 → 우
		등가 판정	== !=	좌 → 우
↓		비트 AND	&	좌 → 우
		비트 XOR	^	좌 → 우
		비트 OR	|	좌 → 우
		논리 AND	&&	좌 → 우
		논리 OR	||	좌 → 우
	조건 연산자		? :	좌 ← 우
낮음	대입 연산자		+ =+ =+ *= /= %=	좌 ← 우
	나열 연산자		,	좌 → 우

- 꼭 기억해야할 연산자 우선순위
  ① ( )
  ② ++ -- !
  ③ * / %
  ④ + -
  ⑤ =

연산자(operator)

- 연산자 우선순위 실험

```
#include <stdio.h>

int main(void) {
    int a = 10+4*3-1;
    printf("%d \n", a);

    int b = 10+4*(3-1);
    printf("%d \n", b);

    int r=4+5*6/(2+1)+15-5*2;
    printf("%d \n", r);
}
```

```
21
18
19
```

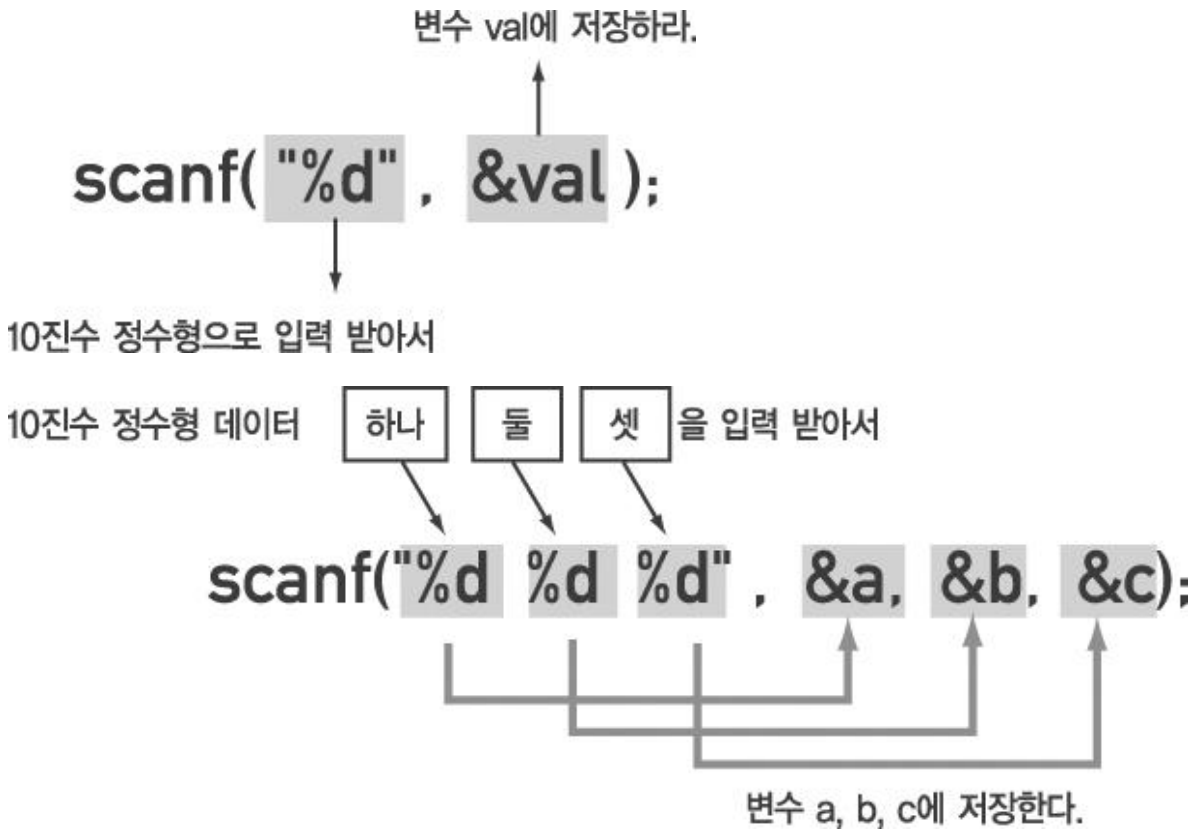
- 연산자 결합방향 실험

```
#include <stdio.h>

int main(void)
{
    int r = 10-1-2-3+4+5;
    printf("%d \n", r);

    int a=3, b=4, c=5, d=6;
    a = b = c = d;
    printf("%d %d %d %d", a,b,c,d);
}
```

```
13
6 6 6 6
```

scanf() 함수를 이용한 입력

- 키보드로부터 정수 입력을 위한 scanf 함수의 호출



OJ에 제출

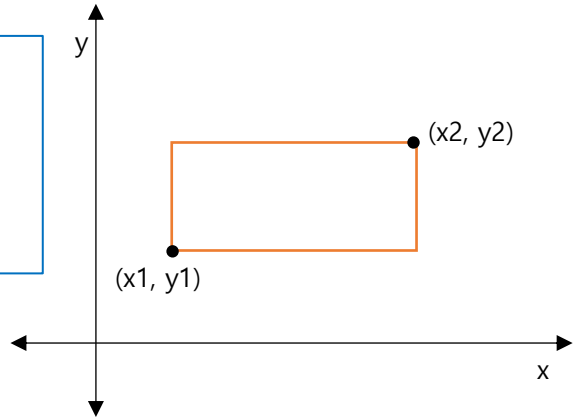
```c
#include <stdio.h>

int main() {
   int n1, n2;

   printf("input n1: ");
   scanf("%d", &n1);
   printf("input n2: ");
   scanf("%d", &n2);
   printf("%d + %d = %d\n",n1, n2, n1+n2);

   printf("input two nums:\n");
   scanf("%d %d", &n1, &n2);
   printf("%d * %d = %d\n", n1, n2, n1*n2);

   return 0;
}
```

연습문제 – 직사각형의 넓이

- 문제
  - 두 점의 x, y 좌표를 입력 받아서, 두 점이 이루는 직사각형의 넓이를 계산하여 출력하는 프로그램을 작성하시오. (x1<x2, y1<y2)

  - 실행의 예

```
2 4
4 8
8
```



- 정답

OJ에 제출

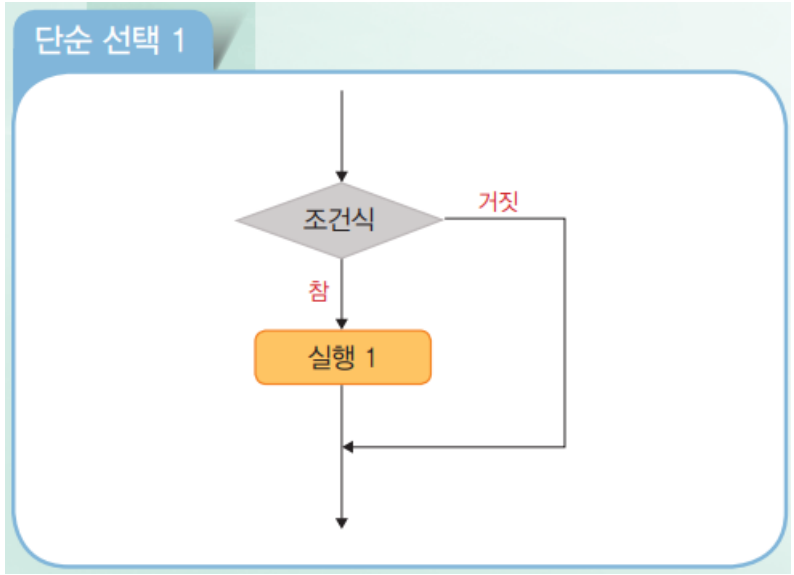
```
#include <stdio.h>

int main() {
   int x1, y1;
   int x2, y2;









}
```

제어문

- 조건문
  - if 문
    - 단순 if
    - if … else
    - if … else if … else
  - switch문
    - case
    - default
- 반복문
  - while 문
  - do…while 문
  - for문
- 기타
  - break 문
  - continue 문

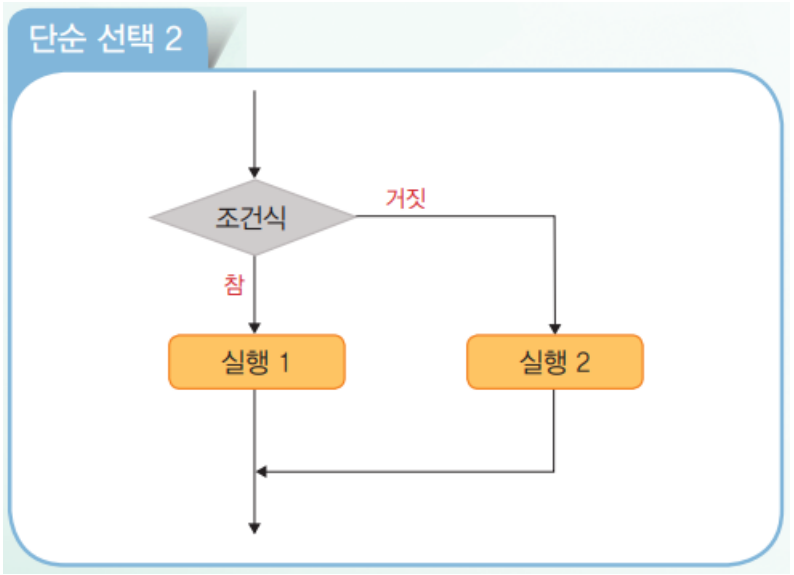
선택 실행 구조

- if 문



- 예시
  - 100점이면 congratulation 출력

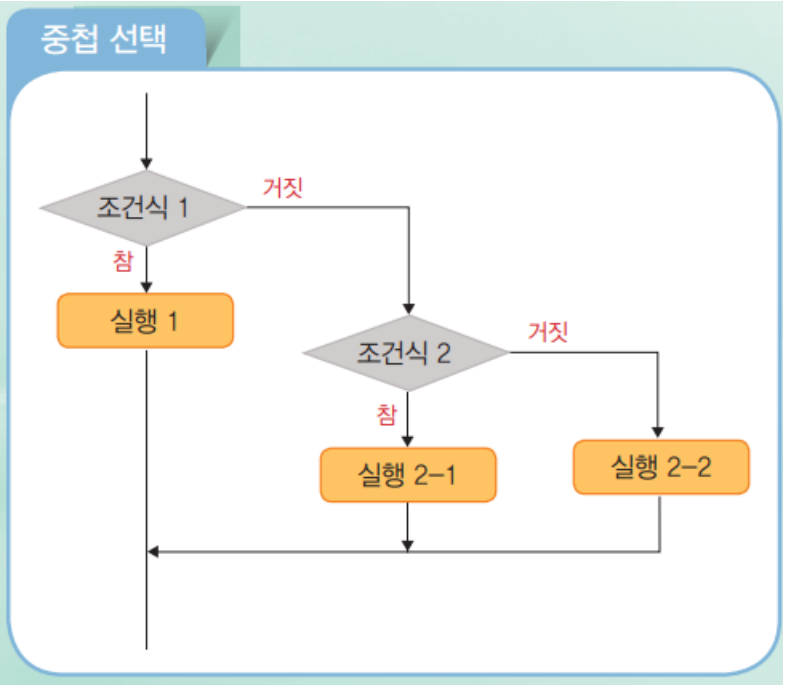
- if … else 문



- 예시
  - 60점 이상이면 "Pass", 아니면 "Fail"

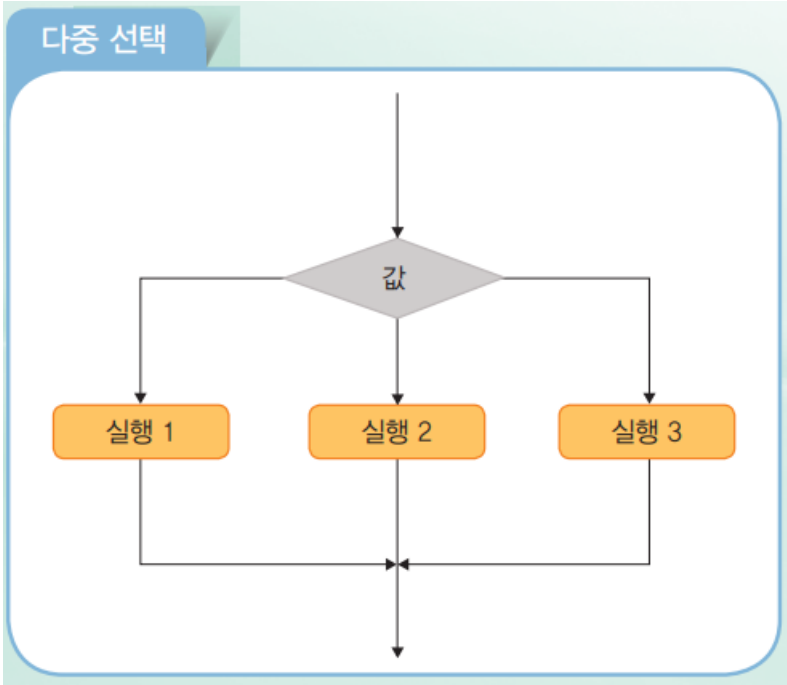
선택 실행 구조

- if … else if … else 문



- 예시
  - 90점 이상이면 A, 80점 이상이면 B,
  - 그 외에는 C

- switch 문



- 예시
  - ↑ : 전진, ← : 좌회전, → : 우회전

조건문 – if

- 단순 if 문
  - 특정 조건을 만족하는 경우 해야 할 일이 있을 때 사용
  - 처리해야 할 명령어가 2개 이상인 경우 반드시 중괄호로 묶어야 함

```
input score: 100
Perfect!
You good!
Test passed.
```

```
input score: 60
You good!
Test passed.
```

```c
#include <stdio.h>

int main() {
    int score;
    printf("input score: ");
    scanf("%d", &score);

    if(score == 100)
        printf("Perfect!\n");

    if(score >= 60) {
        printf("You good!\n");
        printf("Test passed.\n");
    }
    return 0;
}
```

연산자(operator)

- 비교 연산자(관계 연산자)
  - 두 피연산자의 관계(크다, 작다 혹은 같다)를 따지는 연산자
  - true(논리적 참, 1), false(논리적 거짓, 0) 반환

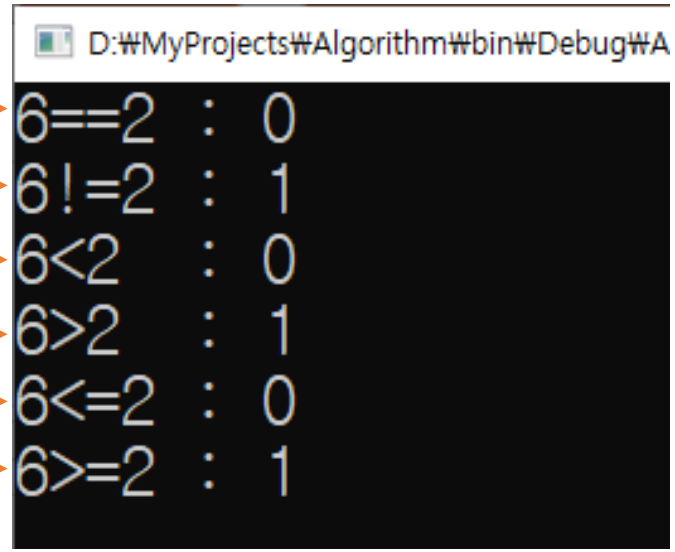
연산자	연산의 예	의미	결합성
==	a == b	a와 b가 같은가	→
!=	a != b	a와 b가 같지 않은가	→
<	a < b	a가 b보다 작은가	→
>	a > b	a가 b보다 큰가	→
<=	a <= b	a가 b보다 작거나 같은가	→
>=	a >= b	a가 b보다 크거나 같은가	→

연산자(operator)

- 비교 연산자(관계 연산자)

```
#include <stdio.h>

int main() {
    int a=6, b=2;

    printf("%d==%d : %d\n", a, b, a==b);
    printf("%d!=%d : %d\n", a, b, a!=b);
    printf("%d<%d  : %d\n", a, b, a<b);
    printf("%d>%d  : %d\n", a, b, a>b);
    printf("%d<=%d : %d\n", a, b, a<=b);
    printf("%d>=%d : %d\n", a, b, a>=b);
}
```

- 모든 연산자는 계산 결과를 반환한다.
- 비교연산자는 참(true) 이면 1을, 거짓(false)이면 0을 반환한다.



조건문 – if

- if … else 문
  - 특정 조건을 만족하는 경우 A를 하고 만족하지 않는 경우 B를 수행할 때 사용
  - 점수를 입력 받아 60점 이상이면 'You passed!' 를 아니면 'Failed.' 'Retry it.' 을 출력하는 프로그램을 작성하시오.

```
#include <stdio.h>

int main()
{
    int score;
    printf("input score: ");
    scanf("%d", &score);

    if(score >= 60)
        printf("You passed!\n");
    else {
        printf("Failed.\n");
        printf("Retry it.\n");
    }
    return 0;
}
```

조건문 – if

- if … else if … else 문

  - 점수를 입력 받아 90점 이상이면 'A', 80점 이상이면 'B', 70점 이상이면 'C', 그 밖의 경우 'F'를 출력하는 프로그램을 작성하시오.

OJ에 제출

```
#include <stdio.h>

int main() {
    int score;
    char grade;
    printf("input score: ");
    scanf("%d", &score);

    if(score >= 90)
        grade = 'A';
    else if(score >= 80)
        grade = 'B';
    else if(score >= 70)
        grade = 'C';
    else
        grade = 'F';

    printf("grade: %c\n", grade);
    return 0;
}
```

연습문제

- BMI 계산
  - 체질량지수는 자신의 몸무게(kg)를 키의 제곱(m)으로 나눈 값입니다.
  - 몸무게(kg단위)와 키(cm단위)를 입력받아 BMI를 계산하여 소수점 둘째 자리까지 출력하고,
  - BMI 수치에 따른 결과를 출력하시오.
    - 18.5 미만이면 '저체중'
    - 18.5 ~ 23미만이면 '정상'
    - 23.0 ~ 25 미만이면 '과체중'
    - 25.0 이상부터는 '비만'

OJ에 제출

```
#include <stdio.h>
int main() {
    double w;  // 몸무게
    double h;  // 키
    double bmi;

    scanf("%lf", &w);
    scanf("%lf", &h);









}
```

연산자(operator)

- 논리 연산자
  - and, or, not을 표현하는 연산자
  - true(1), false(0) 반환

연산	C연산자	연산의 예	의미	결합성
AND	&&	a && b	true면 true 리턴	→
OR	||	a || b	하나라도 true면 true 리턴	→
NOT	!	!a	true면 false를, false면 true 리턴	→

조건문 – if

- 필기/실기 모두 60점 이상이어야 합격

```
#include <stdio.h>

int main() {
    int pilgi, silgi;
    printf("필기와 실기 점수를 입력: ");
    scanf("%d %d", &pilgi, &silgi);

    if(pilgi>=60 && silgi>=60)
        printf("You passed!\n");
    else {
        printf("Failed.\n");
        printf("Retry it.\n");
    }
    return 0;
}
```

- 필기/실기 둘 중 하나만 60점 이상이면 합격

```
#include <stdio.h>

int main() {
    int pilgi, silgi;
    printf("필기와 실기 점수를 입력: ");
    scanf("%d %d", &pilgi, &silgi);

    if(pilgi>=60 || silgi>=60)
        printf("You passed!\n");
    else {
        printf("Failed.\n");
        printf("Retry it.\n");
    }
    return 0;
}
```

조건문 – if

- 점수가 60점 미만이 아니면 합격

```
#include <stdio.h>

int main() {
    int score;
    printf("input score: ");
    scanf("%d", &score);

    if( !(score<60) )
        printf("You passed!\n");
    else
        printf("Failed.\n");

    return 0;
}
```

- 필기가 60점미만이 아니고, 실기도 60점 미만이 아니면 합격

```
#include <stdio.h>

int main(){
    int pilgi, silgi;
    printf("필기와 실기 점수를 입력: ");
    scanf("%d %d", &pilgi, &silgi);

    if( !(pilgi<60) && !(silgi<60))
        printf("You passed!\n");
    else
        printf("Failed.\n");

    return 0;
}
```

연습문제

- 두 개의 정수와 한 개의 사칙 연산자를 입력받아 사칙연산 결과를 처리하는 프로그램을 작성하시오.
- 입력되는 모든 숫자는 정수이고, 가운데 연산자는 + - * / % 이외는 없다.

```
예: 10 + 5
계산할 수식을 입력하세요
9 * 5
9 * 5 = 45
```

```c
#include <stdio.h>

int main() {
    int n1, n2, res;
    char op;
    printf("예: 10 + 5\n");
    printf("계산할 수식을 입력하세요\n");
    scanf("%d %c %d", &n1, &op, &n2);









    printf("%d %c %d = %d\n", n1, op, n2, res);
    return 0;
}
```

반복문

- 반복문의 기능
  - 특정 영역을 특정 조건이 만족되는 동안에 반복 실행하기 위한 문장

- 세 가지 형태의 반복문이 제공됨
  1) while문에 의한 반복
     - 몇 번 반복해야 하는지 모를 때 사용, ex) 답 맞출 때까지 계속
  2) do ~ while문에 의한 반복
     - 일단 한번은 실행하고 그 결과에 따라 다시 반복할 수도 있을 때 사용, ex) 메뉴 입력
  3) for문에 의한 반복
     - 반복 횟수가 정해져 있는 경우 주로 사용, ex) 10번 출력

반복문 - while

- 형식

```
while(반복조건)
    반복할 문장;
```

```
while(반복조건) {
    반복할 문장1;
    반복할 문장2;
          :
}
```

- 반복조건이 참인 동안 반복할 문장을 실행

- 예시

```
int n = 1;
while(n < 5) {
    printf("%d \n", n);
    n++;  // n 1증가
}
```

n
5

```
1
2
3
4
```

반복문 - while

- 5회 반복 방법1

```
int c = 0;
while(c < 5) {
    printf("%d \n", c);
    c++;
}
```

```
0
1
2
3
4
```

- 5회 반복 방법2

```
int c = 1;
while(c <= 5) {
    printf("%d \n", c);
    c++;
}
```

```
1
2
3
4
5
```

반복문 - while

- 퀴즈

```c
#include <stdio.h>
int main() {
    int num = 10; // 10부터 시작

    puts("Rocket lunch countdown..");
    while(num > 0) {  // 0보다 크면
        printf("%2d \n", num);
        num--;  // 1씩 감소하면서…
    }
    printf("last num:%2d \n", num);
}
```

- 결과

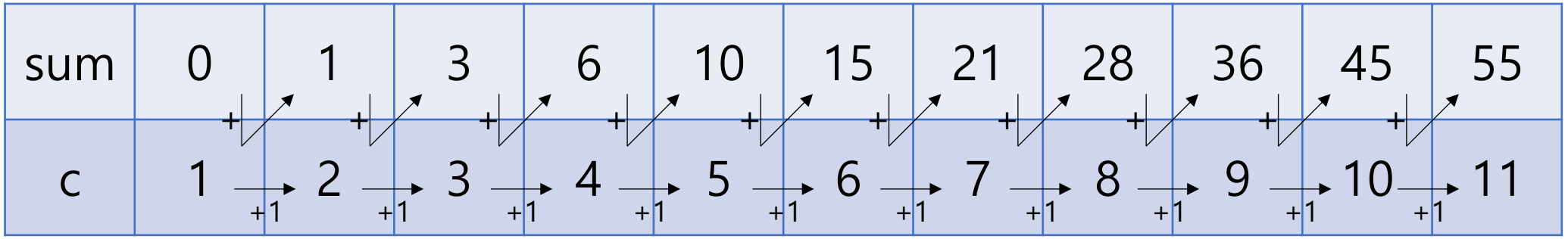
```
Rocket lunch countdown..
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
```

- 카운트 다운 숫자는 얼마까지 출력될까?
- 종료 직전 num 값은? → 0

반복문 - while

- 1부터 10까지 누계 구하기

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10



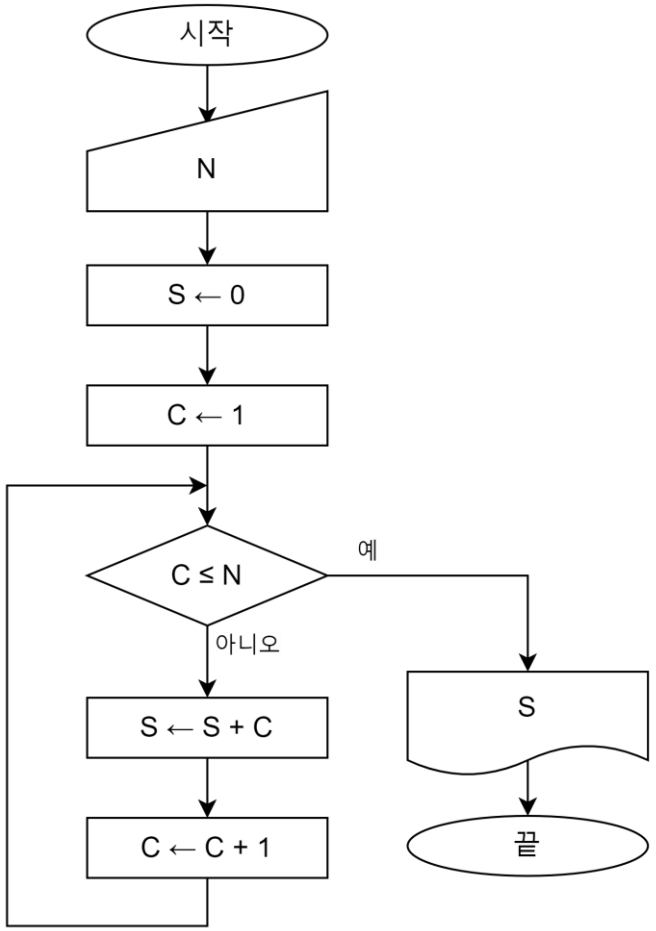
[초기값]
sum = 0
c = 1

```
while(c < 11)
    sum = sum + c
    c = c + 1
```

```
while(c <= 10)
    sum = sum + c
    c = c + 1
```

반복문 - while

- 1부터 10까지 누계 구하기



- 소스코드

```
#include <stdio.h>
int main() {
    int sum=0, c=1;
    while(c <= 10) {
        sum=sum+c;
        c++;
   }
   printf("%d \n", sum);
}
```

- 출력

```
55
```

STEP	sum	c
1	0	1
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

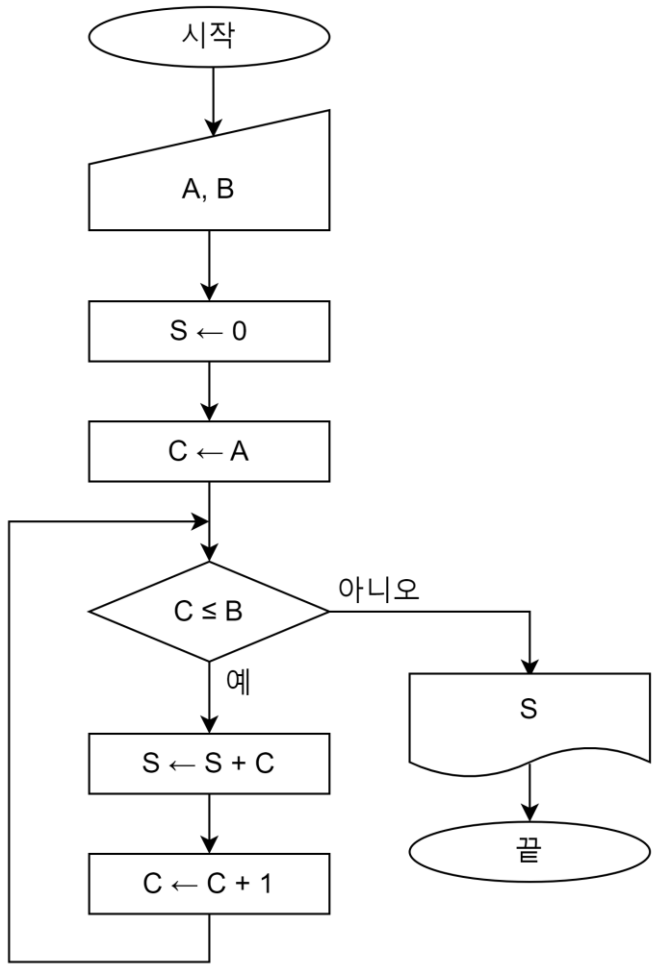
연습문제

- while문을 사용하여 두 정수 a와 b를 입력 받아 a부터 b까지 누계를 구하는 프로그램을 작성하시오.

  (a < b 라고 가정)

  - 예

```
1 10
55
```




```c
#include <stdio.h>

int main() {
    int a, b;

    return 0;
}
```

반복문 – do ... while

- 문법

```
do {
    반복할 문장1;
    반복할 문장2;
        :
}
while(반복조건);
```

- while문은 조건을 먼저 확인하고 반복 할지 결정하고,
- do…while문은 일단 한 번 해보고 반복 할지 결정한다.

- 비교

```
int main() {
    int n = 1;
    while(n < 1) {
        printf("%d \n", n);
        n++;
    }
}
```

```
int main() {
    int n = 1;
    do {
        printf("%d \n", n);
        n++;
    }
    while(n < 1);
}
```

소인수로 분해하기

- 문제

  양의 정수 한 개가 입력되었을 때, 그 수를 소인수로 분해하여 출력하는 프로그램을 작성하시오.

- 입력

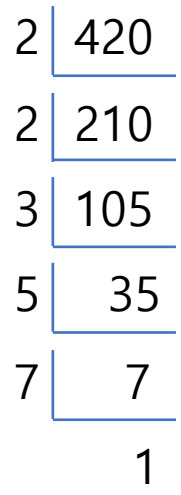
  양의 정수 한 개가 입력된다. (2이상)

- 출력

  그 수를 소인수로 분해하여 오름차순으로 출력한다.

- 입력과 출력의 예

입력 예	출력 예
420	2 2 3 5 7



- 프로그램

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

    int d=2;
    do {









    }
    while(d<=n);
}
```

중첩된 while

- 중첩된 while

```
while(반복조건1) {
    while(반복조건2) {


    }
}
```

  - 들여쓰기를 사용하여 반복 내용의 시작과 끝을 명확히 하자!

- 00:00 부터 ~ 05:59 까지 시간 출력

```
#include <stdio.h>

int main() {
    int hour=0;

    puts("clock time");
    while(hour < 6) {
        int min = 0;
        while(min < 60) {
            printf("%02d:%02d ", hour, min);
            min++;
        }
        printf("\n");
        hour++;
    }
    return 0;
}
```

반복문 - for

- for문 형식

```
for(①초기식; ②조건식; ④증감식) {
    ③반복할 문장;
}
```

- 실행순서

1) ①초기식 ②조건식 ③반복할 문장 ④증감식
2) ②조건식 ③반복할 문장 ④증감식
3) :
4) ②조건식

- 예시

```
for(int i=0; i<3; i++) {
    printf("%d\n", i);
}
```

- 실행순서

①초기식	②조건식	③반복문장	④증감식	i
i=0	i<3	printf(0)	i++	1
	i<3	printf(1)	i++	2
	i<3	printf(2)	i++	3
	i<3			

반복문 - for

- 5회 반복 방법1

```
for(int i=0; i<5; i++) {
    printf("%d\n", i);
}
```

- 출력

```
0
1
2
3
4
```

- 5회 반복 방법2

```
for(int i=1; i<=5; i++) {
    printf("%d\n", i);
}
```

- 출력

```
1
2
3
4
5
```

반복문 - for

- 1부터 15사이 3의 배수 출력

```
#include <stdio.h>
int main() {
   for(int i=3; i<=15; i=i+3)
        printf("%d", i);
}
```

- 출력

```
3
6
9
12
15
```

- 10부터 1까지 카운트다운

```
#include <stdio.h>
int main() {
   puts("Rocket launch countdown…");
   for(int i=10; i>=1; i--)
        printf("%d ", i);
}
```

- 출력

```
Rocket launch countdown…
10 9 8 7 6 5 4 3 2 1
```

반복문 – for 문과 while 문 비교

- for 문

```c
// 1부터 5까지의 합계를 구하는 프로그램
#include <stdio.h>

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

    for(n=1; n<=5; n++) {
        sum= sum + n;
        printf("sum of 1 to %d: %2d \n", n, sum);
    }
    return 0;
}
```

- while 문

```c
// 1부터 5까지의 합계를 구하는 프로그램
#include <stdio.h>

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

    n=1;
    while(n<=5) {
        sum = sum + n;
        printf("sum of 1 to %d: %2d \n", n, sum);
        n++;
    }
    return 0;
}
```

3의 배수 게임

- 문제

  3의 배수 게임을 하던 정올이는 3의 배수 게임에서 작은 실수를 계속해서 벌칙을 받게 되었다.
  3의 배수 게임의 왕이 되기 위한 수련 프로그램을 작성해 보자.

  ** 3의 배수 게임이란?
  여러 사람이 순서를 정해 순서대로 수를 부르는 게임이다.
  만약 3의 배수를 불러야 하는 상황이라면, 그 수 대신 "박수"를 친다.

- 입력

  첫 줄에 하나의 정수 n이 입력된다.
  (n은 50미만의 자연수이다)

- 출력

  1부터 n까지 순서대로 공백을 두고 수를 출력하는데, 3의 배수(3, 6, 9 ...)인 경우 수 대신 영문 대문자 X 를 출력한다.

- 입력과 출력의 예

입력 예	출력 예
7	1 2 X 4 5 X 7

공약수 찾기

- 문제
  - 입력된 두 자연수의 공약수를 모두 출력하는 프로그램을 작성하시오.
- 입력
  - 첫 번째 줄에 두 자연수 a와 b가 공백으로 분리되어 입력된다.
    (1 ≤ a, b ≤ 2,100,000,000)
- 출력
  - a와 b의 공약수를 작은 수부터 큰 수 순서로 공백으로 분리하여 출력한다.
- 예시

```
8 24
1 2 4 8
```

- 프로그램

```
#include <stdio.h>
int main() {
    int a, b;
    scanf("%d %d", &a, &b);




}
```

약수의 합 구하기

- 문제

  한 정수 n을 입력 받아서 n의 모든 약수의 합을 구하는 프로그램을 작성하시오.

  예를 들어 10의 약수는 1, 2, 5, 10이므로 이 값들의 합인 18이 10의 약수의 합이 된다.

- 입력

  첫번째 줄에 정수 n이 입력된다.

  (단, 1 <= n <= 100,000)

- 출력

  n의 약수의 합을 출력한다

- 입력과 출력의 예

입력 예	출력 예
5	6

입력 예	출력 예
10	18

- 고찰

  n의 약수들을 어떻게 알아낼 수 있을까?

약수의 합 구하기

- 문제

  한 정수 n을 입력 받아서 n의 모든 약수의 합을 구하는 프로그램을 작성하시오.

  예를 들어 10의 약수는 1, 2, 5, 10이므로 이 값들의 합인 18이 10의 약수의 합이 된다.

- 입력

  첫번째 줄에 정수 n이 입력된다.

  (단, 1 <= n <= 100,000)

- 출력

  n의 약수의 합을 출력한다

- 답안 예시

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int ans = 0;









    printf("%d\n", ans);
    return 0;
}
```

반복문 – 중첩된 for

1) 한 학급 1번 부터 30번까지 출력한다.

```
int main() {
    for(int n=1; n<=30; n++) {
        printf("%4d ", n);
    }
}
```

2) 열 개 학급에 대하여 출력한다.

```
int main() {
    for(int c=1; c<=10; c++) {
        printf("[%d반]\n", c);
        for(int n=1; n<=30; n++) {
            printf("%4d ", n);
        }
        printf("\n");
    }
}
```

3) 세 개 학년에 대하여 출력한다.

반복문 – 중첩된 for

- 중첩된 for 문을 이용하여 구구단 출력하기

```
#include <stdio.h>

int main() {
   for(int d=1; d<=5; d++) {   // 1단부터 5단까지
      printf(" %d 단\n", d);
      for(int x=1; x<=9; x++) {      // x1부터 x9까지
         printf("%d x %d = %2d \n", d, x, d*x);
      }
      printf("\n");
   }
   return 0;
}
```

```
 1 단
1 x 1 =  1
1 x 2 =  2
1 x 3 =  3
1 x 4 =  4
1 x 5 =  5
1 x 6 =  6
1 x 7 =  7
1 x 8 =  8
1 x 9 =  9

 2 단
2 x 1 =  2
2 x 2 =  4
2 x 3 =  6
2 x 4 =  8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

 3 단
3 x 1 =  3
```

반복문 – 중첩된 for

- 연습문제

  삼각형의 밑변 길이 정수 a를 입력 받아 중첩된 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성하시오.

*	1개
**	2개
***	3개
****	4개
*****	5개
******	:
*******	a개

- 정답

```
#include <stdio.h>

int main() {




}
```

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

반복문 – 중첩된 for

- 연습문제

삼각형의 밑변 길이 정수 a를 입력 받아 중첩된 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성하시오.

```
*
**
***
****
*****
******
*******
```

공백+별=a

∴ 공백=a-별

- 정답

```
#include <stdio.h>

int main() {




}
```

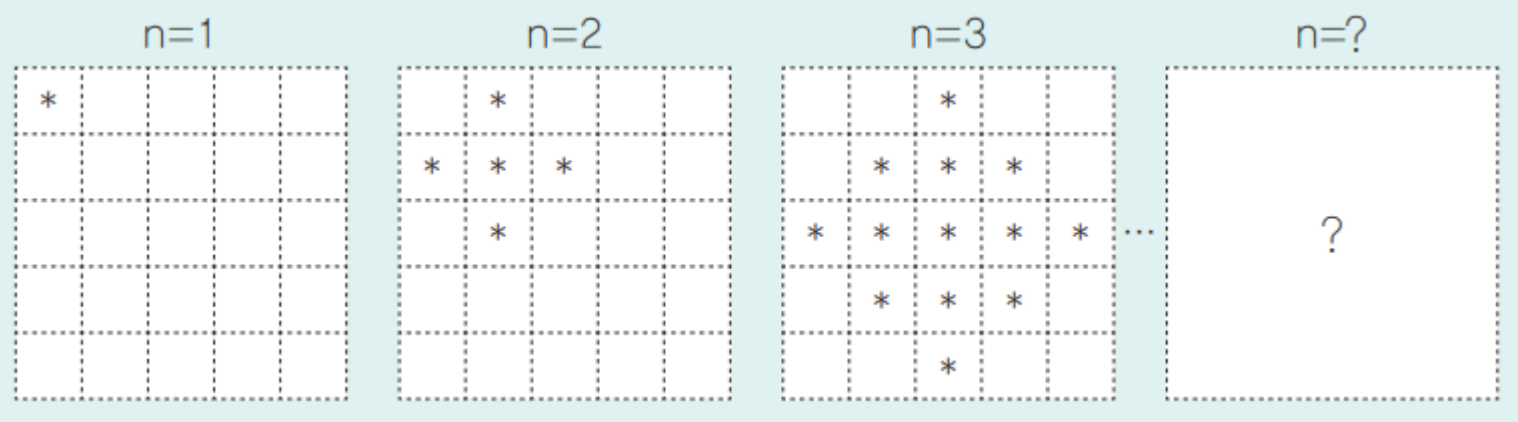
```
#include <stdio.h>
```

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

중첩된 for문 활용

- 연습문제

  삼각형의 밑변 길이 정수 a를 입력 받아 중첩된 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성 하시오.



- 입력설명

  첫 번째 줄에 자연수 n이 입력된다. (1 ≤ n ≤ 15)

		i	i	i	i	i	i	i	i	i
		1	2	3	4	5	6	7	8	9
j	1				*					
j	2			*	*	*				
j	3		*	*	*	*	*			
j	4	*	*	*	*	*	*	*		
j	3		*	*	*	*	*			
j	2			*	*	*				
j	1				*					

a=4

공백개수=a-j

별개수=2*j-1

제어문 – break

- break 문
  - switch, for, while, do ~ while문의 영역을 빠져 나오기 위해 사용
  - 가장 가까운 루프를 벗어난다.

- 사용 예
  - 1+2+3+…+n 의 합이 처음으로 100이상이 될 때, 그 때의 합과 n을 구하는 프로그램을 작성하시오.

```
#include <stdio.h>

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

   for(n=1; true; n++) {
      sum = sum + n;
      if(sum >= 100)
         break;
   }
   printf("n: %d, sum: %d \n", n, sum);
   return 0;
}
```

소수 판별

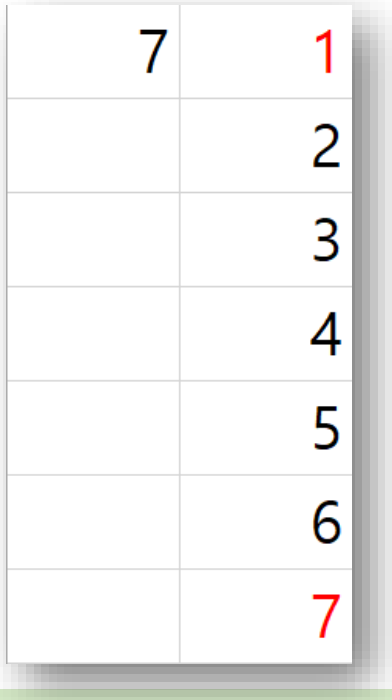
- 문제

  3 이상의 자연수(n)가 입력되었을 때, 소수 여부를 판별하는 프로그램을 작성해 보자.

  (3 ≤ n ≤ 1,000,000)

- 입력과 출력 예시

입력 예	출력 예
41	prime
111	composite



- 프로그램

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", n);




}
```

제어문 – continue

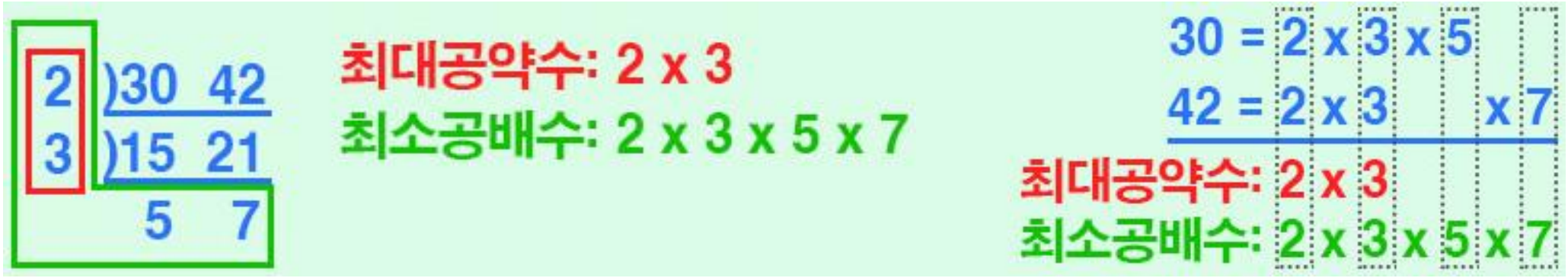
- continue 문
  - 반복문 내에서 사용되며, 남겨진 반복내용을 중단하고 다음 반복을 시작한다.

- 사용 예
  - 1부터 20까지의 정수 중에서 홀수만을 출력하시오.
  - (for, continue 문을 사용할 것.)

```
#include <stdio.h>

int main() {
   int i;
   for(i=1; i<=20; i++)  {
      if(i%2==0)
         continue;
      printf("%3d ", i);
   }
   return 0;
}
```

최대공약수와 최소공배수

- 최대공약수
  - Greatest Common Divisor, GCD
  - 공약수: 여러 수의 공통된 약수
  - 최대공약수: 여러 수의 공약수 중 최대인 수
- 최소공배수
  - Lowest Common Multiple, LCM
  - 공배수: 여러 수의 공통된 배수
  - 최소공배수: 공배수 중 최소인 수



- G = gcd(30, 42) = 6
- L = lcm(30, 42) = 210

- A = 30, B = 42
- A × B = G × L
- 30 × 42 = 6 × 210

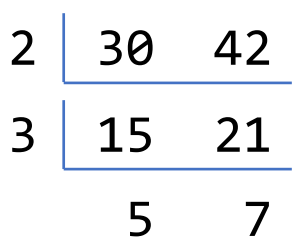
최대공약수 구하기

- 문제

  두 수 a, b를 입력 받아 최대공약수를 출력하는 프로그램을 작성하시오.

  예를 들어, 12과 16의 최대공약수는 4이다.

  ```
  30 42
  6
  ```



- 고찰
  - while 문이 적합한가?
  - do … while 문이 적합한가?

```c
#include <stdio.h>
int main() {
    int a, b;
    scanf("%d %d", &a, &b);

    int d=2, G=1;

    printf("%d\n", G);
}
```

최대공배수 구하기

- 문제

  두 수 a, b를 입력 받아 최대공배수를 출력하는 프로그램을 작성하시오.

  예를 들어, 6과 8의 최소공배수는 24이다.

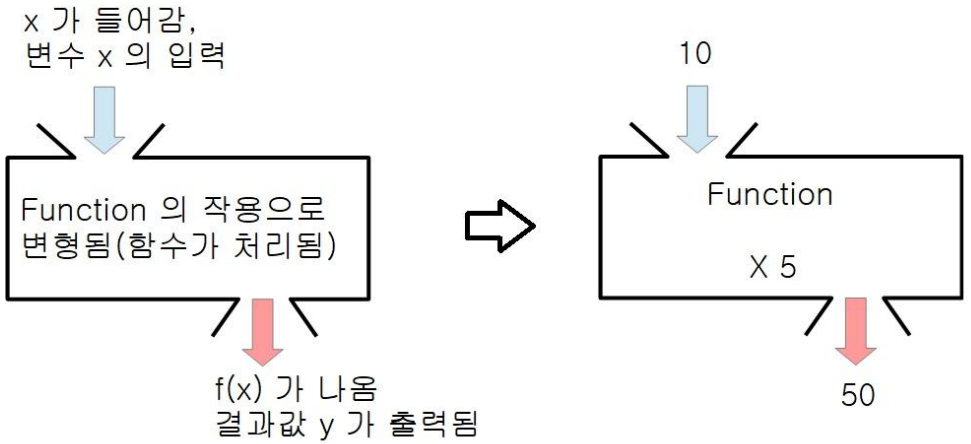
  ```
  6 8
  24
  ```

- 전략
  - 방금 전에 만든 최대공약수 프로그램을 약간 개조하자!

  - $A \times B = G \times L$

함수

- 함수(function) 란?
  - 특정한 처리.기능을 수행하는 코드를 하나로 묶어 둔 것.
  - 특정 인자를 받아 결과값을 반환하는 개체를 말하기 때문에 서브루틴(subroutine)이라고도 한다.



- 함수 사용의 효과
  - 코드들을 기능 단위로 묶을 수 있기 때문에 프로그램을 이해하고 만들기 쉽게 한다.
- 함수의 종류
  - 내장 함수
  - 사용자정의 함수
  - 매크로 함수
- 함수=프로시저=메소드

내장 함수

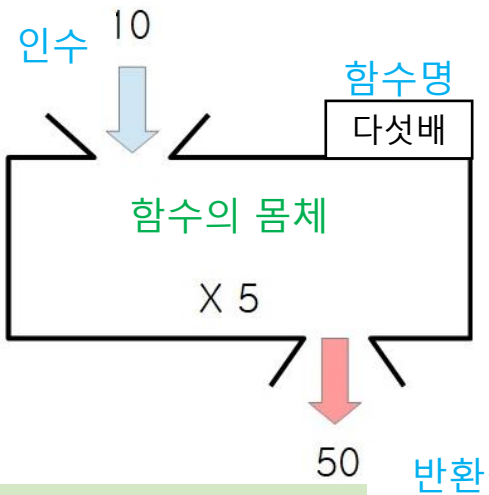
- 헤더파일의 종류

종류	기능	내장함수
stdio.h	표준 입출력 함수 등을 정의	**printf( )**, **scanf( )**, **gets( )**, getchar( ), **puts( )**, putchar( ), fgetc( ), fgets( ), fputc( ), fputs( ), fopen( ), fclose( ) 등
conio.h	직접 콘솔 입출력 함수 등을 정의	getch( ), getche( ), putch( ), cgets( ) 등
math.h	수학 함수와 매크로 정의	sin( ), cos( ), tan( ), exp( ), log( ), **sqrt( )**, **abs( )**, fabs( ), **pow( )**, fmod( ) 등
string.h	문자열 처리 함수 정의	**strlen,(), strcat( )**, **strcpy( )**, **strcmp( )**, strncat( ) 등
ctype.h	문자 검사 매크로 정의	isalpha( ), islower( ), isupper( ), tolower( ), toupper( ) 등

사용자 정의 함수

- 형식

```
[함수의 리턴형] 함수명([인수1,인수2…])
{
    문장1;
    문장2;
     …
     …
    문장n;
    [return] [리턴값]
}
```



- 예

```
[리턴형] 함수명 (인수)
int    main ( ) {
     함수의 몸체
}

int add(int a, int b) {
    int sum = a + b;
    return sum;
}

char upper(char ch) {
     return ch-32;
}
```

사용자 정의 함수

- 함수의 형태
  - 인수
    - 없는가?
    - 있는가? 있다면 한 개인가, 두 개인가?
  - 리턴값
    - 없는가?
    - 있는가? (한 개만 리턴 가능)
  - 다양한 형태의 함수 모양이 나올 수 있음

- 형태

인수	리턴	형태
X	X	void func() void func(void)
X	O	int func() double func(void)
O	X	void func(int ar) void func(char c)
O	O	int func(int ar) int func(int a, int b)

사용자 정의 함수

- Case1: 인수 X, 리턴값 X
  - 기능: "Copyright" 출력
  - 함수명: output
  - 인수: 없음
  - 리턴값: 없음

- 함수 구현

```
void output() {
    printf("---------------\n");
    printf(" function test\n");
    printf("---------------\n");
    return;
}
```

사용자 정의 함수

- Case2: 인수 O, 리턴 O
  - 기능: x+y 값을 구한다
  - 함수명: add
  - 인수: x, y 2개

    x:int, y:int
  - 리턴값: x+y

    리턴형: int

- 함수 구현

```
int add(int x, int y) {
    int sum = x + y;
    return sum
}
```

사용자 정의 함수

- 함수의 호출

```
#include <stdio.h>
int add(int x, int y) {
    int sum = x + y;
    return sum;
}

int main() {
    int a=5, b=4;
    int sum=0;

    sum = add(a, b);
    printf("%d \n", sum);
}
```

- 메모리에서 일어나는 일
  - 지역변수는 함수 호출 시 생성 되고, 함수가 종료되면 자동으로 파괴된다.

소속	변수	값
add	sum	9
	y	5
	x	4

copy

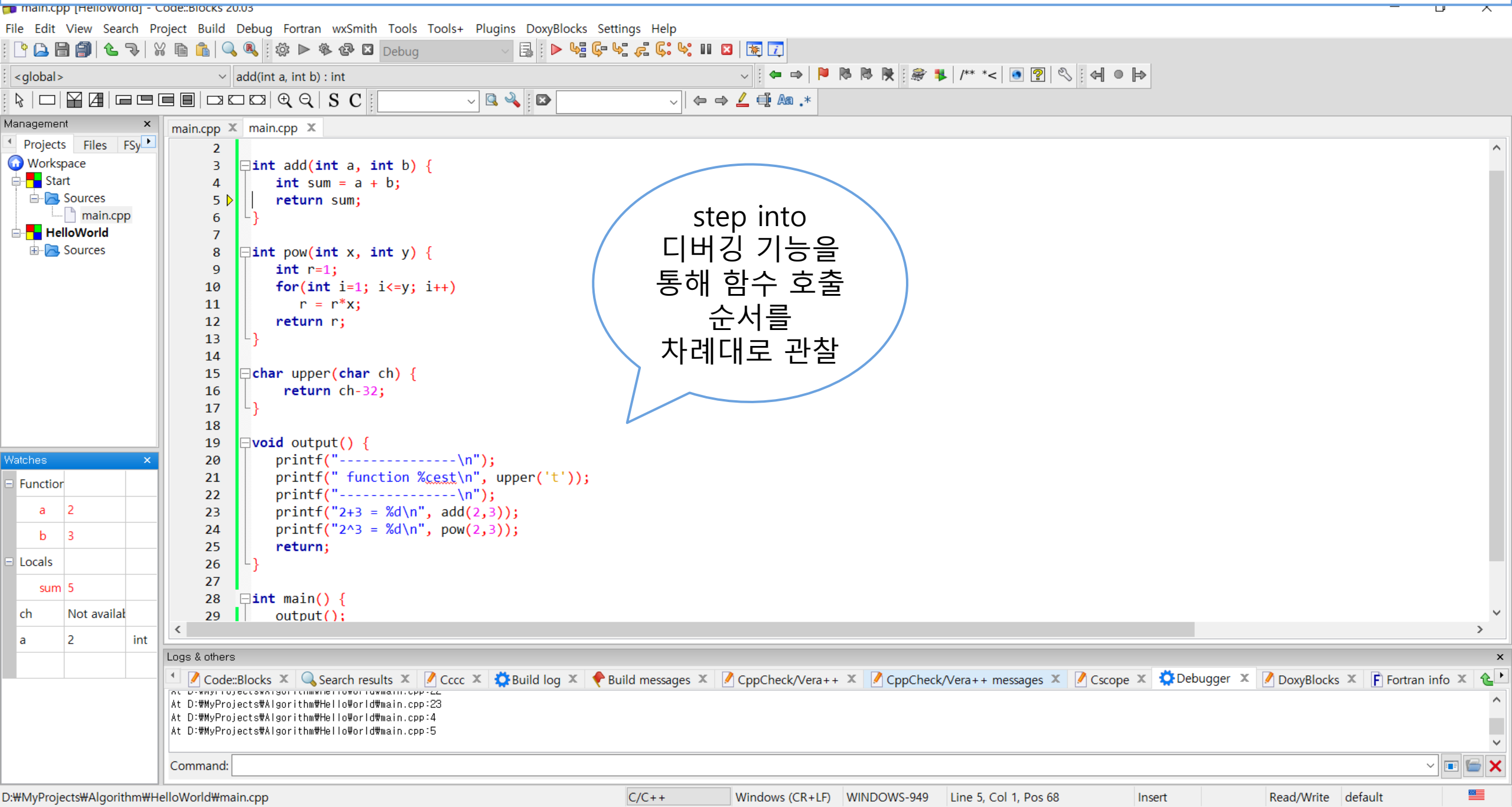
소속	변수	값
main	sum	9
	b	5
	a	4

사용자 정의 함수

```c
#include <stdio.h>

int add(int a, int b) {
   int sum = a + b;
   return sum;
}

int pow(int x, int y) {
   int r=1;
   for(int i=1; i<=y; i++)
      r = r*x;
   return r;
}

char upper(char ch) {
    return ch-32;
}
```

```c
void output() {
   printf("---------------\n");
   printf(" function %cest\n", upper('t')); //함수호출
   printf("---------------\n");
   printf("2+3 = %d\n", add(2,3)); //함수호출
   printf("2^3 = %d\n", pow(2,3)); //함수호출
   return;
}

int main() {
   output();  //함수호출
}
```

Debugging: Step into [shift]+F7



사용자 정의 함수

함수의 프로토타입을 미리 알려준다

```c
#include <stdio.h>

int add(int a, int b);
int pow(int x, int y);
char upper(char ch);

void output() {
   printf("---------------\n");
   printf(" function %cest\n", upper('t'));
   printf("---------------\n");
   printf("2+3 = %d\n", add(2,3));
   printf("2^3 = %d\n", pow(2,3));
   return;
}

int main() {
   output();
}
```

```c
int add(int a, int b) {
   int sum = a + b;
   return sum;
}

int pow(int x, int y) {
   int r=1;
   for(int i=1; i<=y; i++)
      r = r*x;
   return r;
}

char upper(char ch) {
    return ch-32;
}
```

Logs & others

Code::Blocks | Search results | Cccc | Build log

File	Line	Message
D:₩MyProjects...	5	error: 'upper' was not declared in this scope
D:₩MyProjects...	5	note: suggested alternative: '_popen'
D:₩MyProjects...	7	error: 'add' was not declared in this scope
D:₩MyProjects...	8	error: 'pow' was not declared in this scope
D:₩MyProjects...	8	note: suggested alternative: 'putw'

지역변수 / 전역변수

- 지역변수
  - 함수 안에서 선언된 변수
  - 해당 함수 안에서만 사용가능
  - 초기값이 쓰레기 값이다
  - 함수가 호출되면 생성되고 함수가 종료되면 사라진다
  - 동일한 이름의 전역/지역변수가 존재하면 지역변수가 우선한다
  - 스택에 저장
- 전역변수
  - 함수 외부에서 선언된 변수
  - 어느 함수에서든 사용가능
  - 초기값이 `0` 이다
  - 프로그램이 실행 중이면 항상 존재한다
  - 프로그램을 이해하기 어렵게 만드므로 꼭 필요한 경우에만 사용하자
  - 전역공간에 저장

지역변수 / 전역변수

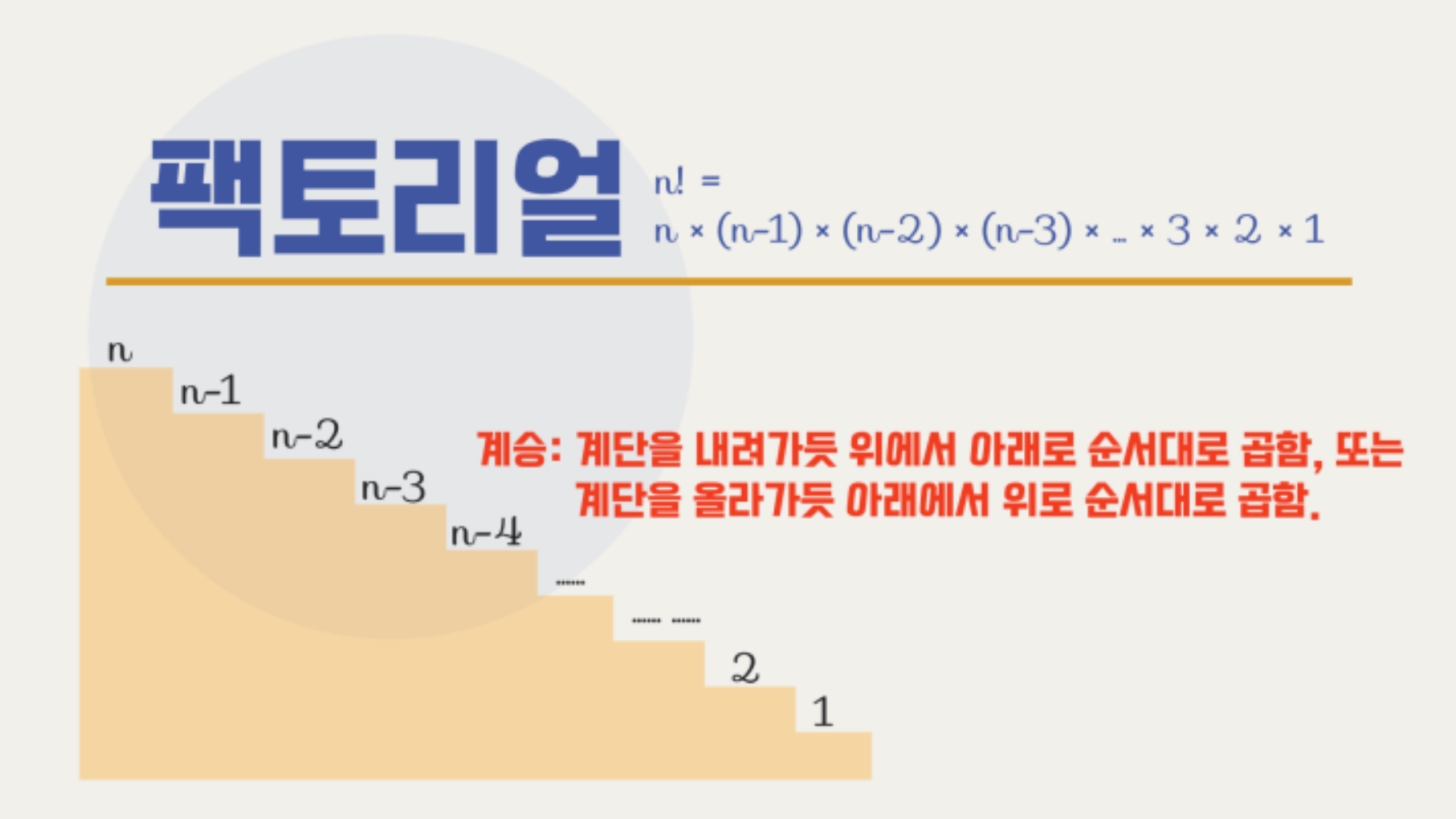
- 지역변수 예시

```c
#include <stdio.h>
// add의 sum과 main의 sum은 동명이인

int add(int a, int b) {
    int sum = a + b;
    return sum;
}

int main(){
    int sum=0;
    add(1, 2);
    printf("%d\n", sum);
}
```

- 전역변수 예시

```c
#include <stdio.h>
// add의 sum과 main의 sum은 동일변수

int sum;
void add(int a, int b) {
    sum = a + b;
}

int main(){
    add(1, 2);
    printf("%d\n", sum);
}
```

팩토리얼 계산


팩토리얼
n! =
n × (n-1) × (n-2) × (n-3) × … × 3 × 2 × 1
n
n-1
n-2
n-3
n-4
……
…… ……
2
1
계승: 계단을 내려가듯 위에서 아래로 순서대로 곱함, 또는
계단을 올라가듯 아래에서 위로 순서대로 곱함.

팩토리얼 계산

- 비 재귀적 해결
  - ex) 팩토리얼 계산

```
n! = n x (n-1) x (n-2) x … x 1

5! = 5 x 4 x 3 x 2 x 1 = 120

3! = 3 x 2 x 1 = 6
```

- 답안

```
#include <stdio.h>
int factorial(int n) {



                    ?



    return r;
}
int main() {
    printf("6! = %d\n", factorial(6));
}
```

재귀함수

- 재귀함수
  - 실행 도중 자기 자신을 호출(재귀 호출)하는 함수
  - ex) 팩토리얼 계산

```
5! = 5 x 4 x 3 x 2 x 1
5! = 5 x 4!
        4! =4x3!
              3!=3x2!
                    2!=2x1!
                          1!=1
```

$$f(n) \begin{cases} n \times f(n-1) & \dots\ n \geq 2 \\ 1 & \dots\ n = 1 \end{cases}$$

- 함수 예시
  - 탈출조건이 없으면 무한루프가 되므로 유의

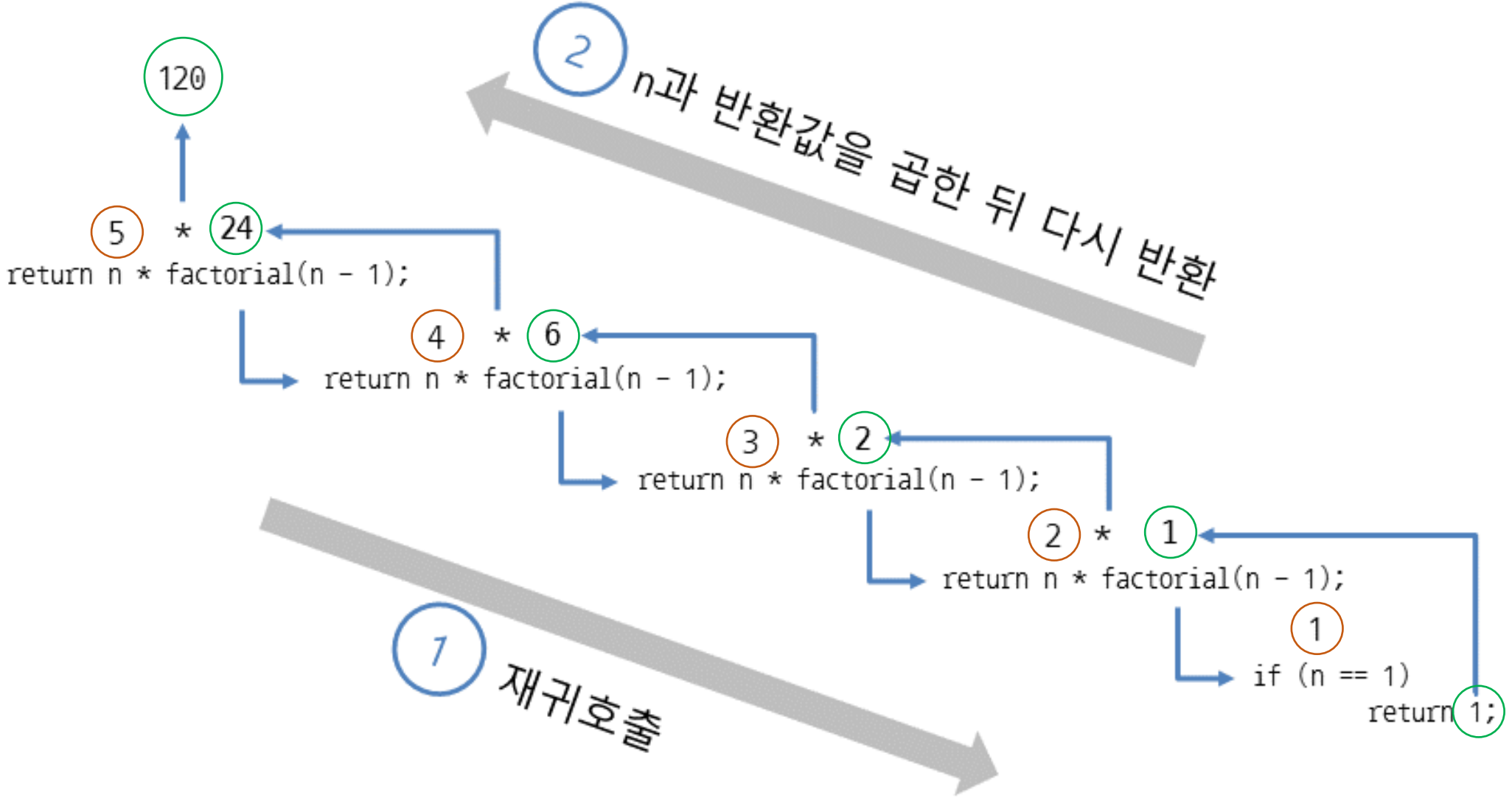
```
int factorial(int n) {
   if(n >= 2)
      return n*factorial(n-1);
   else
      return 1;
}

// 3항 조건 연산자를 활용하여
// 아래와 같이 표현해도 동일한 효과
int factorial(int n) {
   return (n>=2)? n*factorial(n-1) : 1;
}
```

재귀함수

factorial(5)

계산과정

묘사

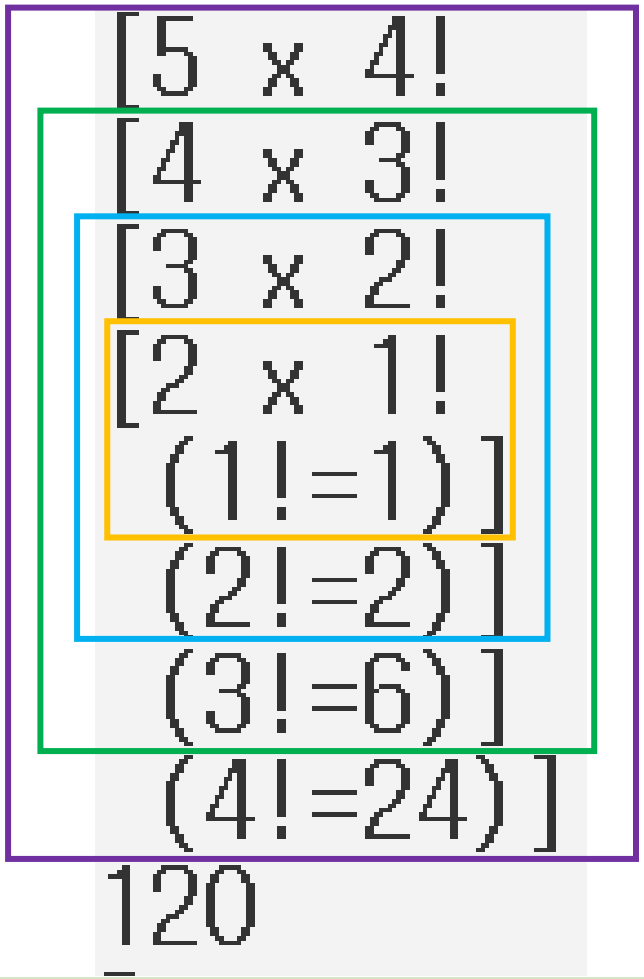


재귀함수

- 재귀함수 호출 관찰

```
#include <stdio.h>

int fact(int n) {
    if(n >= 2) {
        printf("[%d x %d!\n", n, n-1);
        int f = fact(n-1);
        printf(" (%d!=%d)]\n", n-1, f);
        return n*f;
    }
    else
        return 1;
}

int main() {
    printf("%d", fact(5));
}
```

- 함수 예시



재귀함수

```
#include <stdio.h>
int factorial(int n) {
   printf("%d! = ", n);
   if(n >= 2) {
      printf("%d * %d!\n", n, n-1);
      int ans = n*factorial(n-1);
      printf("%d! returned %d\n", n, ans);
      return ans;
   }
   else
      printf("returned 1\n") ;
      return 1;
}

int main() {
    printf("6! = %d\n", factorial(6));
}
```

```
6! = 6 * 5!
5! = 5 * 4!
4! = 4 * 3!
3! = 3 * 2!
2! = 2 * 1!
1! = returned 1
2! returned 2
3! returned 6
4! returned 24
5! returned 120
6! returned 720
6! = 720
```

재귀 함수 – 연습문제1

- 피보나치 수 찾기
  - 첫째 항 및 둘째 항이 1이며, 그 뒤의 모든 항은 바로 앞 두 항의 합인 수열
  - 처음 여섯 항은 각각 1, 1, 2, 3, 5, 8 이다.
  - 숫자 K를 입력 받아 K번째에 해당하는 피보나치 수를 출력하는 알고리즘을 작성하시오.

- 함수 구현

```
int fibo(int n) {



}
```

재귀 함수 – 연습문제2

- 계단을 오르는 방법
  - 계단을 한 번에 한 칸 또는 두 칸 만 오를 수 있다고 할 때 n 칸으로 되어 있는 계단 전체를 오르는 방법은 몇 가지가 있는가?

  - 힌트1
    - 1칸 계단: 1가지 방법
    - 2칸 계단: 2가지 방법
    - 3칸 계단은?

  - 힌트2: n칸 계단에 오르는 방법
    - n-2칸 까지 올라온 다음 두 칸 오른다 +
    - n-1칸 까지 올라온 다음 한 칸 오른다

- 함수로 표현

예를 들어,

f(n) : n개의 계단일 때 오르는 방법의 수

f(1) = 1

f(2) = 2

f(3) = f(2) + f(1)

f(4) = f(3) + f(2)

f(5) = f(4) + f(3)

:

f(n) = f(n-1) + f(n-2)

연습문제 풀이: 계단오르기

- 고찰

계단	오르는 방법		방법 개수	
(1)	①	①	1	1
(2)	①+① ②	(1)+① ②	1 1	2
(3)	①+② ①+①+①, ②+①	(1)+② (2)+①	1 2	3
(4)	①+①+②, ②+② ①+②+①, ①+①+①+①, ②+①+①	(2)+② (3)+①	2 3	5
(5)	①+②+②, ①+①+①+②, ②+①+② ①+①+②+①, ②+②+①, ①+②+①+①, …	(3)+② (4)+①	3 5	8
(6)	생략 생략	(4)+② (5)+①	5 8	13

- 점화식 표현

```
f(1) = 1
f(2) = 2
f(3) = 3
f(n) = f(n-2)+f(n-1)
```

연습문제 풀이: 계단오르기

```
#include <stdio.h>

int count(int stairs) {




}

int main() {
   int stairs;
   scanf("%d", &stairs);
   printf("%d\n", count(stairs));
}
```

- 함수로 표현

예를 들어,

f(n) : n개의 계단일 때 오르는 방법의 수

f(1) = 1

f(2) = 2

f(3) = f(2) + f(1)

f(4) = f(3) + f(2)

f(5) = f(4) + f(3)

:

f(n) = f(n-1) + f(n-2)

배열

- 배열
  - 같은 형식의 여러 데이터를 하나의 변수에 긴 띠 모양으로 저장하여 사용하는 자료의 집합체
  - 줄줄이 연결된 타입이 동일한 변수들의 집합

- 선언
  - 형식

    데이터형 배열명[원소의 수]

  - 선언 예

    int kor[5];

    kor변수 5개 만듦

    kor[0] ~ kor[4]

  - 배열의 구조
    - 0번 인덱스부터 시작됨에 유의

kor[0]	kor[1]	kor[2]	kor[3]	kor[4]

배열

- 대입

```
kor[0] = 60;
kor[1] = 60;
kor[2] = 60;
kor[3] = 60;
kor[4] = 60;
```

- 반복문 이용한 대입

```
for(i=0; i<5; i++)
    kor[i] = 60;
```

- 초기화

```
int a[5] = {3,2,7,6,9};
int b[] = {3,6,5,4};
int c[5] = {5,8,3};
int d[5] = {4,};
static int e[5];
```

배열

- 배열의 순회1

```
#include <stdio.h>

int main() {
  int a[10]={1,3,7,6,4,8,9,12,2,10};

  // 0번부터 시작하여 n-1에서 끝남에 유의
  for(int i=0; i<10; i++) {
    printf("%4d", a[i]);
  }
  return 0;
}
```

- 배열의 순회2

```
#include <stdio.h>

int main() {
  int a[10]={1,3,7,6,4,8,9,12,2,10};
  int i;
  // 0번부터 시작하여 n-1에서 끝남에 유의
  i=0;
  while(i<10) {
    printf("%4d", a[i]);
    i++;
  }
  return 0;
}
```

배열

- 배열의 합

```
#include <stdio.h>

void main() {
  int i, sum=0;
  int a[10]={1,3,7,6,4,8,9,12,2,10};

  for(i=0; i<10; i++) {
    sum = sum + a[i];
  }
  printf("sum = %d\n", sum);
}
```

- 피보나치수

```
#include <stdio.h>

void main() {
  int i, fibo[10]={1,1};

  for(i=2; i<10; i++) {
    fibo[i]=fibo[i-1]+fibo[i-2];
  }

  for(i=0; i<10; i++) {
    printf("%4d\n", fibo[i]);
  }
}
```

숫자 목록에서 수 찾기(선형탐색)

- 문제

  n개로 이루어진 정수 목록에서 원하는 수의 위치를 찾으시오.
  단, 입력되는 정수 목록에 같은 수는 없다.

- 입력

  첫 줄에 한 정수 n이 입력된다.
  (2 <= n <= 100,000)
  둘째 줄에 n개의 정수가 공백으로 구분되어 입력된다.
  (입력되는 모든 정수는 21억 보다 작다)
  셋째 줄에는 찾고자 하는 수가 입력된다.

- 출력

  찾고자 하는 원소의 위치를 출력한다.
  없으면 -1을 출력한다.

- 입력과 출력의 예

입력 예	출력 예
8 1 2 3 5 7 9 11 15 11	7

최댓값 찾기

- 문제

  9개의 서로 다른 자연수가 주어질 때, 이들 중 최댓값을 찾고 그 값이 몇 번째 수 인지를 구하는 프로그램을 작성하시오. 예를 들어, 서로 다른 9개의 자연수가 각각 3, 29, 38, 12, 57, 74, 40, 85, 61 라면, 이 중 최댓값은 85이고, 이 값은 8번째 수이다

- 입력

  첫째 줄부터 아홉째 줄까지 한 줄에 하나의 자연수가 주어진다. 주어지는 자연수 는 100보다 작다.

- 출력

  첫째 줄에 최댓값을 출력하고, 둘째 줄에 최댓값이 몇 번째 수인지를 출력한다.

- 입력과 출력의 예

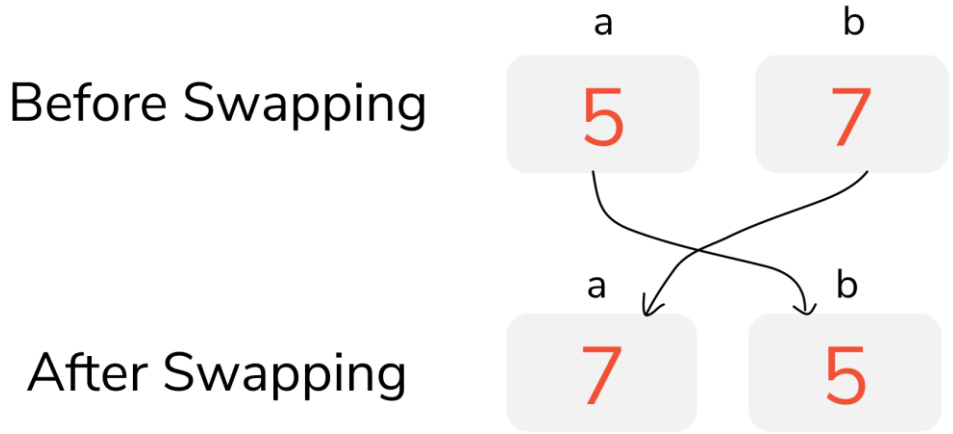
입력 예	출력 예
3 29 38 12 57 74 40 85 61	85 8

- 출처

  한국정보올림피아드(2007 지역본선 초등부)

SWAP

- 두 변수 내용물을 서로 교환하는 연산



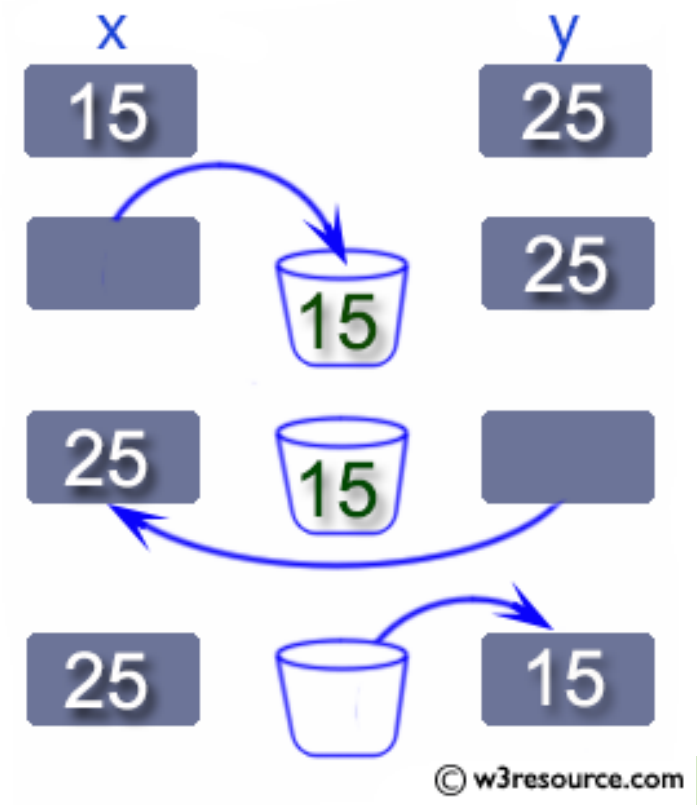
- 잘못된 구현

```
#include <stdio.h>

int main() {
   int a=5, b=7;
   printf("%d %d\n", a, b);

   a=b;
   b=a;

   printf("%d %d\n", a, b);
   return 0;
}
```

```
5 7
7 7
```

SWAP

- 두 변수 내용물을 서로 교환하는 연산
- 임시 변수가 필요함.



- 올바른 구현

```
#include <stdio.h>

int main() {
   int a=5, b=7;
   printf("%d %d\n", a, b);

   int t=a;
   a=b;
   b=t;

   printf("%d %d\n", a, b);
   return 0;
}
```

```
5 7
7 5
```

SWAP

- 고급 구현

```
#include <stdio.h>

// C++의 Generic과 참조자를 사용
template <class T>
inline void SWAP(T& a, T& b)
{
    T temp = a;
    a = b;
    b = temp;
}
```

제네릭과 참조자는 본 수업의 범위를 벗어나는 내용이므로 자세한 설명은 생략한다.

```
int main() {
    int a=5, b=7;
    printf("%d %d\n", a, b);

    SWAP(a, b);

    printf("%d %d\n", a, b);
    return 0;
}
```

```
5 7
7 5
```

STL sort() 함수 사용하기

- sort() 함수의 사용
  - C++의 STL 사용
  - #include <algorithm> 필요
  - sort(배열시작, 배열끝, [비교함수])
  - 기본 정렬 방식은 오름차순
- 비교함수(true일 때 SWAP 됨)

```
// 오름차순용
bool asc_order(int a, int b) {
    return a < b;
}
// 내림차순용
bool desc_order(int a, int b) {
    return a > b;
}
```

```
#include <stdio.h>
#include <algorithm>
using namespace std;

void show_array(int a[], int len) {
    for(int i=0; i<len; i++)
        printf("%5d", a[i]);
    printf("\n\n");
}

int main() {
    int a[10] = {7, 5, 8, 1, 4, 9, 2, 10, 6, 3};
    show_array(a, 10);

    sort(a, a+10);                  // 오름차순 정렬
    show_array(a, 10);

    sort(a, a+10, desc_order);  // 내림차순 정렬
    show_array(a, 10);
    return 0;
}
```

정렬하여 k번째 수 찾기

- 문제

  n개의 정수를 배열에 입력 받아 정렬한 뒤, k번째로 큰 숫자를 찾는 프로그램을 작성하시오. 만약 네 개의 정수 1, 2, 3, 4가 입력되었다면, 3번째로 큰 수는 2이다.

- 입력

  첫 번째 줄에 입력 받을 자료의 개수 n이 입력된다. 두 번째 줄부터 정수 n개가 한 줄에 하나씩 차례대로 입력된다. 마지막 줄에는 k가 입력된다.

- 출력

  입력된 자료들 가운데 k번째로 큰 숫자를 출력한다.

- 입력과 출력의 예

입력 예	출력 예
4 1 2 3 4 3	2

- 고찰

  이 문제를 풀려면 오름차순 정렬을 사용해야 하는가? 내림차순 정렬을 사용해야 하는가?

에라토스테네스의 체

- 에라토스테네스의 체

isPrime[]

idx	0	1	2	3	4	5	6	…
val	1	1	0	0	0	0	0	…

- 소스코드

```
#include <stdio.h>
#define MAX  101
int isPrime[MAX] = {1, 1, 0, };
int main() {
  int cnt=0;
  // 1: 소수아님, 0: 소수
  for(int i=2; i<MAX; i++) {
    if(isPrime[i]==0) { // 현재수만 소수이고
      printf("%5d", i);
      cnt++;
      for(int j=i+i; j<MAX; j+=i) // 배수들은
         isPrime[j]=1;  // 소수아님으로 셋팅
    }
  }
  printf("\n1~100사이 %d개의 소수를 찾아냄\n", cnt);
}
```

에라토스테네스의 체

- 문제

  자연수 a, b 구간 사이에 존재하는 소수의 개수를 알아내는 프로그램을 작성하시오.

- 입력

  두 자연수 a와 b가 공백으로 분리되어 입력된다. (1 ≤ a, b ≤ 500,000,000)

- 출력

  a, b 구간 사이에 존재하는 소수의 개수를 출력한다.

- 입력과 출력의 예

입력 예	출력 예
1 10	4

```
#include <stdio.h>
#define MAX  50000000
char isPrime[MAX+1] = {1, 1, 0, };

int main() {
    int a, b, cnt=0;
    scanf("%d %d", &a, &b);









    printf("%d", cnt);
}
```

다차원 배열

- 2차원 배열의 선언
  - 1차원 배열을 여러 개 겹쳐 놓은 것
  - 행과 열의 평면 구조를 가진 배열
  - 선언

    데이터형 배열명[행 수][열 수]

  - 선언 예

    int a[4][5];

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]

- 2차원 배열의 초기화

int a[2][3] = {3, 2, 7, 6, 9, 8};

3	2	7
6	9	8

int b[2][3] = {5,8,3,7};

5	8	3
7	쓰레기	쓰레기

int c[2][3] = {4, };

4	0	0
0	0	0

int d[2][3] = { {3, }, {7,6,9} };

3	0	0
7	6	9

int e[][3] = { {3,2,6}, {7,6,9} };

다차원 배열

- 2차원 배열의 순회(행 우선)

```
#include <stdio.h>
#define ROW 3
#define COL 4

int main() {
    int a[ROW][COL] = {
        {1,2,3,4},{5,6,7,8},{9,10,11,12} };

    for(int r=0; r<ROW; r++) {      //행 순회
        for(int c=0; c<COL; c++) { //열 순회
            printf("%4d", a[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

1	2	3	4
5	6	7	8
9	10	11	12

```
   1   2   3   4
   5   6   7   8
   9  10  11  12
```

- 2차원 배열의 순회(열 우선)

```
#include <stdio.h>
#define ROW 3
#define COL 4

int main() {
    int a[ROW][COL] = {
        {1,2,3,4},{5,6,7,8},{9,10,11,12} };

    for(int c=0; c<COL; c++) {      //열 순회
        for(int r=0; r<ROW; r++) { //행 순회
            printf("%4d", a[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

1	2	3	4
5	6	7	8
9	10	11	12

```
   1   5   9
   2   6  10
   3   7  11
   4   8  12
```

격자판의 최댓값

- 문제

  <그림1>과 9x9 격자판에 쓰여진 81개의 자연수가 주어질 때, 이들 중 최댓값을 찾고 그 최댓값이 몇 행 몇 열에 위치한 수인지 구하는 프로그램을 작성하시오.

	1열	2열	3열	4열	5열	6열	7열	8열	9열
1행	3	23	85	34	17	74	25	52	65
2행	10	7	39	42	88	52	14	72	63
3행	87	42	18	78	53	45	18	84	53
4행	34	28	64	85	12	16	75	36	55
5행	21	77	45	35	28	75	90	76	1
6행	25	87	65	15	28	11	37	28	74
7행	65	27	75	41	7	89	78	64	39
8행	47	47	70	45	23	65	3	41	44
9행	87	13	82	38	31	12	29	29	80

<그림 1>

예를 들어, 왼쪽과 같이 81개의 수가 주어질 경우에는 이들 중 최댓값은 90이고, 이 값은 5행 7열에 위치한다.

출처: 한국정보올림피아드(2007 지역예선 중고등부)

- 입력

  첫째 줄부터 아홉째 줄까지 한 줄에 아홉 개씩 자연수가 주어진다. 주어지는 자연수는 100보다 작다.

- 출력

  첫째 줄에 최대값을 출력하고, 둘째 줄에 최댓값이 위치한 행 번호와 열번호를 빈칸을 사이에 두고 차례로 출력한다. 최댓값이 두 개 이상인 경우 행 숫자가 가장 작은 위치를 출력한다.

입력 예	출력 예
3 23 85 34 17 74 25 52 65 10 7 39 42 88 52 14 72 63 87 42 18 78 53 45 18 84 53 34 28 64 85 12 16 75 36 55 21 77 45 35 28 75 90 76 1 25 87 65 15 28 11 37 28 74 65 27 75 41 7 89 78 64 39 47 47 70 45 23 65 3 41 44 87 13 82 38 31 12 29 29 80	90 5 7

격자판의 최댓값

- 문제

```
#include <stdio.h>

#define ROW 9
#define COL 9

int a[ROW][COL];

void input() {
    for(int r=0; r<ROW; r++)
        for(int c=0; c<COL; c++)
            scanf("%d", &a[r][c]);
}
```

```
int main() {
    input();

    int mr, mc, max=-1;









    printf("%d\n", max);
    printf("%d %d\n", mr+1, mc+1);
    return 0;
}
```

알고리즘의 효율성

- 이해의 복잡도
  - difficulty
  - 알고리즘 이해와 구현에 필요한 시간과 노력의 양

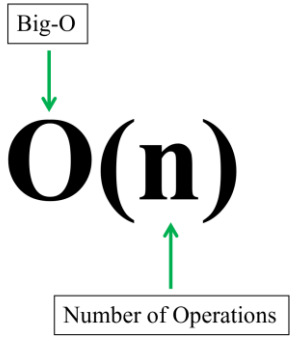
- 시간 복잡도
  - time complexity
  - 문제를 해결하는데 걸리는 시간과의 함수 관계
  - 반복문, 중첩된 반복문의 구조와 개수에 의해 결정

- 공간 복잡도
  - space complexity
  - 문제를 해결하기 위해 필요한 메모리(저장) 공간의 양
  - 변수 및 배열의 개수와 크기에 의해 결정
  - 함수가 호출될 때마다 사용되는 스택 공간이 늘어남

알고리즘의 시간 복잡도

- 빅오(O)표기법
  - 빅오 표기법은 알고리즘의 성능 평가 방법 중 가장 많이 사용하는 방법 중 하나
  - 가장 많이 사용하는 이유는 최악의 성능을 표시하기 때문
  - 최악의 성능 지표는, 적어도 이 정도의 성능은 보장한다는 의미
  - 실행 횟수를 점근적 표기법으로 표시

- 표기 형식



최소 n번은 연산해야 답이 나온다.

- 실행 횟수 계산
  - 프로그램은 첫번째 줄부터 마지막 줄까지 차례로 실행된다고 가정.
  - 헤더 파일은 알고리즘의 성능에 영향을 주지 않는다.
  - 함수 진입, 함수 반환은 알고리즘 성능에 영향을 주지 않는다.

알고리즘의 시간 복잡도

- 프로그램 예시

```
#define N 100                    // 영향을 주지 않는다.
#include <stdio.h>               // 영향을 주지 않는다.

void main(int)                   // 영향을 주지 않는다.
{
    int sum = 0;                 // 실행 횟수: 1회
    int i;                       // 실행 횟수: 1회

    for(i=1; i<=N; i++) {        // 실행 횟수: N+1회
        sum = sum + i;           // 실행 횟수: N회
    }

    printf("sum:%d\n",sum);      // 실행 횟수: 1회
    // 총 횟수: 1 + 1 + N+1 + N + 1 = 2N + 4회
}
```

- 실행 횟수 계산
  - 상수항은 무시
    - $O(101) \rightarrow O(1)$
    - $O(2N+1) \rightarrow O(N)$
  - 지배적이지 않은 항은 무시
    - $O(N^2+N) \rightarrow O(N^2)$
    - $O(N+logN) \rightarrow O(N)$
    - $O(100 \times 2^N + 500N^2) \rightarrow O(2^N)$

- 예시 프로그램의 Big-O: $O(N)$

빅오 표기의 종류

- $O(1)$
  - 상수시간(constant time)
  - 데이터 양과 상관없이 문제 해결에 항상 정해진 시간이 걸림
  - 평가: 최상의 알고리즘
  - 알고리즘 예
    - 정수의 홀짝 판별
    - 가우스의 누계 구하기
      - (시작수+마지막수) x n / 2

- $O(logN)$
  - 로그시간(logarithmic)
  - 데이터 양이 증가 함에 따라 실행 시간이 로그 함수 그래프로 나타남
  - 데이터가 많이 늘어나도 실행시간은 약간만 증가하는 특징
  - 평가: 좋은 알고리즘
  - 알고리즘 예
    - 이진탐색
      - 10개 일때, $\log_2 10 = 3.x$
      - 100개 일때, $\log_2 100 = 6.x$
      - 1000개 일때, $\log_2 1000 = 9.x$

빅오 표기의 종류

- $O(N)$
  - 선형시간(linear time)
  - 데이터 양의 증가에 따라 실행 시간이 일차 함수 그래프로 나타남
  - 테이터 증가량과 정비례하여 실행 시간이 증가하는 특징
  - 평가: 준수한 알고리즘
  - 알고리즘 예
    - 정렬되지 않은 배열에서 최댓값 찾기

- $O(N\ logN)$
  - 선형 로그 시간(linearithmic time)
  - 데이터 양의 증가에 따라 실행 시간이 일차 함수 + 로그함수 형태의 그래프로 나타남
  - 데이터의 증가량보다 실행시간이 더 많이 증가하는 특징
  - 평가: 봐줄만한 알고리즘
  - 알고리즘 예
    - 힙 정렬
    - 자이델(Seidel)의 다각형 삼각

빅오 표기의 종류

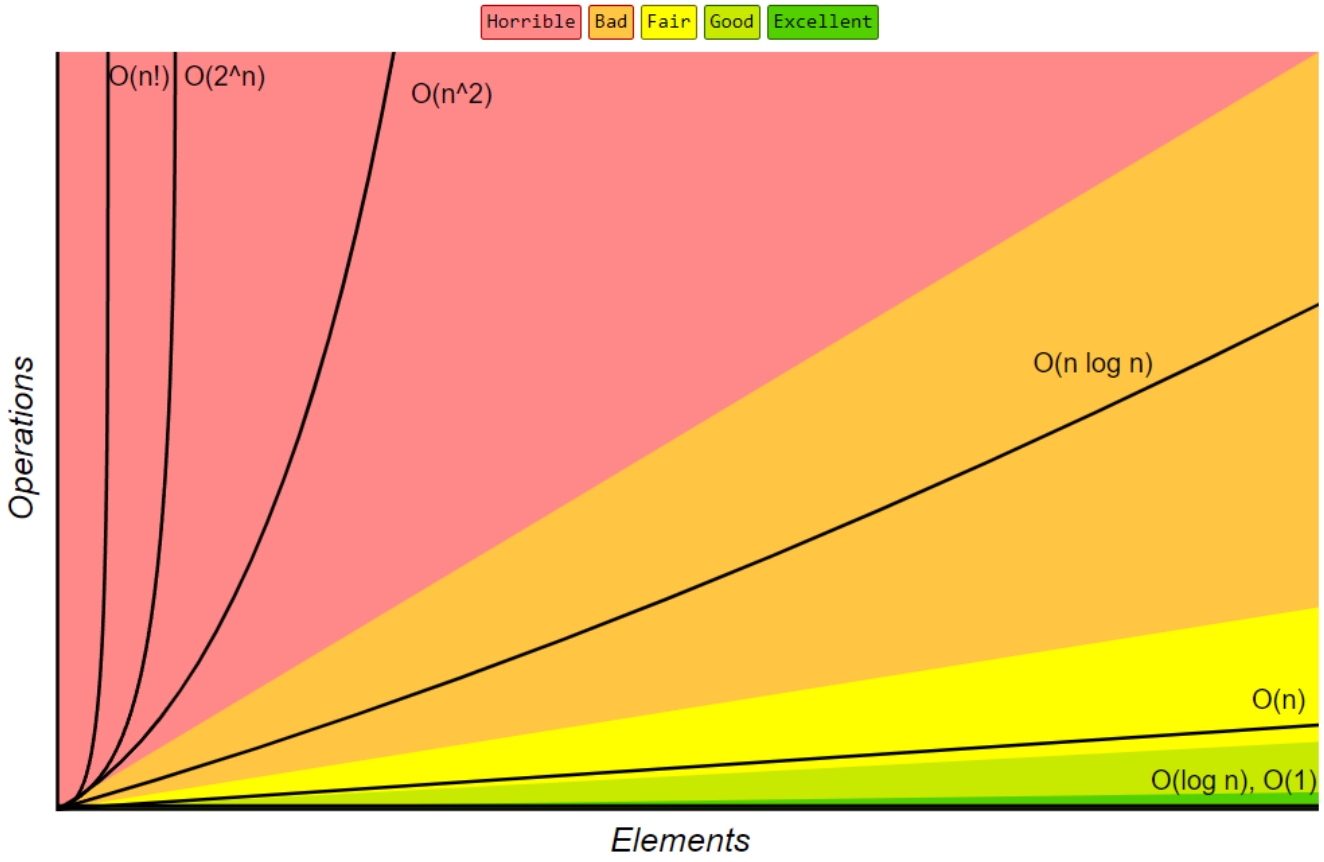
- $O(N^2)$
  - 이차식 시간(quadratic time)
  - 데이터 양의 증가에 따라 실행 시간이 이차 함수(N^2) 그래프로 나타남
  - 테이터 증가량에 제곱으로 비례하여 실행 시간이 증가하는 특징
  - 평가: 나쁜 알고리즘
  - 알고리즘 예
    - 선택정렬
    - 버블정렬
- $O(N^3)$
  - 삼차식 시간(cubic time)
  - 데이터 양의 증가에 따라 실행 시간이 삼차 함수(N^3) 그래프로 나타남
  - 테이터 증가량과 정비례하여 실행 시간이 증가하는 특징
  - 평가: 끔찍한 알고리즘
  - 알고리즘 예
    - 행렬 2개의 무식한 곱셈

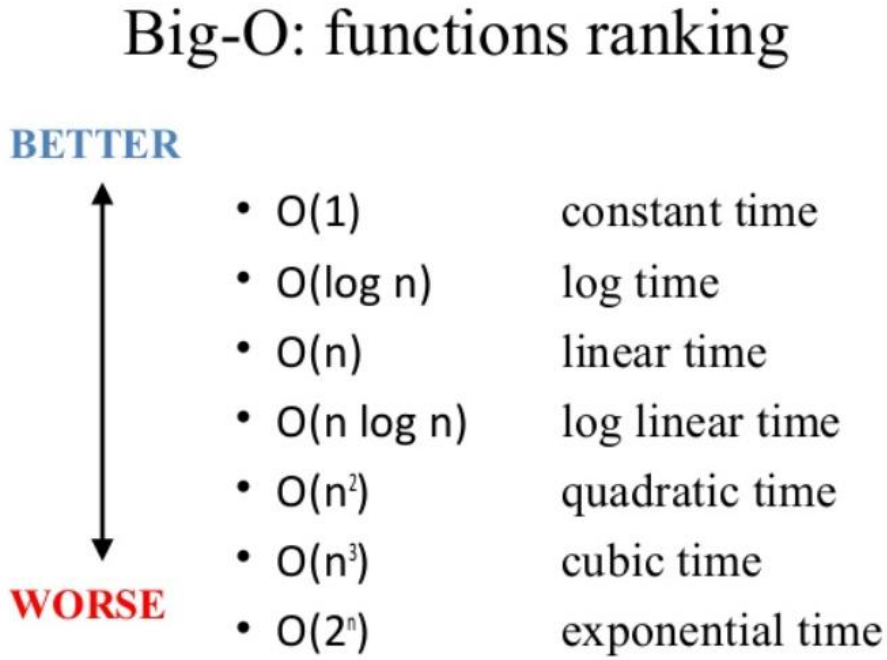
빅오 표기의 종류

- $O(2^N)$
  - 지수 시간(exponential time)
  - 데이터 양의 증가에 따라 실행 시간이 지수 함수(2^N) 그래프로 나타남
  - 테이터 증가량에 따라 실행 시간이 지수 형태로 증가하는 특징
  - 평가: 최악의 알고리즘
  - 알고리즘 예
    - 2^N을 재귀 호출로 계산
- $O(N!)$
  - 계승 시간(factorial time)
  - 데이터 양의 증가에 따라 실행 시간이 팩토리얼 함수($N!$) 그래프로 나타남
  - 평가: 노코멘트
  - 알고리즘 예
    - 브루트포스 탐색을 통한 외판원 문제 해결방법

빅오 표기의 종류

- 알고리즘 성능 비교
  - $O(1) > O(\log N) > O(N) > O(N \log N) > O(N^2) > \ \ldots \ > O(2^N) > O(N!)$





시간제한 피하기

- 주어진 입력 N의 크기에 따른 허용 시간 복잡도

N의 크기	시간복잡도
N ≤ 11	O(N!)
N ≤ 25	$O(2^N)$
N ≤ 100	$O(N^4)$
N ≤ 500	$O(N^3)$
N ≤ 3,000	$O(N^2 \log N)$
N ≤ 5,000	$O(N^2)$
N ≤ 1,000,000	O(NlogN)
N ≤ 10,000,000	O(N)
N > 10,000,000	O(logN), O(1)

- 활용 방법
  - 컴퓨터는 대략 1초에 1억회의 연산 수행한다고 가정하고 왼쪽 표를 얻어냄
  - 시간 제한은 대부분 1~5초
  - 입력 데이터가 5000개 이하로 주어진다면 $O(N^2)$ 또는 그보다 빠른 알고리즘을 설계하여 문제를 풀어야 함.
  - 입력 데이터가 25개 이하로 주어진다면 $O(2^N)$ 알고리즘만 되어도 통과 가능할 것임.