

# C++\_기초 문법

*Visual Studio 2022*

# C언어와 C++ 언어 비교

## ● C 언어

- **Ken Thompson, Dennis Ritchie(켄 톰슨, 데니스 리치)**
- AT & T 벨연구소, 1970년
- 유닉스(UNIX) 운영체제에서 사용하기 위해 만든 언어
- 고급언어이지만 메모리를 직접 접근하는 저급언어 특징, 운영체제나 시스템 프로그래밍에 적합하고, 하드웨어나 임베디드 시스템에서 많이 사용됨
- **절차적 프로그래밍** 기법
- 단점 : 다른 고급언어에 비해 이해가 쉽지 않고, 개발 기간이 오래 걸리며, 대형 프로젝트에 적합하지 않다.

# C언어와 C++ 언어 비교

## ● C++ 언어

- **Bjarne Stroustrup(비야네 스트루스트룹)**
- AT & T 벨연구소, 1983년
- C언어를 확장한 프로그래밍 언어
- 절차적 프로그래밍 기법을 지원하면서 추가적으로 **객체지향 프로그래밍 기법**
- 캡슐화, 정보은닉, 상속과 다형성
- 템플릿(template)을 통해 일반화프로그래밍 기법 제공
- 대규모 소프트웨어 개발에 적합하다.(은행 업무, 게임, 임베디드 프로젝트 등)

# C언어와 C++ 언어 비교

## ● C++에서 개선된 내용

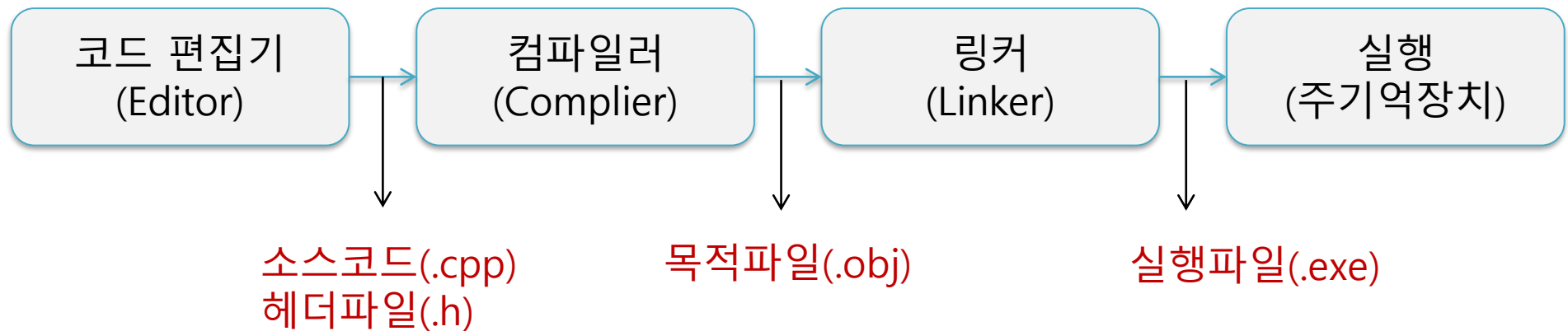
- **메모리 관리** : C 언어는 메모리 크기, 위치 등을 개발자가 직접 관리해야 했다. 하지만 C++은 자동으로 메모리를 할당하거나 해제할 수 있는 **new, delete** 키워드를 지원한다.
- **레퍼런스(참조)**: 포인터를 더 편리하게 사용할 수 있다.
- **객체 지향 언어의 다형성** : 가상함수, 연산자 오버로딩 기능 추가됨

# C언어와 C++ 언어 비교

## ● C++ 활용 분야

- **네이티브 애플리케이션 개발** : 플랫폼에 최적화된 기계어 코드를 만들어 냈으므로 엑셀, 포토샵 같은 애플리케이션 개발
- **과학 기술 계산**: 계산과 통신 성능을 특정 문제에 맞게 최적화 할 수 있는데, 분산, 병렬, 수치 해석 등 풍부한 라이브러리가 존재함
- **임베디드 프로그래밍** : 메모리와 하드웨어를 직접 다룰 수 있어 메모리 사용이 제한적인 임베디드 프로그래밍에 적합하다. (OS의 펌웨어와 IOT 마이크로컨트롤러 개발 프로그램)
- **게임 프로그래밍**: 성능이 중요한 게임 개발에 적합하며, 대규모 게임이나 고사양 게임에 적합(게임 엔진)

# C++ 프로그래밍 단계



## ➤ 빌드(build) 과정

- **전처리(preprocessing)** : #include(지시자), #define 등 실제값으로 대체하여 컴파일을 위한 최종 소스파일을 만드는 과정
- **컴파일(compile)** : 소스파일을 기계어로 변환하여 목적파일을 만든다.
- **링크(link)** : 실제로 외부(다른파일, 라이브러리)에 존재하는 함수나 전역 변수들을 찾아서 연결함

# C 프로그램의 구성 요소

- 세미콜론

- 문자의 끝은 항상 세미콜론(;)으로 끝난다

- 주석문

- ① 주석(Comment) : 소스코드에 대한 설명
- ② 컴파일 되지 않는다.
- ③ 주석 처리 방법

```
/*  
파일명: hello.cpp  
만든이: 김기용  
프로그램: Hello~ World 테스트  
*/
```

↑  
여러 줄 주석 처리

```
//std(클래스)는 namespace(이름 공간)  
//cout, endl은 함수  
std::cout << "Hello~ World!" << std::endl;  
std::cout << "안녕~ 세상" << std::endl;
```

↑  
한 줄 주석 처리

# C 프로그램의 구성 요소

- C와 C++ 의 입출력 비교

구분	C언어	C++
헤더 파일	<stdio.h>	<iostream>
출력문	printf()	cout <<
입력문	scanf()	Cin >>



# 〈iostream〉 헤더 파일

- C++ 표준 라이브러리 클래스

**cout** : 표준 출력을 담당하는 ostream 클래스의 객체이다.

**cin** : 표준 입력을 담당하는 istream 클래스의 객체이다.

**iostream** : 입출력에 필요한 클래스와 전역 객체들을 정의한 헤더파일이다.

cout << 데이터

데이터를 cout 스트림으로 출력한다.

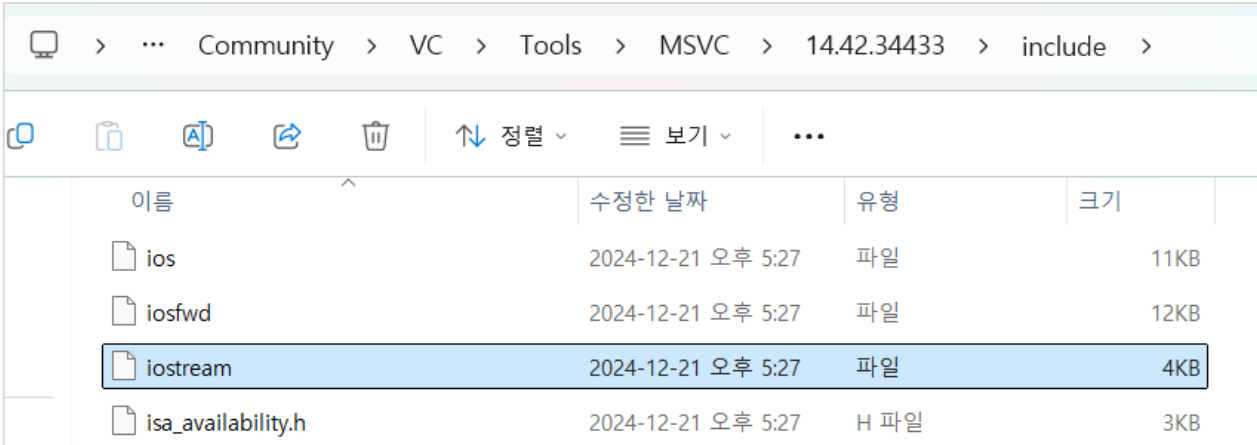
<< 연산자

- 정수를 왼쪽으로 쉬프트(shift)하는 연산자.
- 오른쪽 피연산자 데이터를 왼쪽 스트림 객체에 삽입한다.

# 〈iostream〉 헤더 파일

- <iostream> 헤더 파일의 위치는?

C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.42.34433\include



이름	수정한 날짜	유형	크기
ios	2024-12-21 오후 5:27	파일	11KB
iosfwd	2024-12-21 오후 5:27	파일	12KB
iostream	2024-12-21 오후 5:27	파일	4KB
isa_availability.h	2024-12-21 오후 5:27	H 파일	3KB

# 실습 예제

## ● printData.cpp

```
#include <iostream> //입출력을 위한 헤더파일
/*
파일명: hello.cpp
만든이: 김기용
프로그램: Hello~ World 테스트
*/

int main()
{
    /*
    std(소속을 의미하는 이름공간) > cout(클래스의 출력 객체)
    :: - 범위 연산자
    endl은 함수 - 줄바꿈 기능
    << - 시프트 연산자: 오른쪽 데이터를 왼쪽 스트림 객체에 넣는다.
    */
    std::cout << "Hello~ World!!" << std::endl;
    std::cout << "안녕~ 세계야!!" << std::endl;
}
```

# 실습 예제

## ● printData.cpp

```
//사칙 연산
std::cout << 4 + 5 << std::endl;
std::cout << 4 - 5 << std::endl;
std::cout << 4 * 5 << std::endl;
//- 4/5는 int형으로 반환되므로 소수 이하 손실됨
std::cout << 4.0 / 5.0 << std::endl;
}
```

```
Hello~ World!!
안녕~ 세계야!!
9
-1
20
0.8
```

# 변수(Variable)

- 변수란?

- 프로그램 내부에서 사용하는 데이터를 저장해두는 메모리 공간
- 한 순간에 한 개의 값을 저장한다.(배열-여러 개 저장)

- 변수의 선언 및 사용

- 자료형 변수이름;
- 자료형 변수이름 = 초기값;

변수 선언문

```
char ch;
```

```
int year = 2019;
```

```
double rate = 0.05;
```

이름

ch

year

rate

공간(값)



자료형

char

int

double

# 자료형(data type)

## ● 자료형이란?

- 데이터를 저장하는 공간의 유형

자료형		용량 (bytes)	주요 용도	범위
불리언	<b>bool</b>	1	true, false 표현	0 ~ 1
문자	<b>char</b>	1	문자 또는 작은 정수 표현	-128~127
정수	<b>short</b>	2	정수 표현	-32768~32767
	<b>int</b>	4	큰 범위의 정수 표현	-2147483648~2147483647
	<b>long</b>	8	큰 범위의 정수 표현	$-2^{63} \sim (2^{63}-1)$
실수	<b>float</b>	4	실수 표현	$10^{-38} \sim 10^{38}$
	<b>double</b>	8	정밀한 실수 표현	$10^{-380} \sim 10^{380}$
문자열	<b>string</b>		문자열 표현	크기가 정해지지 않음

▶ bool과 string은 C에는 없고 C++에 추가된 자료형이다.

# 변수의 선언과 사용

## ● variableEx.cpp

```
#include <iostream>
#include <string> //string 자료형 사용

using namespace std; //이름공간 선언: 이후 std 생략
/*
    변수와 자료형
    ++에 추가된 자료형 - bool, string
*/

int main()
{
    //문자형 자료
    char ch1 = 'A';
    char ch2 = 65;
    char ch3[] = "나"; //c언어 배열
    string str1 = "나"; //c++ string 자료형
    string str2 = "apple";
```

# 변수의 선언과 사용

```
cout << ch1 << endl;
cout << ch2 << endl;
cout << ch3 << endl;
cout << str1 << ", " << str2 << endl;

//bool 자료형 - 참(1), 거짓(0)
int n1 = 10, n2 = 20;
bool b1 = (n1 < n2);
bool b2 = (n1 == n2);

cout << b1 << ", " << b2 << endl; //1, 0

//조건 연산자
int result1 = (n1 > n2) ? 1 : 0;
string result2 = (n1 > n2) ? "true" : "false";

cout << result1 << ", " << result2 << endl; //0, false

return 0;
}
```



# 산술 연산자

## ● operatorEx.cpp

```
//사칙 연산
int n1 = 4, n2 = 5;
int add, sub, mul;
float div;

add = n1 + n2;
sub = n1 - n2;
mul = n1 * n2;
div = (float)n1 / n2; //강제 형변환: 정수 -> 실수

cout << add << endl; //9
cout << sub << endl; //-1
cout << mul << endl; //20
cout << div << endl; //0.8
```

# 산술 연산자

## ● operatorEx.cpp

```
//몫과 나머지  
int bread = 20, people = 3;  
int share, remainder;  
  
share = bread / people;  
remainder = bread % people;  
  
cout << "몫: " << share << endl;  
cout << "나머지: " << remainder << endl;
```

# 논리 연산자

## ● operatorEx.cpp

```
//논리 연산자
int a = 3, b = 4, c = 5;
int res1, res2, res3;

res1 = (a < b) && (b == c); // 1 && 0 = 0
res2 = (a < b) || (b == c); // 1 || 0 = 1
res3 = !(b > c); // !0 = 1

cout << res1 << endl;
cout << res2 << endl;
cout << res3 << endl;
```

# 상수(constant)

- 상수(constant)

- 한번 설정해 두면 그 프로그램이 종료 될 때까지 결코 변경될 수 없는 값
- 상수를 숫자로 직접 나타내는 것보다 이름을 붙여 사용하는 것이 좋다.
- 상수를 만드는 방법

1. **#define** 상수이름 상수값 – 매크로 상수로 정의(전처리, 컴파일 되지 않음)

```
#define PI 3.141592
```

2. **const** 자료형 상수이름 = 상수값(권장 – 유지보수 문제 등)

```
const double PI = 3.141592;
```

*//PI = 3.14 (변경할 수 없다.)*

# 상수(constant)

```
#include <iostream>
using namespace std;

#define PI 3.141592 //매크로 상수
int main()
{
    //원의 넓이 계산
    int radius = 6;
    const double PI2 = 3.141592; //원주율 상수
    double circleArea1, circleArea2;

    //PI = 3.14; //상수는 수정할 수 없음

    //1. 매크로 상수 활용
    cout << fixed; //소수 자리수 설정
    cout.precision(6);
    circleArea1 = PI * radius * radius;
    cout << "원의 넓이: " << circleArea1 << endl;
```

# 상수(constant)

```
//2. const 상수 키워드 활용(권장)
circleArea2 = PI2 * radius * radius; //정밀 계산

cout << fixed; //소수 자리수 설정
cout.precision(6);
cout << "원의 넓이: " << circleArea2 << endl;

return 0;
```

```
}
```

# 데이터 입력 받기

## ● 데이터 입력

cin >> 변수

cin 스트림으로 읽어온 데이터를 저장할 변수를 지정함

```
int number;    //학번
string name;   //이름
int eng;       //영어 점수
int math;      //수학 점수
double avg;    //평균

cout << "학번을 입력하세요: ";
cin >> number;
cout << "이름을 입력하세요: ";
cin >> name;
cout << "영어점수를 입력하세요: ";
cin >> eng;
cout << "수학점수를 입력하세요: ";
cin >> math;
```

# 데이터 입력 받기

## ● 데이터 입력

```
cout << "\n=== 학생의 정보 출력 ===\n";  
cout << "학번: " << number << endl;  
cout << "이름: " << name << endl;  
cout << "영어점수: " << eng << endl;  
cout << "수학점수: " << math << endl;
```

```
avg = (double)(math + eng) / 2;  
cout << "평균점수: " << avg << endl;
```

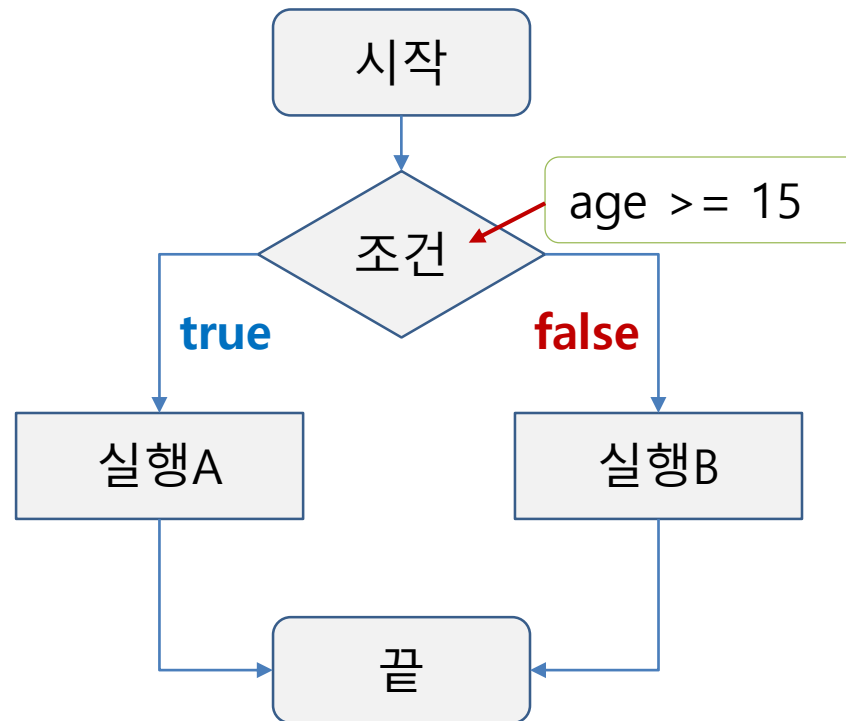
```
학번 입력 : 101  
이름 입력 : 김코딩  
영어점수 입력 : 85  
수학점수 입력 : 90  
=== 학생의 정보 출력 ===  
학번 : 101  
이름 : 김코딩  
영어점수 : 85  
수학점수 : 90  
평균점수 : 87.5
```



# 조건문(if문)

## ❖ 조건문이란?

- 특정한 조건에 의해서 프로그램 진행이 분기되는 구문
- if문, switch~case문이 대표적이다.



# 조건문(if ~ else 문)

## ▪ if문

```
if(조건식){  
    수행문;  
}
```

//조건식이 참이면 수행문 실행

## ▪ if-else 문

```
if(조건식){  
    수행문1;  
}  
else{  
    수행문2  
}
```

//조건식이 참이면 수행문1 실행,  
아니면 수행문2 실행

# 조건문(if ~ else 문)

## ■ 숫자와 문자 비교

```
//숫자 비교
int n1 = 10;
int n2 = 10;

if (n1 == n2)
{
    cout << "두 수는 같습니다.\n";
}
else
{
    cout << "두 수는 같지 않습니다.\n";
}
```

# 조건문(if ~ else 문)

## ■ 숫자와 문자 비교

```
//문자 비교
string str1 = "사과";
string str2 = "귤";

cout << str1.compare("사과") << endl; //0
cout << str1.compare("귤") << endl;  //1

if (str1.compare(str2) == 0) {
    cout << "두 문자열이 일치합니다.\n";
}
else {
    cout << "두 문자열이 일치하지 않습니다.\n";
}
```

# if ~ else if ~ else문

## ▪ if - else if - else 문(다중 조건문)

```
If(조건 1){  
    수행문1;  
}  
else if(조건 2)  
    수행문2  
}  
else{  
    수행문3  
}
```

//조건 1이 참이면 수행문1 실행, 조건2가 참이면 수행문2  
실행, 조건1,2가 모두 거짓이면 수행문3 실행

# if ~ else if ~ else문

## ■ 학점 계산하기

```
/*
점수에 따른 학점 산출하기
점수: 90 ~ 100 -> 'A'
점수: 80 ~ 90 -> 'B'
점수: 70 ~ 80 -> 'C'
점수: 70 미만 -> 'F'
*/

int score;
char grade = 'A';

cout << "점수를 입력하세요: ";
cin >> score;

if (score >= 90 && score <= 100)
{
    grade = 'A';
}
```

```
else if (score >= 80)
{
    grade = 'B';
}
else if (score >= 70)
{
    grade = 'C';
}
else
{
    grade = 'F';
}

cout << "학점은 " << grade << "입니다." << endl;
```

# 조건문(switch - case)

## ▪ switch-case문

조건식의 결과가 정수 또는 문자 값이고 그 값에 따라 수행문이 결정될때 if~else if ~ else문을 대신하여 switch-case문을 사용

```
switch(변수){  
    case 변수값:  
        실행문  
        break;  
    ...  
    default:  
        실행문  
}
```

//변수값에 해당하는 case 이면 실행문  
수행, 해당 값이 없으면 default 수행

# 조건문(switch - case)

## ▪ swtich-case문

```
/*
| 엘리베이터 타기 : 1 ~ 3층까지 있는 건물
*/
int floor;

cout << "가고 싶은 층을 누르세요: ";
cin >> floor;

switch (floor)
{
case 1:
|   cout << "1층을 눌렀습니다.\n";
|   break;
case 2:
|   cout << "2층을 눌렀습니다.\n";
|   break;
case 3:
|   cout << "3층을 눌렀습니다.\n";
|   break;
default:
|   cout << "건물에 없는 층입니다.\n";
|   break;
}
```



# 반복문(while문)

## ● 반복문

- 주어진 조건이 만족할 때까지 수행문을 반복적으로 수행함

- while, for 문이 대표적임

- **while 문**

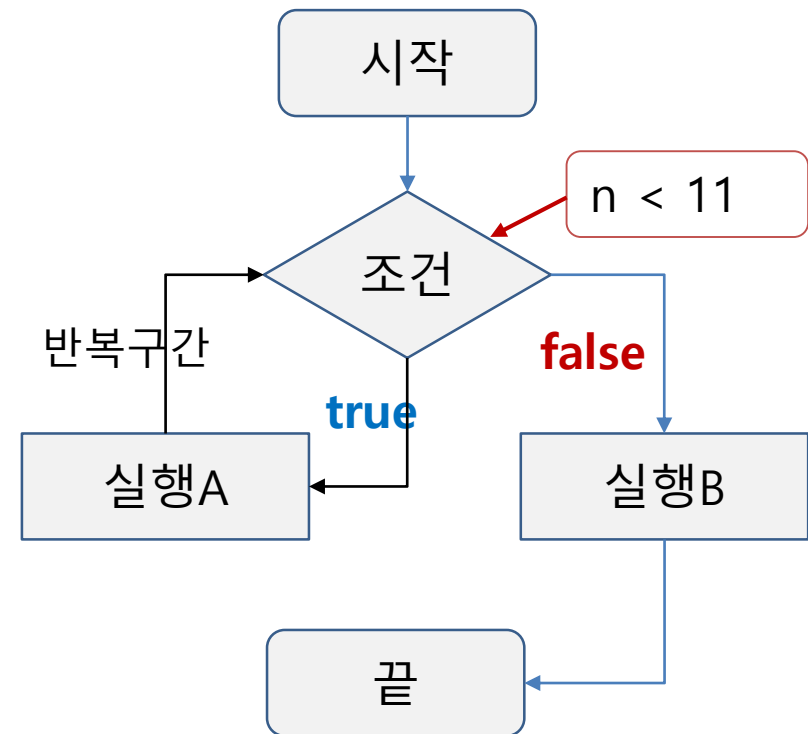
조건식이 참인 동안 반복 수행

```
while(조건식){
```

```
    수행문1;
```

```
    ...
```

```
}
```



# 기타 제어 -break 문

## ▪ break 문

반복문에서 break 문을 만나면 더 이상 반복을 수행하지 않고  
반복문을 빠져 나옴

```
while(조건식){  
    if(조건식){  
        break;  
    }  
}
```

while(1){ } – 무한 반복문  
1이면 참, 1이 아니면 거짓

# 반복문(while문)

## ▪ break문 예제

```
계속 반복할까요?(y/n 입력) Y
계속 반복!
계속 반복할까요?(y/n 입력) y
계속 반복!
계속 반복할까요?(y/n 입력) k
잘못된 입력입니다! 다시 입력하세요!
계속 반복할까요?(y/n 입력) n
반복 중단!
```

```
/*
    키보드 'y' or 'Y' 키를 누르면 - "계속 반복" 출력
    키보드 'n' or 'N' 키를 누르면 - "반복 중단" 출력
    그 이외의 키는 "잘못된 입력입니다. 다시 입력하세요!"
*/
string key; //입력할 키 변수

while (1)
{
    cout << "계속 반복할까요?(y/n 입력) ";
    cin >> key;
```

# 반복문(while문)

## ▪ break문 예제

```
//compare() - 문자 비교 일치하면 0, 일치하지 않으면 1
if (key.compare("y") == 0 || key.compare("Y") == 0)
{
    cout << "계속 반복!\n";
}
else if (key.compare("n") == 0 || key.compare("N") == 0)
{
    cout << "반복 중단!\n";
    break;
}
else
{
    cout << "잘못된 입력입니다! 다시 입력하세요!\n";
}
}
```

# 반복문(for문)

## ■ for 문

주로 조건이 횟수인 경우에 사용하는 반복문이다.  
초기화식, 조건식, 증감식을 한꺼번에 작성

```
for(초기화식; 조건식; 증감식){  
    수행문;  
}
```

## ■ for문 수행 과정

```
int n;  
    ① → ② ← ④  
for(n = 1; n <= 5; n++) {  
    ③ ↘ ↗  
    cout << n << endl ;  
}
```

# 반복문(for문)

## ▪ for문 예제

```
/*  
  1 ~ 10 까지 출력하기  
*/  
  
for (int n = 1; n <= 10; n++)  
{  
    cout << n << " ";  
}
```

```
/*  
  1 ~ 10 까지 합계 계산하기  
*/  
int sum = 0; //합계 변수 초기화  
  
for (int n = 1; n <= 10; n++)  
{  
    sum += n;  
    cout << "n=" << n << ", sum=" << sum << endl;  
}  
  
cout << "합계 : " << sum << endl;
```

# 구구단 출력

## ■ 구구단 출력 프로그램

```
/*  
    단을 입력받아 구구단 출력하기  
*/  
int dan;  
  
cout << "단 입력: ";  
cin >> dan;  
  
for (int n = 1; n < 10; n++)  
{  
    cout << dan << " x " << n << " = " << (dan * n) << "\n";  
}
```

```
단 입력 : 8  
8 x 1 = 8  
8 x 2 = 16  
8 x 3 = 24  
8 x 4 = 32  
8 x 5 = 40  
8 x 6 = 48  
8 x 7 = 56  
8 x 8 = 64  
8 x 9 = 72
```

# 반복문(중첩 for문)

## ■ 중첩된 반복문(Nested Loop)

	열1	열2	열3	열4	열5
행1					
행2					
행3					
행4					
행5					

가가가가가  
가가가가가  
가가가가가  
가가가가가  
가가가가가

\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*



# 반복문(중첩 for문)

## ■ 중첩된 반복문(Nested Loop)

```
// "가" 출력
int i, j;

for (i = 1; i <= 5; i++)
{
    for (j = 1; j <= 5; j++)
    {
        cout << "가";
    }
    cout << "\n";
}
```

```
// 별 찍기
for (i = 1; i <= 5; i++)
{
    for (j = 1; j <= 5; j++)
    {
        cout << "*";
    }
    cout << "\n";
}
```

# 반복문(중첩 for문)

## ■ 구구단 전체 출력하기

```
//전체 구구단
for (int i = 2; i < 10; i++) {
    cout << "[" << i << "]" << "단" << endl;
    for (int j = 1; j < 10; j++) {
        cout << i << " x " << j << " = " << (i * j) << endl;
    }
    cout << endl;
}
```

# 실습 문제 1 - 조건문

입장객 수에 따른 좌석 줄 수를 계산하는 프로그램을 작성하세요.

[파일이름: Seat.cpp, 변수명 - customer(입장객 수), column(열), row(줄)]

👉 실행 결과

```
입장객 수 입력 : 20  
좌석 열 수 입력 : 5  
4개의 줄이 필요합니다.
```

👉 실행 결과

```
입장객 수 입력 : 23  
좌석 열 수 입력 : 5  
5개의 줄이 필요합니다.
```