



C++_기초 문법

Visual Studio 2022

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;
```

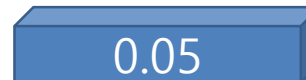
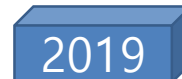
이름

ch

year

rate

공간(값)



자료형

char

int

double

자료형(data type)

● 자료형이란?

- 데이터를 저장하는 공간의 유형
- 사용할 데이터의 종류에 따라 메모리 공간을 적절하게 설정해 주는 것

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

※ 정수형 양수 표현범위를 2배로 늘릴 때는 자료형 앞에 **unsigned**를 붙일수 있는데. 이 경우 동일한 공간으로 0을 포함한 양수만을 표현하게 된다.

예) char: 128~127 → unsigned char는 0~255 범위 표현

변수의 선언과 사용

```
#include <iostream>

using namespace std; //이름 공간 설정

int main()
{
    //정수형 변수 n1, n2 선언 및 초기화
    int n1 = 4;
    int n2 = 5;

    //출력
    cout << "두 수의 합: " << n1 + n2 << endl;
    cout << "두 수의 차: " << n1 - n2 << endl;
    cout << "두 수의 곱: " << n1 * n2 << endl;
    cout << "두 수의 나누기: " << (double)n1 / n2 << endl;
    cout << "자료형의 크기: " << sizeof(n1) << "bytes" << endl;

    cout << "===== " << endl;
```

```
두 수의 합: 9
두 수의 차: -1
두 수의 곱: 20
두 수의 나누기: 0.8
자료형의 크기: 4bytes
=====
두 수의 곱: 2.31
두 수의 나누기: 0.524
자료형의 크기: 8bytes
=====
A
A

나
자료형의 크기: 1bytes
자료형의 크기: 3bytes
자료형의 크기: 6bytes
문자열의 길이: 3bytes
문자열의 길이: 6bytes
=====
banana
안녕하세요
자료형의 크기: 40bytes
자료형의 크기: 40bytes
문자열의 길이: 6bytes
문자열의 길이: 10bytes
```

변수의 선언과 사용

```
double n3 = 1.1;
double n4 = 2.1;

cout << "두 수의 곱: " << n3 * n4 << endl;

cout.precision(3); //소수 자리수 설정
cout << "두 수의 나누기: " << n3 / n4 << endl;
cout << "자료형의 크기: " << sizeof(n3) << "bytes" << endl;
cout << "===== " << endl;

//문자형 변수 ch 선언 및 초기화
char ch1 = 'A';
char ch2 = 65; //아스키 코드값
char ch3 = '\n';
char ch4[] = "나"; //배열 - 문자열 끝에 '\0' 포함

cout << ch1 << endl;
cout << ch2 << endl;
cout << ch3 << endl;
cout << ch4 << endl;
```

변수의 선언과 사용

```
cout << "자료형의 크기: " << sizeof(ch1) << "bytes" << endl;
cout << "자료형의 크기: " << sizeof(ch4) << "bytes" << endl;
cout << "===== " << endl;

//문자열(c++ 에서 추가된 자료형-string)
string s1 = "banana";
string s2 = "안녕하세요";

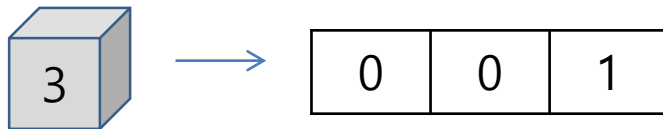
cout << s1 << endl;
cout << s2 << endl;
//string은 동적 메모리이므로 컴파일러에 따라 크기가 다르다.
cout << "자료형의 크기: " << sizeof(s1) << "bytes" << endl;
cout << "자료형의 크기: " << sizeof(s2) << "bytes" << endl;
cout << "문자열의 길이: " << size(s1) << "bytes" << endl;
cout << "문자열의 길이: " << size(s2) << "bytes" << endl;

return 0;
}
```

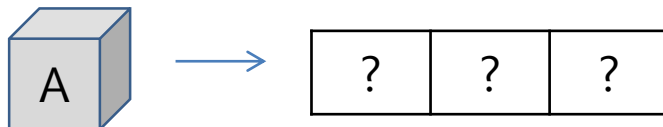
아스키 코드

- 아스키 코드(ASCII Code)가 왜 필요한가?

10진수를 2진수로 변환



문자를 2진수로 변환



저장하는 문자에 해당하는 숫자를 지정하고 메모리에 저장할때는 그 숫자를 비트 단위로 바꾸어 저장

- 아스키 코드(ASCII Code)

아스키 코드는 미국 [ANSI](#)에서 표준화한 정보교환용 7비트 부호체계이다.

000(0x00)부터 127(0x7F)까지 총 128개의 부호가 사용된다.

영문 키보드로 입력할 수 있는 모든 기호들이 할당되어 있는 부호 체계이다.

아스키 코드

- 아스키 코드(ASCII Code)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u

아스키 코드 vs 유니코드

❖ 유니 코드(Uni code)

전 세계의 모든 문자를 다루도록 설계된 표준 문자 전산 처리 방식. 주요 구성 요소는 ISO/IEC 10646 Universal Character Set과 UCS, UTF 등의 인코딩 방식, 문자 처리 알고리즘 등이다.

유니코드를 사용하면 **한글과 간체자, 아랍 문자** 등을 통일된 환경에서 깨뜨리지 않고 사용할 수 있다.

초창기에는 문자 코드는 ASCII의 로마자 위주 코드였고, 1바이트의 남은 공간에 각 나라가 자국 문자를 할당하였다.

이런 상황에서 다른 국가에 이메일을 보냈더니 글자가 깨졌던 것. 이에 따라 **2~3바이트의** 넉넉한 공간에 세상의 모든 문자를 할당한 결과물이다.
현재는 유니코드가 아스키 코드를 포함하며, 표준이 되었다.

형 변환(Type Conversion)

- **묵시적 형변환(자동)**

2) 연산 중에 자동 변환되는 경우

```
예) int num1=100;           // 정수
    double num2=3.14;       // 실수
    cout << iNum + fNum <<; // 정수 + 실수 -> 실수
```

- **명시적 형변환(강제)**

큰 자료형에서 작은 자료형으로 변환

변환 자료형을 명시해야 하고(괄호사용), 자료의 손실이 발생

```
예) double dNum = 12.34;
    int iNum = (int)dNum;
```

형 변환(Type Conversion)

//자동 형변환(묵시적 형변환)

```
int iNum = 20;
```

```
float fNum = 0.9f;
```

```
cout << iNum << endl;
```

```
cout << fNum << endl;
```

```
cout << iNum + fNum << endl;
```

//강제 형변환(명시적 형변환)

```
double dNum = 1.2;
```

```
int iNum2, iNum3;
```

```
iNum2 = (int)(dNum + fNum);
```

```
iNum3 = (int)dNum + (int)fNum;
```

```
cout << iNum2 << endl;
```

```
cout << iNum3 << endl;
```

//평균 계산하기

```
int count = 2;
```

```
int total = 185;
```

```
double avg;
```

```
avg = (double)total / count;
```

```
cout << "평균: " << avg << 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.
- 상수 이름은 대문자로 관례적으로 사용.
- 상수를 만드는 방법
 1. **#define 상수이름 상수값** – 매크로 상수로 정의(전처리, 컴파일되지 않음)

```
#define PI 3.141592
```

2. **const 자료형 상수이름 = 상수값**

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

//PI = 3.14 (변경할 수 없다.)

상수(constant)

- 상수(constant)

```
#include <iostream>
#define PI 3.14 //원주율 상수 선언

using namespace std;

int main()
{
    //상수 선언 및 초기화
    const int MIN_VAL = 1;
    const int MAX_VAL = 100;

    //MAX_VAL = 999; 수정 불가

    cout << MIN_VAL << " ~ " << MAX_VAL << endl;

    //원의 넓이 계산
    //const double PI = 3.14; //원주율
    int radius = 5; //반지름
    double circleArea; //원의 면적

    //면적 계산
    circleArea = radius * radius * PI;


    cout << "원의 면적: " << circleArea << endl;
```

대입 및 부호 연산자

■ 대입 연산자

- 변수에 값을 대입하는 연산자
- 연산의 결과를 변수에 대입
- 우선 순위가 가장 낮은 연산자
- 왼쪽 변수(lvalue)에 오른쪽 값(rvalue)를 대입

total = num1 + num2



```
char ch1 = 'A';  
char ch2 = 65;  
char ch3 = '\n';  
char ch4[] = "나";
```

산술 및 증감 연산자

■ 산술 연산자 / 증감 연산자

연산자	기 능	연산 예
+	두 항을 더합니다.	5+3
-	앞 항에서 뒤 항을 뺍니다.	5-3
*	두 항을 곱합니다.	5*3
/	앞 항에서 뒤 항을 나누어 몫을 구합니다.	5/3
%	앞 항에서 뒤 항을 나누어 나머지를 구합니다.	5%3

연산자	기 능	연산 예
++	항의 값에 1을 더합니다.	val = ++num; // num = num+1; val = num++;
--	항의 값에서 1을 뺍니다.	val = --num // num = num-1; val = num--;

산술 및 증감 연산자

```
//증감 연산자
```

```
int a = 99;
```

```
int b = 9;
```

```
cout << a++ << endl; //99, 후치 연산자
```

```
cout << a << endl; //100
```

```
cout << ++a << endl; //101, 전치 연산자
```

```
cout << a << endl; //101
```

```
cout << a-- << endl; //101
```

```
cout << a << endl; //100
```

```
cout << --a << endl; //99
```

```
cout << a << endl; //99
```

```
//몫과 나머지
```

```
int share, remainder;
```

```
share = a / b;
```

```
remainder = a % b;
```

```
cout << "몫: " << share << endl; //11
```

```
cout << "나머지: " << remainder << endl; //0
```


비교 및 논리 연산자

■ 관계(비교) 연산자 / 논리 연산자

연산자	기 능	연산 예
>	왼쪽 항이 크면 참을, 아니면 거짓을 반환합니다.	num > 3;
<	왼쪽 항이 작으면 참, 아니면 거짓을 반환합니다.	num < 3;
>=	왼쪽 항이 크거나 같으면 참, 아니면 거짓을 반환합니다.	num >= 3;
<=	왼쪽 항이 작거나 같으면 참, 아니면 거짓을 반환합니다.	num <= 3;
==	두 개의 항 값이 같으면 참, 아니면 거짓을 반환합니다.	num == 3;
!=	두 개의 항 값이 다르면 참, 아니면 거짓을 반환합니다.	num != 3

조건 연산자

■ 논리 연산자 / 조건 연산자

연산자	기 능	연산 예
&& (논리 곱)	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	(7<3) && (5>2)
 (논리 합)	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	(7>3) (5<2)
! (부정)	참은 거짓으로, 거짓은 참으로 바꿉니다.	!(7>3)

연산자	기 능	연산 예
조건식?결과1:결과2;	조건식이 참이면 결과1, 조건식이 거짓이면 결과2가 선택됩니다.	int num = (5>3)?10:20;

비교 및 논리 연산자

//비교 연산자

```
cout << "a == b: " << (a == b) << endl;  
cout << "a != b: " << (a != b) << endl;  
cout << "a >= b: " << (a >= b) << endl;  
cout << "a <= b: " << (a <= b) << endl;
```

//논리 연산자

```
cout << "(a == b) && (a > b): " << ((a == b) && (a > b)) << endl;  
cout << "(a == b) || (a > b): " << ((a == b) || (a > b)) << endl;  
cout << "!(a == b): " << (!(a == b)) << endl;
```

//조건 연산자

```
cout << "((a > b) ? 'T' : 'F'):" << ((a > b) ? 'T' : 'F') << endl;
```

복합대입자

■ 복합대입 연산자

연산자	기 능	연산 예
+=	두 항의 값을 더해서 왼쪽 항에 대입합니다.	num += 2; num=num+2
-=	왼쪽 항에서 오른쪽 항을 빼서 그 값을 왼쪽 항에 대입합니다.	num -= 2; num=num-2
*=	두 항의 값을 곱해서 왼쪽 항에 대입합니다.	num *= 2; num=num*2
/=	왼쪽 항을 오른쪽 항으로 나누어 그 몫을 왼쪽 항에 대입합니다.	num /= 2; num=num
%=	왼쪽 항을 오른쪽 항으로 나누어 그 나머지를 왼쪽 항에 대입합니다.	num %= 2; num=num%2

복합대입자

```
//복합 대입 연산자
int c = 10;

c += 1; //c = c + 1
cout << "c = " << c << endl;

c -= 1; //c = c - 1
cout << "c = " << c << endl;

c *= 2; //c = c * 1
cout << "c = " << c << endl;

c /= 2; //c = b / 2
cout << "c = " << c << endl;
```

비트 연산자

■ 비트 연산자

연산자	기 능	연산 예
&	$a \& b$	1 & 1 -> 1을 반환, 그 외는 0
	$a b$	0 0 -> 0을 반환, 그 외는 1
~	$\sim a$	a가 1이면 0, 0이면 1을 반환
<<	$a < < 2$	a를 2비트 만큼 왼쪽으로 이동
>>	$a > > 3$	a를 2비트 만큼 오른쪽으로 이동

비트 연산자

■ 비트 논리연산자

```
int num1 = 5;  
int num2 = 10;  
int result = num1 & num2;
```



```
num1 : 0 0 0 0 0 1 0 1  
& num2 : 0 0 0 0 1 0 1 0  
-----  
result : 0 0 0 0 0 0 0 0
```

■ 비트 이동 연산자

```
int num = 5;  
num << 2;
```



```
num      : 0 0 0 0 0 1 0 1  
num << 2 : 0 0 0 1 0 1 0 0
```

비트 연산자

```
//비트 논리 연산자
int num1 = 5;    //00000101
int num2 = 10;   //00001010
int result1, result2;

result1 = num1 & num2; //00000000
result2 = num1 | num2;  //00001111

cout << "result1 = " << result1 << endl;
cout << "result2 = " << result2 << endl;

//비트 이동 연산자
int num3 = 4;    //00000100

cout << "(num3 << 1) = " << (num3 << 1) << endl; //00001000
cout << "(num3 >> 1) = " << (num3 >> 1) << endl;  //00000010
```


데이터 입력 받기

● 데이터 입력

cin >> 변수이름

```
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;
```

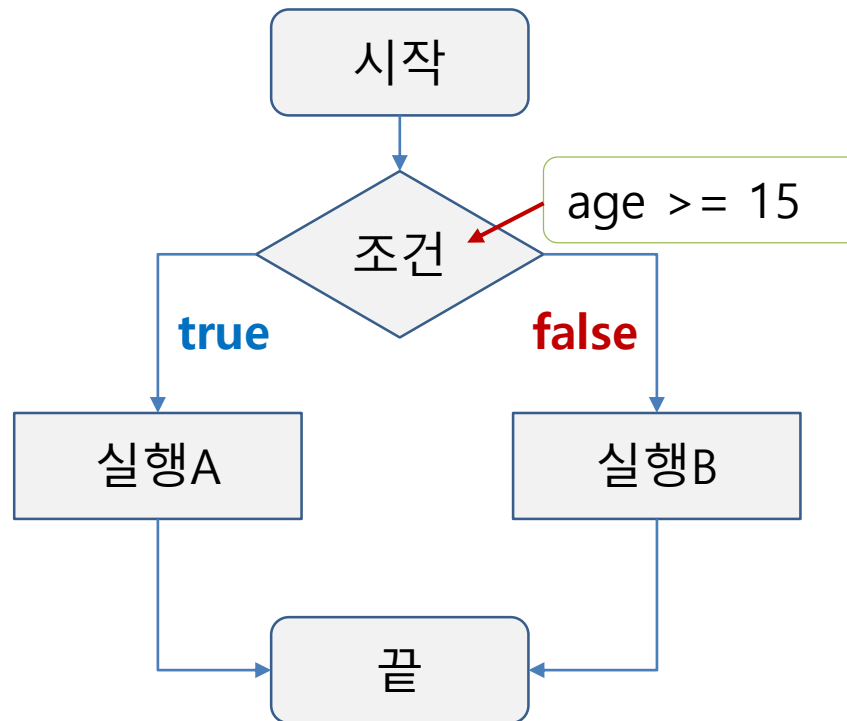
```
cout << "\n=== 학생의 정보 출력 ===\n";
cout << "학번: " << number << endl;
cout << "이름: " << name << endl;
cout << "영어점수: " << eng << endl;
cout << "수학점수: " << math << endl;

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

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

```
/*  
    나이가 15세 이상이면 "관람가", 아니면 "관람불가"  
*/  
int age = 16;  
  
//if문  
/*if (age >= 15)  
{  
    cout << "관람가입니다." << endl;  
}*/  
  
//if ~ else문  
if (age >= 15)  
{  
    cout << "관람가입니다." << endl;  
}  
else  
{  
    cout << "관람불가입니다." << endl;  
}  
cout << "나이는 " << age << "세 이군요!" << endl;
```

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";
}
```

```
//문자 비교
string str1 = "orange";
string str2 = "apple";

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

if ~ else if ~ else문

▪ if - else if - else 문

```
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

```
/*  
    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;  
}
```

조건문(SWITCH - CASE)

▪ case문 동시에 사용하기

```
/*  
    12달 중에 31일, 30일, 28인 달 선택  
*/  
int month = 10;  
int day;  
  
switch (month)  
{  
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:  
        day = 31;  
        break;  
    case 2:  
        day = 28;  
        break;  
    case 4: case 6: case 9: case 11:  
        day = 30;  
        break;  
    default :  
        day = 0;  
        break;  
}  
  
cout << month << "월은 " << day << "일까지 있습니다.\n";
```


반복문(while문)

● 반복문

- 주어진 조건이 만족할 때까지 수행문을 반복적으로 수행함

- while, for 문이 대표적임

- **while 문**

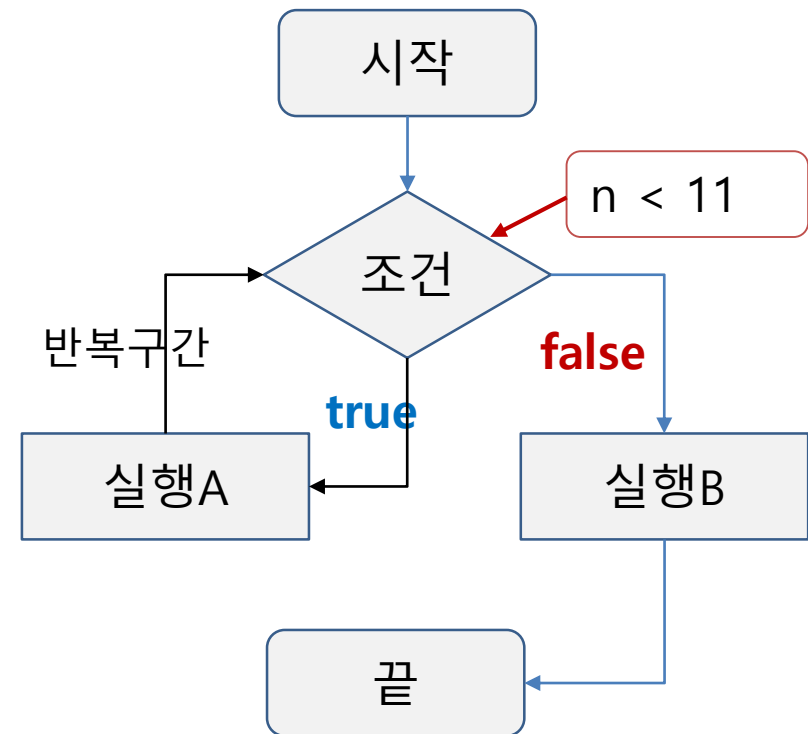
조건식이 참인 동안 반복 수행

```
while(조건식){
```

```
    수행문1;
```

```
    ...
```

```
}
```



반복문(while문)

▪ while문 예제

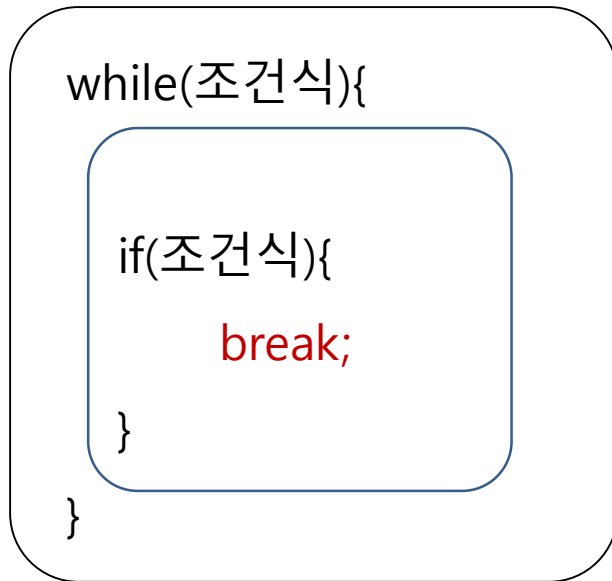
```
/*  
1 ~ 10 까지 출력하기  
*/  
  
int n = 0;  
  
while (n < 10)  
{  
    n++;  
    cout << n << " ";  
}
```

```
/*  
1 ~ 10 까지의 합계 구하기  
*/  
int n = 1;  
int sum = 0;  
  
while (n <= 10)  
{  
    sum += n;  
    cout << "n=" << n << ", sum=" << sum << endl;  
    n++;  
}  
  
cout << "합계 : " << sum << endl;
```

기타 제어 -break 문

▪ break 문

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



while(1){ } – 무한 반복문

1이면 참, 1이 아니면 거짓

반복문(while문)

▪ break문 예제

```
/*
 1 ~ 10 까지 출력하기
 break문 사용
*/
int n = 0;

while (1)
{
    n++;
    if (n > 10)
        break; //반복문 탈출
    cout << n << " ";
}
```

```
/*
 1 ~ 10 까지의 합계 구하기
*/
int n = 1;
int sum = 0;

while (1)
{
    if (n > 10)
        break; //반복문 탈출
    sum += n;
    //cout << "n=" << n << ", sum=" << sum << endl;
    n++;
}

cout << "합계 : " << sum << endl;
```

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(while문)

▪ 커피 자판기 프로그램

- 동전 500원을 넣으면 커피가 나온다.
- 500원을 초과하면 거스름돈이 나오고 커피가 나온다.
- 500원보다 작으면 커피가 나오지 않음
- 커피는 총 5개이고 모두 소진되면 판매를 중지한다.

```
int money;
int coffee = 5;
while (1)
{
    cout << "돈을 넣어주세요: ";
    cin >> money;
    if (money == 500)
    {
        cout << "커피가 나옵니다.\n";
        coffee -= 1;
    }
}
```

반복문(while문)

▪ 커피 자판기 프로그램

```
    else if (money > 500)
    {
        cout << "커피가 나오고, 거스름돈 " << (money - 500) << "원을 돌려줍니다.\n";
        coffee -= 1;
    }
    else
    {
        cout << "돈을 돌려주고 커피는 나오지 않습니다.\n";
    }

    if (coffee == 0)
    {
        cout << "커피가 모두 소진되어 판매를 중지합니다.\n";
        break;
    }
}

return 0;
}
```


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

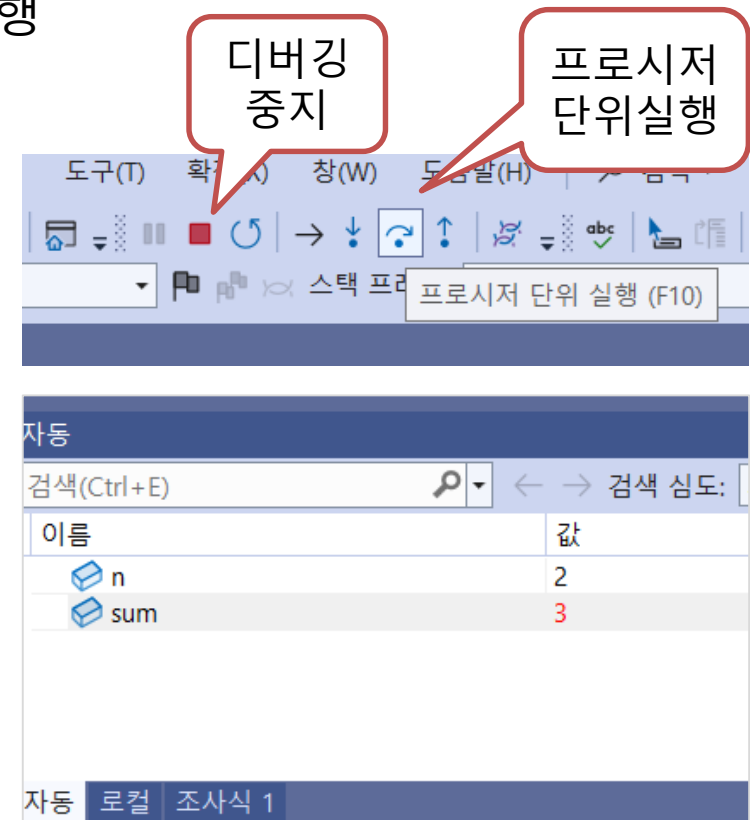
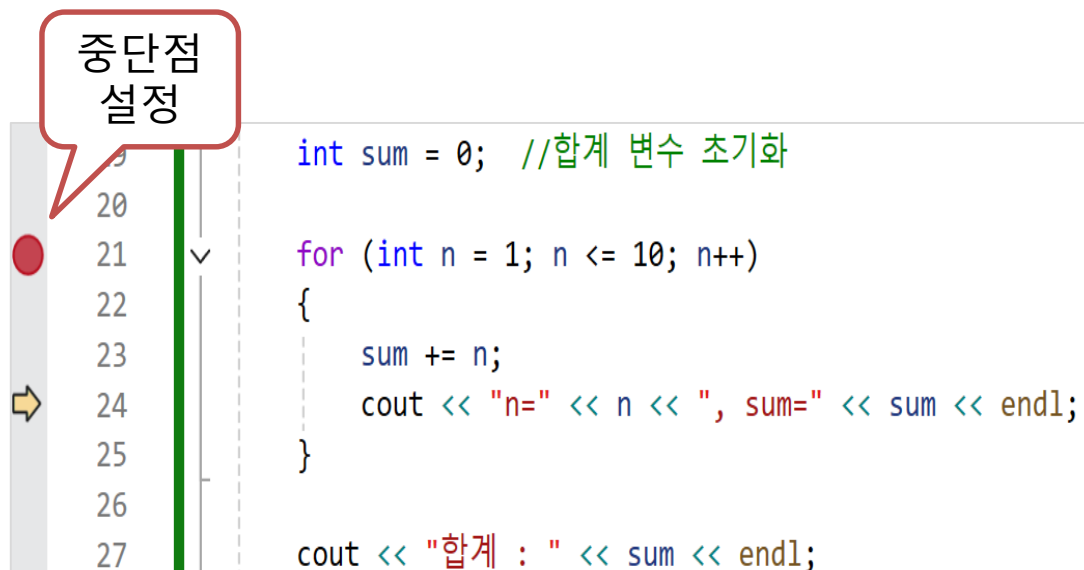
for (int n = 1; n <= 10; n++)
{
    sum += n;
    cout << "n=" << n << ", sum=" << sum << endl;
}

cout << "합계 : " << sum << endl;
```

디버깅(Debugging)

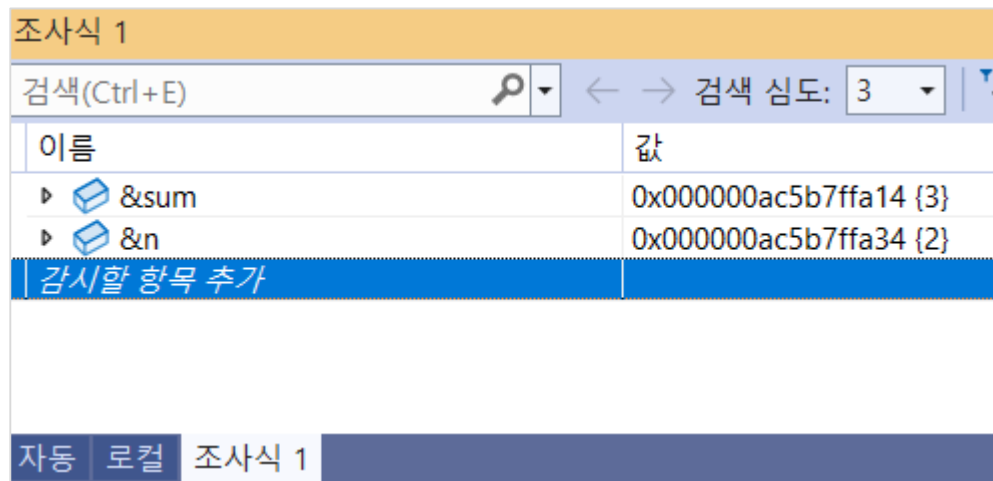
◆ 디버깅 작업

- 중단점 설정(F9) : 디버그 > 중단점 설정
- 실행(F10) : 디버그 > 프로시저 단위 실행



디버깅(Debugging)

◆ 디버깅 작업



조사식1 > 검사할 항목 추가
&sum 입력 : 메모리 주소 보기

구구단 출력

■ 구구단 출력 프로그램

```
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
```

```
/*
   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";
}
```

반복문(중첩 for문)

■ 중첩된 반복문(Nested Loop)

	열1	열2	열3	열4	열5
행1					
행2					
행3					
행4					
행5					

[illegible]

```

*****
*****
*****
*****
*****

```

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

```
/*  
    구구단 2 ~ 9단까지 출력하기  
*/  
for (i = 2; i <= 9; i++)  
{  
    for (j = 1; j <= 9; j++)  
    {  
        cout << i << " x " << j << " = " << (i * j) << "\n";  
    }  
    cout << "\n";  
}
```


배열(Array)

❖ 배열은 왜 사용해야 할까?

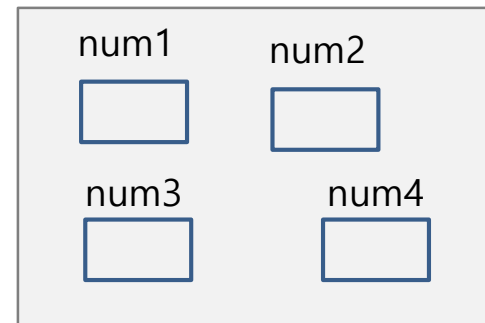
- 정수 10개를 이용한 프로그램을 할 때 10개의 정수 타입의 변수를 선언

`int num1, int num2, int num3... num10;`

정보가 흩어진 채 저장되므로

비효율적이고 관리하기 어렵다.

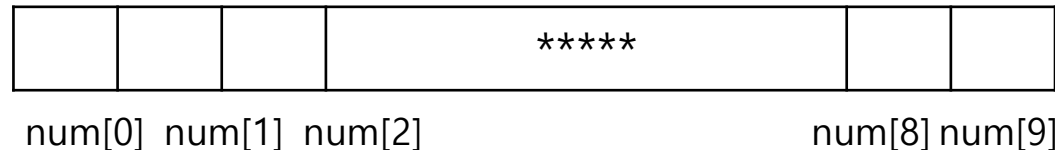
메모리



- 배열은 동일한 자료형의 변수를 한꺼번에 순차적으로 관리할 수 있다.

`int num[10];`

배열 이름
1개



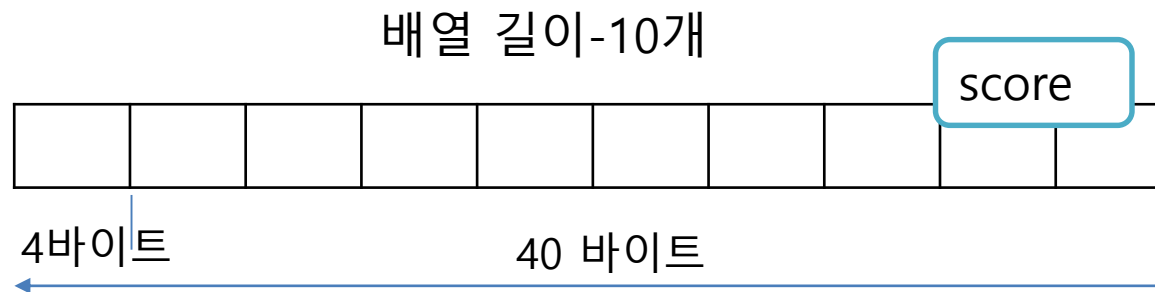
배열(Array)

- 배열이란?

여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.
배열 변수는 []안에 설정한 값만큼 메모리를 할당하여 저장한다.

- 배열 변수의 선언과 사용

```
int score[10];
```



배열(Array)

- 정수형 배열 관리

```
/*
| 정수형 배열 선언, 저장 및 출력
*/
int arr[3];

//저장
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;

/*
| 정수형 배열 선언 및 초기화
*/
//int arr[3] = { 1, 2 };
int arr[3] = { 1, 2, 3 };

//배열의 크기 - size()
cout << size(arr) << endl;
```

```
//출력
for (int i = 0; i < size(arr); i++)
{
    cout << arr[i] << " ";
}
cout << "\n===== \n";

//요소 수정 - 1번 인덱스 수정
arr[1] = 5;

for (int i = 0; i < size(arr); i++)
    cout << arr[i] << " ";
```

배열(Array)

■ 정수형 배열의 연산

```
/*
    성적의 합계와 평균, 최대값, 최소값
*/
int score[4] = { 70, 90, 80, 65 };
int total = 0;    //합계
double avg;       //평균
int max, min;     //최대값, 최소값

//배열의 요소 개수
cout << size(score) << endl;

/*for (int i = 0; i < size(score); i++)
{
    cout << score[i] << " ";
}*/

// 합계 계산
for (int i = 0; i < size(score); i++)
{
    total += score[i];
}
cout << "합계 : " << total << endl;
```

```
// 평균 계산 : 합계 / 개수
avg = total / (double)size(score);
cout << "평균 : " << avg << endl;

//최대값
max = score[0]; //첫째 요소 최대값으로 설정
for (int i = 1; i < size(score); i++)
{
    if (score[i] > max)
    {
        max = score[i];
    }
}
cout << "최대값 : " << max << endl;
```

배열(Array)

- 문자형 배열 관리

```
/*  
| 문자형 배열 선언, 저장 및 출력  
*/  
char c1, c2, c3; //문자형 변수 선언  
  
c1 = 'B';  
c2 = c1 + 1;  
c3 = c1 - 1;  
  
cout << c1 << " " << c2 << " " << c3 << endl;  
cout << "\n===== \n";
```

```
B C A  
=====
```

A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79

배열(Array)

- 문자형 배열 관리

```
//알파벳 대문자를 저장할 26개 배열 생성
char alphabets[26];
char ch = 'A';

//저장
for (int i = 0; i < 26; i++)
{
    alphabets[i] = ch;
    ch++; //ch = ch + 1;
}

//출력
for (int i = 0; i < 26; i++)
{
    cout << alphabets[i] << " " << (int)alphabets[i] << endl;
}
```

배열(Array)

- 문자형 배열 관리

```
//문자열 배열 관리
string cart = "라면"; //문자열 변수
cout << cart << endl;

string carts[] = {"라면", "빵", "화장지", "생수"};

//배열의 크기
cout << size(carts) << endl;

//2번 요소 조회
cout << carts[2] << endl;

//요소 수정
carts[1] = "쌀";

//전체 요소 출력
for (int i = 0; i < size(carts); i++)
{
    cout << carts[i] << " ";
}
```

2차원 배열

배열의 확장 : 2차원 배열

이정후의 1반 학생들의 키를 배열에 저장

```
int class1[5]
```

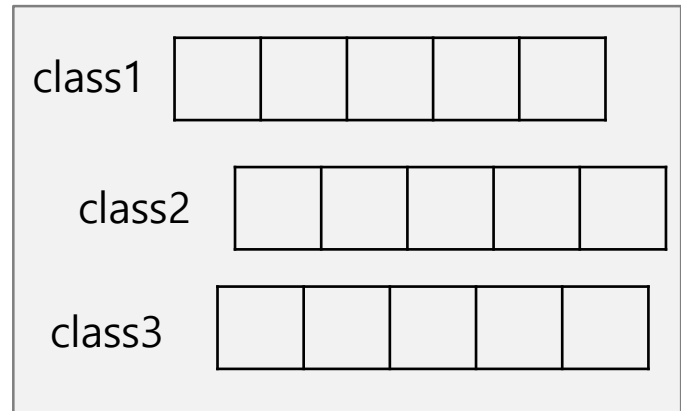
2반과 3반 학생들의 키를 배열에 저장

```
int class1[5]
```

```
int class2[5]
```

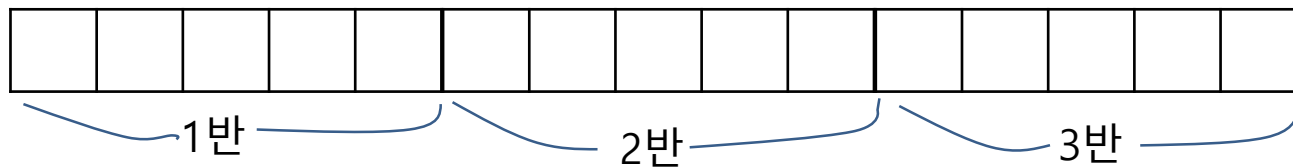
```
int class3[5]
```

메모리



2차원 배열을 사용한 경우

```
int class[3][5]
```



2차원 배열

■ 배열의 확장 : 2차원 배열

1. 지도, 게임 등 평면이나 공간을 구현할 때 많이 사용됨.
2. 이차원 배열의 선언과 구조

```
int arr[2][3];
```

3. 선언과 초기화

```
int arr[2][3] = {  
    {1, 2, 3},  
    {4, 5, 6}  
};
```



arr[0][0]	arr[0][1]	arr[0][2]

arr[1][0] arr[1][1] arr[1][2]

arr[0][0]	arr[0][1]	arr[0][2]
1	2	3
4	5	6

arr[1][0] arr[1][1] arr[1][2]

이차원 배열

■ 정수형 배열 생성 및 출력

```
/*  
    2차원 배열을 생성하고 1 ~ 6까지 저장하기  
*/  
  
int a[2][3];  
int i, j, k = 0;  
  
//저장  
for (i = 0; i < 2; i++)  
{  
    for (j = 0; j < 3; j++)  
    {  
        a[i][j] = k + 1;  
        k++;  
    }  
}
```

```
//출력  
for (i = 0; i < 2; i++)  
{  
    for (j = 0; j < 3; j++)  
    {  
        cout << a[i][j] << " ";  
    }  
}
```

이차원 배열

- 이차원 배열 – 정수형 배열

학생 3명의 2과목 점수

Kim, Lee, Park

이름	수학	영어
Kim	75	80
Lee	85	95
Park	90	100

이차원 배열

- 정수형 배열 생성 및 연산

```
// 정수형 2차원 배열 선언 및 초기화
int a[3][2] = {
    {75, 80},
    {85, 95},
    {90, 100}
};

//특정 요소 조회
cout << "a[0][0]=" << a[0][0] << endl;
cout << "a[1][1]=" << a[1][1] << endl;

//전체 조회
for (int x = 0; x < 3; x++)
{
    for (int y = 0; y < 2; y++)
    {
        cout << "a[" << x << "][" << y << "]=" << a[x][y] << ' ';
    }
    cout << '\n';
}
```

이차원 배열

- 이차원 배열 – 정수형 배열 생성 및 연산

```
//요소의 수 및 합계
int count = 0;
int total = 0;
for (int x = 0; x < 3; x++)
{
    for (int y = 0; y < 2; y++)
    {
        count++;
        total += a[x][y];
    }
}
cout << "배열의 요소 수: " << count << endl;
cout << "배열의 요소의 총합: " << total << 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.
```

실습 문제 2 – 배열의 연산

배열 길이가 5인 정수 배열을 선언하고, 1~10중 홀수만을 배열에 저장한 후 그 합과 평균을 계산하시오. [평균은 실수형 자료형을 사용함]

👉 실행 결과

```
총합 : 25  
평균 : 5
```