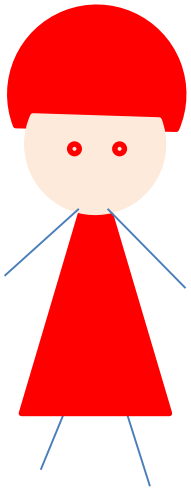
자료형의 종류

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

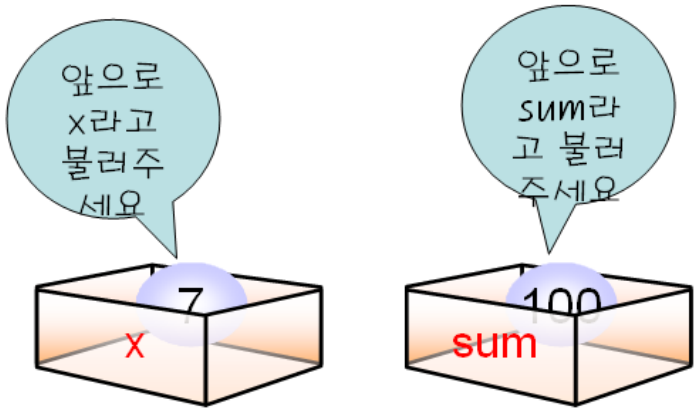
변수의 이름짓기

- 식별자(identifier): 식별할 수 있게 해주는 이름
  - 변수 이름
  - 함수 이름

김도원



이주호



식별자를 만드는 규칙

- 알파벳 문자와 숫자, 밑줄 문자 _로 구성
- 첫 번째 문자는 반드시 알파벳 또는 밑줄 문자 _
- 대문자와 소문자를 구별
- C 언어의 키워드와 똑같은 이름은 허용되지 않는다.

(Q) 다음은 유효한 식별자인가?

sum O
Dollor# X // #기호
_count O
choihee0 O
double X // 키워드
n_pictures O
2hansoo X // 숫자로 시작

키워드

키워드(keyword):

C++언어에서 고유한 의미를 가지고 있는 특별한 단어

예약어(reserved words) 라고도 한다.

C언어의 키워드

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

C++언어의 키워드

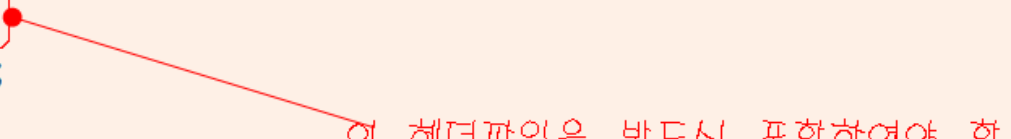
asm	false	protected	try
bool	friend	public	typeid
catch	inline	reinterpret_cast	typename
class	mutable	static_cast	using
const_cast	namespace	template	virtual
delete	new	this	wchar_t
dynamic_cast	operator	throw	
explicit	private	true	

문자열 타입

- **string** 타입을 제공한다. (문자열의 비교는 **==** 연산자로 가능)

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

int main(void)
{
   string s1 = "Good";
   string s2 = "Morning";
   string s3 = s1 + " " + s2 + "!\n";
   cout << s3;
   return 0;
}
```



실행결과

```
Good Morning!
```

cin, cin.get(), cin.getline()

1. cin :
- 문자와 문자열 모두 입력 받을 수 있다.
- 공백, 개행 무시
- space, tab, enter로 구분

```
1  #include<iostream>
2  using namespace std;
3  int main(void)
4  {
5      char c;
6      char s[10];
7
8      cout << "문자 입력: ";
9      cin >> c >> s;
10
11     cout << c << " " << s << endl;
12     return 0;
13 }
```

```
문자 입력: s sunrin
s sunrin
계속하려면 아무 키나 누르십시오 . . .
```
space bar

```
문자 입력: s    sunrin
s sunrin
계속하려면 아무 키나 누르십시오 . . .
```
tab

```
문자 입력: s
sunrin
s sunrin
계속하려면 아무 키나 누르십시오 . . .
```
enter

cin, cin.get(), cin.getline()

2. cin.get() :

- **문자 하나**만 입력 받을 수 있다.
- 공백, 개행 도 입력으로 포함

```
1  #include<iostream>
2  using namespace std;
3  int main(void)
4  {
5      char ch1, ch2;
6
7      ch1 = cin.get();
8      ch2 = cin.get();
9
10     cout << ch1 << " " << ch2 << endl;
11     return 0;
12 }
```

```
ab
a b
계속하려면 아무 키나 누르십시오 . . .
```

a + b

```
a b
a
계속하려면 아무 키나 누르십시오 . . .
```

a + spacebar + b

```
a
a

계속하려면 아무 키나 누르십시오 . . .
```

a + enter+ b

cin, cin.get(), cin.getline()

3. cin.getline() :

- 종결문자를 NULL로 바꿈, 종결문자 생략시 Enter로
- 공백, 개행도 입력으로 받음
- 문자열만 입력받음

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char a[10];
6      cin.getline(a, 10);
7      cout << a << endl;
8
9      return 0;
10 }
```

```
abcde
abcde
계속하려면 아무 키나 누르십시오 . . .
```

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char a[10];
6      cin.getline(a, 10, 'c');
7      cout << a << endl;
8
9      return 0;
10 }
```

```
abcde
ab
계속하려면 아무 키나 누르십시오 . . .
```

#실습(1)

<Enter>키가 입력될 때까지 문자들을 읽고, 입력된 문자 ‘s’의 개수를 화면에 출력하시오.

출력결과 :

```
문장을 입력하세요: sunrin highschool
s의 개수: 2
계속하려면 아무 키나 누르십시오 . . .
```

#실습1정답

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char str[100];
6      int count = 0;
7
8      cout << "문장을 입력하세요: ";
9      cin.getline(str, 100, '₩n');
10
11     for (int i = 0; str[i] != '₩0'; i++){
12         if (str[i] == 's') count++;
13     }
14
15     cout << "s의 개수: "<< count << endl;
16     return 0;
17 }
```

#실습

기호 상수를 만드는 방법

```
1  #include<iostream>
2
3  #define LEN 5
4  using namespace std;
5  int main(void)
6  {
7      const double PI = 3.141592;
8
9      cout << "반지름이 "<< LEN << "인 원의 둘레 : "<< LEN*2*PI << endl;
10     cout << "반지름이 " << LEN << "인 원의 넓이 : " << LEN * LEN * PI << endl;
11
12     return 0;
13 }
```

```
반지름이 5인 원의 둘레 : 30
반지름이 5인 원의 넓이 : 75
계속하려면 아무 키나 누르십시오 . . .
```

Q. 숫자값을 직접 사용하는 것보다 기호 상수를 사용하는 것의 이점은?

기능에 따른 연산자의 분류

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

산술 연산자

- 덧셈, 뺄셈, 곱셈, 나눗셈 등의 사칙 연산을 수행하는 연산자

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

관계 연산자

- 두개의 피연산자를 비교하는 연산자
- 결과값은 참(1) 아니면 거짓(0)

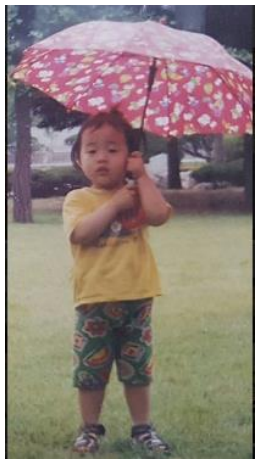
연산자 기호	의미	사용예
==	x와 y가 같은가?	x == y
!=	x와 y가 다른가?	x != y
>	x가 y보다 큰가?	x > y
<	x가 y보다 작은가?	x < y
>=	x가 y보다 크거나 같은가?	x >= y
<=	x가 y보다 작거나 같은가?	x <= y

If(이수근 < 서장훈)
TRUE!!

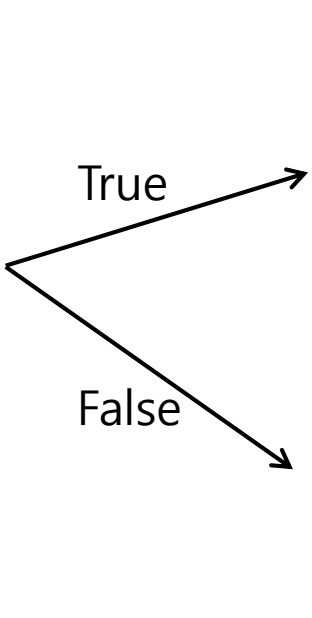
논리 연산자

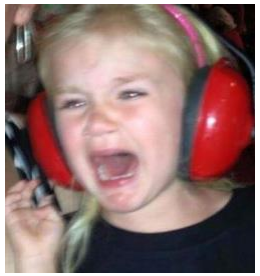
- 여러 개의 조건을 조합하여 참과 거짓을 따지는 연산자
- 결과값은 참(1) 아니면 거짓(0)

연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
||	x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참



&&





#실습(2)

윤년 판단하기

1. 4로 나누어 떨어지는 연도는 윤년으로 한다.(2004, 2008..)

2. 100으로 나누어 떨어지는 연도는 평년으로 한다.(2100,2200..)

3. 400으로 나누어 떨어지는 연도는 윤년이다.(1600, 2000..)

힌트!!

```
(((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
```

#실습2정답

```
#include<iostream>
using namespace std;
int main(void)
{
    int year, i=0;

    while (i < 3){

        cout << "연도를 입력하세요: ";
        cin >> year;

        if (((year % 4 == 0) && (year % 100)) || (year % 400 == 0))
            cout << "윤년입니다." << endl;
        else
            cout << "윤년이 아닙니다." << endl;

        i++;
    }
    return 0;
}
```

```
연도를 입력하세요: 2016
윤년입니다.
연도를 입력하세요: 2017
윤년이 아닙니다.
연도를 입력하세요: 2012
윤년입니다.
계속하려면 아무 키나 누르십시오 . . .
```

우선 순위

Symbol1	연산 형식	결합성
[ ] ( ) . –>후위 ++ 및 후위 —	식	왼쪽에서 오른쪽
전위 ++ 및 전위 — sizeof & * + – ~ !	단항	오른쪽에서 왼쪽
형식 캐스팅	단항	오른쪽에서 왼쪽
* / %	곱하기	왼쪽에서 오른쪽
+ –	더하기	왼쪽에서 오른쪽
<< >>	비트 시프트	왼쪽에서 오른쪽
< > <= >=	관계	왼쪽에서 오른쪽
== !=	같음	왼쪽에서 오른쪽
&	비트 AND	왼쪽에서 오른쪽
^	비트 제외 OR	왼쪽에서 오른쪽
|	비트 포함 OR	왼쪽에서 오른쪽
&&	논리 AND	왼쪽에서 오른쪽
||	논리 OR	왼쪽에서 오른쪽
? :	조건식	오른쪽에서 왼쪽
= *= /= %= += –= <<= >>=&= ^= |=	단순 및 복합 할당2	오른쪽에서 왼쪽
,	순차적 계산	왼쪽에서 오른쪽

우선 순위의 일반적인 지침

① 콤마 < 대입 < 논리 < 관계 < 산술 < 단항

② 괄호 연산자는 가장 우선순위가 높다.

③ 모든 단항 연산자들은 이항 연산자들보다 우선순위가 높다.

④ 콤마 연산자를 제외하고는 대입 연산자가 가장 우선순위가 낮다.

⑤ 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용

```
( x <= 10 ) && ( y >= 20 )
```

⑥ 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.

```
x + 2 == y + 3
```

결합 규칙

- 만약 **같은 우선순위**를 가지는 연산자들이 여러 개가 있으면 어떤 것을 먼저 수행하여야 하는가의 규칙

```
result = x * y % z;
         ① ②
```

#퀴즈

```
#include<iostream>
using namespace std;
int main(void)
{
    int x = 0, y = 0, z = 0;

    1 cout << (2 > 3 || 6 > 7) << endl;
    2 cout << (2 || 3 && 3 > 2) << endl;
    3 cout << (x = y = z = 1) << endl;
    4 cout << (-++x + y--) << endl;

    return 0;
}
```

0

1

1

-1