



C프로그래밍

Lecture 03. 변수와 연산자

동덕여자대학교
데이터사이언스 전공
권 범

목차

- ❖ 01. 연산을 위한 연산자와 값의 저장을 위한 변수
- ❖ 02. C언어의 다양한 연산자 소개
- ❖ 03. 키보드로부터 데이터 입력과 C언어의 키워드
- ❖ 04. 연습 문제

01. 연산을 위한 연산자와 값의 저장을 위한 변수

02. C언어의 다양한 연산자 소개

03. 키보드로부터의 데이터 입력과 C언어의 키워드

04. 연습 문제

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 덧셈 프로그램 구현에 필요한 연산자 (1/2)

```
1  /* simpleadd1.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      3 + 4;      // 3과 4의 합을 명령함
7      return 0;
8  }
```

실행 결과로는 아무것도 나타나지 않습니다.

- ✓ 6번 줄에서 + 기호를 사용하고 있습니다.
- ✓ 컴파일 및 실행 시 문제가 발생하지 않습니다.
- ✓ 따라서 + 기호는 컴퓨터가 인식 가능한 (지원하는) 기호입니다.

- ✓ 실제로 +는 덧셈의 의미를 갖습니다.
- ✓ 따라서 실행으로 인해 3과 4의 합이 진행됩니다.
- ✓ +와 같은 기호를 가리켜 연산자라고 합니다.

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 덧셈 프로그램 구현에 필요한 연산자 (2/2)

```
1  /* simpleadd1.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      3 + 4;      // 3과 4의 합을 명령함
7      return 0;
8  }
```

연산의 결과는?

- ✓ + 연산만 요구를 하였지 그 결과를 출력하기 위한 어떠한 코드도 삽입되지 않았습니다.
- ✓ 따라서 아무런 출력도 이뤄지지 않습니다.
- ✓ 연산의 결과를 저장해 두어야 원하는 바를 추가로 진행할 수 있습니다.
- ✓ 연산 결과 또는 값의 저장을 위해서 C언어에서는 변수(Variable)라는 것을 제공합니다.

실행 결과로는 아무것도 나타나지 않습니다.

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 변수를 이용한 데이터의 저장

“ 변수 ” (Variable)

값을 저장할 수 있는 메모리 공간에 붙여진 이름
혹은 메모리 공간 자체를 의미합니다.

변수라는 것을 선언하면 메모리 공간이 할당되고,
할당된 메모리 공간에 이름이 붙습니다.

변수의 이름

- ✓ 변수의 이름을 통해서 할당된 메모리 공간에 접근이 가능합니다.
- ✓ 값을 저장할 수도 있고, 저장된 값을 참조할 수도 있습니다.

- vary
(동사) 다르다 (달라지다)
- variable
(형용사) 변동이 심한
(명사) 변수

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 변수의 선언 방법 #1

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6      num = 20;
7      printf("%d\n", num);
8      return 0;
9  }
```

int num;

- ✓ int 정수의 저장을 위한 메모리 공간의 할당
- ✓ num 할당된 메모리 공간의 이름은 num

num = 20;

- ✓ 변수 num에 접근하여 20을 대입(저장)

printf("%d\n", num);

- ✓ num에 저장된 값을 참조(출력)

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 변수의 선언 방법 #2

```
1  /* variable_declare.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int a, b;
7      int c = 30, d = 40;
8
9      a = 10;
10     b = 20;
11
12     printf("a: %d, b: %d\n", a, b);
13     printf("c: %d, d: %d\n", c, d);
14     return 0;
15 }
```

6번 줄

- ✓ 선언하고자 하는 변수의 종류(자료형)가 같다면, 한 줄에 걸쳐서 동시에 선언하는 것이 가능합니다.
- ✓ 값이 대입되기 전까지 쓰레기 값(의미 없는 값)이 채워집니다.

7번 줄

- ✓ 변수를 선언하고 동시에 데이터를 저장하는 것도 가능합니다.
- ✓ 이를 두고 "선언과 동시에 초기화 한다"라고 합니다.

9, 10번 줄

- ✓ 변수 a와 b의 값을 초기화합니다.

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 변수의 이름을 지을 때 적용되는 규칙

규칙

- ① 변수의 이름은 알파벳, 숫자, 언더바(_)로 구성됩니다.
- ② C언어는 대소문자를 구분합니다.
따라서 변수 Num과 변수 num은 서로 다른 변수입니다.
- ③ 변수의 이름은 숫자로 시작할 수 없습니다.
- ④ 키워드(Keywords)도 변수의 이름으로 사용할 수 없습니다.
(키워드에 대해서는 잠시 후, 뒤쪽 슬라이드에서 설명합니다.)
- ⑤ 이름에는 공백이 포함될 수 없습니다.

**변수의 이름을 정할 때에는
의미 있는 이름을 짓는 것이 중요합니다!**

적절치 않은 변수의 이름	적절치 않은 이유
<code>int 7ThVal;</code>	변수의 이름이 숫자로 시작합니다.
<code>int phone#;</code>	변수의 이름에 #과 같은 특수문자는 올 수 없습니다.
<code>int your name;</code>	변수의 이름에는 공백이 포함될 수 없습니다.

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 변수의 자료형(Data Type)

- 변수의 두 가지 분류

정수형 변수

- ✓ 정수 값의 저장을 목적으로 선언된 변수
- ✓ 정수형 변수는 char형, short형, int형, long형 변수로 나뉩니다.

실수형 변수

- ✓ 실수 값의 저장을 목적으로 선언된 변수
- ✓ 실수형 변수는 float형 변수와 double형 변수로 나뉩니다.

정수형 변수와 실수형 변수가
나뉘는 이유는 무엇일까요?

정수를 저장하는 방식과
실수를 저장하는 방식이 다르기 때문입니다.

```
#include <stdio.h>

int main(void)
{
    int num1 = 24;
    // num1은 정수형 변수 중 int형 변수
    double num2 = 3.14;
    // num2는 실수형 변수 중 double형 변수

    ...

    return 0;
}
```

01. 연산을 위한 연산자와 값의 저장을 위한 변수

❖ 덧셈 프로그램의 완성

```
1  /* simpleadd2.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 3;
7      int num2 = 4;
8      int result = num1 + num2;
9
10     printf("덧셈 결과: %d\n", result);
11     printf("%d + %d = %d\n", num1, num2, result);
12     printf("%d와(과) %d의 합은 %d입니다.\n", num1, num2, result);
13
14     return 0;
15 }
```

덧셈 결과: 7

3 + 4 = 7

3와(과) 4의 합은 7입니다.

변수를 선언하여 덧셈의 결과를 저장했기 때문에
덧셈 결과를 다양한 형태로 출력할 수 있습니다.

02. C언어의 다양한 연산자 소개

- 01. 연산을 위한 연산자와 값의 저장을 위한 변수
- 03. 키보드로부터의 데이터 입력과 C언어의 키워드
- 04. 연습 문제

02. C언어의 다양한 연산자 소개

❖ 연산자란?

“ 연산자 ” (Operator)

주어진 식을 계산하여 결과를 얻어내는 과정을 연산이라고 하며,
연산을 수행하는 기호를 연산자라고 합니다.



02. C언어의 다양한 연산자 소개


❖ 대입 연산자와 산술 연산자 (1/2)

연산자	연산자의 기능
=	연산자 오른쪽에 있는 값을 왼쪽에 있는 변수에 대입합니다. 예) num = 20;
+	두 피연산자의 값을 더합니다. 예) num = 3 + 4;
-	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺍니다. 예) num = 4 - 3;
*	두 피연산자의 값을 곱합니다. 예) num = 4 * 3;
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눕니다. 예) num = 7 / 3;
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환합니다. 예) num = 7 % 3;

02. C언어의 다양한 연산자 소개

❖ 대입 연산자와 산술 연산자 (2/2)

```
1  /* operator1.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 9;
7      int num2 = 2;
8
9      printf("%d + %d = %d\n", num1, num2, num1 + num2);
10     printf("%d - %d = %d\n", num1, num2, num1 - num2);
11     printf("%d * %d = %d\n", num1, num2, num1 * num2);
12     printf("%d / %d = %d\n", num1, num2, num1 / num2);
13     printf("%d ÷ %d = %d\n", num1, num2, num1 % num2);
14
15     return 0;
16 }
```



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

함수 호출 문장 안에 연산식이 있는 경우,
연산이 이뤄지고 그 결과를 기반으로 함수의 호출이 진행됩니다.

02. C언어의 다양한 연산자 소개

❖ 복합 대입 연산자

```
1  /* operator2.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 2, num2 = 4, num3 = 6;
7
8      num1 += 3;    // num1 = num1 + 3;
9      num2 *= 4;    // num2 = num2 * 4;
10     num3 %= 5;    // num3 = num3 % 5;
11     printf("Result: %d, %d, %d\n", num1, num2, num3);
12     return 0;
13 }
```

Result: 5, 16, 1



02. C언어의 다양한 연산자 소개

❖ 부호의 의미를 갖는 + 연산자와 - 연산자

```
1  /* operator3.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = +2;          // int num1 = 2;와 동일함
7      int num2 = -4;
8
9      num1 = -num1;
10     printf("num1: %d\n", num1);
11     num2 = -num2;
12     printf("num2: %d\n", num2);
13     return 0;
14 }
```

num1: -2
num2: 4

두 연산자를 혼동하지 않도록 주의합시다!

num1=-num2;	// 부호 연산자의 사용
num1-=num2;	// 복합 대입 연산자의 사용

혼동을 최소화하는 띄어쓰기

num1 = -num2;	// 부호 연산자의 사용
num1 -= num2;	// 복합 대입 연산자의 사용

02. C언어의 다양한 연산자 소개


❖ 증가, 감소 연산자 (1/3)

연산자	연산자의 기능
<code>++num</code>	값을 1 증가 후, 속한 문장의 나머지를 진행(선 증가, 후 연산) 예) <code>val = ++num;</code>
<code>num++</code>	속한 문장을 먼저 진행한 후, 값을 1 증가(선 연산, 후 증가) 예) <code>val = num++;</code>
<code>--num</code>	값을 1 감소 후, 속한 문장의 나머지를 진행(선 감소, 후 연산) 예) <code>val = --num;</code>
<code>num--</code>	속한 문장을 먼저 진행한 후, 값을 1 감소(선 연산, 후 감소) 예) <code>val = num--;</code>

02. C언어의 다양한 연산자 소개

❖ 증가, 감소 연산자 (2/3)

```
1  /* operator4.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 12;
7      int num2 = 12;
8
9      printf("num1: %d\n", num1);
10     printf("num1++: %d\n", num1++); // 선 연산, 후 증가
11     printf("num1: %d\n\n", num1);
12     printf("num2: %d\n", num2);
13     printf("++num2: %d\n", ++num2); // 선 증가, 후 연산
14     printf("num2: %d\n", num2);
15     return 0;
16 }
```



```
num1: 12
num1++: 12
num1: 13

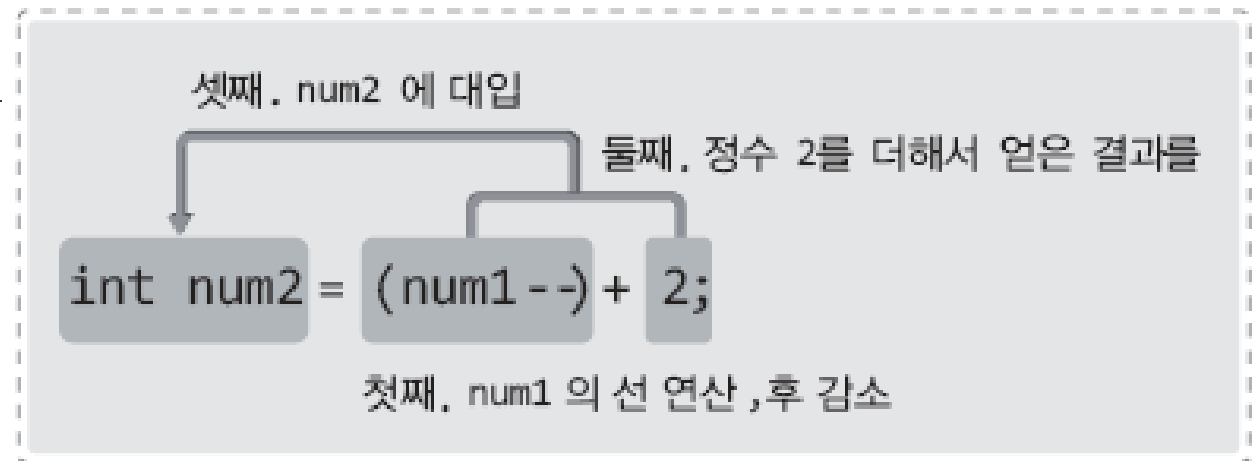
num2: 12
++num2: 13
num2: 13
```

02. C언어의 다양한 연산자 소개

❖ 증가, 감소 연산자 (3/3)

```
1  /* operator5.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 10;
7      int num2 = (num1--) + 2;    // 선 연산, 후 감소
8
9      printf("num1: %d\n", num1);
10     printf("num2: %d\n", num2);
11     return 0;
12 }
```

num1: 9
num2: 12



02. C언어의 다양한 연산자 소개

❖ 관계 연산자 (1/2)


연산자	연산자의 기능
<	예) $n1 < n2$ n1이 n2보다 작은가?
>	예) $n1 > n2$ n1이 n2보다 큰가?
==	예) $n1 == n2$ n1과 n2가 같은가?
!=	예) $n1 != n2$ n1과 n2이 다른가?
<=	예) $n1 <= n2$ n1이 n2보다 작거나 같은가?
>=	예) $n1 >= n2$ n1이 n2보다 크거나 같은가?

연산의 조건을 만족할 경우: 참(True)을 의미하는 1을 반환
연산의 조건을 만족하지 못할 경우: 거짓(False)를 의미하는 0을 반환

02. C언어의 다양한 연산자 소개

❖ 관계 연산자 (2/2)

```
1  /* operator6.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 10;
7      int num2 = 12;
8      int result1, result2, result3;
9      result1 = (num1 == num2);
10     result2 = (num1 <= num2);
11     result3 = (num1 > num2);
12     printf("result1: %d\n", result1);
13     printf("result2: %d\n", result2);
14     printf("result3: %d\n", result3);
15     return 0;
16 }
```



```
result1: 0
result2: 1
result3: 0
```

**C언어는 0이 아닌 모든 값을 참(True)으로 간주합니다.
다만, 1이 참을 의미하는 대표적인 값일 뿐입니다.**

02. C언어의 다양한 연산자 소개


❖ 논리 연산자 (1/2)

연산자	연산자의 기능
&&	예) A && B A와 B 모두 참(True)이면 연산 결과로 참을 반환(논리 AND)
	예) A B A와 B 둘 중 하나라도 참이면 연산 결과로 참을 반환(논리 OR)
!	예) !A A가 참이면 거짓(False), A가 거짓이면 참을 반환(논리 NOT)

02. C언어의 다양한 연산자 소개

❖ 논리 연산자 (2/2)

```
1  /* operator7.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 10;
7      int num2 = 12;
8      int result1, result2, result3;
9      result1 = (num1 == 10) && (num2 == 12);
10     result2 = (num1 < 12) || (num2 > 12);
11     result3 = !num1;
12     printf("result1: %d\n", result1);
13     printf("result2: %d\n", result2);
14     printf("result3: %d\n", result3);
15     return 0;
16 }
```



```
result1: 1
result2: 1
result3: 0
```


02. C언어의 다양한 연산자 소개

❖ 콤마 연산자

- ✓ 콤마(,)도 연산자입니다.
- ✓ 둘 이상의 변수를 동시에 선언하거나, 둘 이상의 문장을 한 행에 삽입하는 경우에 사용하는 연산자입니다.
- ✓ 둘 이상의 인자를 함수로 전달할 때 인자의 구분을 목적으로도 사용합니다.
- ✓ 콤마 연산자는 다른 연산자들과 달리 연산의 결과가 아닌 구분을 목적으로 합니다.

```
1  /* operator8.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 1, num2 = 2;
7      printf("Hello, "), printf("world!\n");
8      num1++, num2++;
9      printf("%d ", num1), printf("%d ", num2), printf("\n");
10     return 0;
11 }
```

Hello, world!
2 3

02. C언어의 다양한 연산자 소개

❖ 연산자의 우선순위와 결합 방향

연산자의 우선순위

- ✓ 연산의 순서에 대한 순위를 의미합니다.
- ✓ 덧셈과 뺄셈보다는 곱셈과 나눗셈의 우선순위가 높습니다.

연산자의 결합 방향

- ✓ 우선순위가 동일한 두 연산자 사이에서의 연산을 진행하는 방향을 의미합니다.
- ✓ 덧셈, 뺄셈, 곱셈, 나눗셈 모두 결합 방향이 왼쪽에서 오른쪽으로 진행된다.

$3 + 4 * 5 / 2 - 10$

- ✓ 연산자의 우선순위에 근거하여 곱셈과 나눗셈이 먼저 진행됩니다.
- ✓ 결합 방향에 근거하여 곱셈이 나눗셈보다 먼저 진행됩니다.

03. 키보드로부터의 데이터 입력과 C언어의 키워드

- 01. 연산을 위한 연산자와 값의 저장을 위한 변수
- 02. C언어의 다양한 연산자 소개
- 04. 연습 문제

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ 키보드로부터의 정수 입력을 위한 scanf 함수의 호출 (1/5)

```
1  /* simpleadd2.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1 = 3;
7      int num2 = 4;
8      int result = num1 + num2;
9
10     printf("덧셈 결과: %d\n", result);
11     printf("%d + %d = %d\n", num1, num2, result);
12     printf("%d와(과) %d의 합은 %d입니다.\n", num1, num2, result);
13
14     return 0;
15 }
```

현재 코드에서 다양한 숫자의 덧셈 결과를 얻고자 하는 경우,
코드의 변경과 그에 따른 컴파일 과정이 필수적입니다.

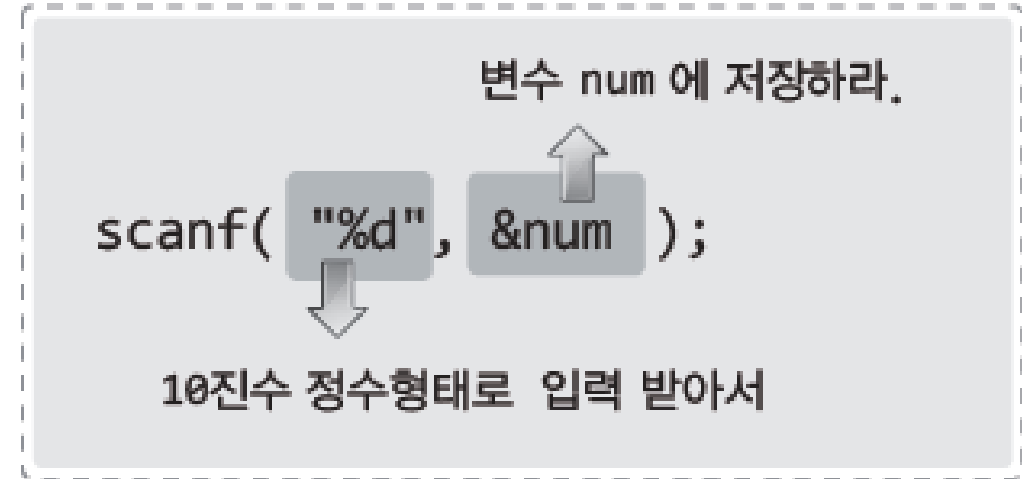
지금부터 사용자로부터 정수를 입력받아서 계산하는 형식으로
이 프로그램을 수정해 보겠습니다.

덧셈 결과: 7
3 + 4 = 7
3와(과) 4의 합은 7입니다.

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ 키보드로부터의 정수 입력을 위한 scanf 함수의 호출 (2/5)

```
1  /* scanf1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num;
8      scanf("%d", &num);
9      return 0;
10 }
```



- ✓ printf 함수에서, 서식 문자 %d는 10진수 정수의 출력을 의미합니다.
- ✓ 반면 scanf 함수에서, 서식 문자 %d는 10진수 정수의 입력을 의미합니다.
- ✓ 변수의 이름 num 앞에 &를 붙인 이유는 이후에 천천히 알게 됩니다.

- &: and를 나타내는 기호, Ampersand(앰퍼샌드)

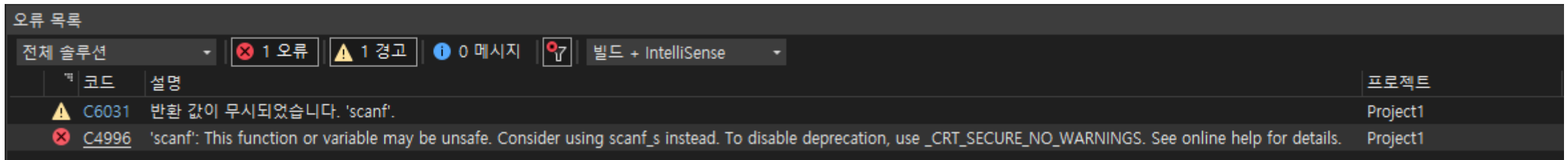
03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ 키보드로부터의 정수 입력을 위한 scanf 함수의 호출 (3/5)

```
1  /* scanf1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num;
8      scanf("%d", &num);
9      return 0;
10 }
```

- ✓ Visual Studio는 scanf 함수보다 안정적인 scanf_s 함수의 사용을 권고합니다.
- ✓ scanf_s 함수를 지원하지 않는 컴파일러도 있고, 또 이는 권고 사항이지 필수 사항이 아니기 때문에 우리 수업에서는 scanf 함수를 사용합니다.
- ✓ Visual Studio 사용자 입장에서는 여러 가지 방법으로 이 경고를 무시할 수 있는데, 여기서는 가장 간단한 방법을 사용하였습니다.

2번 줄을 주석 처리하고, 소스 코드를 실행시키면 아래와 같은 오류 문구가 출력됩니다.



03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ 키보드로부터의 정수 입력을 위한 scanf 함수의 호출 (4/5)

```
1  /* scanf1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num;
8      scanf("%d", &num);
9      return 0;
10 }
```

scanf 함수의 안전과 관련된 경고를 발생시키지 말라는 의미입니다.

다음 두 문장 중에 한 문장을 선택하고,
소스 코드의 첫 번째 줄에 선택한 문장을 삽입 합니다.

- ✓ #define _CRT_SECURE_NO_WARNINGS
- ✓ #pragma warning (disable:4996)

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ 키보드로부터의 정수 입력을 위한 scanf 함수의 호출 (5/5)

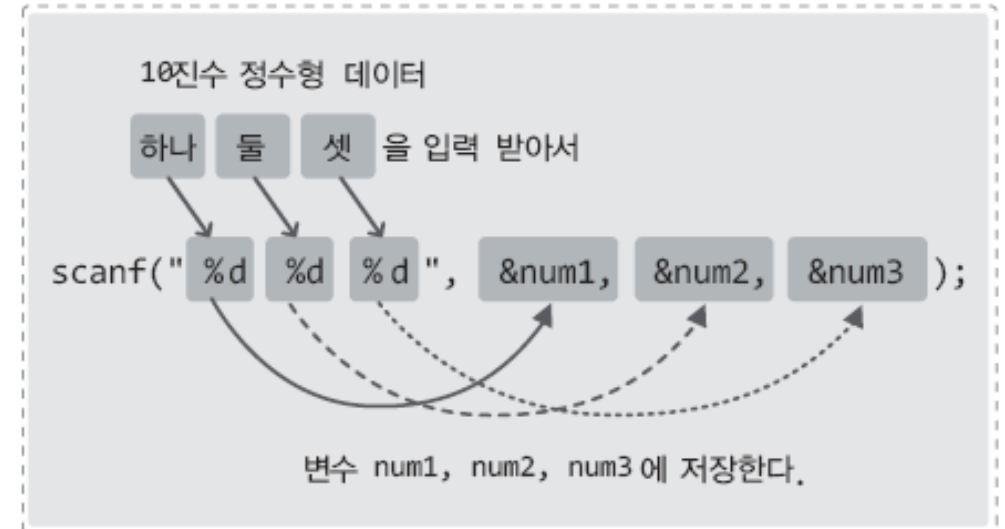
```
1  /* simpleadd3.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int result;
8      int num1, num2;
9
10     printf("첫 번째 정수: ");
11     scanf("%d", &num1);           // 첫 번째 정수 입력
12     printf("두 번째 정수: ");
13     scanf("%d", &num2);           // 두 번째 정수 입력
14
15     result = num1 + num2;
16     printf("%d + %d = %d\n", num1, num2, result);
17     return 0;
18 }
```

첫 번째 정수: 3
두 번째 정수: 4
3 + 4 = 7

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ scanf 함수 호출을 통해 여러 개의 데이터 입력받기 (1/2)

```
1  /* scanf2.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num1, num2, num3;
8      scanf("%d %d %d", &num1, &num2, &num3);
9      return 0;
10 }
```



한 번의 scanf 함수 호출을 통해서
둘 이상의 데이터를 원하는 방식으로 입력 받을 수 있습니다.

- ✓ 키보드로부터 데이터를 하나씩 입력할 때마다 공백을 삽입해 줘야 합니다.
- ✓ 스페이스 바(Space Bar), tab키, enter키를 이용하여 공백을 삽입할 수 있습니다.

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ scanf 함수 호출을 통해 여러 개의 데이터 입력받기 (2/2)

```
1  /* simpleadd4.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int result;
8      int num1, num2;
9
10     printf("숫자 두 개를 입력하세요: ");
11     scanf("%d %d", &num1, &num2);
12
13     result = num1 + num2;
14     printf("%d + %d = %d\n", num1, num2, result);
15     return 0;
16 }
```

숫자 두 개를 입력하세요: 3 4
3 + 4 = 7

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ C언어의 표준 키워드 (1/2)

“ 키워드 ” (Keywords)

C언어의 문법을 구성하는, 그 의미가 결정되어 있는 단어들!
이러한 단어들을 가리켜 키워드(Keywords)라 합니다.

- ✓ int, return과 같은 것들은 이미 기능적 의미가 정해져 있습니다.
- ✓ 이러한 키워드들이 모여서 C언어의 문법 체계를 구성하게 됩니다.
- ✓ 키워드들은 변수나 함수의 이름으로 사용할 수 없습니다.

03. 키보드로부터의 데이터 입력과 C언어의 키워드

❖ C언어의 표준 키워드 (2/2)

C언어에서 표준화된 키워드들			
auto	_Bool	break	case
char	_Complex	const	continue
default	do	double	else
enum	extern	float	for
goto	if	_Imaginary	return
restrict	short	signed	sizeof
static	struct	switch	typedef
union	unsigned	void	volatile
while			

키워드들을 모두를 외울 필요는 전혀 없습니다!

04. 연습 문제

- 01. 연산을 위한 연산자와 값의 저장을 위한 변수
- 02. C언어의 다양한 연산자 소개
- 03. 키보드로부터 데이터 입력과 C언어의 키워드

04. 연습 문제

❖ 연습 문제 1.

- 사용자로부터 두 개의 정수를 입력받아서 뺄셈과 곱셈 연산의 결과를 출력하는 프로그램을 작성하세요.

두 개의 정수 입력: 9 3
뺄셈 결과: 6
곱셈 결과: 27

04. 연습 문제

❖ 연습 문제 1. 정답 및 해설

```
1  /* example1.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num1, num2;
8      int result1, result2;
9
10     printf("두 개의 정수 입력: ");
11     scanf("%d %d", &num1, &num2);
12     result1 = num1 - num2;
13     result2 = num1 * num2;
14     printf("뺄셈 결과: %d\n", result1);
15     printf("곱셈 결과: %d\n", result2);
16
17     return 0;
18 }
```

두 개의 정수 입력: 9 3
뺄셈 결과: 6
곱셈 결과: 27

04. 연습 문제

❖ 연습 문제 2.

- 사용자로부터 세 개의 정수를 입력받은 다음에 곱과 합을 순서대로 진행해서 그 결과를 출력하는 프로그램을 작성하세요. 예를 들어, 사용자로부터 순서대로 입력받은 세 개의 정수가 1, 2, 3이라면 $1 \times 2 + 3$ 의 계산 결과를 출력해야 합니다.

세 개의 숫자를 입력하세요: 1 2 3

$1 * 2 + 3 = 5$

04. 연습 문제

❖ 연습 문제 2. 정답 및 해설

```
1  /* example2.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int n1, n2, n3;
8      int result
9
10     printf("세 개의 숫자를 입력하세요: ");
11     scanf("%d %d %d", &n1, &n2, &n3);
12     result = n1 * n2 + n3;
13     printf("%d * %d + %d = %d\n", n1, n2, n3, result);
14
15     return 0;
16 }
```

세 개의 숫자를 입력하세요: 1 2 3
1 * 2 + 3 = 5

04. 연습 문제

❖ 연습 문제 3.

- 하나의 정수를 입력받아서 제곱 연산을 한 다음 얻어지는 결과를 출력하는 프로그램을 작성하세요.
예를 들어, 4를 입력하면 16이 출력되어야 합니다.

한 개의 숫자를 입력하세요: 4

16

04. 연습 문제

❖ 연습 문제 3. 정답 및 해설

```
1  /* example3.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num;
8
9      printf("한 개의 숫자를 입력하세요: ");
10     scanf("%d", &num);
11     printf("%d * %d = %d\n", num, num, num * num);
12
13     return 0;
14 }
```

한 개의 숫자를 입력하세요: 4
16

04. 연습 문제

❖ 연습 문제 4.

- 입력받은 두 개의 정수를 나누었을 때 발생하는 나머지 값을 출력하는 프로그램을 작성하세요.
몫은 구하지 않아도 됩니다. 예를 들어, 3과 2를 입력하면 나머지 1이 출력되어야 합니다.

두 개의 숫자를 입력하세요: 3 2
나머지: 1

04. 연습 문제

❖ 연습 문제 4. 정답 및 해설

```
1  /* example4.c */
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int num1, num2;
8
9      printf("두 개의 숫자를 입력하세요: ");
10     scanf("%d %d", &num1, &num2);
11     printf("나머지: %d\n", num1 % num2);
12
13     return 0;
14 }
```

두 개의 숫자를 입력하세요: 3 2
나머지: 1

- ❖ 01. 연산을 위한 연산자와 값의 저장을 위한 변수
- ❖ 02. C언어의 다양한 연산자 소개
- ❖ 03. 키보드로부터 데이터 입력과 C언어의 키워드
- ❖ 04. 연습 문제

THANK YOU!

Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr