Fundamentals of Computer System
- Lecture 2 (Introducing C)

민기복

Ki-Bok Min, PhD

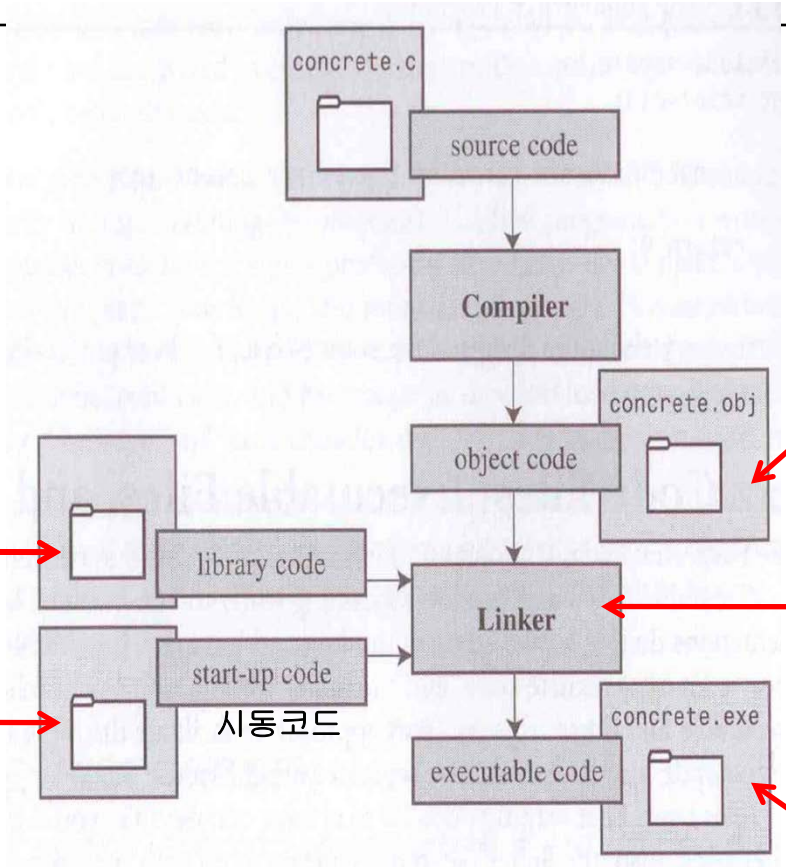
서울대학교 에너지자원공학과 조교수

Assistant Professor, Energy Resources Engineering

SEOUL NATIONAL UNIVERSITY

Last Lecture

SEOUL NATIONAL UNIVERSITY



Object file contains machine language, but not ready to run

Collection of useful functions

Linker bring together object file, startup code and library code

Interface between your computer & operation system (DOS or Linux?).

Executable program!

The compiler we will use – which is MS Visual C++ express ed – starts linker automatically.

Today's content

SEOUL NATIONAL UNIVERSITY

- Putting together a simple C program
- Functions (함수): main(), printf()
- Creating variables, assigning them values, and displaying those values on screen
- The newline character (개행문자), Operator (연산자): =
- Comments in your program, creating programs with more than one functions, and finding program errors
- Keywords – vocabulary of C

1. First C program

SEOUL NATIONAL UNIVERSITY

```c
#include <stdio.h>
int main(void)
{
    printf("hello, world\n");
}
```

last week

this week

```c
#include <stdio.h>
int main(void)                  /* a simple program                  */
{
    int num;                    /* define a variable called num      */
    num = 1;                    /* assign a value to num             */

    printf("I am a simple ");   /* printf() 함수를 사용한다          */
    printf("computer.\n");
    printf("My favorite number is %d because it is first.\n",num);

    return 0;
}
```

1. First C program

SEOUL NATIONAL UNIVERSITY

Include another file

```c
#include <stdio.h>
int main(void)                  /* a simple program                 */
{
    int num;                    /* define a variable called num     */
    num = 1;                    /* assign a value to num            */

    printf("I am a simple ");   /* printf() 함수를 사용한다         */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

- Tells the compiler to include the information found in the file stdio.h.
- stdio.h: a standard part of all C compiler packages, support for keyboard input and for displayinig output
- #include : preprocessor instructions (전처리 지시자)

1. First C program

SEOUL NATIONAL UNIVERSITY

a function name

```c
#include <stdio.h>
int main(void)                  /* a simple program                */
{
    int num;                    /* define a variable called num    */
    num = 1;                    /* assign a value to num           */

    printf("I am a simple ");   /* printf() 함수를 사용한다        */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

- C program consist of one or more *functions (함수)*.
- This particular program (first.c) consists of one function called **main**.
- The parenthesis identify **main()** as a function name
- **int** → **main()** returns an integer, void → main() doesn't have argument (전달인자)
- main() is always the first function called.
- Old compiler → int main() :pre ISO/ANSI C compiler

1. First C program

SEOUL NATIONAL UNIVERSITY

a comment (주석)

```c
#include <stdio.h>
int main(void)
{
    int num;
    num = 1;

    printf("I am a simple ");
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

```c
/* a simple program                          */
/* define a variable called num              */
/* assign a value to num                     */
/* printf() 함수를 사용한다                  */
```

- The symbols /* and */ enclose comments.
- Clarify a program.
- They are intended for the reader only
- Ignored by the compiler

1. First C program

SEOUL NATIONAL UNIVERSITY

Beginning & finishing of the body of the function

```
#include <stdio.h>
int main(void)                  /* a simple program                */
{
    int num;                    /* define a variable called num    */
    num = 1;                    /* assign a value to num           */

    printf("I am a simple ");   /* printf() 함수를 사용한다        */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

- Opening and closing brace marks make up the function

1. First C program

SEOUL NATIONAL UNIVERSITY

```c
#include <stdio.h>
int main(void)
{
    int num;                    /* define a variable called num    */
    num = 1;                    /* assign a value to num           */

    printf("I am a simple ");   /* printf() 함수를 사용한다        */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

A declaration statement
선언 명령문

- Announces that you are using a variable called **num**
- **num** will be an **int** (integer) type

1. First C program

SEOUL NATIONAL UNIVERSITY

```c
#include <stdio.h>
int main(void)
{
    int num;
    num = 1;

    printf("I am a simple ");    /* printf() 함수를 사용한다      */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

an assignment statement
(대입명령문)

- Assigns the value 1 to the variable called num

1. First C program

SEOUL NATIONAL UNIVERSITY

```c
#include <stdio.h>
int main(void)
{
    int num;
    num = 1;

    printf("I am a simple ");  /* printf() 함수를 사용한다 */
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

a function call statement
(함수 호출 명령문)

- Displays the phrase **I am a simple** on your screen
- Leaves the cursor on the same line
- **printf()** is called function and is part of the standard C library
- Using a function in the program is termed *calling a function* (함수호출)

1. First C program

SEOUL NATIONAL UNIVERSITY

```
#include <stdio.h>
int main(void)                  /* a simple program                */
{
    int num;                    /* define a variable called num    */
    num = 1;

    printf("I am a simple ");
    printf("computer.₩n");
    printf("My favorite number is %d because it is first.₩n",num);

    return 0;
}
```

last function call statement
(마지막 함수 호출 명령문)

- Prints the value of **num** embedded in the phrase in quotes.
- **%d** instructs the computer ‘where’ and ‘in what form’ to print the value of **num**

1. First C program

SEOUL NATIONAL UNIVERSITY

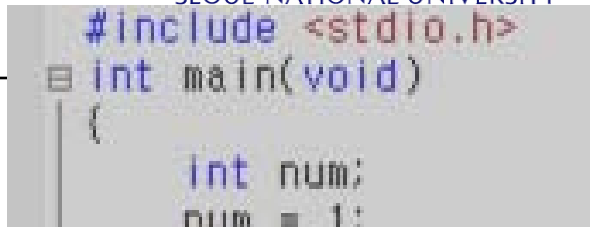
```
#include <stdio.h>
int main(void)                  /* a simple program                */
{
    int num;                    /* define a variable called num    */
    num = 1;                    /* assign a value to num           */

    printf("I am a s                                                */
    printf("computer
    printf("My favor                                        num);

    return 0;
}
```

a return statement
(리턴 명령문)

- a C function can furnish (or return) a number to the agency that used it.
- ISO/ANSI C requirement for a properly written main() function.

2. Program Details
#include directives and Header Files

SEOUL NATIONAL UNIVERSITY



- #include<stdio.h>
  - An example of C preprocessor directive (C 전처리 지시자)
  - Preprocessing (전처리): C compiler perform some preparatory work on source code
  - Same as typing the entire **stdio.h** file (~300 lines) into your file
  - **stdio.h** stands for Standard input/output header
  - Contains information about input and output functions, ex) **printf()**
  - A collection of information at the top of a file: **header**
    - Library: contains actual code for a function,
    - header files: help guide the compiler in putting your program together

2. Program Details
#include directives and Header Files

SEOUL NATIONAL UNIVERSITY

- Why wasn’t this automatically included???

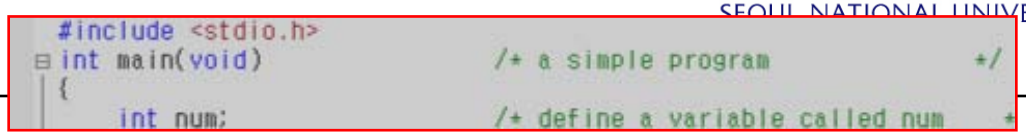
2. Program Details
the main() function

```
#include <stdio.h>
int main(void)
{
    int num;
    num = 1;
```

SEOUL NATIONAL UNIVERSITY

- C always begins execution with the function called **main()**.
- You are free to choose names for other functions, but **main()** must be there to start things.
- **int** is the main() function's return type.
- Parenthesis enclose information being passed along to the function.
  - In this particular example, nothing is being passed along, so 'void'.
  - main() or void main() may or may not work in your compiler. But don't use this *old* format.

2. Program Details
Comments



- Everything between /* and */ is ignored by the compiler.
- Valid and invalid comment

```
/* This is a C comment. */

/* This comment is spread over
   TWO LINES. */

/*

You can do this, too.

*/

/* Hopefully, this works.
```

→ Not valid

2. Program Details

Comments

```
/* a simple program                    */
```

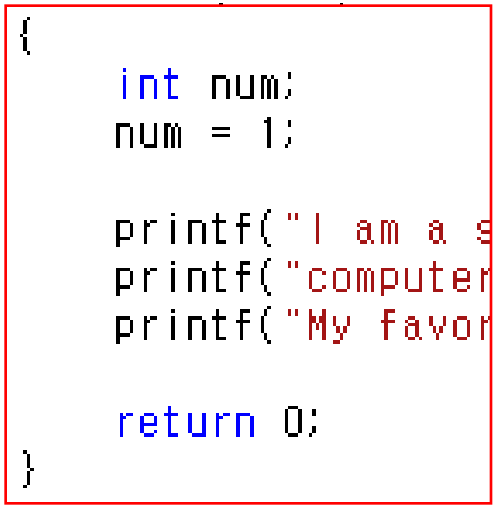
SEOUL NATIONAL UNIVERSITY

// : confined to a single line

Ex)

```
// Here is a comment

int rigue;   //such comments can go here, too.

 /*
I hope this works.
*/
x = 100;
y = 200;
/* now for something else. */
```

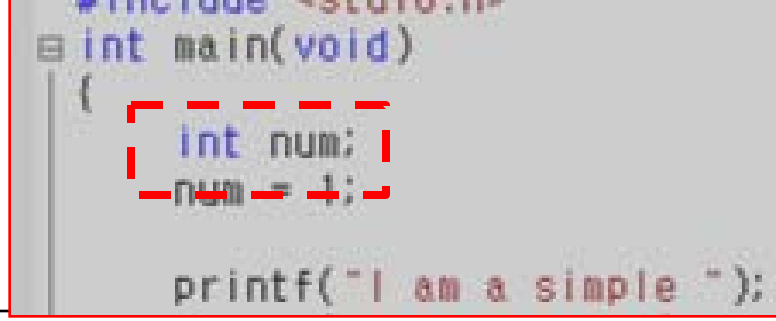
2. Program Details
Braces (중괄호)



```
{
    int num;
    num = 1;

    printf("I am a s
    printf("computer
    printf("My favor

    return 0;
}
```

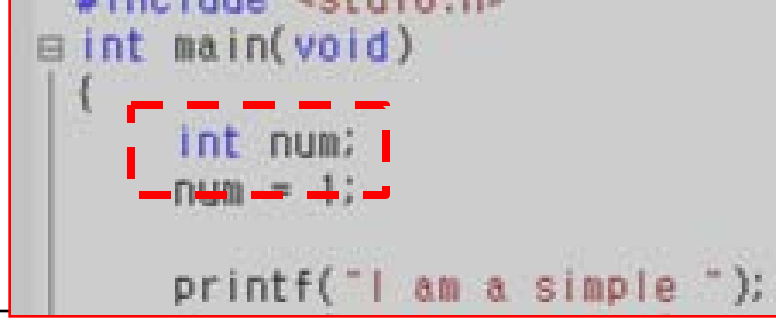
- Only { }
- 중괄호만 쓸 것!!!

2. Program Details
Declaration (선언 명령문)



- One of C's most important features
- Does two things;
  - 1. Somewhere in the function, you have a variable called **num**
  - 2. **int** proclaims num as an integer (정수형) (there are other types, ex) character, floating-point,…)
- Compiler uses this info for suitable storage space in memory
- **;** identifies the line as **C statement**.
- **int** is a keyword – reserved for C, so don't use it for your name of a function or a variable.

2. Program Details
Declaration (선언 명령문)



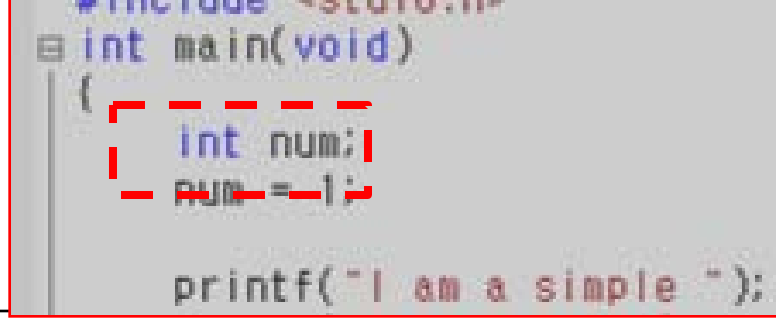
- num is called an *identifier* (식 별 자) – name selected for a variable, a function, or some other entity. Declaration connects a particular identifier with a particular location in computer memory
- All variables must be declared *before* they are used.

Ex)

```
int main()                          // C99 rules
{
//    some statements
      int doors;
      doors = 5;     // first use of doors
//    more statements
      int dogs;
      dogs =3                       //first use of dogs
      // other statements
}
```

As long as it is declared before its first use → okay
But, it is a good practice to declare variables from the beginning.

2. Program Details
Declaration (선언 명령문)



- The maximum number of characters for variables: 63

The_maximum_number_OF_characters_for_variables_is_63_the_rest_will_be

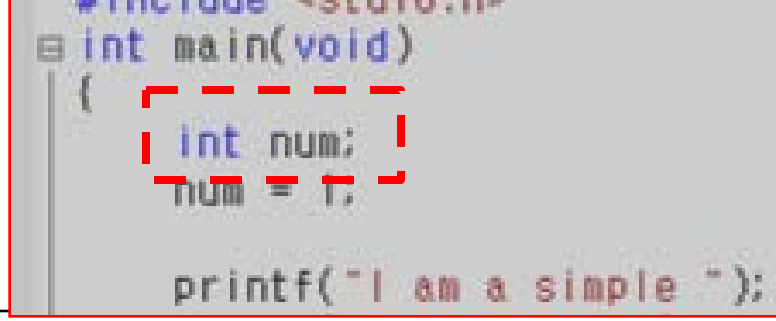
- Use lowercase or uppercase letters, digits, & underscore.
- The first character cannot be a digit.

The rest will be simply ignored

  - Valid names: snu2 Snu2 Hot_Tub _kcab
  - Invalid names: $Z]** 2cat Hot-Tub don't

- Avoid starting with _ (not an error but confusing with other C library identifiers)
- Case sensitive

2. Program Details
Declaration (선언 명령문)

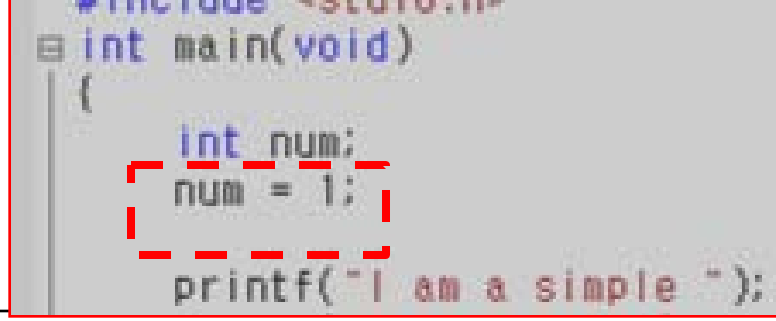


- Four reasons to declare variables
  - Easier for a reader to grasp what the program is about
  - Encourages you to do some planning before you plunging into writing a program
  - Prevent hard-to-find bugs,

  Ex) Radius1 = 20.4
  Circum = 6.28 * Radiusl

  - Compiler won't work without declaration!

2. Program Details
Assignment (대입 명령문)



- Assign the value 1 to the variable **num**.
- You can assign **num** a different value later – that's why is it a *variable*
- Assigns the value from the right side to the left side.
- Statement is completed with semicolon ;

2. Program Details
The printf() function

SEOUL NATIONAL UNIVERSITY

```
printf("I am a simple ");    /* printf( ) 함수를 사용한다      */
printf("computer.₩n");
printf("My favorite number is %d because it is first.₩n",num);
```

- () → printf is a function
- I am a simple is passed to printf().
- Printf("I am a simple");

  Argument (전달인자)

- When the program reaches this line, control is to function is executed. After function is finished, control is returned to the original (*calling*) function – **main**() in this case.

2. Program Details
The printf() function

SEOUL NATIONAL UNIVERSITY

```
printf("I am a simple ");    /* printf( ) 함수를 사용한다      */
printf("computer.₩n");
printf("My favorite number is %d because it is first.₩n",num);
```

printf("computer. \n");

Newline character (개행문자) – a escape sequence
Start a new line at the left = pressing 'enter' key.

printf("My favorite number is %d because it is first.\n",num)

- % alert the program that a variable is going to be printed
- d tells it to print as a *decimal* integer

2. Program Details
Return statement

```
num = 1;                    /*

printf("I am a simple ");
printf("computer.\n");
printf("My favorite number i

return 0;
```

- int main(void) → **main()** function is supposed to return an integer
- Return 0;
- Even if you don't do this, it will work.
- Don't worry too much about this.

3. The structure of a simple program
A function consists of a header and a body

SEOUL NATIONAL UNIVERSITY

Header

Body

```
{
...
return 0;
 }
```

```
#include <stdio.h>
int main(void)

{
int q;
q = 1;
printf("%d is neat. \n", q);
return 0;
}
```

Preprocessor instructions (전처리 지시자)

Function name (함수이름)

Declaration statement (선언명령문)

Assignment statement (대입명령문)

Function statement (함수호출명령문)

4. Tips on making a readable program
Style matters

SEOUL NATIONAL UNIVERSITY

- A readable program is much easier to understand, to correct or modify
  - Choose meaningful variables
  - Use comments as much as possible
  - Use blank lines to separate one section from another
  - Use one line per statement – C has a free-form format and allows for multiple statements per line

```
#include <stdio.h>

int main(void)  { int q; q = 1; printf("%d is neat. \n", q); return 0; }
```

This program works okay but this a bad program!!

5. Taking another step
Calculation & multiple variables

SEOUL NATIONAL UNIVERSITY

```c
// fathm_ft.c -- 2 fathoms를 feet로 변환한다
#include <stdio.h>
int main(void)
{
    int feet, fathoms;

    fathoms = 2;
    feet = 6 * fathoms;
    printf("%d fathoms는 %d feet다!₩n", fathoms, feet);
    printf("맞아, %d feet야.₩n", 6 * fathoms);

    return 0;
}
```

Begins with a comment

Declares two variables instead of just one,
Use comma ,

calculation. feet = 6 × 2

Printing multiple variables

Printing a value (not just a variable)

5. Taking another step
Multiple functions

SEOUL NATIONAL UNIVERSITY

prototype

Function call

butler() executed here!!!

Function definition

```c
/* two_func.c -- 하나의 파일에서 두 개의 함수를 사용하는 프로그램 */
#include <stdio.h>
void butler(void);      /* ISO/ANSI C 함수 프로토타입 */
int main(void)
{
    printf("butler 함수 밖에 있느냐?\n");
    butler();
    printf("그래. 차 한 잔 내오고, CD-ROM 드라이브도 가져오너라.\n");

    return 0;
}
void butler(void)       /* 함수 정의의 시작 */
{
    printf("부르셨습니까? 주인님!\n");
}
```

- **butler**() function appeared three times: prototype, function call, function definition
- void butler(void);

no return value

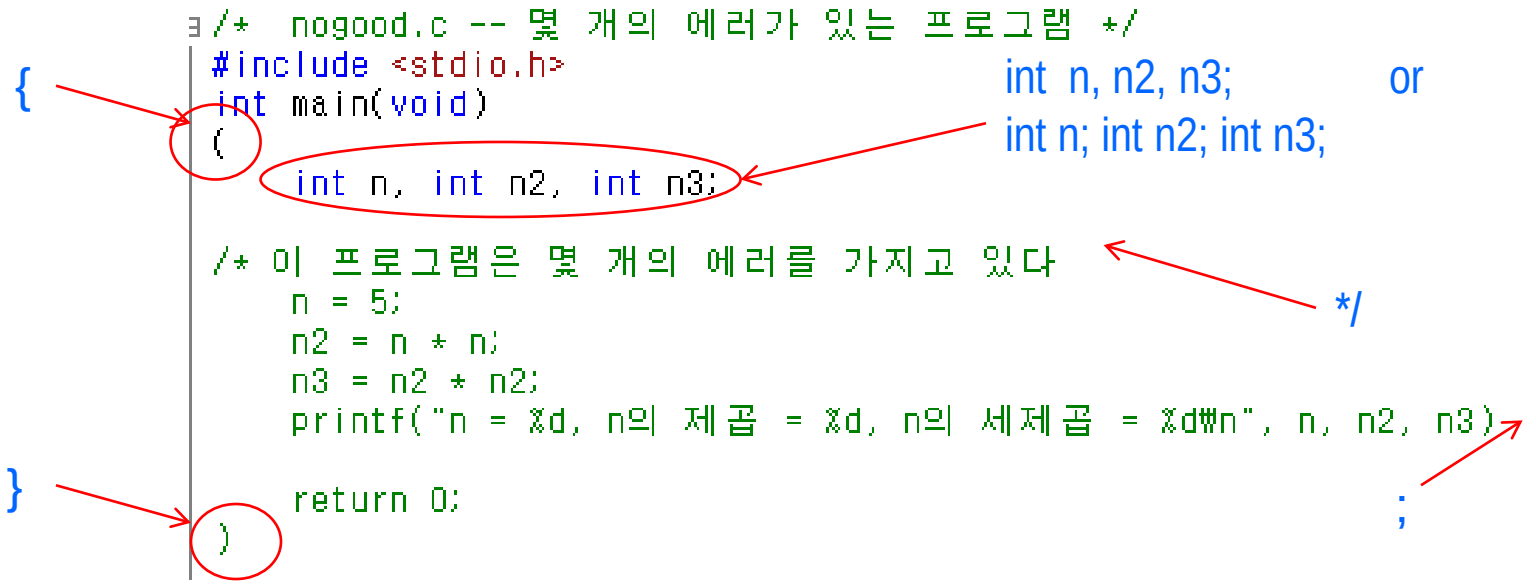
no argument

- Prototype: - a declaration telling the compiler that you are using a function.
  - specifies properties of the function
- The location of function definition does not matter. But it's nice to put **main**() first.

5. Taking another step
Debugging – Syntax errors

SEOUL NATIONAL UNIVERSITY

- Bugs: program errors.
- Debugging: finding and fixing the errors.
- Syntax error = Grammatical error in C, compiler will find these (but don't just rely on that)~ grammatical error in English



5. Taking another step
Debugging – semantic errors

SEOUL NATIONAL UNIVERSITY

- Semantic errors: errors in meaning
- Compiler does not detect semantic errors ← don't violate C rules.
- You have to find these errors.

```
/*  nogood.c -- 몇 개의 에러가 있는 프로그램 */
#include <stdio.h>
int main(void)
(
    int n, int n2, int n3;

/* 이 프로그램은 몇 개의 에러를 가지고 있다
    n = 5;
    n2 = n * n;
    n3 = n2 * n2;
    printf("n = %d, n의 제곱 = %d, n의 세제곱 = %d₩n", n, n2, n3)

    return 0;
)
```

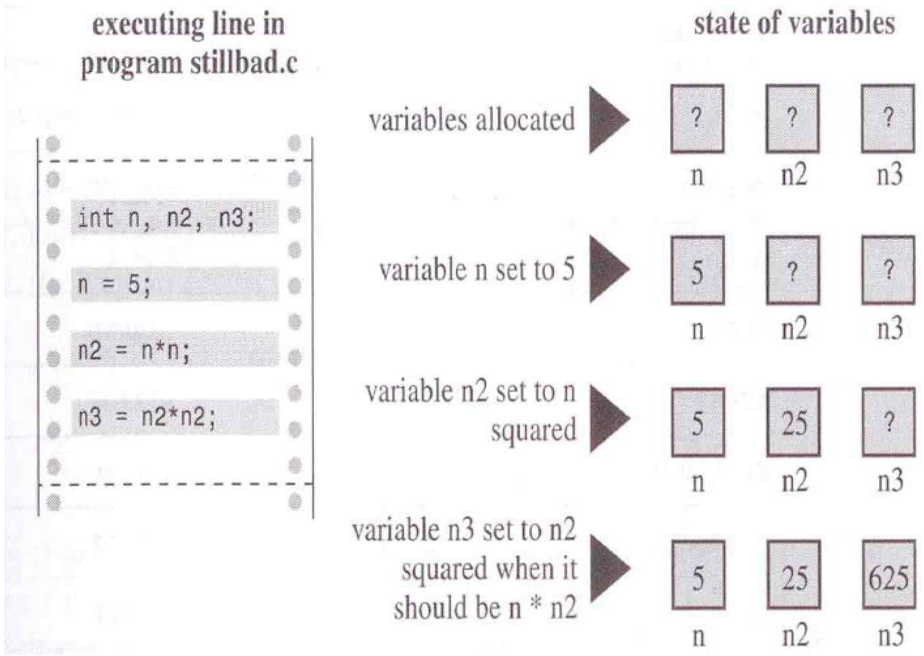
N3 = n2 * n;

5. Taking another step
Debugging – Program state

SEOUL NATIONAL UNIVERSITY

- Tracing the state
  - Follow the program steps one by one
  - Use extra (temporary) printf() to monitor the selected variables at key points
  - Use debugger

Tracing a program



5. Taking another step
Keywords and reserved identifiers

SEOUL NATIONAL UNIVERSITY

- Keywords : vocabulary of C – not your vacabulary and we can't use them as identifier

  Ex) int float do void if long …

  - There will be syntax errors if you use them.
- Reserved identifiers: no syntax errors but you shouldn't use. Reserved

  Ex) printf(), _Bool (those beginning with _)

Summary

SEOUL NATIONAL UNIVERSITY

- Putting together a simple C program
- Functions (함수): main(), printf()
- Creating variables, assigning them values, and displaying those values on screen
- The newline character (개행문자), Operator (연산자): =
- Comments in your program, creating programs with more than one functions, and finding program errors
- Keywords – vocabulary of C