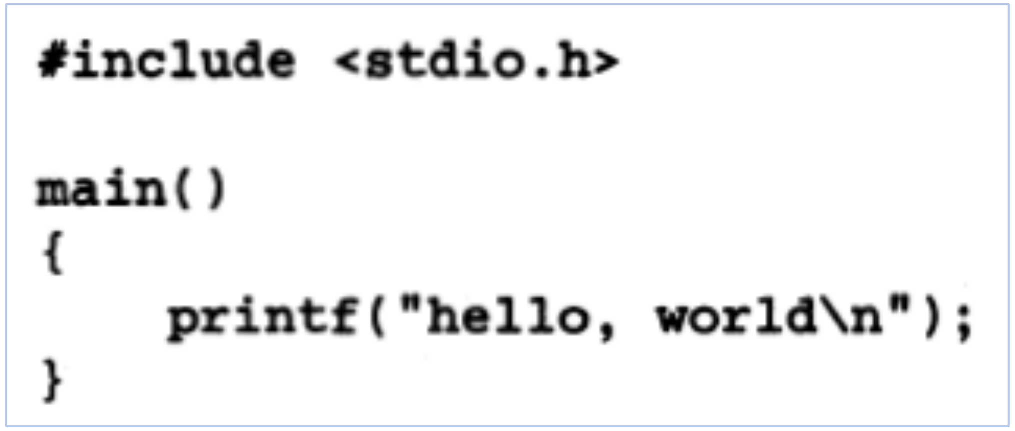
A Tutorial Introduction

백 윤 철

Contents

Getting Started

- In C, the program to print "`hello, world`"



```
#include <stdio.h>

main()
{
     printf("hello, world\n");
}
```

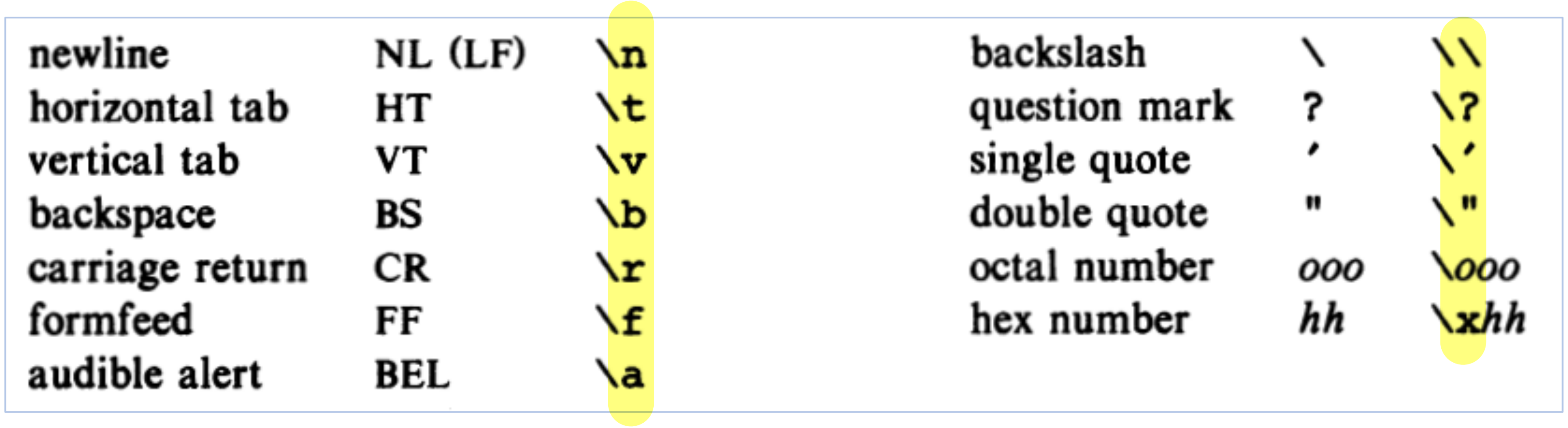
- Program begins executing at the beginning of `main()`.
- `main()` will usually call other functions to help perform its job, some that you wrote, and others from libraries that are provided for you.

Getting Started

- One method of communicating data between functions is for the calling function to provide a list of values, called arguments, to the function it calls.
- A function is called by naming it, followed by a 괄호 안의 parenthesized list arguments, so this calls the function `printf()` with the argument `“hello, world\n”.`
- The sequence `\n` in the string is C notation for the *newline character*.

Getting Started

- Escape sequences

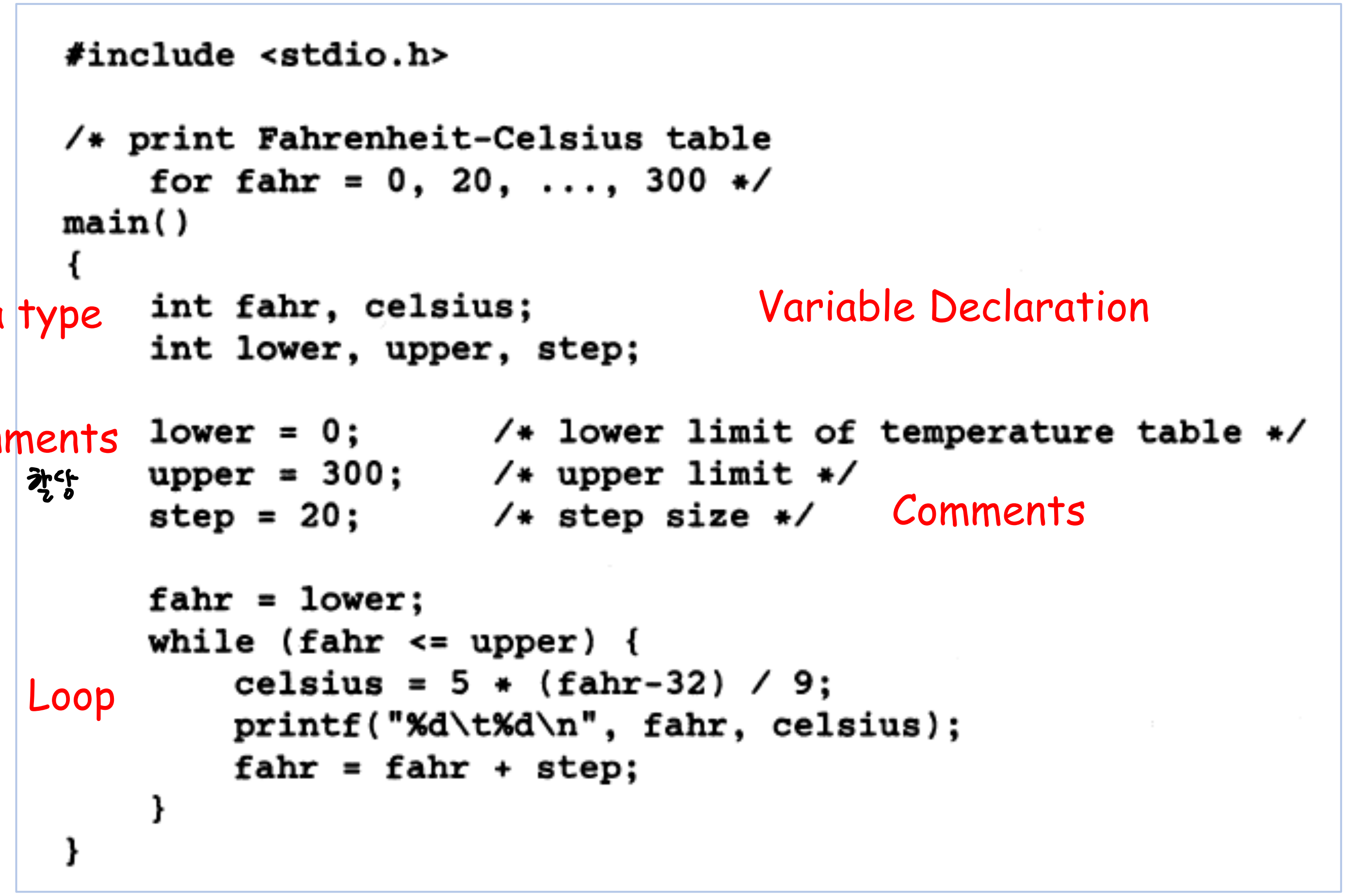


newline	NL (LF)	`\n`	backslash	`\`	`\\`
horizontal tab	HT	`\t`	question mark	`?`	`\?`
vertical tab	VT	`\v`	single quote	`’`	`\’`
backspace	BS	`\b`	double quote	`"`	`\"`
carriage return	CR	`\r`	octal number	*ooo*	`\`*ooo*
formfeed	FF	`\f`	hex number	*hh*	`\x`*hh*
audible alert	BEL	`\a`			

Variables and Arithmetic Expressions

- Fahrenheit - Celsius temperatures.

fahr	0
celsius	-17
lower	0
upper	300
step	20



```
#include <stdio.h>

/* print Fahrenheit-Celsius table
    for fahr = 0, 20, ..., 300 */
main()
{
    int fahr, celsius;
    int lower, upper, step;

    lower = 0;      /* lower limit of temperature table */
    upper = 300;    /* upper limit */
    step = 20;      /* step size */

    fahr = lower;
    while (fahr <= upper) {
        celsius = 5 * (fahr-32) / 9;
        printf("%d\t%d\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

Data type

Variable Declaration

Assignments

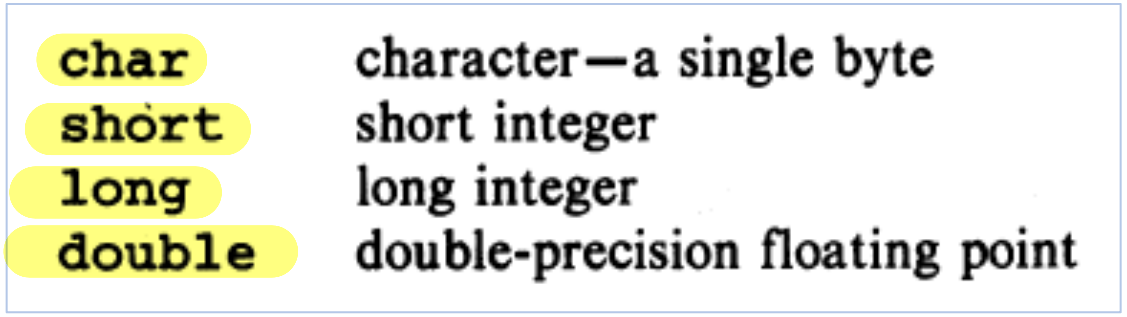
할당

Comments

Loop

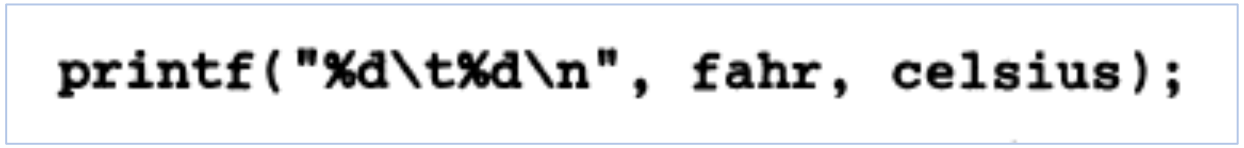
Variables and Arithmetic Expressions

- Basic Data Types



char	character—a single byte
short	short integer
long	long integer
double	double-precision floating point

- printf()– standard library function



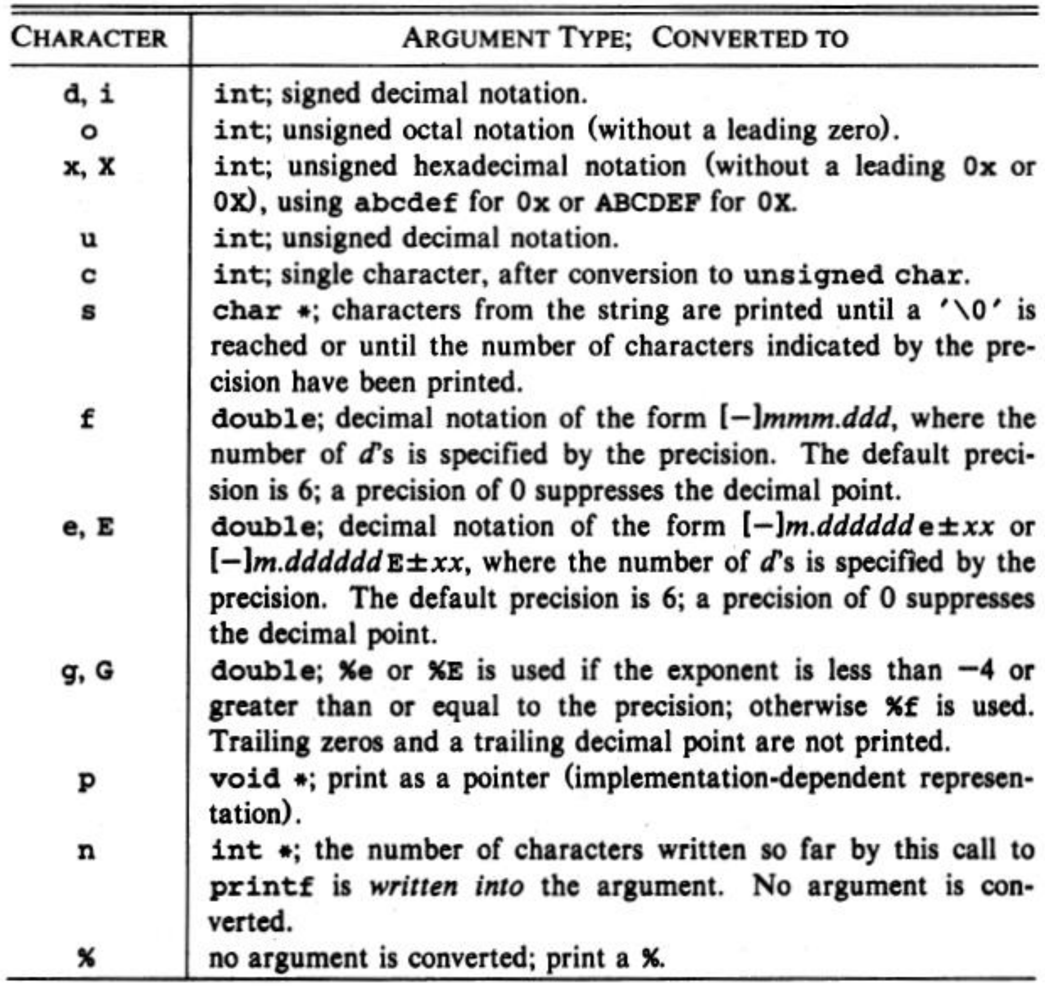
```
printf("%d\t%d\n", fahr, celsius);
```

  - %d printf() conversion
  - \t escape sequence

Variables and Arithmetic Expressions

- `printf()` conversions

TABLE B-1. PRINTF CONVERSIONS

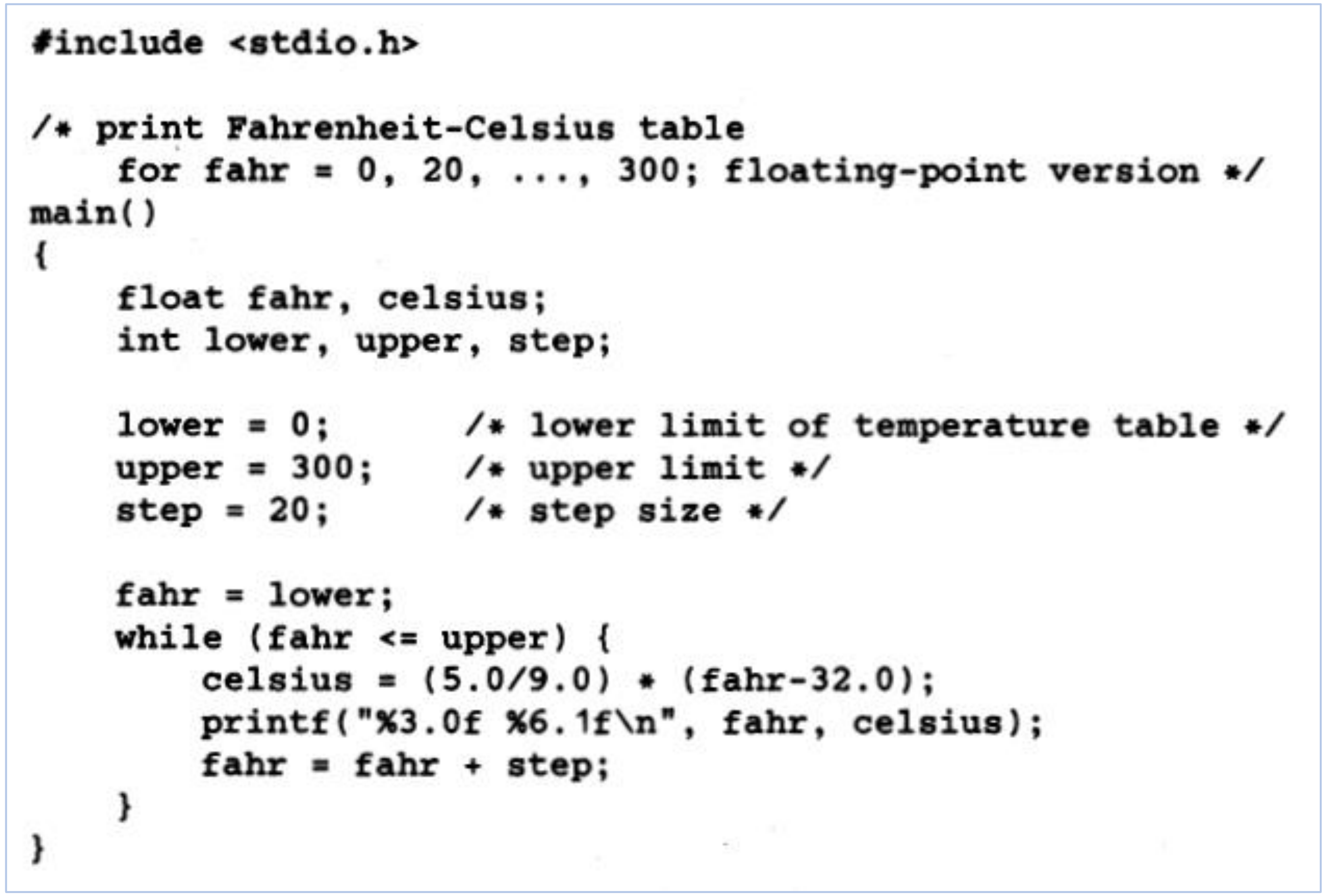


CHARACTER	ARGUMENT TYPE; CONVERTED TO
`d`, `i`	`int`; signed decimal notation.
`o`	`int`; unsigned octal notation (without a leading zero).
`x`, `X`	`int`; unsigned hexadecimal notation (without a leading `0x` or `0X`), using `abcdef` for `0x` or `ABCDEF` for `0X`.
`u`	`int`; unsigned decimal notation.
`c`	`int`; single character, after conversion to `unsigned char`.
`s`	`char *`; characters from the string are printed until a `'\0'` is reached or until the number of characters indicated by the precision have been printed.
`f`	`double`; decimal notation of the form [−]*mmm.ddd*, where the number of *d*'s is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
`e`, `E`	`double`; decimal notation of the form [−]*m.dddddd*`e`±*xx* or [−]*m.dddddd*`E`±*xx*, where the number of *d*'s is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
`g`, `G`	`double`; `%e` or `%E` is used if the exponent is less than −4 or greater than or equal to the precision; otherwise `%f` is used. Trailing zeros and a trailing decimal point are not printed.
`p`	`void *`; print as a pointer (implementation-dependent representation).
`n`	`int *`; the number of characters written so far by this call to `printf` is *written into* the argument. No argument is converted.
`%`	no argument is converted; print a `%`.

Variables and Arithmetic Expressions

산술의

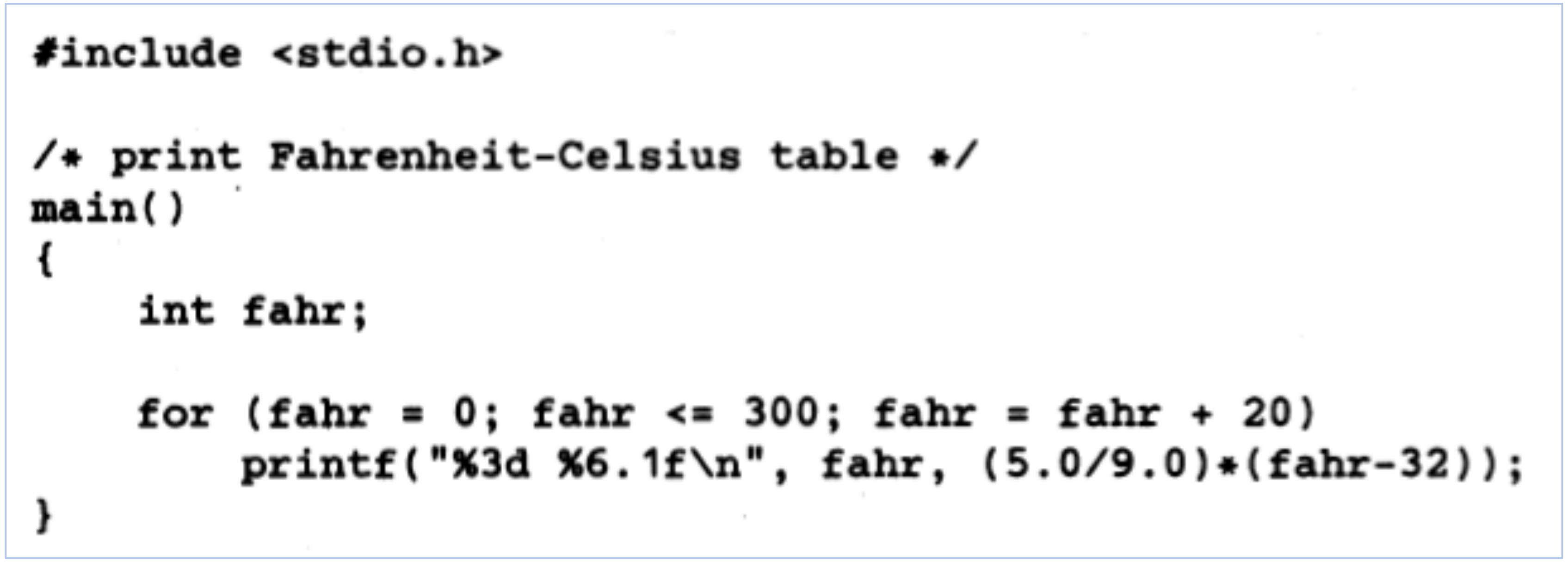
- floating-point arithmetic version



```
#include <stdio.h>

/* print Fahrenheit-Celsius table
    for fahr = 0, 20, ..., 300; floating-point version */
main()
{
    float fahr, celsius;
    int lower, upper, step;

    lower = 0;       /* lower limit of temperature table */
    upper = 300;     /* upper limit */
    step = 20;       /* step size */

    fahr = lower;
    while (fahr <= upper) {
        celsius = (5.0/9.0) * (fahr-32.0);
        printf("%3.0f %6.1f\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

For statements

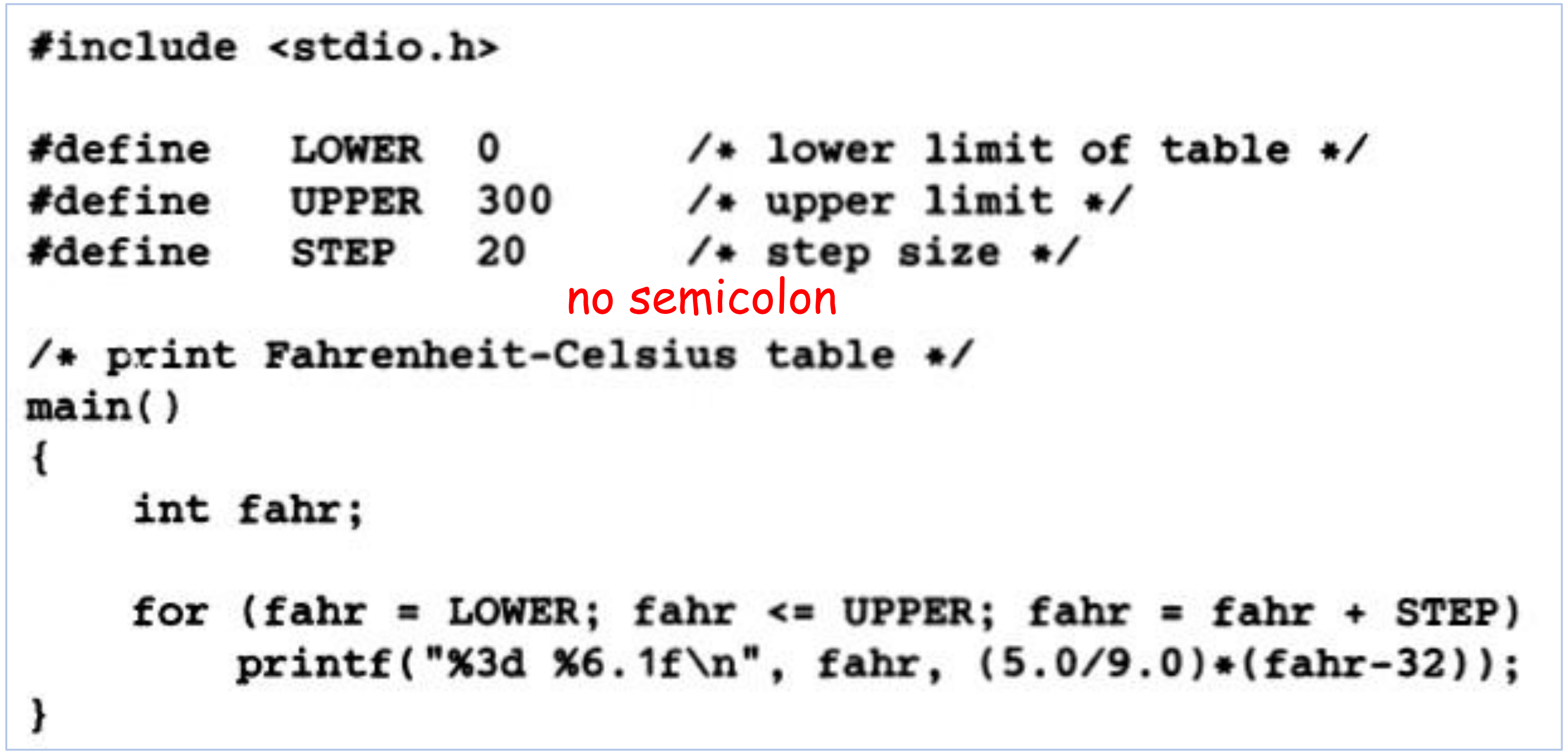
- for (initialization; test; increment step){}



```
#include <stdio.h>

/* print Fahrenheit-Celsius table */
main()
{
    int fahr;

    for (fahr = 0; fahr <= 300; fahr = fahr + 20)
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
}
```

Symbolic Constants

- #define name replacement-text



```c
#include <stdio.h>

#define   LOWER  0        /* lower limit of table */
#define   UPPER  300      /* upper limit */
#define   STEP   20       /* step size */
```

no semicolon

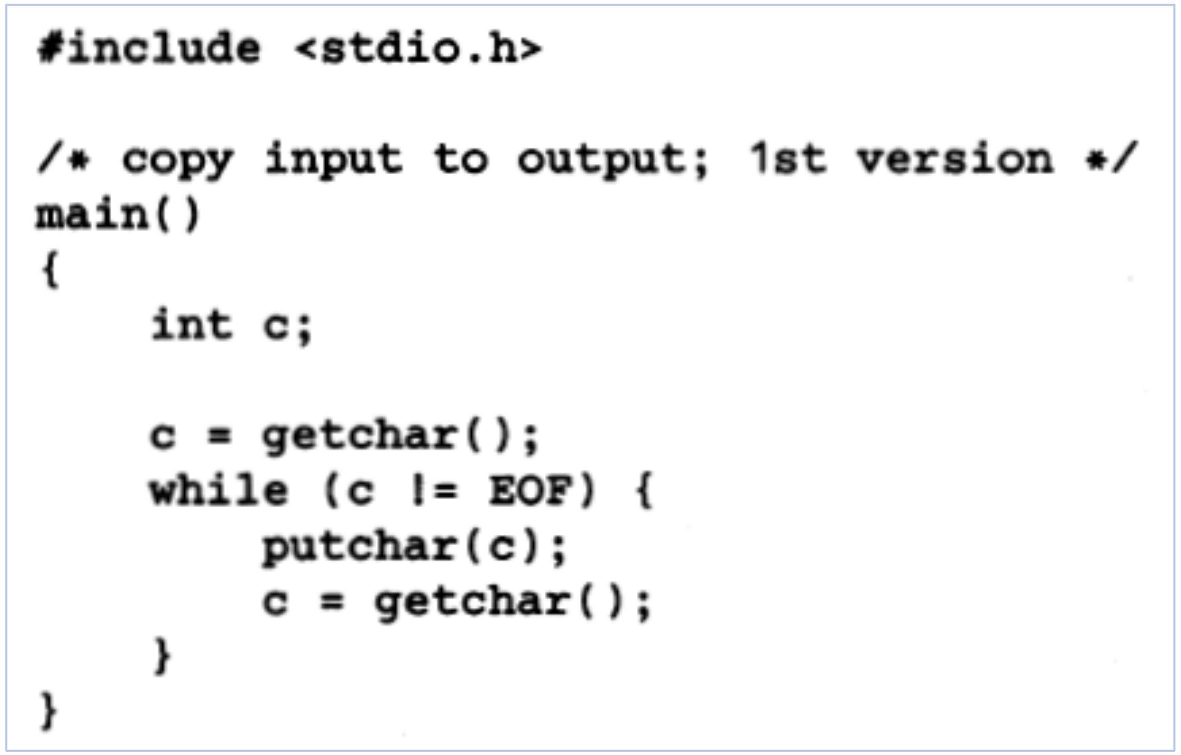
```c
/* print Fahrenheit-Celsius table */
main()
{
    int fahr;

    for (fahr = LOWER; fahr <= UPPER; fahr = fahr + STEP)
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
}
```

Character input, Output

- The standard library provides several functions for reading or writing one character at a time, of which `getchar()` and `putchar()` are the simplest.
- `getchar()` reads the next input character from a text stream and returns that as it its value.
- The function `putchar()` prints a character each time it is called.

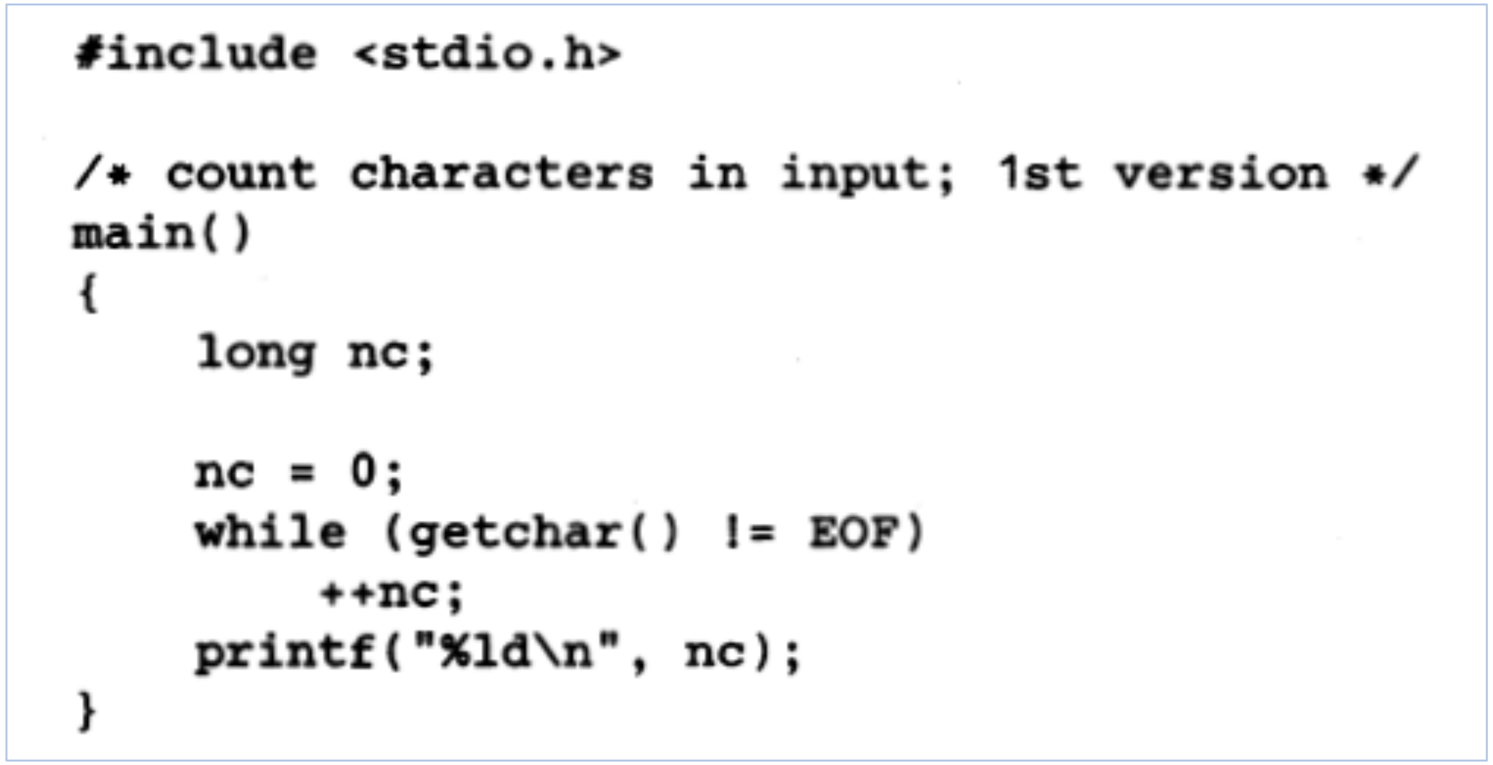
File Copying

- This is the program that copies its input to output one character at a time.
- EOF (End Of File) is an integer defined in <stdio.h>



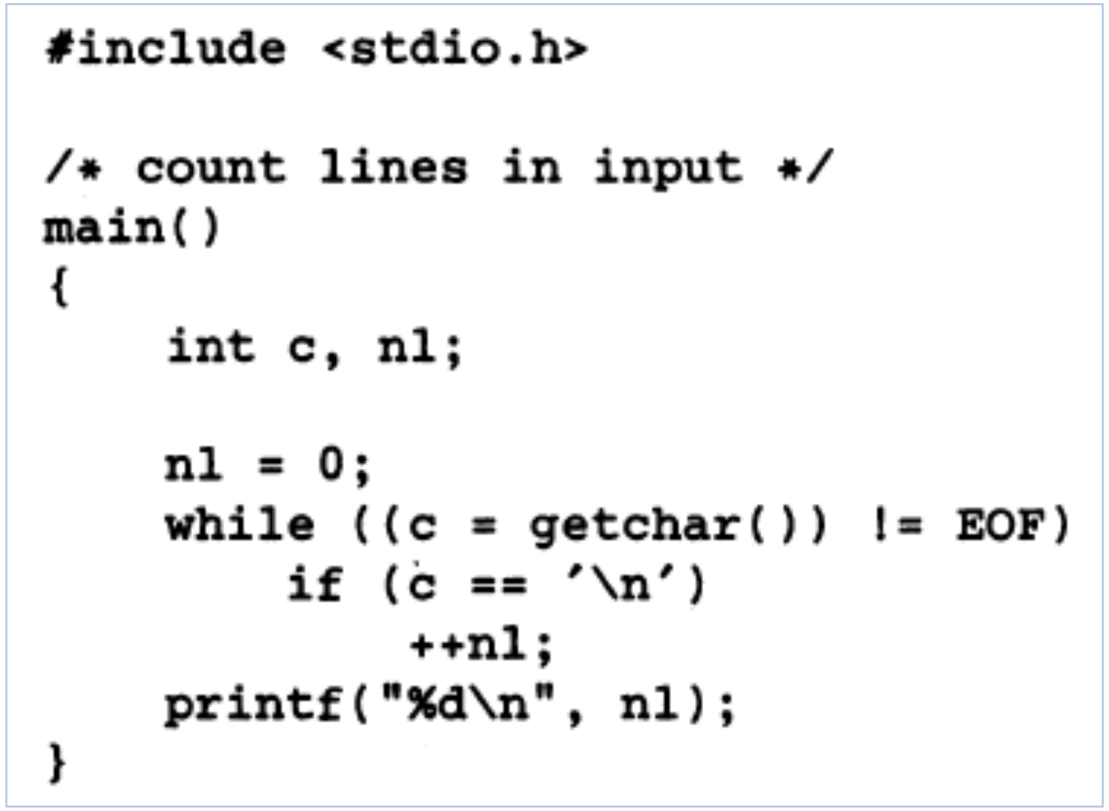
```
#include <stdio.h>

/* copy input to output; 1st version */
main()
{
    int c;

    c = getchar();
    while (c != EOF) {
        putchar(c);
        c = getchar();
    }
}
```

Character Counting



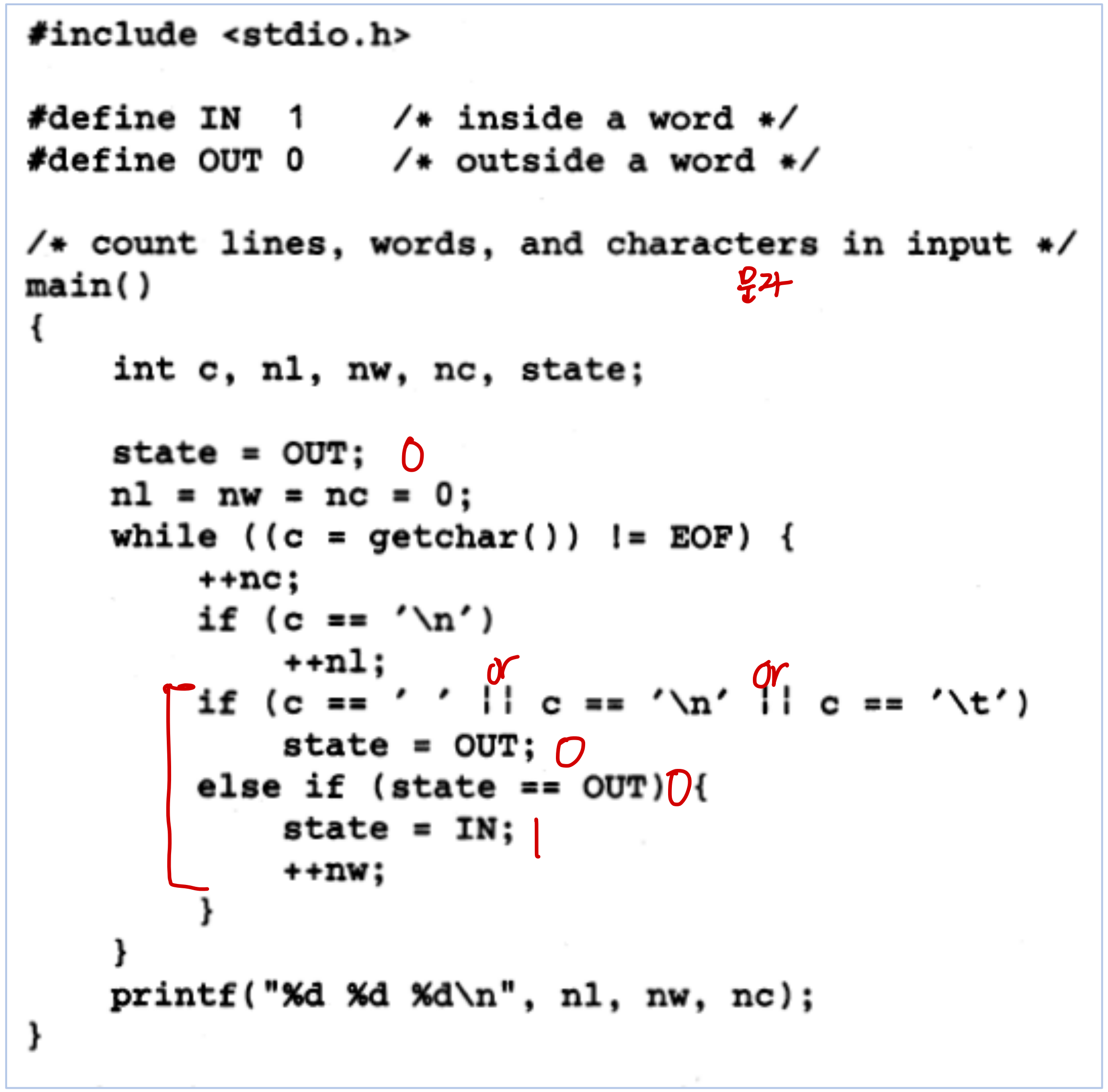
```
#include <stdio.h>

/* count characters in input; 1st version */
main()
{
    long nc;

    nc = 0;
    while (getchar() != EOF)
        ++nc;
    printf("%ld\n", nc);
}
```

Line Counting



```
#include <stdio.h>

/* count lines in input */
main()
{
    int c, nl;

    nl = 0;
    while ((c = getchar()) != EOF)
        if (c == ’\n’)
            ++nl;
    printf("%d\n", nl);
}
```

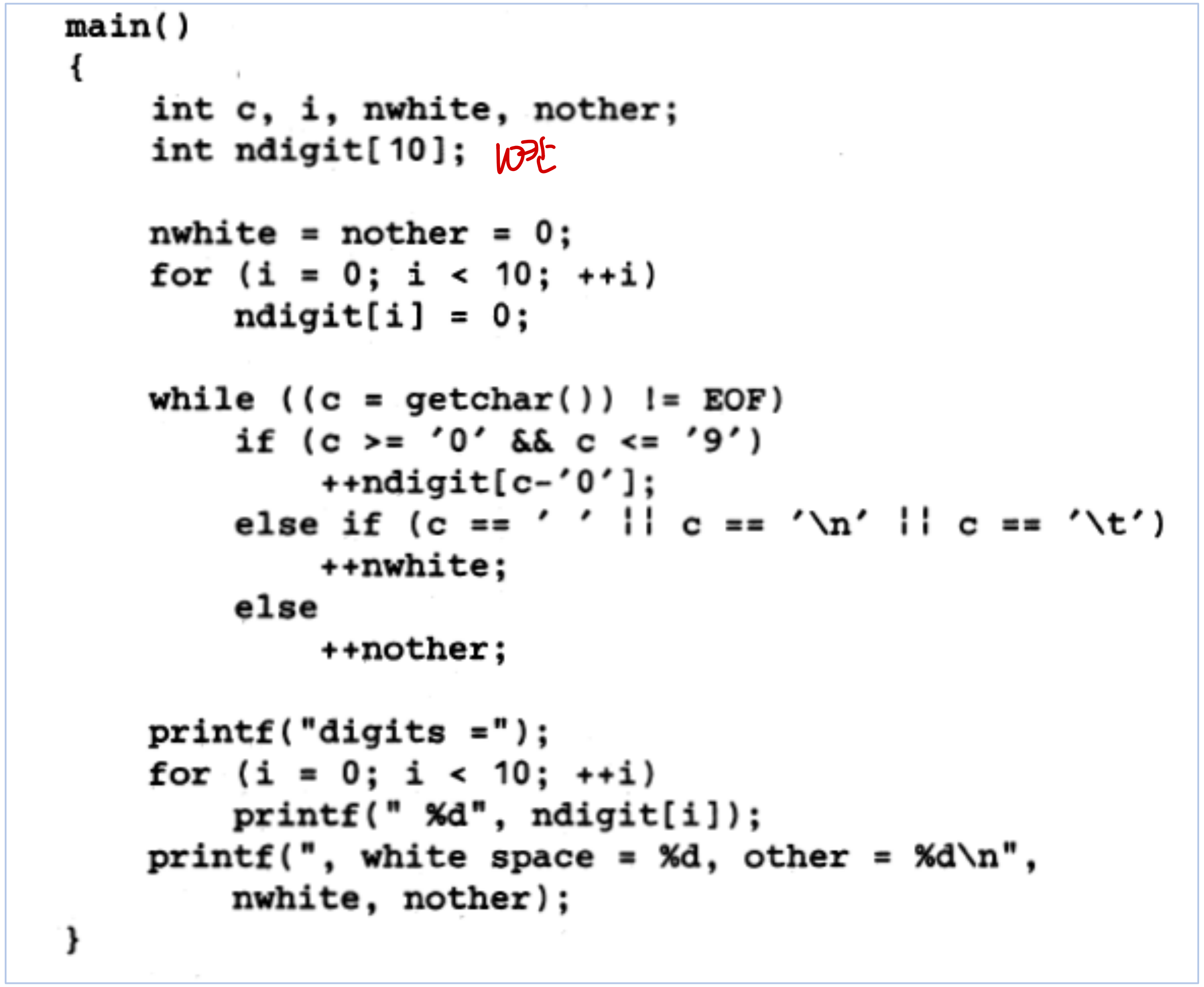
Word Counting



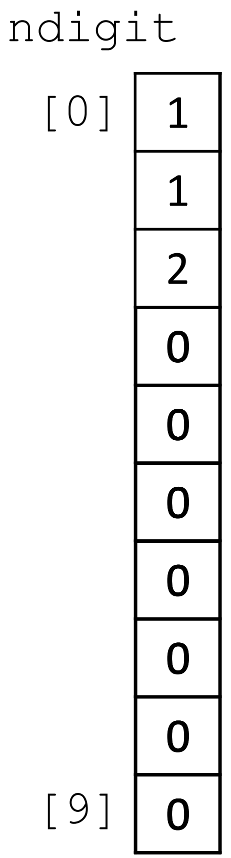
```
#include <stdio.h>

#define IN   1     /* inside a word */
#define OUT  0     /* outside a word */

/* count lines, words, and characters in input */
main()                                   문자
{
    int c, nl, nw, nc, state;

    state = OUT;  0
    nl = nw = nc = 0;
    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n')
            ++nl;            or              or
        if (c == ' ' || c == '\n' || c == '\t')
            state = OUT;  0
        else if (state == OUT)0{
            state = IN;  1
            ++nw;
        }
    }
    printf("%d %d %d\n", nl, nw, nc);
}
```

Arrays

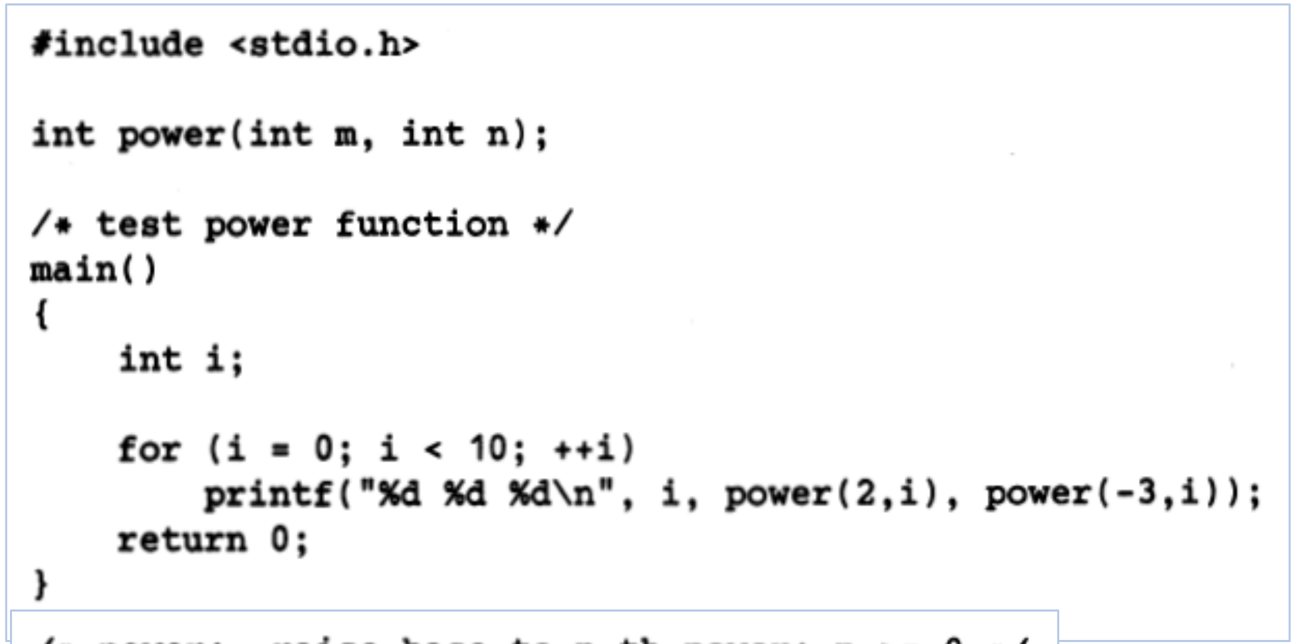
```
int a = 2012;
```



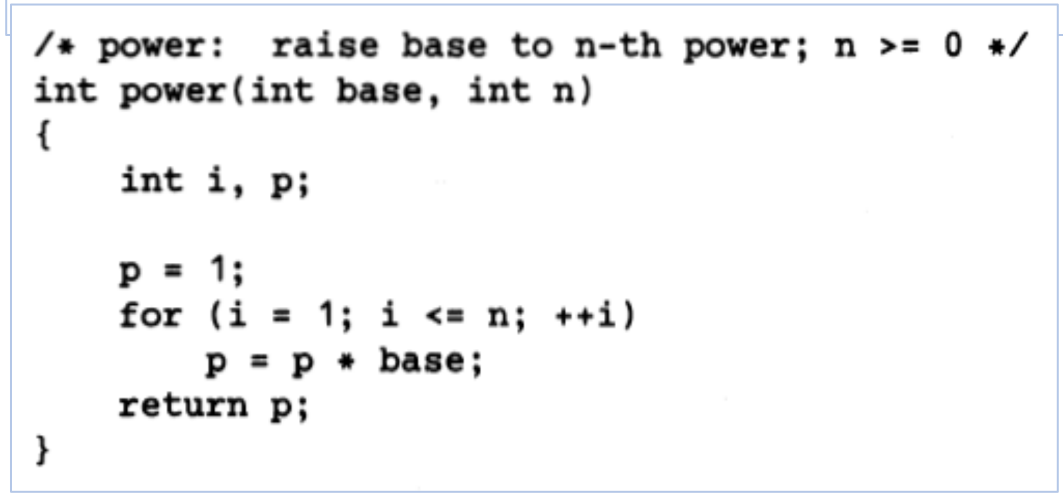
```
main()
{
    int c, i, nwhite, nother;
    int ndigit[10]; 10칸

    nwhite = nother = 0;
    for (i = 0; i < 10; ++i)
        ndigit[i] = 0;

    while ((c = getchar()) != EOF)
        if (c >= '0' && c <= '9')
            ++ndigit[c-'0'];
        else if (c == ' ' || c == '\n' || c == '\t')
            ++nwhite;
        else
            ++nother;

    printf("digits =");
    for (i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]);
    printf(", white space = %d, other = %d\n",
        nwhite, nother);
}
```



Functions



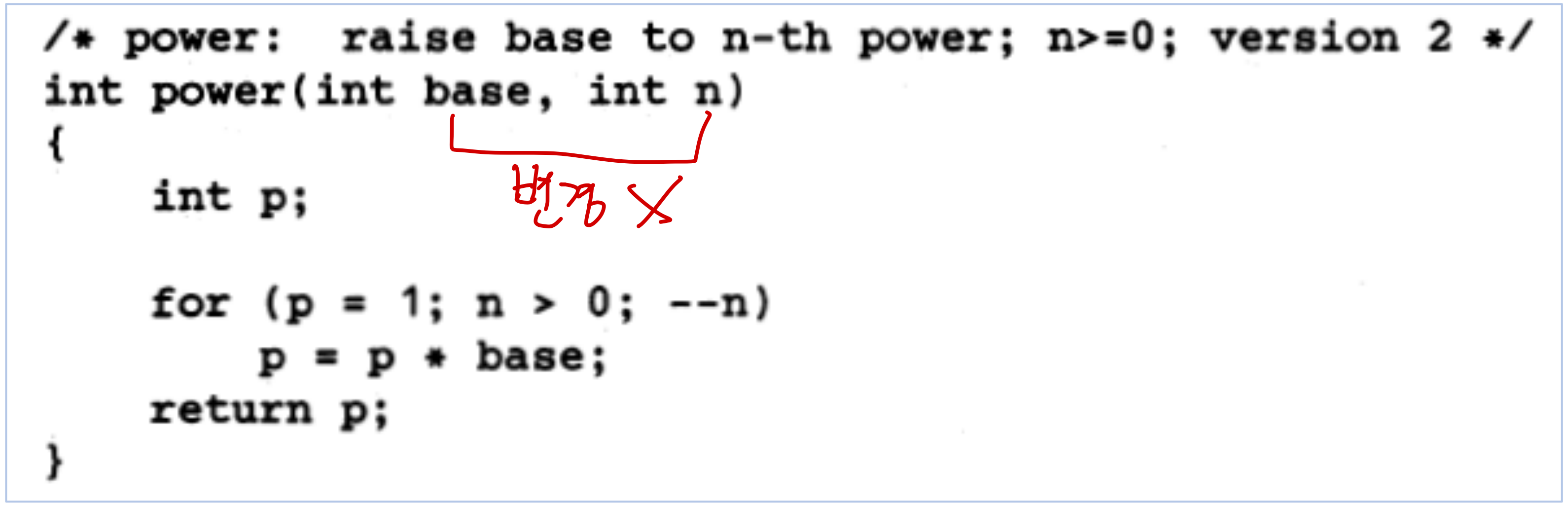
```
#include <stdio.h>

int power(int m, int n);

/* test power function */
main()
{
    int i;

    for (i = 0; i < 10; ++i)
        printf("%d %d %d\n", i, power(2,i), power(-3,i));
    return 0;
}
```



```
/* power:  raise base to n-th power; n >= 0 */
int power(int base, int n)
{
    int i, p;

    p = 1;
    for (i = 1; i <= n; ++i)
        p = p * base;
    return p;
}
```

Arguments – Call by value

- The main distinction is that in C the called function cannot directly alter a variable in the calling function; it can only alter its private, temporary copy.

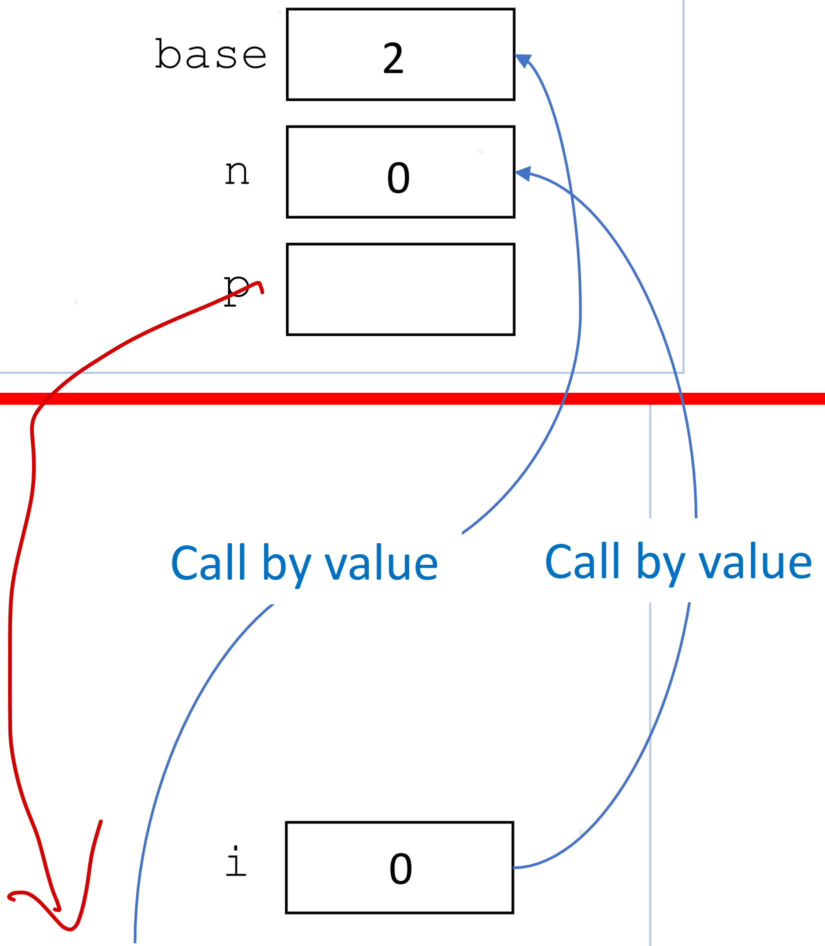


```
/* power:  raise base to n-th power; n>=0; version 2 */
int power(int base, int n)
{
    int p;

    for (p = 1; n > 0; --n)
        p = p * base;
    return p;
}
```

변경 X

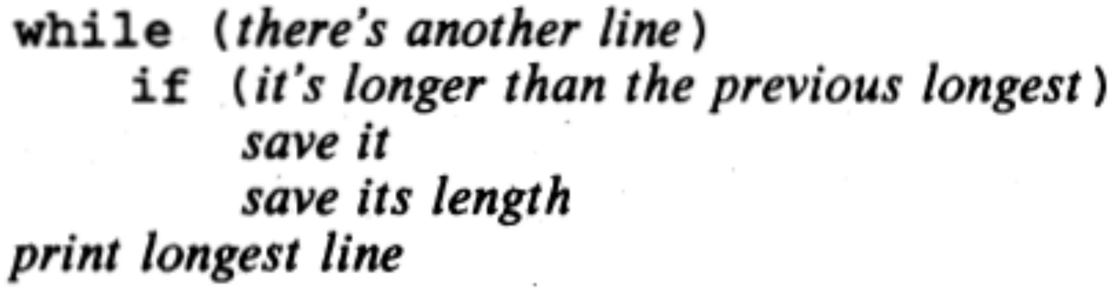
```c
/* power:  raise base to n-th power; n>=0; version 2 */
int power(int base, int n)
{
    int p;

    for (p = 1; n > 0; --n)
        p = p * base;
    return p;
}
```

```c
#include <stdio.h>

int power(int m, int n);

/* test power function */
main()
{
    int i;

    for (i = 0; i < 10; ++i)
        printf("%d %d %d\n", i, power(2,i), power(-3,i));
    return 0;
}
```



Character Arrays

- This program reads a set of text lines and prints the longest.



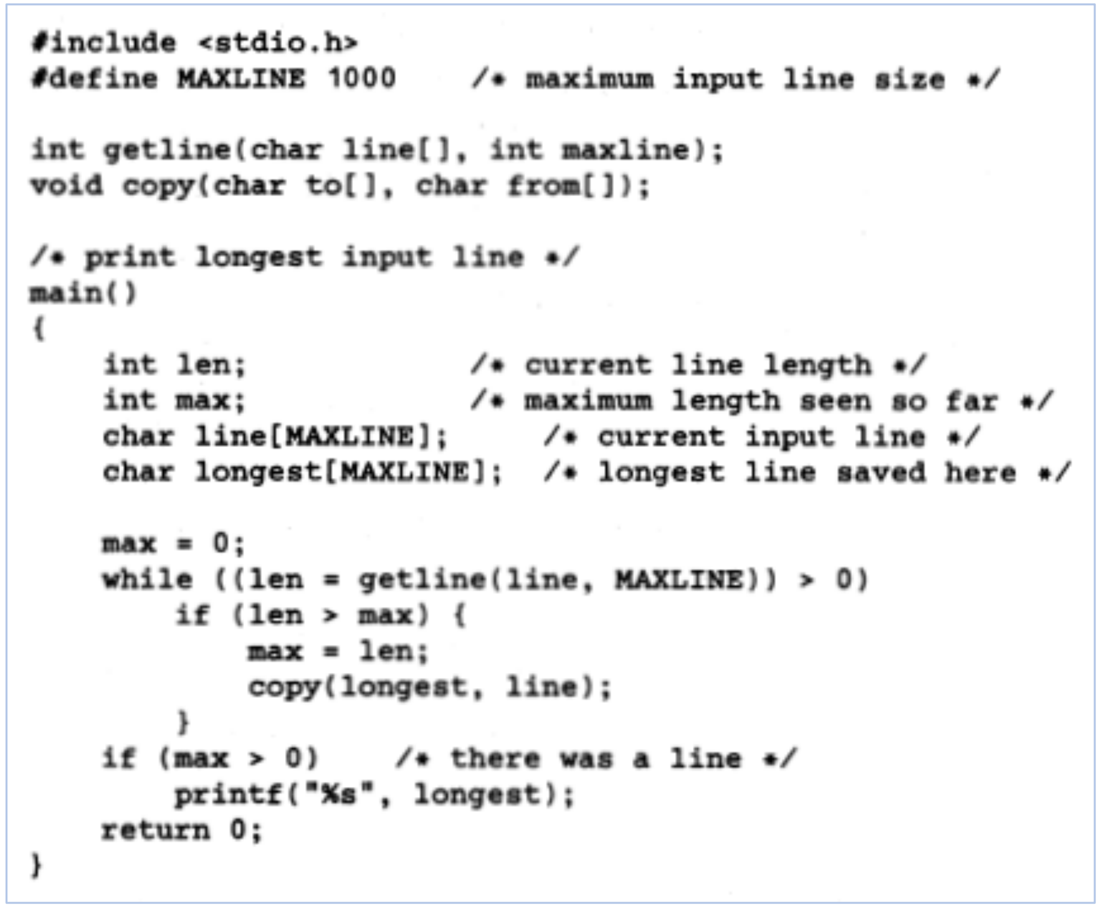
```
while (there's another line)
    if (it's longer than the previous longest)
        save it
        save its length
print longest line
```

1 Function `getline()` to fetch the next line input.

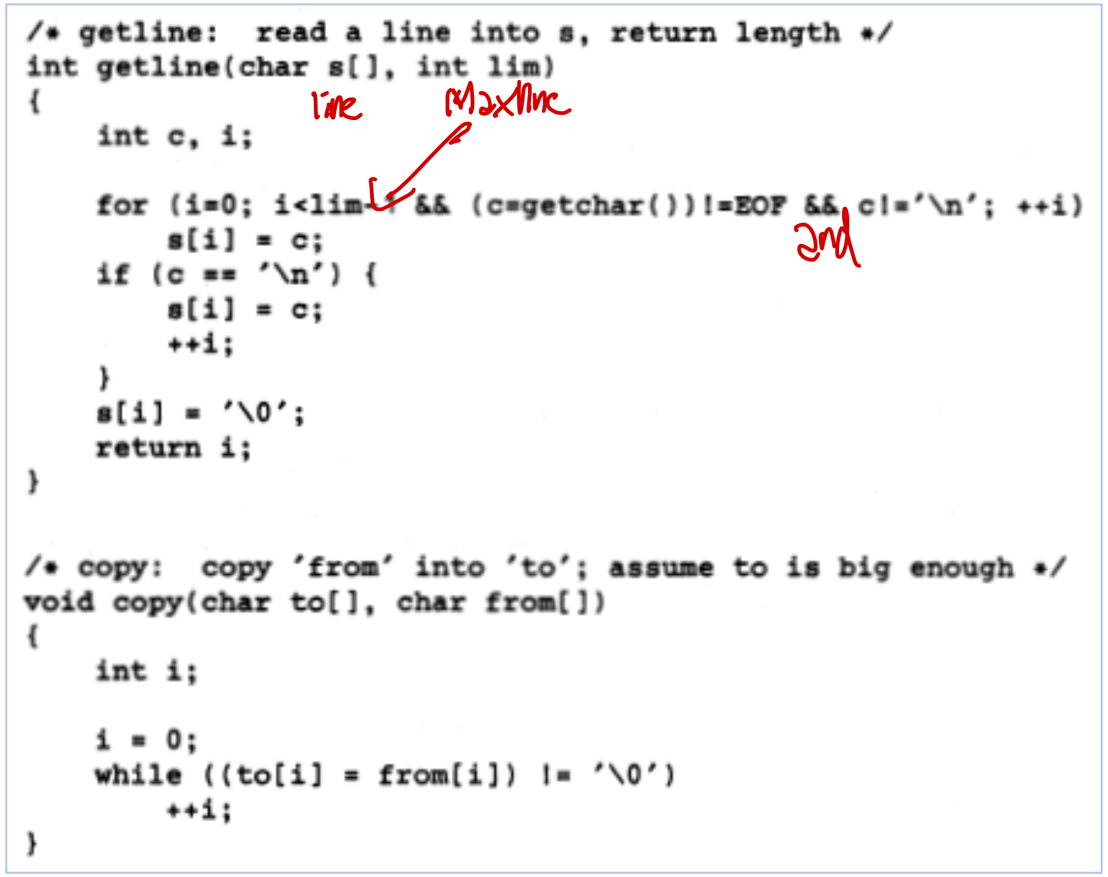
2 Function `copy()` to copy the new line to a safe place.

3 We need a main program to control `getline()` and `copy()`.

Character Arrays



```
#include <stdio.h>
#define MAXLINE 1000    /* maximum input line size */

int getline(char line[], int maxline);
void copy(char to[], char from[]);

/* print longest input line */
main()
{
    int len;              /* current line length */
    int max;              /* maximum length seen so far */
    char line[MAXLINE];      /* current input line */
    char longest[MAXLINE];  /* longest line saved here */

    max = 0;
    while ((len = getline(line, MAXLINE)) > 0)
        if (len > max) {
            max = len;
            copy(longest, line);
        }
    if (max > 0)    /* there was a line */
        printf("%s", longest);
    return 0;
}
```



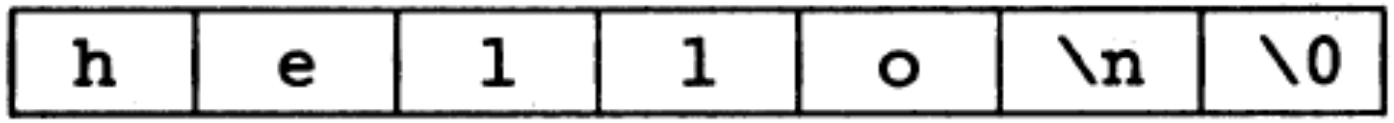
```
/* getline:  read a line into s, return length */
int getline(char s[], int lim)
{
    int c, i;

    for (i=0; i<lim-1 && (c=getchar())!=EOF && c!='\n'; ++i)
        s[i] = c;
    if (c == '\n') {
        s[i] = c;
        ++i;
    }
    s[i] = '\0';
    return i;
}

/* copy:  copy 'from' into 'to'; assume to is big enough */
void copy(char to[], char from[])
{
    int i;

    i = 0;
    while ((to[i] = from[i]) != '\0')
        ++i;
}
```

line Maxline

2nd

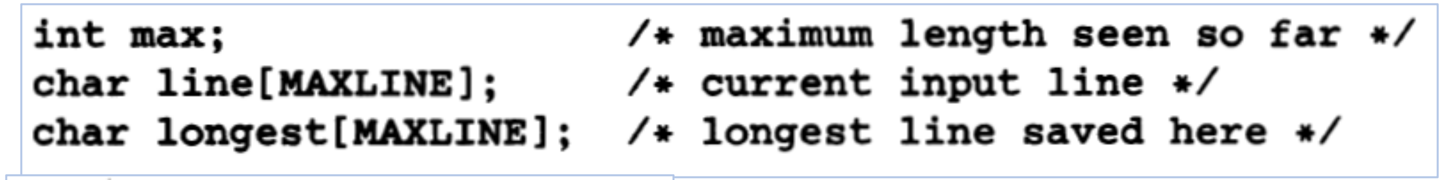
Character Arrays

- `getline()` puts the character ‘\0’ at the end of the array
- String constant “`hello\n`”

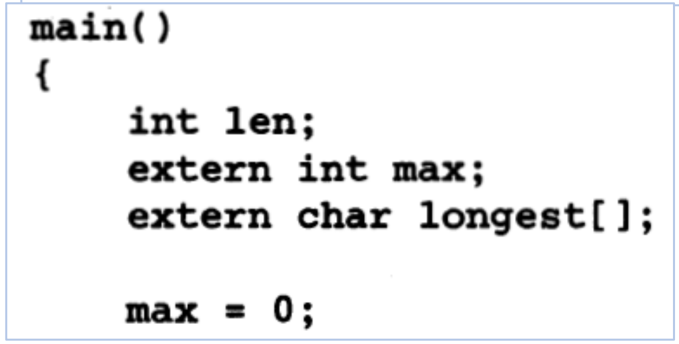


External Variables and Scope

- Automatic variables
  - auto
  - Defined inside of a function
- External variables
  - Defined outside of functions
  - Can be used by `extern` keyword



```
int max;                /* maximum length seen so far */
char line[MAXLINE];     /* current input line */
char longest[MAXLINE];  /* longest line saved here */
```



```
main()
{
    int len;
    extern int max;
    extern char longest[];

    max = 0;
```

정리

- Variables and Arithmetic Expressions
- The for statement
- Symbolic Constants
- Character Input and Output
- Arrays
- Functions
- Arguments-Call by Value
- Character Arrays
- External Variables and Scope

끝