Practice of Information Processing

（IMACU）

Forth lecture(part 1)：Previous Exercises

Makoto Hirota

Contents of this lecture

- Example answers of previous exercises
- Exercise：Algorithm using array
  - Sorting

Exercise 3-1: `if` statements

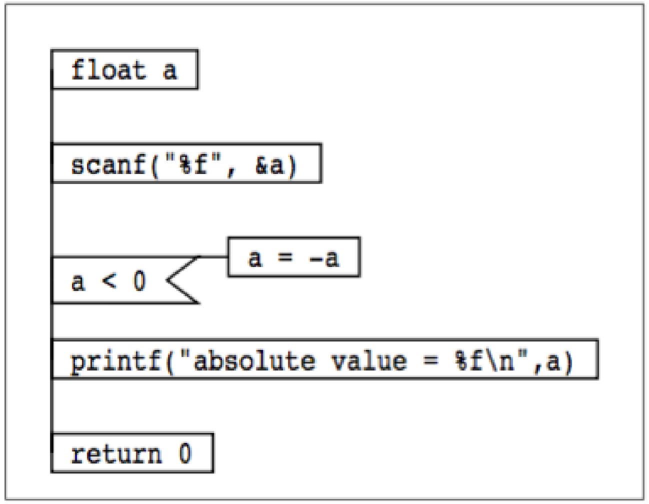
- abs.c:
  - Create a program that displays the absolute value of the entered real number (Hint: `if` statement)
- divisor.c
  - Create a program that determines whether B is a divisor of A for the two input integers A and B.
    - When B is a divisor of A, "B is a divisor of A." is displayed.
    - When B is not a divisor of A, "B is not a divisor of A." is displayed.

    (Hint: `if – else` statement)
- rank.c
  - Create a program that distinguishes A(excellent) / B(great) / C(good) / D(bad) from the entered points and displays them. Judgment should be made as follows.
    - The score is an integer from 0 to 100
    - 0 ~ 59 → D / 60 ~ 74 → C/ 75 ~ 84 → B / 85 ~ 100 →A

  (Hint: `if – if else – else` statement)

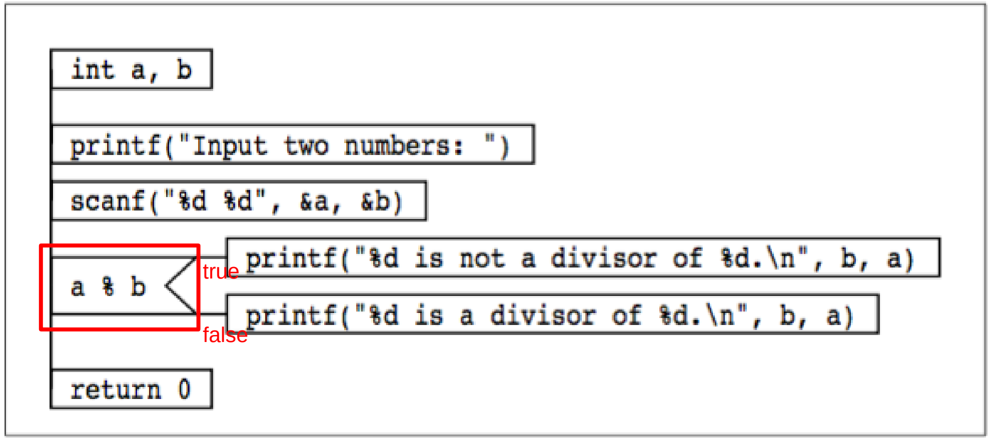
Exercise 3-1：Example of answers

abs.c



```
float a
scanf("%f", &a)
a < 0  →  a = -a
printf("absolute value = %f\n",a)
return 0
```

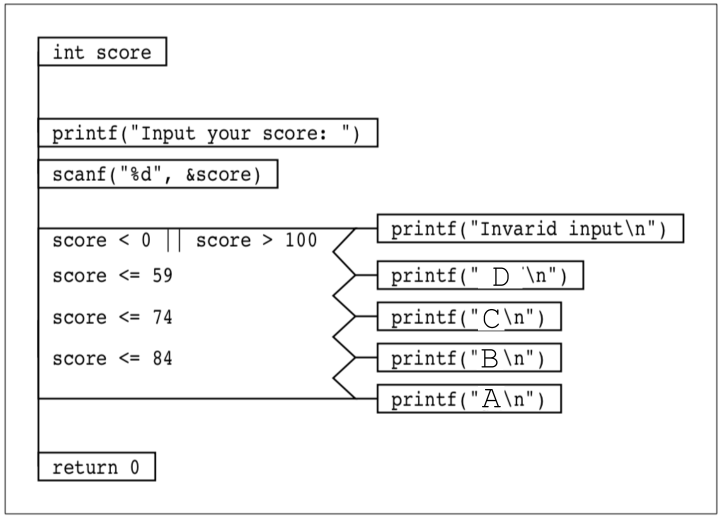
divisor.c



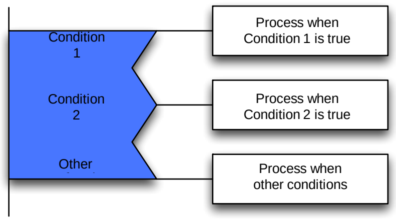
```
int a, b
printf("Input two numbers: ")
scanf("%d %d", &a, &b)
a % b  true  → printf("%d is not a divisor of %d.\n", b, a)
       false → printf("%d is a divisor of %d.\n", b, a)
return 0
```

Taking advantage of the fact that the if statement has a remainder of 0 and if (0) is "false",
Residual 0 → "False" processing,
other than 0 → "True" processing

rank.c

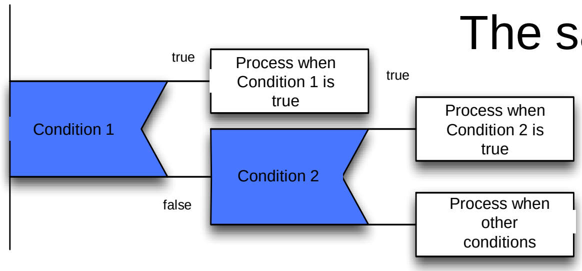


```
int score
printf("Input your score: ")
scanf("%d", &score)
score < 0 || score > 100  → printf("Invarid input\n")
score <= 59               → printf(" D \n")
score <= 74               → printf("C\n")
score <= 84               → printf("B\n")
                          → printf("A\n")
return 0
```



Condition 1
Condition 2
Other
Process when Condition 1 is true
Process when Condition 2 is true
Process when other conditions

The same meaning



true
false
Condition 1
Process when Condition 1 is true
true
Condition 2
Process when Condition 2 is true
Process when other conditions

Exercise 3-2: `switch` statement

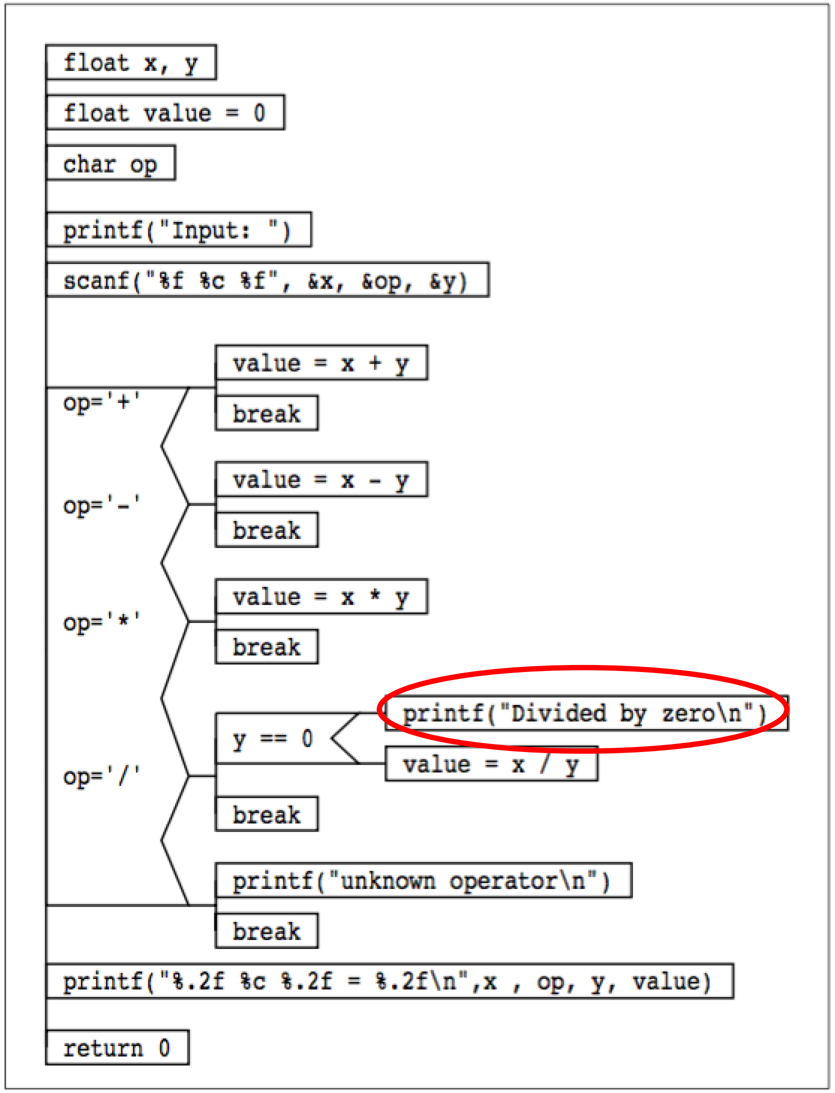
- operator.c
  - Create a calculator program that performs the four arithmetic operations (+,-, *, /) of the two input real numbers, referring to sample.c in the first lecture.
    - Input and output examples:
      - Input “12 + 3” → Display: “15.00”
      - Input “7.5 -10” → Display: “-2.50”
      - Input "2 * 5" → Display: "10.00"
      - Input “10 / 2.5” → Display: “4.00”
    - Tips
      - The operators of the four arithmetic operations are read as char type characters, and the processing is switched as the value of the expression in the switch statement.
      - To enter 3 inputs from the console

```
– scanf(”%? %? %?”, &x, &op, &y);
          ⇧  ⇧  ⇧
```

What should I choose for the format specification?

The variable op is of type char
‘+’, ‘-’, ‘*’, ‘/’ are included

Exercise 3-2



```
float x, y
float value = 0
char op
printf("Input: ")
scanf("%f %c %f", &x, &op, &y)
op='+'
    value = x + y
    break
op='-'
    value = x - y
    break
op='*'
    value = x * y
    break
op='/'
    y == 0
        printf("Divided by zero\n")
        value = x / y
    break
    printf("unknown operator\n")
    break
printf("%.2f %c %.2f = %.2f\n",x , op, y, value)
return 0
```

```
#include <stdio.h>

int main() {
    /**** variable declaration****/
    float x,y;
    float value = 0;
    char op;

    /**** processing contents****/
    printf(“Input :");
    scanf("%f %c %f",&x, &op, &y); /*** formatted ***/

    switch(op){
        case '+';
            value = x + y;
            break;
        case '-';
            value = x - y;
            break;
        case '*';
            value = x + y;
            break;
        case '/';
            if(y == 0){
                printf("Divided by zero¥n");
            }
            else{
                value = x / y;
            }
            break;
        default:
            printf("unknown operator¥n");
            break;
    }
    printf(“%.2f %c %.2f = %.2f¥n",x , op, y, value);
    return 0;
}
```

* Supplement
Whitespace characters (spaces, line breaks, tabs) cannot be entered with scanf (). This space is only used as a delimiter.
It holds even if you enter "12 + 3↵" with
scanf (“% f% c% f”, & x, & op, & y);
Or You can also enter by pressing the enter key three times
“12 ↵
+ ↵
3 ↵ ”

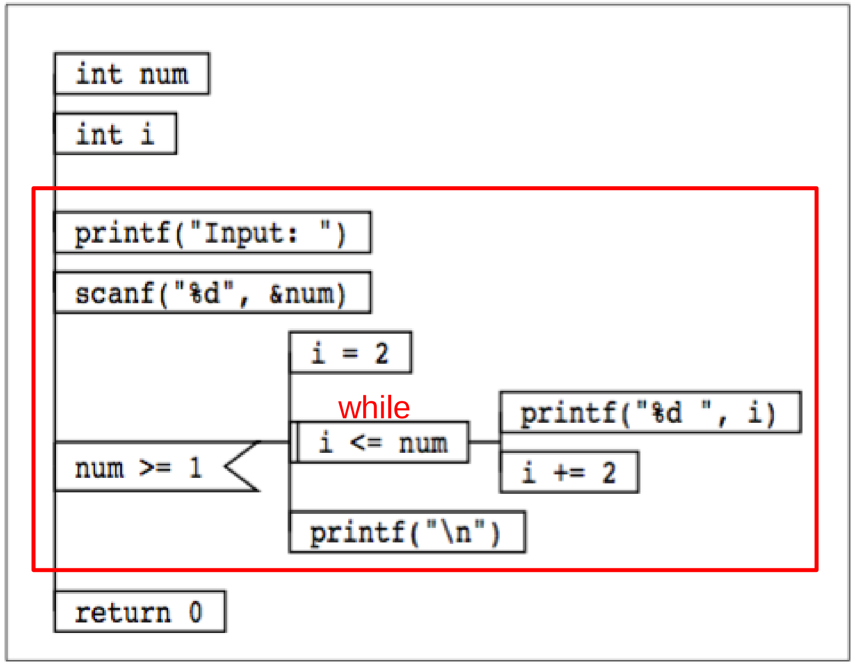
Exercise 3-3: `while` statement

- even_list.c
  - Create a program that displays all positive even numbers less than or equal to a for the input positive integer a.
  - Example: Input: 13 → Output: “2 4 6 8 10 12”
  - Implement using a while statement
- even_list_loop.c
  - Modify odd_list1.c to create a program that allows you to repeatedly enter positive integers from the keyboard.
  - When terminating the program, a value of 0 (zero) or less shall be entered.

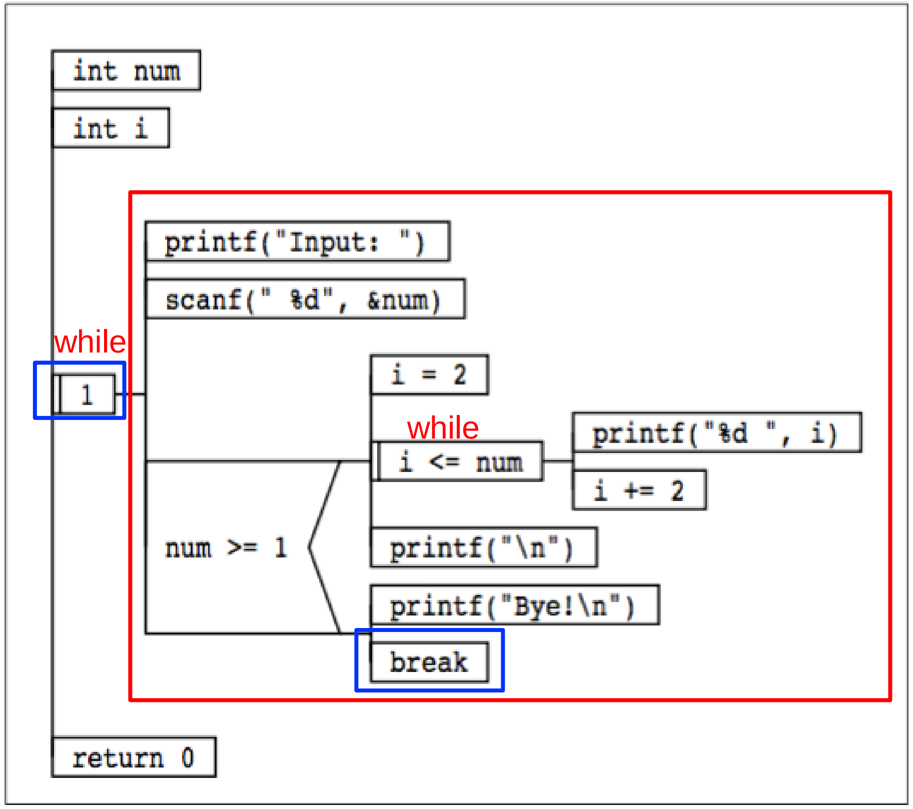
(Note) If you cannot terminate the program, you can forcibly terminate it by pressing the ctrl key + ‘c’.

Exercise 3-3: Example of answers

even_list.c



even_list_loop.c



Exercise 3-4: Input to a one-dimensional array array_input.c

- Create a program array_input.c that gets 5 integers into array `a` from the keyboard, and that outputs the values of each array element and the sum.

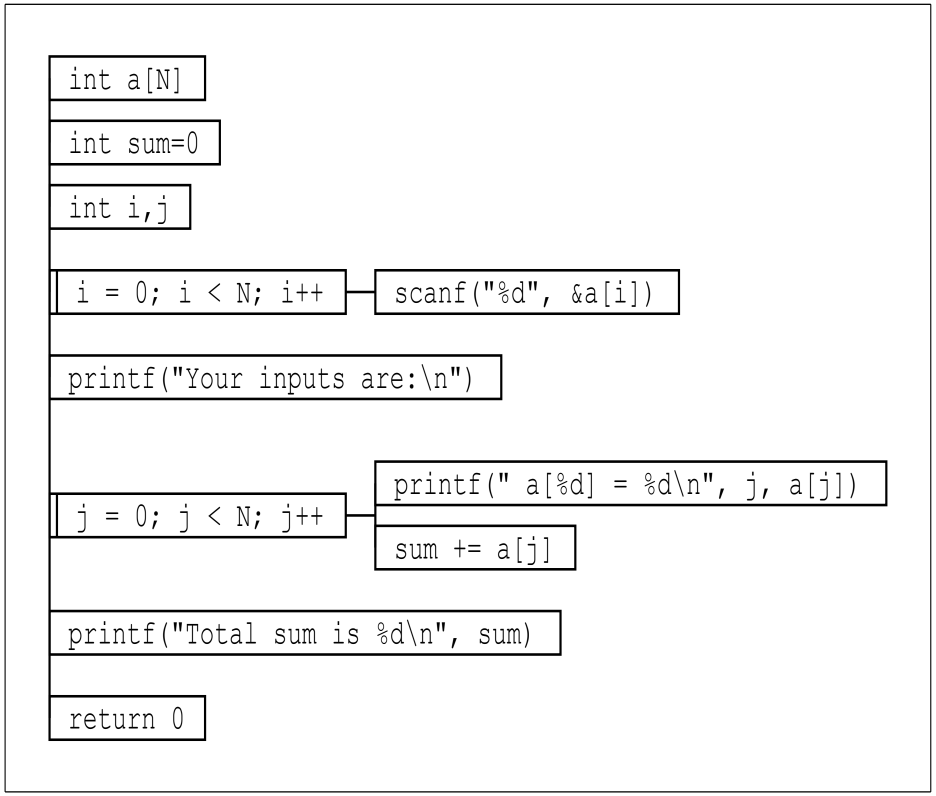
  <example>

```
$ ./array_input
10 ↵ (Enter key)
20 ↵
30 ↵
40 ↵
50 ↵
Your inputs are:
a[0]=10
a[1]=20
a[2]=30
a[3]=40
a[4]=50
Total sum is 150
```

Example answer： array_input.c

Macro definition

#define N 5



In this example, the variables `i` and `j` are used properly for the two `for` statements, but they can be the same. (Because it is initialized first in the for statement)

```c
#include <stdio.h>

#define N 5 /* Macro definition */

int main(){
    /**** variable declaration****/
    int a[N];
    int sum=0;
    int i, j;

    /**** processing contents****/
    /* initialization of array */
    for (i = 0; i < N; i++){
        scanf("%d", &a[i]);
    }

    printf("Your inputs are:¥n");

    for (j = 0; j < N; j++){
        printf("a[%d] = %d¥n", j, a[j]);
        sum += a[j]; /*sum of array*/
    }

    /* display of sum*/
    printf("Total sum is %d:¥n", sum);

    return 0;
}
```

Exercise 3-5: Turn over a string reverse.c

- Create a program reverse.c that converts the character string input from the keyboard in reverse order from the end and displays it.

  <example>

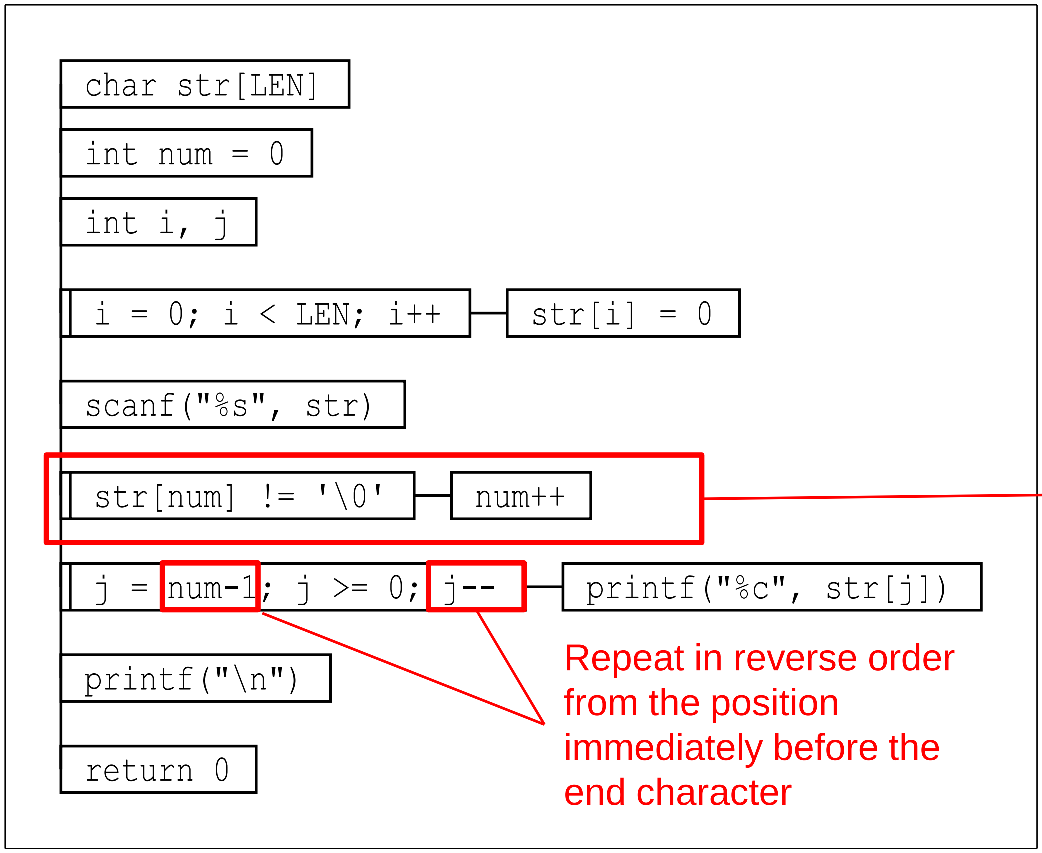
```
$ ./reverse
hello ↵ (Enterキー)
olleh
```

- If you can, make reverse.c works even when the string includes spaces. (Hint: string2.c)

  Ex. “Hello World” → “dlroW olleH”

Example answer：reverse.c

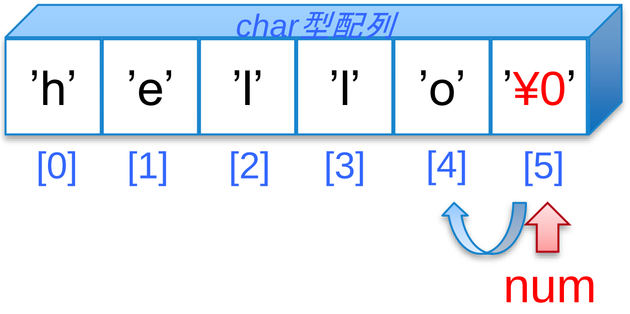
Macro definition
#define LEN 100 （ OK if it is larger enough than the word you enter ）



```
char str[LEN]
int num = 0
int i, j
i = 0; i < LEN; i++ ── str[i] = 0
scanf("%s", str)
str[num] != '\0' ── num++
j = num-1; j >= 0; j-- ── printf("%c", str[j])
printf("\n")
return 0
```

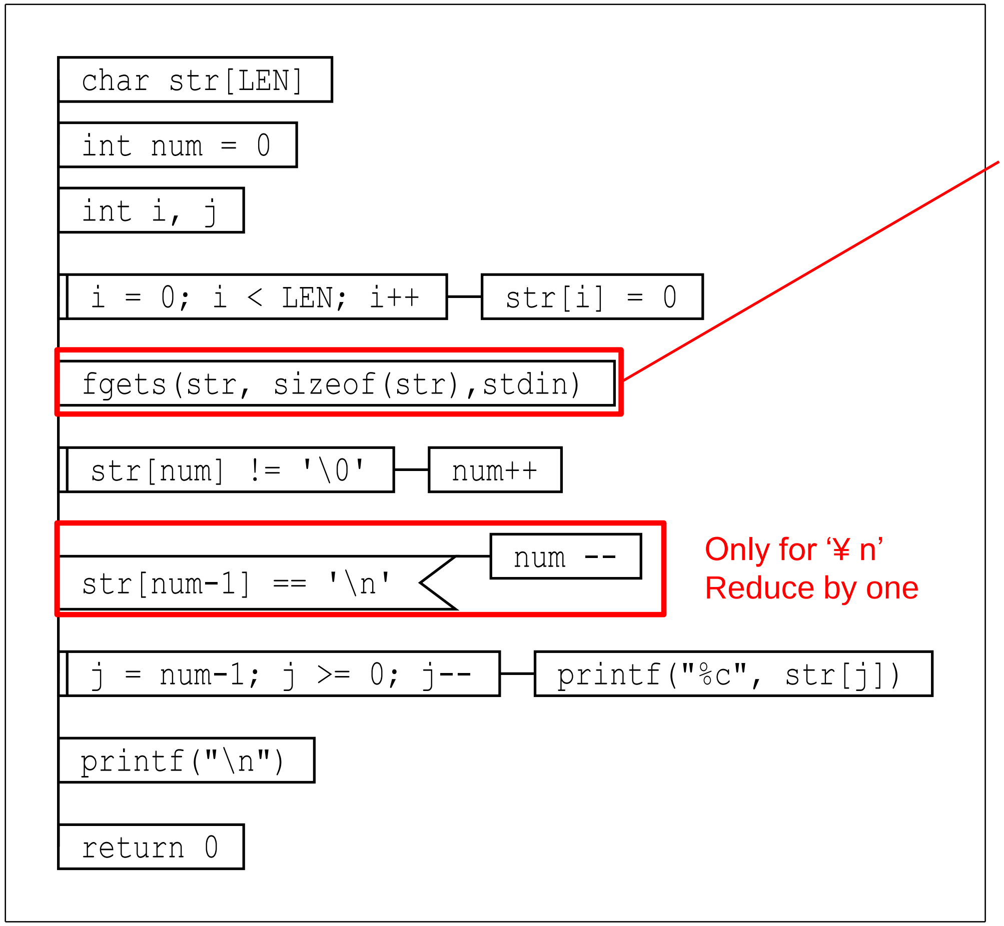
Repeat in reverse order from the position immediately before the end character

while statement
If not a terminating character, increment num
= Find the position of the terminating character



Make reverse.c deal with blank characters

Macro definition
#define LEN 100



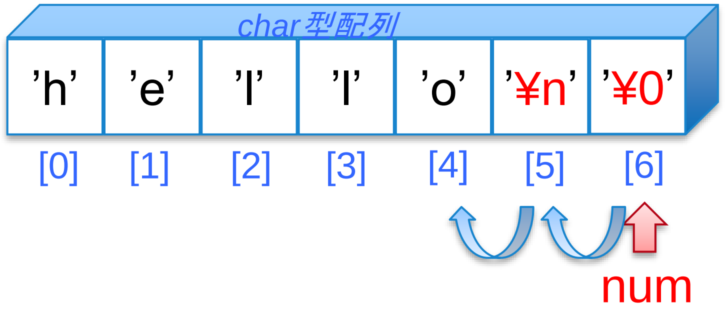
```
char str[LEN]
int num = 0
int i, j
i = 0; i < LEN; i++ ── str[i] = 0
fgets(str, sizeof(str),stdin)
str[num] != '\0' ── num++
str[num-1] == '\n' < num --
j = num-1; j >= 0; j-- ── printf("%c", str[j])
printf("\n")
return 0
```

Use the function fgets instead of the function scanf

Only for '¥ n'
Reduce by one

fgets function
Read one line from the target file (stdin in this example) up to the specified size).

Note that the newline character '¥ n' is also included at the end.



Ex. reverse.c

```
#include <stdio.h>

#define LEN 100 /* Macro definition */

int main() {
    /**** variable declaration****/
    char str[LEN];
    int num=0;
    int i, j;

    /**** processing contents****/
    /* initialization of array */
    for (i = 0; i < LEN; i++){
        str[i] = 0;
    }

    /* input of strings */
    scanf("%s",str);

    /* find terminal character position*/
    while (str[num] != '¥0'){
        num++;
    }
    /* reverse display*/
    for (j = num-1; j>= 0; j--){
        printf("%c", str[j]);
    }

    printf("¥n"); /* change line at last*/

    return 0;
}
```

'blank' char

```
#include <stdio.h>

#define LEN 100 /* Macro definition */

int main() {
    /**** variable declaration****/
    char str[LEN];
    int num=0;
    int i, j;

    /**** processing contents****/
    /* initialization of array */
    for (i = 0; i < LEN; i++){
        str[i] = 0;
    }

    /* input of strings (without scanf)*/
    fgets(str, sizeof(str), stdin);

    /* find terminal character position*/
    while (str[num] != '¥0'){
        num++;
    }

    /* fgets take 'return' char and in the
    case of 'return' char appears before
    the terminal char, decrement num by 1 */
    if (str[num-1] == '¥n'){
        num --;
    }

    /* reverse display */
    for (j = num-1; j>= 0; j--){
        printf("%c", str[j]);
    }

    printf("¥n"); /* change line at last*/

    return 0;
}
```

Exercise 3-6: Matrix calculation for multidimensional arrays

- Create a program matrix_multiple.c that finds the product C of the 4-by-3 matrix A and the 3-by-4 matrix B.
- C = A × B

$$A=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}$$

$$B=\begin{pmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 1 & 2 & 3 & 4\end{pmatrix}$$

The matrix values have already been entered in the sample source matrix_multiple.c, so please use that.

Points:
1. Be aware of which is the row and which is the column when coding

$$\texttt{int A}\underbrace{\texttt{[4]}}_{\text{row}}\underbrace{\texttt{[3]}}_{\text{column}}$$

2. Matrix subscripts start at 0

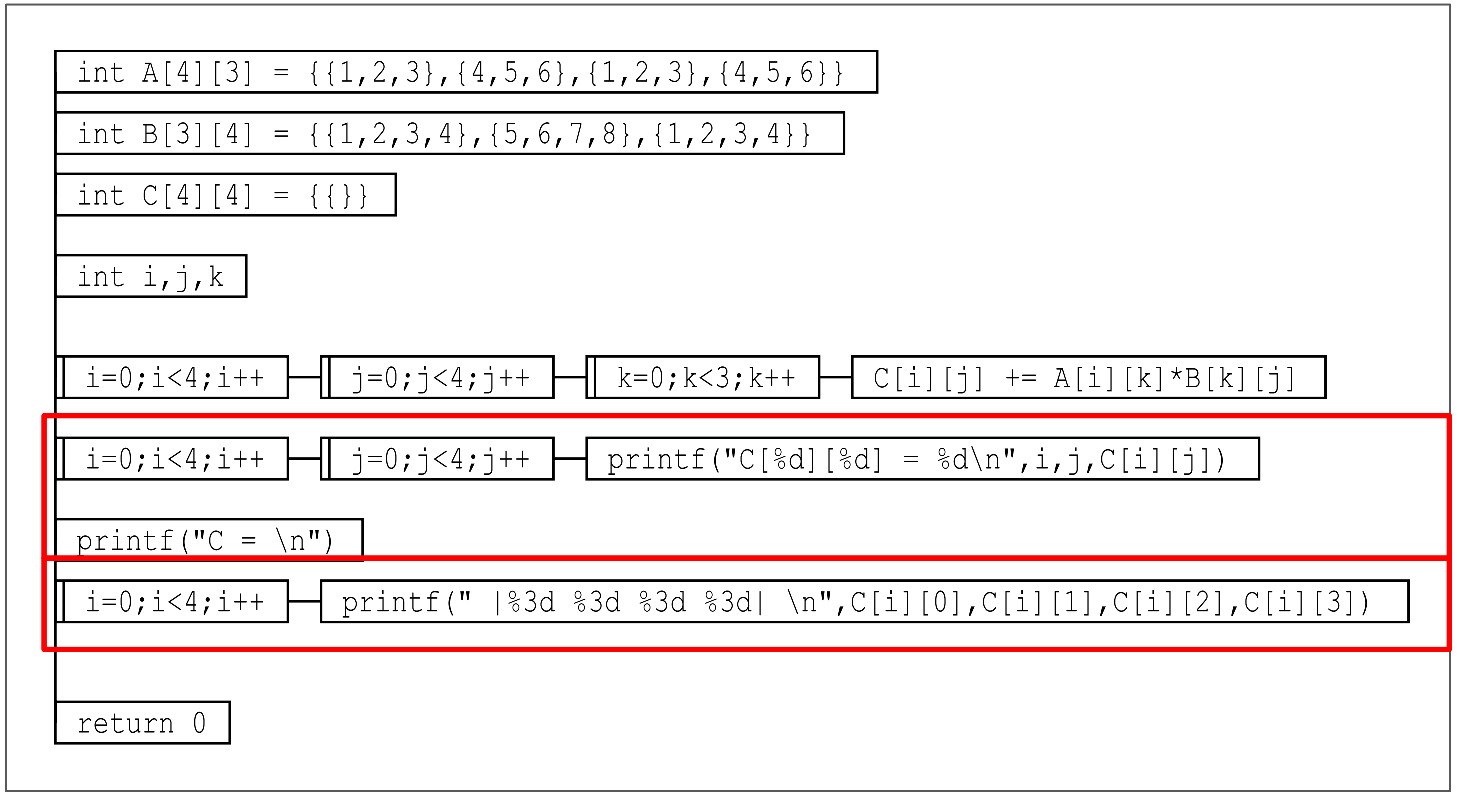
<example>

```
$ ./matrix_multiple
C[0][0] = 14
C[0][1] = 20
C[0][2] = 26
C[0][3] = 32
C[1][0] = 35
C[1][1] = 50
C[1][2] = 65
C[1][3] = 80
C[2][0] = 14
C[2][1] = 20
C[2][2] = 26
C[2][3] = 32
C[3][0] = 35
C[3][1] = 50
C[3][2] = 65
C[3][3] = 80
```

```
$ ./matrix_multiple
C =
   | 14  20  26  32|
   | 35  50  65  80|
   | 14  20  26  32|
   | 35  50  65  80|
```

Tips：matrix_multiple.c

PAD expression



```
int A[4][3] = {{1,2,3},{4,5,6},{1,2,3},{4,5,6}}
int B[3][4] = {{1,2,3,4},{5,6,7,8},{1,2,3,4}}
int C[4][4] = {{}}

int i,j,k

i=0;i<4;i++ ── j=0;j<4;j++ ── k=0;k<3;k++ ── C[i][j] += A[i][k]*B[k][j]

i=0;i<4;i++ ── j=0;j<4;j++ ── printf("C[%d][%d] = %d\n",i,j,C[i][j])

printf("C = \n")
i=0;i<4;i++ ── printf(" |%3d %3d %3d %3d| \n",C[i][0],C[i][1],C[i][2],C[i][3])

return 0
```

①When outputting each component individually

②When outputting in a matrix style

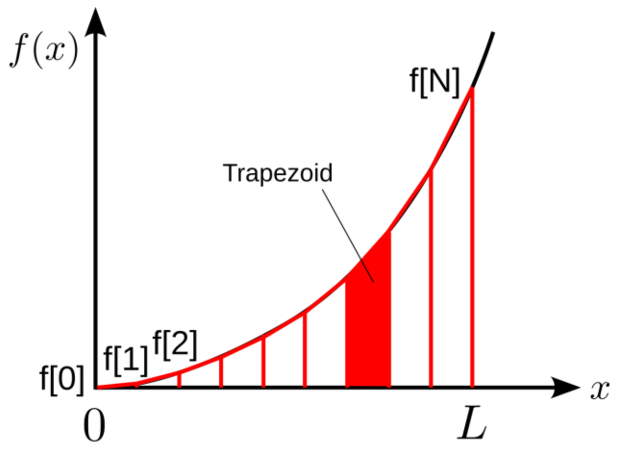
Either one of ① and ② is sufficient

Example answer : matrix_multiple.c

```c
#include <stdio.h>

int main() {
    /**** variable declaration****/
    int A[4][3]={{1,2,3},{4,5,6},{1,2,3},{4,5,6}};
    int B[3][4]={{1,2,3,4},{5,6,7,8},{1,2,3,4}};
    int C[4][4]={{}};

    int i,j,k;

    /**** processing contents****/
    for (i=0;i<4;i++){
        for (j=0;j<4;j++){
            for (k=0;k<3;k++){
                 C[i][j] += A[i][k]*B[k][j];
            }
        }
    }

    printf("C = ¥n");
    for (i=0;i<4;i++){
        printf("|%3d  %3d  %3d  %3d| ¥n",C[i][0],C[i][1],C[i][2],C[i][3]);
    }

    }
    return 0;
}
```

Exercise 3-7: Numerical integration trapez.c

- Create a program trapez.c in which
  1. The values of a function $f(x) = x^2$ at $x = iL/N$ $(i = 0,1,2,\dots,N)$ are stored in an array f[0], f[1], f[2],…, f[N]. (Choose a real number $L > 0$ and an integer $N > 0$ freely)
  2. The integral $\int_0^L f(x)dx$ is calculated approximately by using the trapezoidal rule.



- Check the error of the result in comparison with the analytical answer.

Exercise 3-7: trapez.c model answer

```c
#include <stdio.h>
#define N 100

int main()
{
    int i;
    double x,f[N+1],ans;
    double L=1.0;
    double dx=L/N; // Grid width

    for(i=0;i<=N;i++){
        x=i*dx;
        f[i]=x*x; // function
    }

    ans=0.0;
    for(i=0;i<N;i++){
        // Area of a trapezoid
        ans+=0.5*(f[i]+f[i+1])*dx;
    }

    printf("%d %30.26f %10.6e¥n",
           N,ans,ans-L*L/3.0);
    return 0;
}
```

Be careful.

Better answer

```c
    ans=0.5*f[0];
    for(i=1;i<N;i++){
        ans+=f[i];
    }
    ans+=0.5*f[N];
    ans*=dx;
```

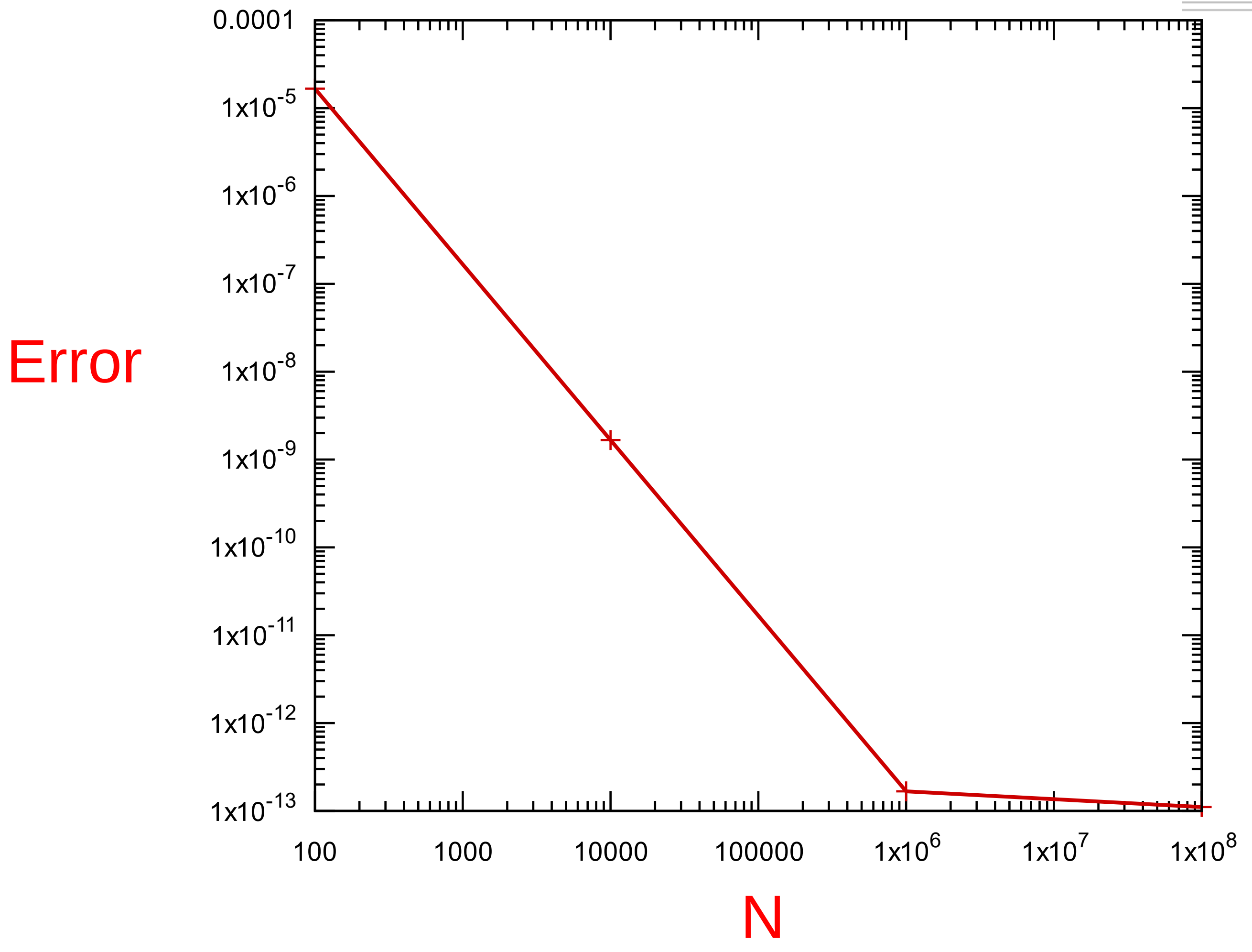
Calculation cost is reduced!

Output:

```
 100   0.33335000000000003517186542 1.666667e-05
```

N Answer Error

Exercise 3-7: Numerical error



- The error decreases in proportion to $1/N^2$.
- However, the decrease stops when $N > 10^6$. (round-off error)

Recall that 10^{-16} is the maximum accuracy of double precision.

Practice of Information Processing

（IMACU）

Forth lecture(part 2)：Exercise using array

Makoto Hirota

Contents of this lecture

- Example answer of previous exercise
- Exercise：Algorithm using array
  - Sorting
  - Mathematical function

Exercise：Algorithm using array

- Sort
  - Sorting the elements of a given array in ascending (or descending) order according to a certain rule.
  - As the size of the array increases, the amount of calculation becomes enormous unless it is sorted efficiently, so various algorithms have been proposed.
  - The calculation cost of a typical algorithm is the order of O (n log n) at best and O (n^2) at worst.
- Sort algorithm
  - bubble sort
  - selection sort
  - Insertion sort
  - shell sort
  - quick sort
  - counting sort

  ･･･（and others）

- Ease of implementation
- Average calculation cost
- Worst-case calculation cost
- Memory usage etc.

will change depending on algorithm

Reference
Wikipedia: “Sorting algorithm”

Sort(1) Bubble sort: sort_bubble.c

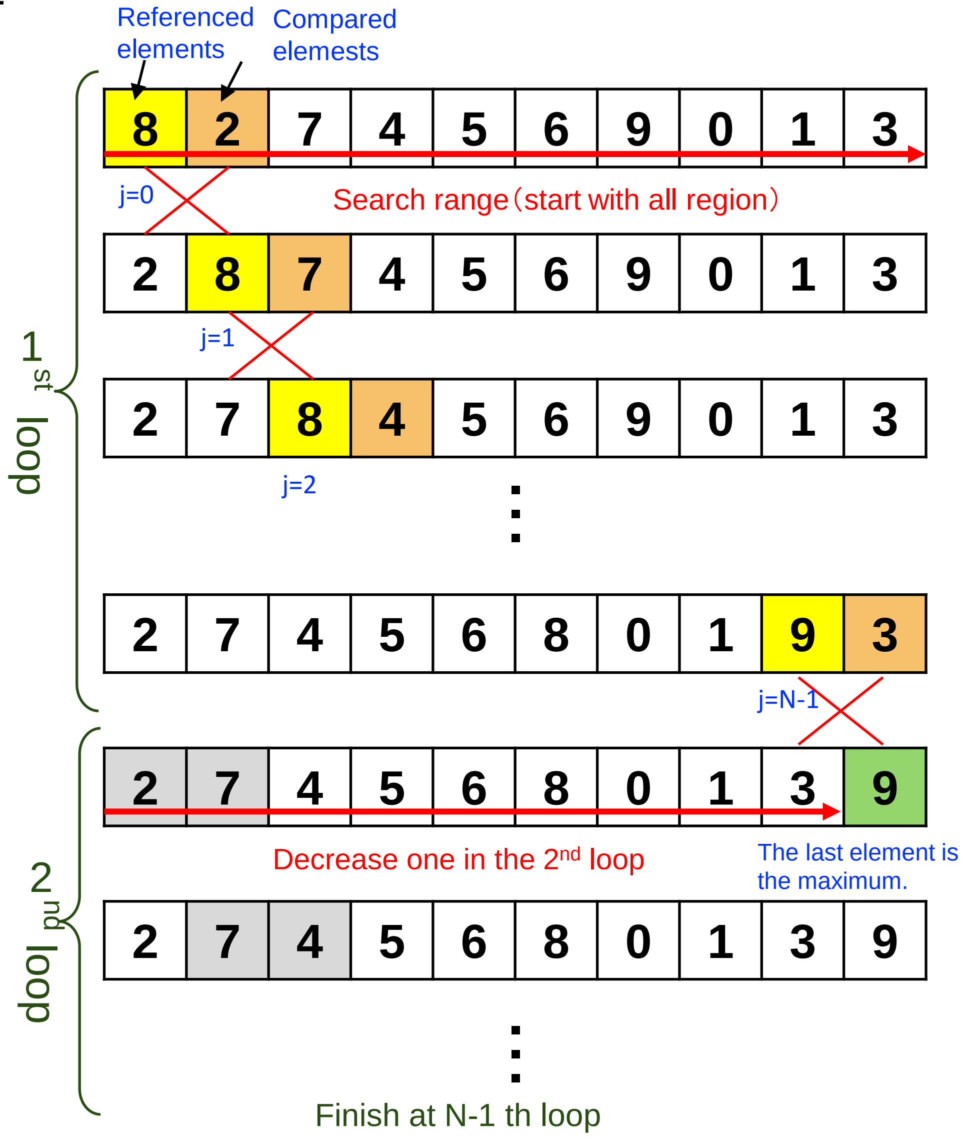
- Compile and execute source file sort_bubble.c

A program that sorts in ascending order

algorithm:

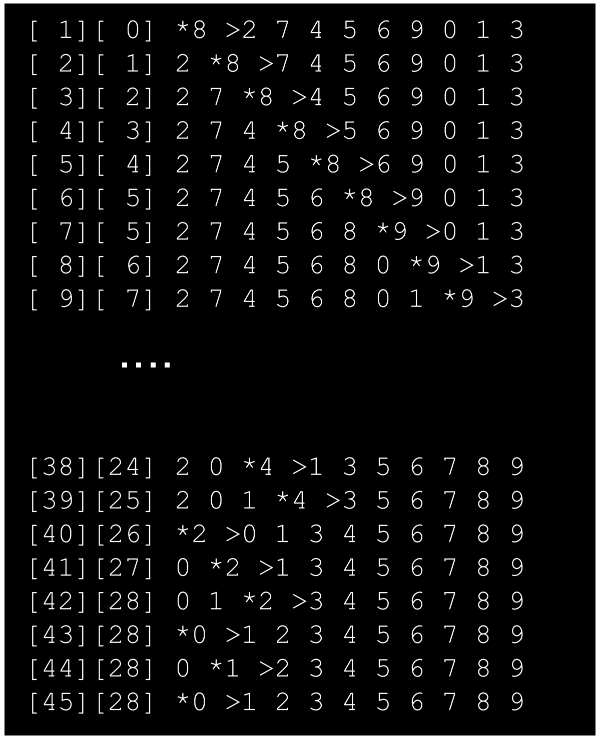
- Compare adjacent elements in order from the first.
- If the reference element is larger than the comparison target, replace it.
- Repeat the comparison while reducing the exploration range by one

```
int A[NUM] = {8,2,7,4,5,6,9,0,1,3};

int i, j, temp;

for (i = 0; i < NUM-1; i++) {    ← repeat N-1 loop
    for (j = 0; j < NUM-i-1; j++) ← With decreasing
                                  the range to sort
        if (A[j] > A[j+1]) {      ← If the referenced
                                  element is greater
    temp = A[j];
            A[j] = A[j+1];        ←「exchange」
            A[j+1] = temp;
        }
    }
}
```



Exercise 4-1: Visualization of bubble sort

- Modify sort_bubble.c and create a program sort_bubble1.c that visualizes bubble sort.
- Also, count and display the number of comparisons and the number of replacements.
  - ex)
    - "*" Before the reference element
    - ">" Before the element to be compared
    - [Comparison count] and [Replacement count] are displayed at the beginning of the array.



```
[ 1][ 0] *8 >2 7 4 5 6 9 0 1 3
[ 2][ 1] 2 *8 >7 4 5 6 9 0 1 3
[ 3][ 2] 2 7 *8 >4 5 6 9 0 1 3
[ 4][ 3] 2 7 4 *8 >5 6 9 0 1 3
[ 5][ 4] 2 7 4 5 *8 >6 9 0 1 3
[ 6][ 5] 2 7 4 5 6 *8 >9 0 1 3
[ 7][ 5] 2 7 4 5 6 8 *9 >0 1 3
[ 8][ 6] 2 7 4 5 6 8 0 *9 >1 3
[ 9][ 7] 2 7 4 5 6 8 0 1 *9 >3

    ....

[38][24] 2 0 *4 >1 3 5 6 7 8 9
[39][25] 2 0 1 *4 >3 5 6 7 8 9
[40][26] *2 >0 1 3 4 5 6 7 8 9
[41][27] 0 *2 >1 3 4 5 6 7 8 9
[42][28] 0 1 *2 >3 4 5 6 7 8 9
[43][28] *0 >1 2 3 4 5 6 7 8 9
[44][28] 0 *1 >2 3 4 5 6 7 8 9
[45][28] *0 >1 2 3 4 5 6 7 8 9
```

- The origin of "bubble" is from the appearance that the larger values move (emerge) to the edge in order.
- Bubble sort always makes n (n -1) / 2 comparisons

Reference: Visualization video of various sorts

- For those who want to quickly grasp the image of various sorting algorithms: Sortable video

  15 Sorting Algorithms in 6 Minutes

  https://www.youtube.com/watch?v=kPRA0W1kECg

  The Sound of Sorting - "Audibilization" and Visualization of Sorting Algorithms

  http://panthema.net/2013/sound-of-sorting/

- For those who want to relax and enjoy:

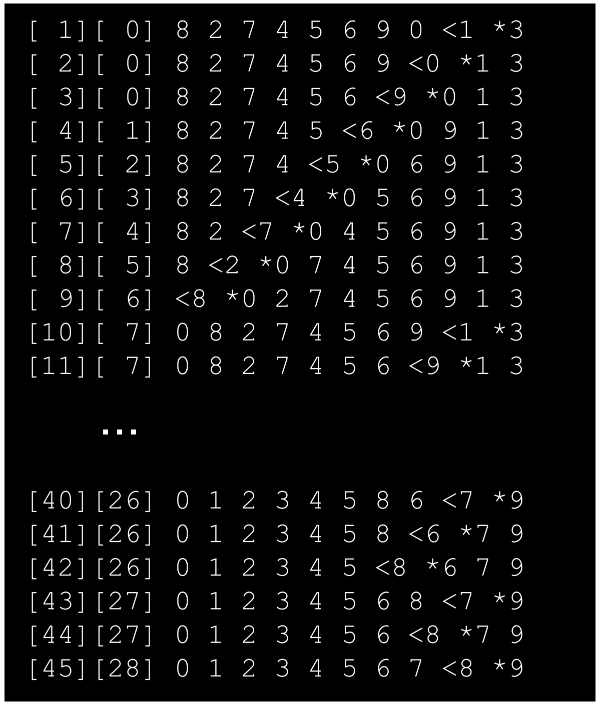
  Bubble-sort with Hungarian folk dance
  https://www.youtube.com/watch?v=lyZQPjUT5B4

Exercise 4-2: Inversion of bubble sort sort_bubble2.c

- Modify sort_bubble1.c to create a program sort_bubble2.c that starts the comparison from the back of the array and replaces the smaller elements in the forward.

ex）

"*" Before the reference element
“<" Before the element to be compared
[Comparison count] and
[Replacement count] are displayed at the beginning of the array.

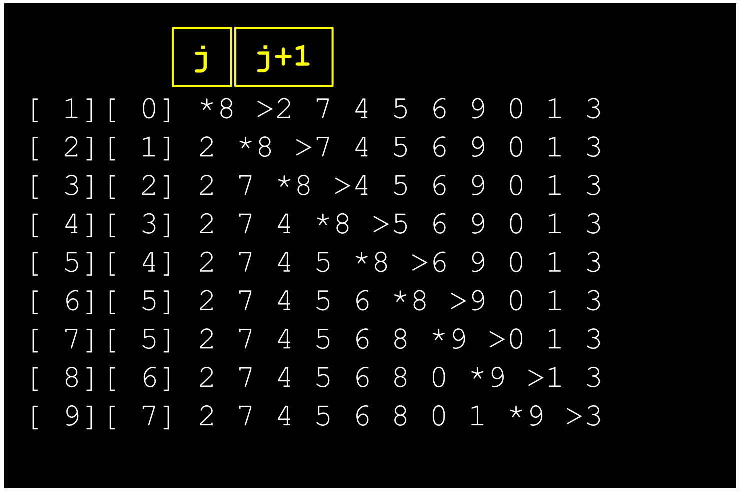


```
[ 1][ 0] 8 2 7 4 5 6 9 0 <1 *3
[ 2][ 0] 8 2 7 4 5 6 9 <0 *1 3
[ 3][ 0] 8 2 7 4 5 6 <9 *0 1 3
[ 4][ 1] 8 2 7 4 5 <6 *0 9 1 3
[ 5][ 2] 8 2 7 4 <5 *0 6 9 1 3
[ 6][ 3] 8 2 7 <4 *0 5 6 9 1 3
[ 7][ 4] 8 2 <7 *0 4 5 6 9 1 3
[ 8][ 5] 8 <2 *0 7 4 5 6 9 1 3
[ 9][ 6] <8 *0 2 7 4 5 6 9 1 3
[10][ 7] 0 8 2 7 4 5 6 9 <1 *3
[11][ 7] 0 8 2 7 4 5 6 <9 *1 3

    ...

[40][26] 0 1 2 3 4 5 8 6 <7 *9
[41][26] 0 1 2 3 4 5 8 <6 *7 9
[42][26] 0 1 2 3 4 5 <8 *6 7 9
[43][27] 0 1 2 3 4 5 6 8 <7 *9
[44][27] 0 1 2 3 4 5 6 <8 *7 9
[45][28] 0 1 2 3 4 5 6 7 <8 *9
```

Exercise 4-2: Tips sort_bubble2.c

sort_bubble1.c

A[j] > A[j+1] then exchange

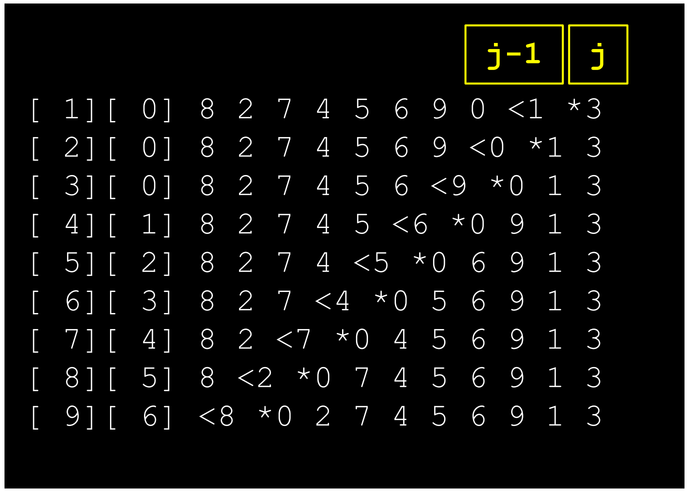


```
[ 1][ 0] *8 >2 7 4 5 6 9 0 1 3
[ 2][ 1] 2 *8 >7 4 5 6 9 0 1 3
[ 3][ 2] 2 7 *8 >4 5 6 9 0 1 3
[ 4][ 3] 2 7 4 *8 >5 6 9 0 1 3
[ 5][ 4] 2 7 4 5 *8 >6 9 0 1 3
[ 6][ 5] 2 7 4 5 6 *8 >9 0 1 3
[ 7][ 5] 2 7 4 5 6 8 *9 >0 1 3
[ 8][ 6] 2 7 4 5 6 8 0 *9 >1 3
[ 9][ 7] 2 7 4 5 6 8 0 1 *9 >3
```

```
for (i = 0; i < (NUM - 1); i++) {
      for (j = 0; j < NUM-i-1 ; j++) {

            /* compare and exchange */
            if (A[j] > A[j+1]) {

            }
      }
}
```

sort_bubble2.c

A[j-1] > A[j] then exchange



```
[ 1][ 0] 8 2 7 4 5 6 9 0 <1 *3
[ 2][ 0] 8 2 7 4 5 6 9 <0 *1 3
[ 3][ 0] 8 2 7 4 5 6 <9 *0 1 3
[ 4][ 1] 8 2 7 4 5 <6 *0 9 1 3
[ 5][ 2] 8 2 7 4 <5 *0 6 9 1 3
[ 6][ 3] 8 2 7 <4 *0 5 6 9 1 3
[ 7][ 4] 8 2 <7 *0 4 5 6 9 1 3
[ 8][ 5] 8 <2 *0 7 4 5 6 9 1 3
[ 9][ 6] <8 *0 2 7 4 5 6 9 1 3
```

```
for (i = 0; i < (NUM - 1); i++) {
      for (j = [ ? ] ; j > [ ? ] ; j--) {

            /* compare and exchange */
            if (A[j-1] > A[j]) {

            }
      }
}
```

Debug using `printf`

- "Debugging" is to find and fix a program error (bug).
- If a program doesn't work, use the `printf` function to check the progress of processing frequently and identify the bug.

Example: When you want to check whether `if` block works,

```
int A[NUM] = {8,2,7,4,5,6,9,0,1,3};
int i, j, temp;

for (i = 0; i < NUM-1; i++) {
    for (j = 0; j < NUM-i-1; j++) {

        if (A[j] > A[j+1]) {
            printf("larger¥n");
            temp = A[j];
            A[j] = A[j+1];
            A[j+1] = temp;
        }
    }

}
```

← display “larger” if we enter `if` block

sort_bubble.c

If you display more detailed information, the situation becomes easier to understand.

```
printf(“larger: A[%d]=%d > A[%d]=%d¥n",
                                j,A[j],j+1,A[j+1]);
```

```
larger: A[0]=8 > A[1]=2
larger: A[1]=8 > A[2]=7
larger: A[2]=8 > A[3]=4
larger: A[3]=8 > A[4]=5
larger: A[4]=8 > A[5]=6
larger: A[6]=9 > A[7]=0
...
```

Exercise 4-3: Selection sort sort_select.c

Submission required

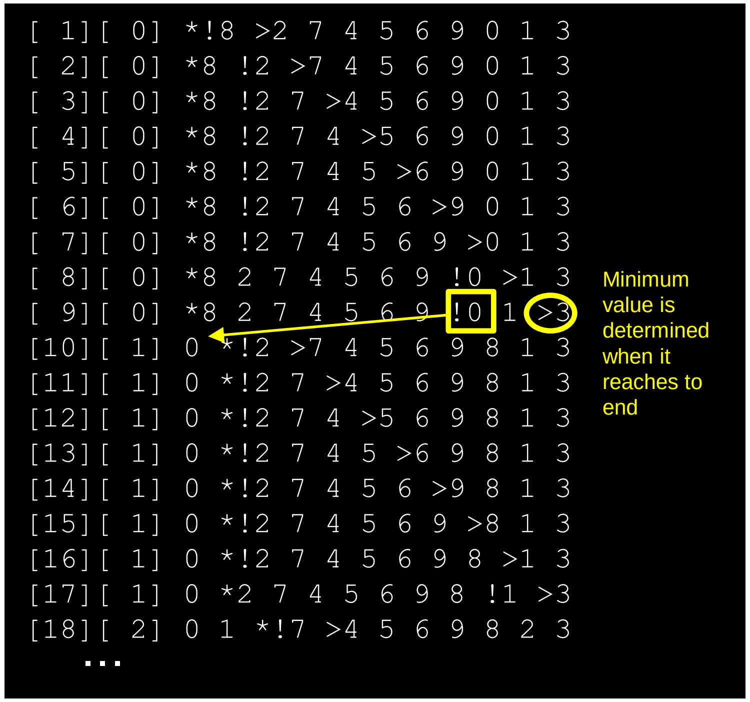
- Replace bubble sort with selection sort.
  - The program name is sort_select.c
  - Visualize and compare the difference with bubble sort
  - (Check the number of comparisons and the number of replacements)

Algorithm of selection sort

1. Find the smallest value in the data column and exchange it for the first element.
2. Next, find the smallest value in the second and subsequent data columns and exchange it for the second element.
3. Repeat this until the end of the data string

(Example)
"*" For the element to be replaced (0, 1, 2 ...)
"!" For the element of the minimum value at the time before comparison
">" For the element to be compared
(want to show that "!" And “>" are compared)

[Comparison count] and [Replacement count] are displayed at the beginning of the array.



```
[ 1][ 0] *!8 >2 7 4 5 6 9 0 1 3
[ 2][ 0] *8 !2 >7 4 5 6 9 0 1 3
[ 3][ 0] *8 !2 7 >4 5 6 9 0 1 3
[ 4][ 0] *8 !2 7 4 >5 6 9 0 1 3
[ 5][ 0] *8 !2 7 4 5 >6 9 0 1 3
[ 6][ 0] *8 !2 7 4 5 6 >9 0 1 3
[ 7][ 0] *8 !2 7 4 5 6 9 >0 1 3
[ 8][ 0] *8 2 7 4 5 6 9 !0 >1 3
[ 9][ 0] *8 2 7 4 5 6 9 !0 1 >3
[10][ 1] 0 *!2 >7 4 5 6 9 8 1 3
[11][ 1] 0 *!2 7 >4 5 6 9 8 1 3
[12][ 1] 0 *!2 7 4 >5 6 9 8 1 3
[13][ 1] 0 *!2 7 4 5 >6 9 8 1 3
[14][ 1] 0 *!2 7 4 5 6 >9 8 1 3
[15][ 1] 0 *!2 7 4 5 6 9 >8 1 3
[16][ 1] 0 *!2 7 4 5 6 9 8 >1 3
[17][ 1] 0 *2 7 4 5 6 9 8 !1 >3
[18][ 2] 0 1 *!7 >4 5 6 9 8 2 3
   ...
```

Let's check the difference in the number of replacements compared to bubble sort

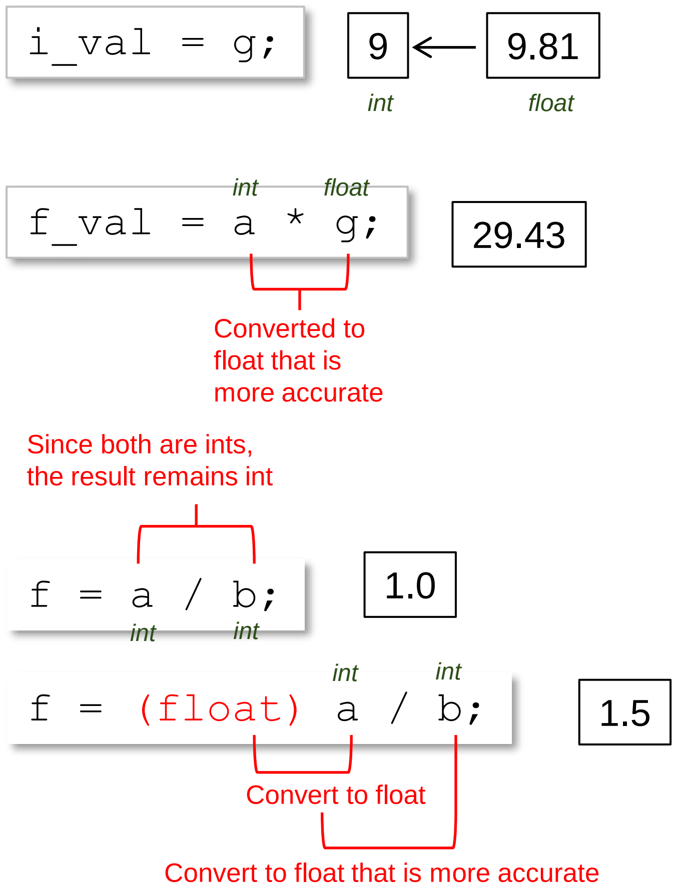
Ref: Select-sort with Gypsy folk dance
https://www.youtube.com/watch?v=Ns4TPTC8whw

Cast (sample program cast.c)

- Implicit type conversion
  - In C, the compiler chooses a safe type and converts it automatically.
  - Rule (1): Automatic conversion at the time of substitution
    - If the left side type and the right side type are different, it is converted to the left side type.
  - Rule (2): Automatic conversion in expression
    - When constants or variables of different types appear in the expression, unify the types to the one with higher precision.
- Explicit type conversion (cast)
  - When you want to forcibly convert to another type when performing an operation between variables of different types, specify the type you want to convert in "()".
    - The cast is just a temporary conversion and does not change the type of the originally declared variable.

```
int a = 3;
int b = 2;
float g = 9.81;
float f_val;
int   i_val;
```

declared.



```
i_val = g;
```

```
f_val = a * g;
```

```
f = a / b;
```

```
f = (float) a / b;
```

9
9.81
int
float
int
float
29.43
Converted to float that is more accurate
Since both are ints, the result remains int
1.0
int
int
int
int
1.5
Convert to float
Convert to float that is more accurate

Usage of mathematical functions math_calc.c

- Compile and execute the sample program math_calc.c that uses mathematical functions. Open the source with an editor and check the contents.
  - In order to use math functions, you need to add math.h to #include in your program.

```
#include <stdio.h>
#include <math.h> //header for mathematical functions
```

  - In addition, when compiling, you need to add a link option (-lm) to the Math library..

```
$ gcc -o math_calc math_calc.c -lm
```

Mathematical functions

- Absolute value
  - fabs(x)

    （abs(x) for int type）
- Trigonometric and hyperbolic functions
  - sin(x), cos(x), tan(x)
  - asin(x), acos(x), atan(x)（care for range of output）
  - sinh(x), cosh(x), tanh(x)
- Exponential / logarithmic function etc.
  - exp(x) : Exponential e^x
  - log(x) : log natural $\log_e(x)$
  - log10(x) : log base 10 $\log_{10}(x)$
  - pow(x, y) : power x^y
  - sqrt(x) : square root

Mathematical function arguments and return values are all double type

Constant defined in math.h

```
M_PI:    π
M_PI_2:  π/2
M_E:     e  (base of log natural)
```

etc

Exercise 4-4: Polynomial approximation

Submission required

- Modify the polynomial program poly.c to calculate the 3rd and 5th order McLaughlin expansions of the sine function sin (x). Create a graph with Excel, and compare the degree of approximation.
  - The calculation range of x is between -2π and 2π.
- Submit the followings.
  1. poly_sin3.c that expands sin (x) into 3rd order McLaughlin series
  2. poly_sin5.c which expands sin (x) into 5th order McLaughlin series
  3. Make a graph of these functions using any plot software

Hints

- poly.c: A trick to reduce the amount of polynomial calculation

n-1 degree polynomial $c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1x + c_0$

If calculated as it is, (n-1) + (n-2) + .. + 1 = n (n-1) / 2 multiplications are required.

Transformation $\Rightarrow$ $(((c_{n-1}\textcircled{x} + c_{n-2})\textcircled{x} + c_{n-3})x + \cdots + c_1)x + c_0$

Only need to multiply (n-1) times

Since the same shape is repeated, the process of starting with the highest-order coefficient c_{n-1} and multiplying by x to add the coefficient of the next order is repeated until the constant term (0th order).

- Mclaughlin expansion of sin（x）

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

3rd 5th

Find the coefficients of each order and manually enter them in the array c [].

Find the series in the second exercise, factorial.c

Note: Array c [0] = c0, c [5] = c5

Pay attention to “implicit type conversion"

Caution

"implicit type conversion"

$\underset{\text{int}}{1} / \underset{\text{int}}{6} = 0.166667 \Rightarrow 0$

Since integers are considered int types, solutions for operations between int types are treated "implicitly" as int types.

When constants and variables of different types appear in the expression, they are unified to the one with higher precision.

Add a decimal point when you want to get a real variable.

poly_sin3.c

```
float c[N] = {0, 1, 0, -1./6};
```

(N=4)

poly_sin5.c

```
float c[N] = {0, 1, 0, -1./6, 0, 1./120};
```

(N=6)

Without a decimal point, 0 will be assigned

Previous lecture: Redirect

- You can easily save the output of the program to a file by using the method "Redirect (>)" on the console terminal.
  - Redirection is a method of changing the standard output destination of a program to a file.
  - The standard output is where printf etc. are displayed. In the terminal, the console terminal is the standard output destination.

```
$ ./score > score.txt
$ cat score.txt  ← cat is command to show the contents
```

The output contents of the execution result are saved in a file called score.txt. Please open it in an editor

Summary

- Example answers of previous exercises
- Exercise：Algorithm using array
  - Sorting
  - Mathematical functions

Midterm Assignment

- Follow the instructions in Assignment01.pdf.

Next lecture:

- Function and procedure
  - Definition and call of function
  - Scope (range) of variables
  - Recursive call