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.
  - A typical algorithm requires an order of O (n log n) at best and O (n^2) at worst for the amount of calculation.
- 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

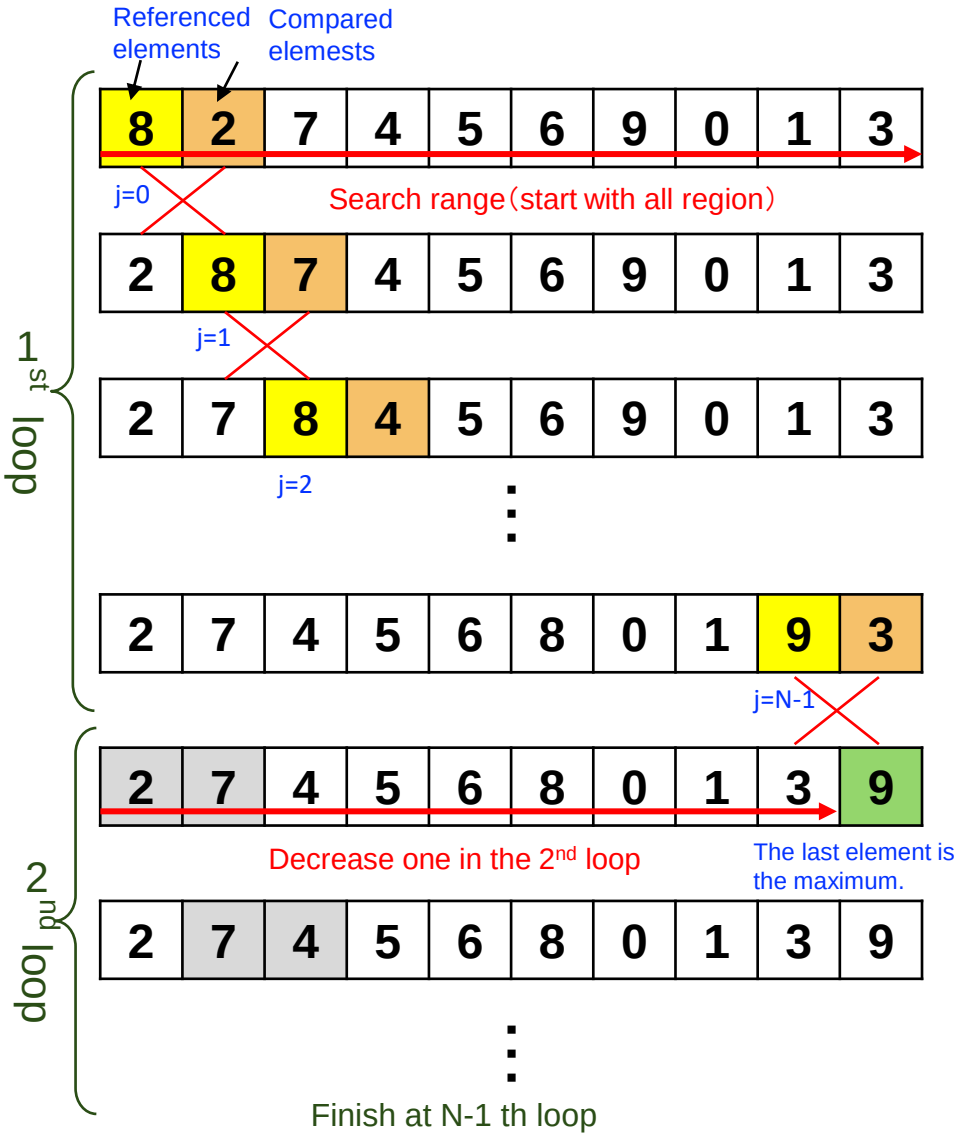
algorithm:

- 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
  ※People dancing in other sorts
  Be careful for overlooking. Time flies

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

```
          j   j+1
[ 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

```
                          j-1   j
[ 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 that doesn't work, use the printf function to view the situation and find the bug.

Example: When you want to check `if` block works, put it in an `if` statement

```
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 go to `if` block

sort_bubble.c

If you display the conditions at that time in detail, 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

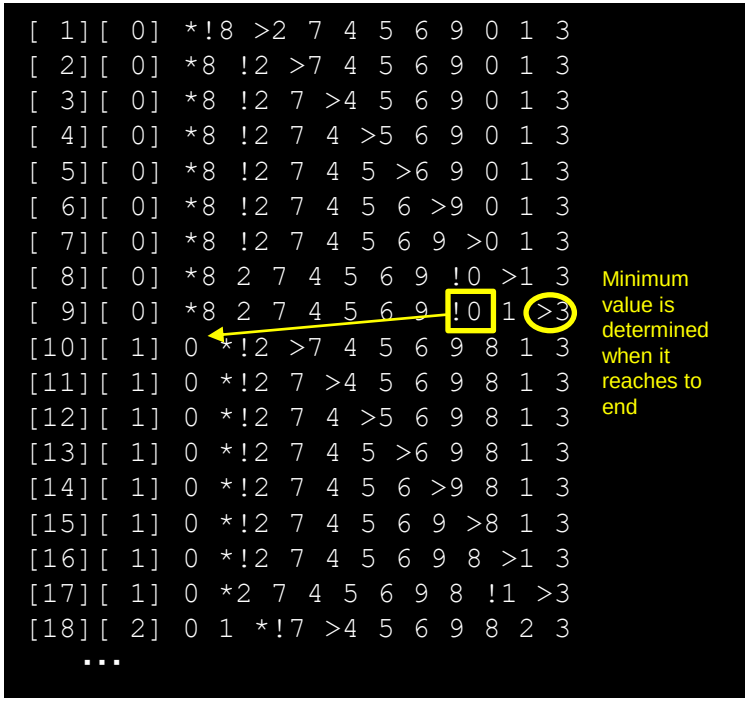
- Survey the principle of selection sort on the Web and replace sort_bubble1.c with selection sort.
  - The program name is sort_select.c
  - Visualize and compare the difference with bubble sort
  - (Let's check the number of comparisons and the number of replacements)

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.



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;
```

9 ← 9.81

int *float*

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

int *float*

29.43

Converted to float that is more accurate

Since both are ints, the result remains int

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

int *int*

1.0

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

int *int*

1.5

Convert to float

Convert to float that is more accurate

Reference: Use 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..

last

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

[Important] In the Linux environment used in the exercise, the link option (-lm) must be added at the end.

Restrictions on ubuntu linux version 11.04 and later.
With the “--no-as-needed” option, you can compile even if -lm is in front of the source file.

Mathematical functions

- Absolute value
  - fabs(x)

    （abs(x) for int type）
- Trigonometric function
  - 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
  - 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

Mathematical function arguments and return values are all double type

Constant defined in math.h

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

etc

Exercise 4-4: Polynomial approximation

- Modify the polynomial program poly.c in the source code, 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 is x: -2π to 2π.
- Submission:
  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 softwares

Exercise 4-4: Explanation

- 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.

Formula transformation $(((c_{n-1}x + c_{n-2})x + c_{n-3})x + \cdots + c_1)x + c_0$

Only need to multiply (n-1) times

Since the same shape is repeated, the process of starting with the highest-order coefficient cn-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

Exercise 4-4: Note "Implicit type conversion when specifying coefficients"

Caution Coefficient specification for each order: Note "implicit type conversion"

When the assignment to the coefficient array is specified by an integer expression

1 / 6 = 0.166667 ⇒ 0

int int

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.

Countermeasures Add a decimal point when assigning an integer value formula to a real variable

poly_sin3.c

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

（N=4）

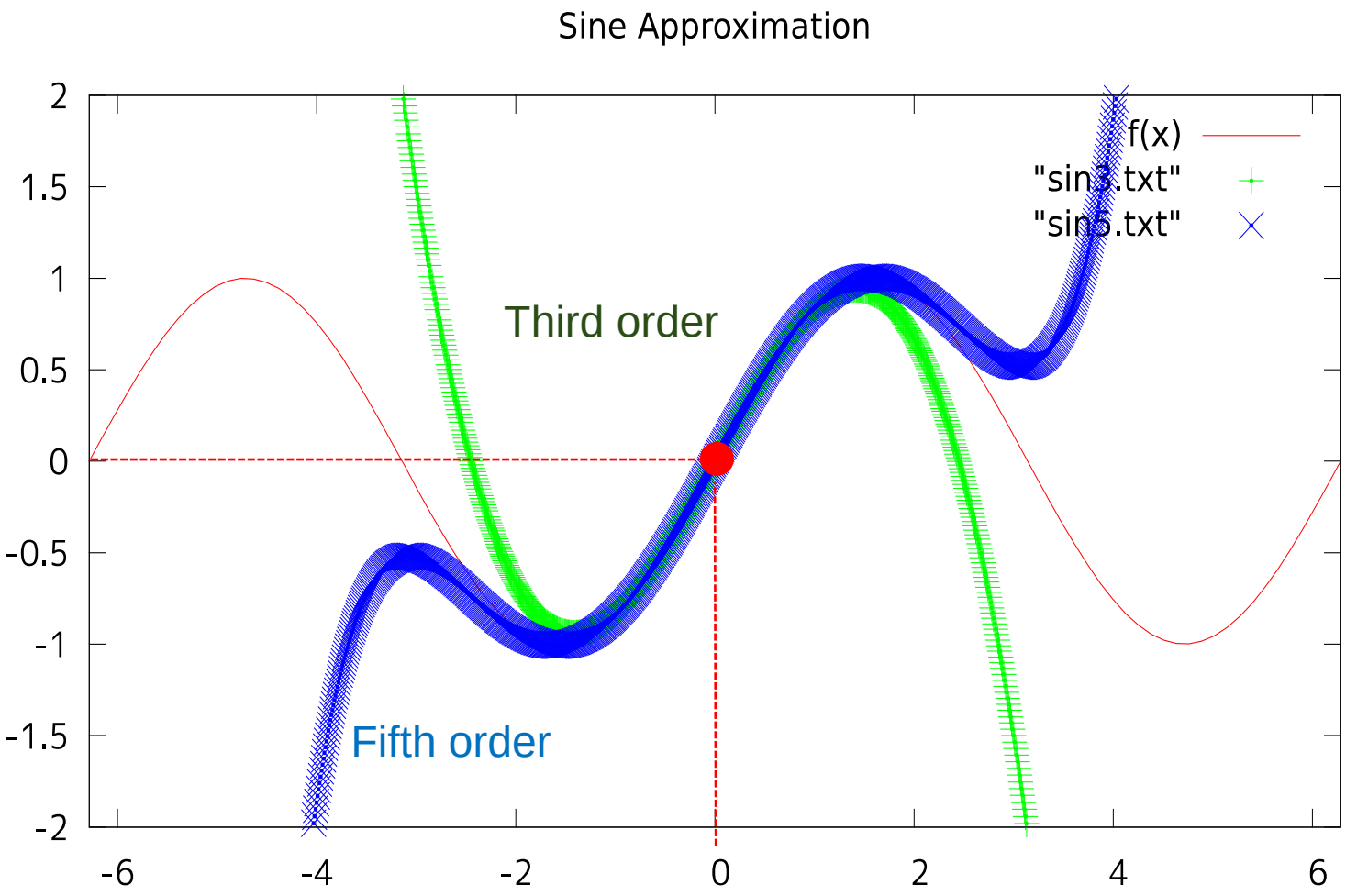
poly_sin5.c

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

（N=6）

Without a decimal point, 0 will be assigned

Exercise 4-4: Example of graph



McLaughlin expansion is approximation around $x=0$.
More accurate with increasing order

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

Midterm Assignment

- Follow the instructions in Assignment01.pdf.
- Problems:
  - Mandatory: Score sheet display program score_sheet.c
  - Optional problem: score_sheet_ex.c (Try this if you can afford to do it)
- Submission (Deadline: November 7 (submit even if late))
  - score_sheet.c(program)
  - report01.txt or .pdf (report)
    - Please give a brief explanation of how your program works.
    - Result of redirecting the execution result (example when one appropriate ID is entered)
- (If you have tackled the optional problem, please also submit following)
  - score_sheet_ex.c
  - report01_ex.txt or .pdf (report)
    - Only explain the difference from score_sheet.c. Please include the result of the redirection.

Next lecture:

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