COMP20005 Workshop Week 7

Preparation:

open `grok`, `jEdit`, and `minGW` (or `Terminal` if yours is a `Mac`)
download this slide set (`ws7.pdf`) from github.com/anhvir/c205 if you like

1	**Arrays:** concept, representation, usage
2	**Discussion 1:** Exercise 3.06 revisited → *an advantage of using arrays*
3	**Discussion 2:** Ex 7.4 → *Conventional ways to work with arrays* → *Redirection:* reading data from a file (instead of from the keyboard) → *Selection Sort* (don't confuse with *Insertion* Sort)
4	**A Case Study** based on exercise 7.5 → *Using* `struct` *and* `typedef`
5	**Assignment 1:** Q&A
6	**Lab:** • do your assignment 1 and test the submission, or • do exercises 7.1, 7.7 – 7.10

Arrays

Situation:

We need to manipulate a series of number like `1, 4, 10, 8, 7, 9`

Mathematically speaking, we are working with the sequence

$$x_1, x_2, x_3, x_4, x_5, x_6$$

where

$x_1 = 1$

$x_2 = 4$

...

$x_6 = 9$

With $i=3$ we can say that x_i has the value of `10`. We even can:

```
set S to 0
for i from 1 to 6
    add x_i to S
```

Quite convenient!

Can we do a similar thing in C?

Arrays

Situation:

We need to manipulate a series of number like `1, 4, 10, 8, 7, 9`

Mathematically speaking, we are working with the sequence

$x_1, x_2, x_3, x_4, x_5, x_6$

Can we do a similar thing in C?

YES. Here, we will declare `x` as an *array* of 6 `int` elements.

```
int x[6]= {1, 4, 10, 8, 7, 9};
```

then:

the first element of `x` is referred to as `x[0]`

the second element of `x` is referred to as `x[1]`

the `i`-th element of `x` is referred to as `x[i]`, with 0 `..` 5

Note: we don't have `x[6]`, the last element is `x[6-1]`

arrays: declaration & use

	statements	variables in memory (*after* LHS statements)
1	`int i, A[5];` /* equivalent to declaring 6 variables, each is of data type **`int`** */	
2	`A[0] = 10;` `i= A[0] * 2;`	
3	`i= 2;` `A[i]= 20;`	

4	`for (i=0; i<=5; i++) {`
	`  A[i]= i*i;`
	`}`
		

5	`for (i=0; i<3; i++) {`
	`  scanf ( "%d", &A[i] );`
	`}` /* supposing that input from keyboard is **10 20 30** */
		

arrays…

	statements	variables in memory (*after* LHS statements)
1	`int i, sum=0,` `A[5]= {0,1,2,3,4};`	
2	`for (i=0; i<5; i++)` `sum += A[i];`	

arrays…

	statements	variables in memory (*after* LHS statements)						
		i	sum	A[0]	A[1]	A[2]	A[3]	A[4]
1	`int i, sum=0, A[5]= {0,1,2,3,4};`		0	0	1	2	3	4
2	`for (i=0; i<5; i++) sum += A[i];`	5	10	0	1	2	3	4
3	`for (i=0; i<4; i++) { A[i+1]= A[i]; }`	4	10	0	0	0	0	0

arrays…

	statements	variables in memory (*after* LHS statements)						
1	`int i, sum=0,` `A[5]= {0,1,2,3,4};`	i	sum	A[0]	A[1]	A[2]	A[3]	A[4]
			0	0	1	2	3	4
2	`for (i=0; i<5; i++)` `sum += A[i];`	5	10	0	1	2	3	4

3	`for (i=0; i<4; i++) {`
	`A[i+1]= A[i];`
	`}`
		0	10	0	0	0	0	0

Notes: No operation with *whole arrays* is allowed. With declaration:

```
int A[3]={10,20,30}, B[3];
```

we cannot write:

```
B= A;
if (A==B) A= A+B;
```

Arrays : using in C

X

```
#define SIZE 5
...

int X[SIZE]= {1, 2, 3};
int n= 3;
```

In computer memory, an array is stored as *a block of contiguous cells*, one cell for one array's element.

Essentially, an array is defined by 4 objects:

- `X:` the array's name, which is actually a pointer to the memory block
- `int:` the data type of each element of the array
- `SIZE`: an `int` constant representing the array's size
- `n`: a buddy `int` variable, representing the number of elements that are currently employed

Arrays as function arguments

- With a function prototype, say:

```
int change_array(int A[], int n);
```

we should note that:

  - the formal parameter `A[]` is an array of `int`, but no size is specified in "`int A[]`",
  - instead, there is another parameter, `n`, which specifies the *current* size of `A[],`
  - the array formal parameter `A[]` is an array, is a pointer, so it can imply *both input and output* of function `change_array`.

- With the call "`change_array(B, 10)`":
  - the action in formal parameter `A[]` is actually happen to `B[].`

- With the above function and the declarations:

```
int B[10]= {1,2,4,2,3,3,1,1,9,8};
int n= 10;
```

What do the following calls mean:

```
change_array(B, 10);
change_array(&B[0], 10);
change_array(B, 8);
```

Discussion 1: Exercise 3.06 revisited

Then, group work with:

- Exercise 7.1 and 3.06 from grok W7
- If have time, do an exercise from W7X (we will discuss 7.4 soon)

PLEASE: use `jEdit` and `gcc`.

NOTES:

For 7.1: Save your time by simplifying the main function to

```
int A[8]= {0,0,0,0,0,0,1,0};
printf("all_zero(A,5))= %d, all_zero(A,8)= %d\n",
           all_zero(A,5), all_zero(A,8));
/* should print out 0 and 1, why? */
```

For 3.06: Start with copy the solution of 3.06 (from grok W3), then change it with the use of array,

Discussion 2: Exercise 7.4

Write a program that reads as many as 1,000 integer values, and counts the frequency of each value in the input:

```
./program
Enter as many as 1000 values, ^D to end
1 3 4 6 4 3 6 10 3 5 4 3 1 6 4 3 1
17 values read into array
Value Freq
    1    3
    3    5
    4    4
    5    1
    6    3
   10    1
```

How?

Discussion 2: Exercise 7.4

Write a program that reads as many as 1,000 integer values, and counts the frequency of each value in the input:

So we need to:

1. *Input value for an array*
2. *Sort an array in increasing order*
3. *Count and print out frequencies*

We will do together steps 1 and 2, and will demonstrate how to read data from a file (instead of from the keyboard).

Please use `jEdit` *and do together with Anh (e.g. you should at least follow Anh's speed in your own* `jEdit` *window). If your* `jEdit`/`gcc` *are not ready, you can employ* `grok`*, but it will be inconvenient.*

Discussion 3: typedef and struct for exercise 7.5

Suppose that a set of "student number, mark" pairs are provided, one pair of numbers per line, with the lines in no particular order. Write a program that reads this data and outputs the same data, but ordered by student number. For example:

823678 66
765876 94
864876 48
785671 68
854565 89

On this input your program should output:

Enter as many as 1000 "studnum mark" pairs, ^D to end
5 pairs read into arrays
studnum mark
765876 94
785671 68
823678 66
854565 89
864876 48

Hint: use two parallel arrays, one for student numbers, and one for the corresponding marks. You may assume that there are at most 1,000 pairs to be handled.

Discussion 3: typedef and struct for exercise 7.5

Use `typedef` *to define a new data type. For example:*

```
typedef int integer;

integer fact(integer n) {
    …
}
```

Discussion 3: typedef and struct for exercise 7.5

Use `typedef` *to define a new data type.*

Use struct to define a multi-component data type. For example:

```
typedef struct{
    int stud_id;
    double mark;
} student_t;

/* return the average mark of n students,
    the pairs (student_id, mark) are stored in array A[] */
double average_mark(student_t A[], int n) {
    …
}
```

Discussion 3 :exampe of using typedef and struct

```
#include <stdio.h>
typedef struct{
    int stud_id;
    double mark;
} student_t;

int main(...) {
    student_t s1= {211111, 99.5), s2;
    student_t A[10];
    int i;
    s2= s1;
    s2.stud_id= 1000001;
    printf("id= %d mark=%f\n", s1.stud_id, s1.mark);
    for (i=0; i<10; i++) {
        scanf("%d %d", &A[i].id, &A[i].mark);
    }
    ...
```

Discussion 3: typedef and struct for exercise 7.5

Suppose that a set of "student number, mark" pairs are provided, one pair of numbers per line, with the lines in no particular order. Write a program that reads this data and outputs the same data, but ordered by student number. For example:

823678 66

765876 94

We can start with, for example:

```
typedef struct{
   int stud_id;
   double mark;
} mark_t;

#define SIZE 30000
int main(…) {
   mark_t unimelb[SIZE];
   int n= 0;
   ...
```

And write functions to:

- input data to an array of `mark_t`
- sort an array of `mark_t`
- ouput data of an array of `mark_t`

Remember to a) create a data file, and b) use redirection for inputting data.

testing

Assignment 1: Q&A

Notes:

- `grok` is great for in-class practice, but
- use `grok` for the assignments might bring some unexpected headache!
- Use `jEdit` and `gcc` for assignments and serious programming projects!

Assignment 1: A reasonable way to start with minGW/Terminal

	command/action	explanation
1	`cd ~`	set your home directory as your *current directory*
2	`mkdir ASS1`	*make a new directory, and of assignment files will be placed in that directory*
3	`cd ASS1`	*change current directory to ASS1*
4	`ls`	*list the content of the current directory, it should be empty*
5	navigate to the assignmen1 FAQ page and download file ass1-skel.c (2nd link of point 1), and all the files listed in point 7. You should download the files to the ASS1 directory.	
6	`ls`	*now you should see the downloaded files*
7	`mv ass1-skel.c ass1.c`	*rename the skeleton file to your assignment*
8	using `jEdit` to do your assignment	
9	`gcc –Wall –o ass1 ass1.c`	*compile the program*
10	`./ass1 <wagons0.tsv >out0.txt`	*run program with redirection*
11	`diff wagon0-out-mac.txt out0.txt`	*check if your output is the same as the expected*

Assignments: advices

- *Be active in the subject's Discussion Forum!*
- *Visit* **LMS→Assignment 1** *frequently!*
- *Read the specifications carefully.*
- *Read the marking rubric carefully and try to maximize your marks!*
- *Read the* ***sample solution to 2015*** *(in LMS.Assignment1, point 6), focusing on* `main()`*. You can learn something from there.*
- *Check your program carefully, at least with all supplied data files. Make sure that your outputs are the same as the expected outputs.*
- *Make as many submissions as you want, only the last one (before deadline) counts. Do verify and read the report from verify to make sure that your program works well on dimefox.*
- *START EARLY, START RIGHT NOW! SUBMIT EARLY, SUBMIT EVERY DAY!*