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 7.1, 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
LMS	• Dicscuss 7.1, 7.4; Do ass1.

Arrays = ?

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 `i` in range `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;`	

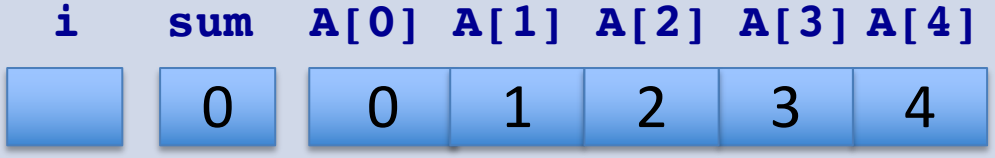
4	`for (i=0; i<5; i++) {`
	`  A[i]= i*i;`
	`}`
		

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)
1	`int i, sum=0,` `  A[5]= {0,1,2,3,4};`	
2	`for (i=0; i<5; i++)` `  sum += A[i];`	

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

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;  // n <= SIZE
// can we?
X[n]= 10;
n++;
X[5]= 100;
```

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 start of the memory block
- `int:` the data type of each element of the array
- `SIZE`: an `int` constant representing the array's capacity
- `n`: a buddy `int` variable, representing the number of elements that are currently employed

Arrays as function arguments

- With a function prototype, say:

```
int sum_array(int A[]);
```

we should note that:

- the formal parameter `A[]` is an array of `int`, but no size or capacity is specified in "`int A[]`", as such this parameter only specify the starting address of the array, it's also equivalent tp "`int *A`".
- as such, there is no way for the above function to compute the sum of A
- we need 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 `sum_array`.

Array Name is a Pointer!

Arrays as function arguments

```
int sum(int A[], int n) {
  int i, s= 0;
  for (i=0; i<n; i++) {
    s += A[i];
  }
  return s;
}
```

- With the above function and the declarations:

```
int B[10]= {1,2,3,4,5,6,7,8,9};
```

For each of the following statements: valid? If yes, what's the output?

```
printf("%d\n", sum(B, 10));
printf("%d\n", sum(B, 5));
printf("%d\n", sum(&B[0], 5));
printf("%d\n", sum(B+0, 5));
printf("%d\n", sum(B+3, 2));
```

DoItTogether: Exercise 7.1 and input/output redirection

Note: If you finish early, do the extended 6.09.

PLEASE: use `jEdit` and `gcc` or similar tools, not grok. Tell Anh if you have to use grok.

NOTES:

For 7.1: (Write function `int all_zero(int A[], int n)` that returns 1 or 0)

Save your time by simplifying the main function to

```
int A[MAX_N], n;
// add: reading the array
printf("all_zero(A,3))= %d, all_zero(A,4)= %d\n",
            all_zero(A+4,5), all_zero(A,MAX_N+1));
/* should print out 0 and 1, why? */
```

and use data:

0 0 0 1 6 0 0 0 0 0 8 7 0

Extra: Extended 6.09 from grok W6: If you finished 6.09 (change coin using functions), come back to that and spend 5 minutes to improve function change_coin by using an array of coin values and a loop.

group work: arrays, sequential search

Task: write a program that input a sequence of at most 100 integers, then:

- compute & print out `mid`= the smallest int that >= the average of the numbers
- search for the appearance of `mid` in the array, output the index of the first appearance
- find the index of the min element, in case of ties, find the smallest index

Step 1 (whole class): develop the main();

Step 2 (group): write other functions

main() function

```
#include <stdio.h>
```

Sample input data:

1 2 4 8 10 20 30 40 50 60 62 63 65 68 69 71 72 75 79 81 82 83 86 88 92 96 98

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 1 1 3 3 3 3 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].stud_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.

Ass1: Q&A make sure that you understand the tasks

Carefully read the spec

*Read the Discussion Forum and post **new** questions*

Ass1: Maximize Your Mark

Check your code against the marking rubric

Examine the 2020 sample solution: you don't need to understand, but you still can learn something from here

Questions on marking rubric?

Assignment 1: understanding

Understanding the requirements for

- Stage 1: 8 marks
- Stage 2: 8 marks
- Stage 3: 4 amrks

TESTING IN YOUR COMPUTER

- *Using redirection when running/testing your program:*

```
./myass1 < meals0.tsv > out_0.txt
```

- *Your program's output must be the same as the expected, ie. the command*

```
diff out_0.txt meals0-out-mac.txt
```

must give empty output (that is, no difference).

- *Remember that your code might work well on the 2 supplied data sets, but fail on some other…*

SUBMITTING:

- *Wait for the verification report*
- *Read the verification report carefully. Your program might work perfectly in your computer but fail in the testing computer(s).*
- *Try to submib early to avoid unexpected technical problems.*

Assignment 1: marking rubric

use of magic numbers, -0.5; unhelpful #defines, -0.5; #defines not in upper case, -0.5; bad choice for function names, -0.5; bad choices for variable names, -0.5; absence of function prototypes, -0.5; inconsistent bracket placement, -0.5; inconsistent indentation, -0.5; lack of whitespace (visual appeal), -0.5; lines >80 chars, -0.5; excessive commenting, -0.5; insufficient commenting, -0.5; use of constant subscripts in 2d arrays, -1.0; “rubbish” program, -5.0 ---------- comment at end of source code that says "programming is fun", +0.5; overall care and presentation, +0.5;	**total mark: up to 5**

Assignment 1: marking rubric

global variables, -1.0; main program too long or too complex, -1.0; other functions too long or too complex, -0.5; overly complex function argument lists, -0.5; insufficient use of functions, -0.5; duplicate code segments, -0.5; overly complex algorithmic approach, -0.5; unnecessary duplication/copying of data, -0.5; other structural issue (minor), -0.5; other structural issue (major), -1.0;	**Structural: up to 6** **Stage 1 : +2** **Stgae 2: +4**

Assignment 1: marking rubric

errors in compilation that prevent testing, -3.0; runtime segmentation fault on test1 with no output generated, -2.0; runtime segmentation fault on test2 with no output generated, -2.0; <various deduction for wrong output> -0.5 each wrong output format -0.5 ---------- no Authorship Declaration at top of program, -5.0; incomplete or unsigned Authorship Declaration at top of program, -3.0; significant overlap detected with another submission, -10.0; use of external code without attribution (major), -10.0; use of external code without attribution (minor), -2.0;	**Execution: total mark: up to 9** **Stage 1: +1** **Stage 2: +4** **Stage 3: +4** **----------** **deducted from the whole work**

LAB: do Assignment 1 OR exercises in W7 / W7X

Notes:

- `grok` is great for in-class practice, but
- use `grok` for the assignments might bring some unexpected inconvenience and even 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 <meals0.tsv >out0.txt`	*run program with redirection*
11	`diff out0.txt meals0-out.txt`	*check if your output is the same as the expected*