

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: understanding, Q&A, arrays and pointers

2

Classwork based on: Ex 7.1, Ex 7.4

→ *Conventional ways to work with arrays*

→ *function all_zero*

→ *Redirection:* reading data from a file (instead of from the keyboard)

→

→ *Discussion on how to do 7.4*

Assignment 1: Q&A, marking rubric

Lab:

- do your assignment 1 and test the submission, or
- do exercises 7.1, 7.7 – 7.10

LMS

- Discuss 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	<code>int i, A[5];</code> /* equivalent to declaring 6 variables, each is of data type <code>int</code> */	<div> <div>i</div> <div>A[0]</div> <div>A[1]</div> <div>A[2]</div> <div>A[3]</div> <div>A[4]</div> </div>
2	<code>A[0] = 10;</code> <code>i = A[0] * 2;</code>	
3	<code>i = 2;</code> <code>A[i] = 20;</code>	
4	<code>for (i=0; i<5; i++) {</code> <code>A[i] = i*i;</code> <code>}</code>	
5	<code>for (i=0; i<3; i++) {</code> <code>scanf ("%d", &A[i]);</code> <code>}</code> /* supposing that input from keyboard is 10 20 30 */	

arrays...

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

arrays...

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

arrays...

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

Arrays : using in C



```
#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

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 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

Array Name is a Pointer! a constant 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));
```

DoltTogether: Exercise 7.1 and beyond

PLEASE: If you can, use `jEdit` and `gcc` or similar tools, not `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 8 7 0

Note: We will extend the task. We will cover more than just exercise 7.1, and we will build the program in a serious “assignment-style”.

Your task: Start a new program on `jEdit` (or `grok playground` if you have to), copy and paste the above code into the `main()` function. Then continue the work together with Anh.

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 \geq 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`
- output 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 marks

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 submit early to avoid unexpected technical problems.*

Assignment 1: marking rubric

<p>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;</p> <p>absence of function prototypes, -0.5;</p> <p>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;</p> <p>use of constant subscripts in 2d arrays, -1.0; “rubbish” program, -5.0</p> <p>-----</p> <p>comment at end of source code that says "programming is fun", +0.5; overall care and presentation, +0.5;</p>	<p>total mark: up to 5</p>
---	-----------------------------------

Assignment 1: marking rubric

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

Assignment 1: marking rubric

<p>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</p> <hr/>	<p>Execution: total mark: up to 9 Stage 1: +1 Stage 2: +4 Stage 3: +4</p> <hr/>
<p>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;</p>	<p>deducted from the whole work</p>

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	<code>cd ~</code>	set your home directory as your <i>current directory</i>
2	<code>mkdir ASS1</code>	<i>make a new directory, and of assignment files will be placed in that directory</i>
3	<code>cd ASS1</code>	<i>change current directory to ASS1</i>
4	<code>ls</code>	<i>list the content of the current directory, it should be empty</i>
5	navigate to the assignmen1 FAQ page and download file ass1-skel.c (2 nd link of point 1), and all the files listed in point 7. You should download the files to the ASS1 directory.	
6	<code>ls</code>	<i>now you should see the downloaded files</i>
7	<code>mv ass1-skel.c ass1.c</code>	<i>rename the skeleton file to your assignment</i>
8	using <code>jEdit</code> to do your assignment	
9	<code>gcc -Wall -o ass1 ass1.c</code>	<i>compile the program</i>
10	<code>./ass1 <meals0.tsv >out0.txt</code>	<i>run program with redirection</i>
11	<code>diff out0.txt meals0-out.txt</code>	<i>check if your output is the same as the expected</i>