

COMP20005 Workshop Week 7

Preparation:

open [grok](#) and [VS](#)(or your favorite tools for editing and gcc)

download this slide set ([ws7.pdf](#)) from github.com/anhvir/c205 if you like

1 **Arrays:** understanding, Ex 7.01, Ex 7.04

2 **Assignment 1:** Q&A, marking rubric

Lab:

- do your assignment 1 and make at least one submission,

Important:

Next Week (Week 8) Tuesday Classes will be replaced by:

When: Monday 24/APR 10AM-12AM,

Where: PAR-207to221 Bouverie St-B112

If you are busy, you can also attend only one hour!

Arrays = ?

Conceptually understanding:

- Array= a collection of same-type data containers
- When declaring an array, we need 4 elements:
 - data type
 - name
 - capacity
 - buddy variable n for the current size

Array in computer memory

- Array occupies a block of must-be-defined “capacity” cells in memory

arrays: a collection of variables under a common name

	statements	in memory (after running LHS)
1	<code>int i, A[5];</code>	<code>i A[0] A[1] A[2] A[3] A[4]</code> 
2	<code>A[0] = 10; i= A[0] * 2;</code>	
3	<code>i= 2; A[i]= 20;</code>	
4	<code>for (i=0; i<5; i++) { A[i]= i*i; }</code>	
5	<code>for (i=0; i<3; i++) { scanf ("%d", &A[i]); } /* supposing input 10 20 30 */</code>	 3

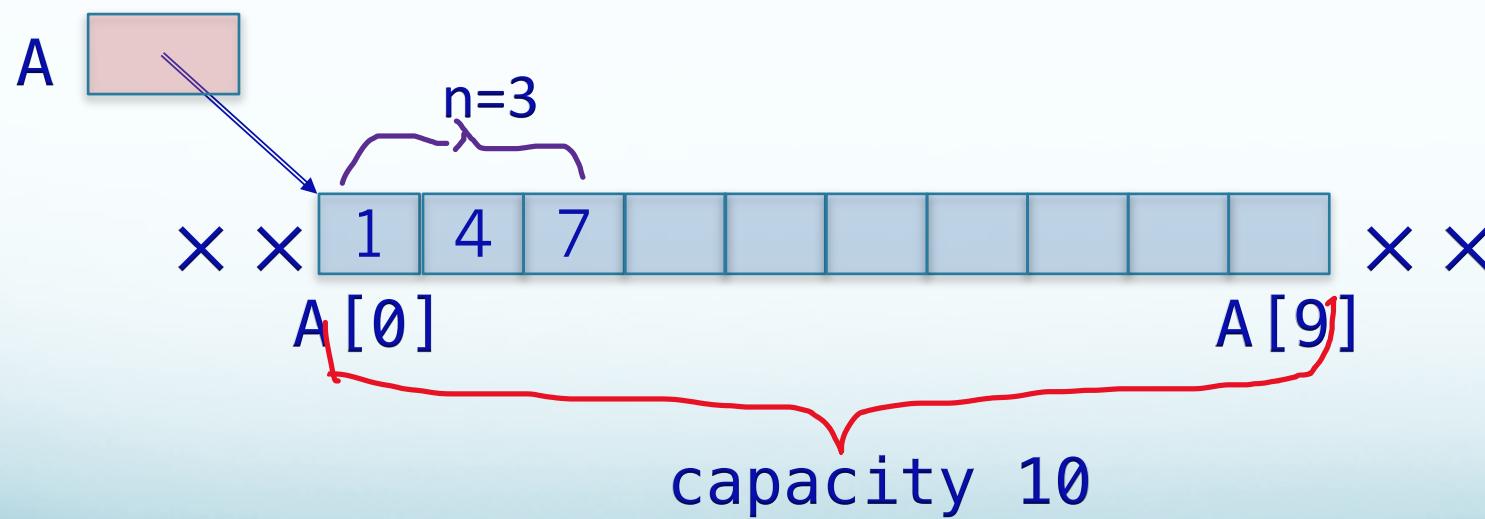
arrays...

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

Chorus of the day: Array names are constant pointers!

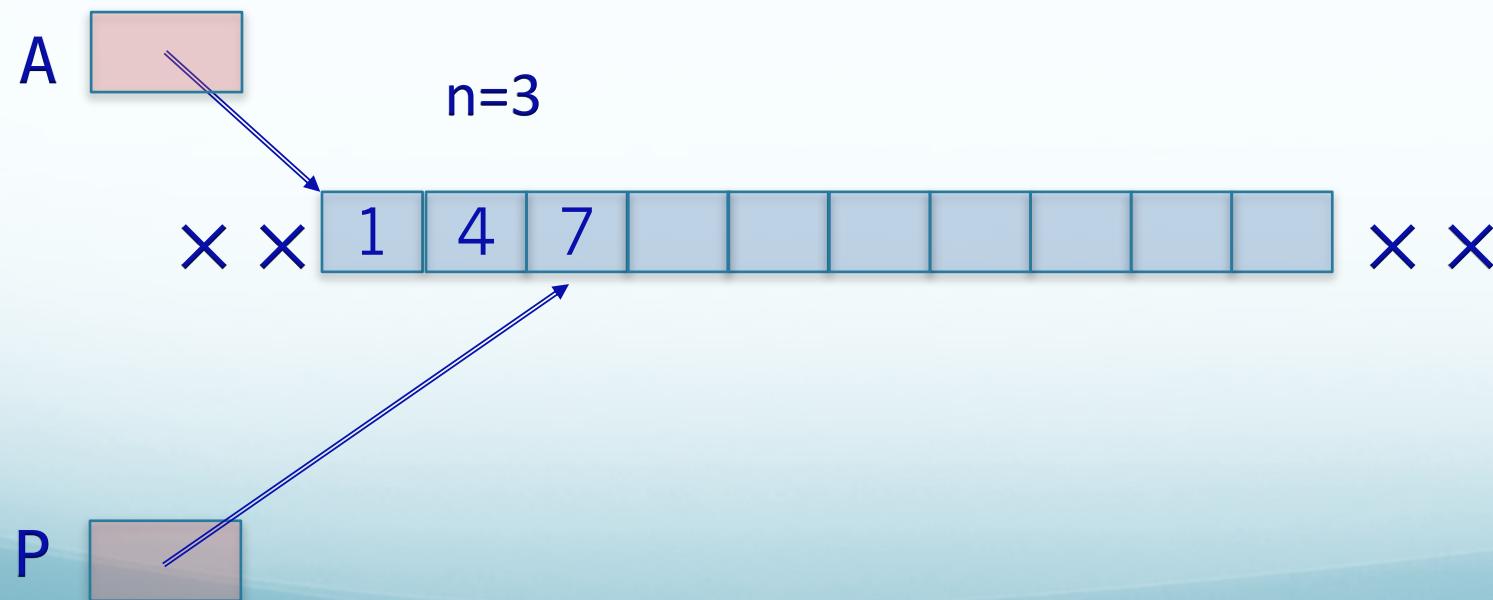
```
#define CAP 10  
int A[CAP] = {1,4,7};  
int n= 3;
```

- the system remember the constant pointer **A**, but won't remember the value **10**,
- check ourselves and don't (wrongly) use **A[-1]**, **A[10]** or **A[11]** – a very infamous **Segmentation Fault** error!



Chorus of the day: *Array names are constant pointers!*

```
#define CAP 10
int A[CAP] = {1,4,7};
int n= 3;
int *p;           // p is a variable pointer
p= A+2;          // p now is an array of capacity 8
                  // p[0] is the same as A[2] or (A+2)[0]
```



Passing an Array to a Function

same as

```
int foo(int *A, int n)
```

Example of function

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

Example: using function

```
int B[]={1,2,3}, s;  
s= foo(B, 3);
```

Pass:

- the **name of the array**
- the **current number** of elements

What function can do:

use value **n** and array **B**

modify the elements of array **B**

Check your understanding

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

With 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));
```

Discussion & DoTogether: Exercise 7.01, 7.04

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

7.04: 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 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?

spec

skeleton code

sample data & expected output

marking rubric

grok or installed editor + gcc

sample solution of 2020

Discussion Forum

testing on GradeScope

Marking Rubric: The importance of Style & Structure

Stage	Presentation	Structure	Execution	max accumulated mark
1	+5	+2	+1	8
2		+4	+4	16
3		+0	+4	20
all	5	6	9	

Failed code could even get 11

Absolutely correct program could get only 9

Correct and good Stage 1+2 alone could get 16

ASS1 Marking Rubric: a few keys in Presentation [Q&A next week]

use of magic numbers, -0.5;
#defines not in upper case, -0.5;
unhelpful **#define**, -0.5
bad choices for variable names, -0.5;
bad choice for function names, -0.5;

use **#define** for meaningful constants
#define-ed names should be in upper case
variable names in lower case (except single-letter array name)
variable names should be expressive

absence of function prototypes, -0.5;

add function prototypes before main()
main() should be the first function body!

inconsistent bracket placement, -0.5;
inconsistent indentation, -0.5;

you can use sample programs in lectures as the model

excessive commenting, -0.5;
insufficient commenting, -0.5;

each function header should have a comment
add comments for non-trivial code segment

lack of whitespace (visual appeal), -0.5;
lines >80 chars, -0.5;

use of constant subscripts in 2d arrays, -1.0;
other issue: minor -0.5, major -1.0
empty program, or helloworld, or etc, -5.0;

•comment at end that says "programming is fun", +0.5;
•overall care and presentation, +0.5;

DO IT !
Easy way to get back 0.5 or 1 mark

ASS1 Marking Rubric: a few keys in Structure

global variables, -1.0;

Global variables are NOT allowed!

main program too long or too complex, -1.0;

functions too long or too complex, -0.5;

overly complex function argument lists, -0.5;

function should not be long
and should not have too many arguments

insufficient use of functions, -0.5;

duplicate code segments, -0.5;

when having a few line, or a complicated line similarly-duplicated, think about creating a new function!

overly complex algorithmic approach, -0.5;

avoid too many levels of nesting `if` and loops
don't make the markers wonder too much to understand your code!

unnecessary duplication/copying of data, -0.5;

For example, think carefully before you copy an array!

other structural issue: minor -0.5, major -1.0;

ASS1 Marking Rubric: a few keys in Execution

failure to compile, -4.0;
unnecessary warning messages in compilation, -1.0;

Your program might be compiled OK in your computer.
But it might have compiler errors or warnings on other machines.

→ carefully read the verification report after a submission (compiler messages are at the beginning of the report)

incorrect Stage 1 layout or values in any test, -0.5 or -1.0 or -1.5;

incorrect Stage 2 layout or values in any test, -0.5 or -1.0 or -1.5;

incorrect Stage 3 layout or values in any test, -0.5 or -1.0 or -1.5;

Again, check the verification report!

When testing compare your outputs with expected outputs using command **diff**:

```
./ass1 < argyle-00050.tsv > out1.txt  
diff out1.txt test1-out.txt
```

Desirable outcome: EMPTY output from command **diff**.

If you have non-empty output:

- + The lines starting with **<** is from the first file of the **diff**
- + The lines starting with **>** is from the second file
- + You can just do the submit to see the output of **diff**

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 many from here*

Special Attention from the marking rubric:

missing Authorship Declaration at top of program, -5.0;

incomplete or unsigned Authorship Declaration at top of program, -3.0;

use of external code without attribution (minor), -2.0;

use of external code without attribution (major), -10.0;

significant overlap detected with another student's submission, -10.0;

Assignment 1: how to start

	using VS/gcc or equivalent	using grok
preparation	<ul style="list-style-type: none">make a new folder, say <code>ass1</code>copy <code>ass1-skel.c</code> to, say, <code>ass1.c</code> in the <code>ass1</code> folderopen <code>ass1.c</code> and sign in the Declarationdownload all files listed in Test data into folder <code>ass1</code>	using Assignment1
programing	fill in your code in <code>ass1.c</code>	fill in your code in <code>program.c</code>
testing	see demonstration	follow the instructions given in LMS
testing on dimefox	see demonstration	
submitting	<code>LMS → Assignments → Assignment 1 → Start Assignment → (upload .c file)</code>	

A1: Test your code by doing a submission

When: when your code is free compiler errors

How:

- Submit on GradeScope & Wait for the verification report
- Read the verification report carefully, pay attention on:
 - compiler messages at the start of the report, if any
 - the differences between your and the expected outputs

Notes:

- You can submit as many times as you want
- Submit early to avoid unexpected technical problems
- Your program might work perfectly in your computer but fail in the testing computer(s).

A1: Test your code in your system or in grok

TESTING IN YOUR COMPUTER

Using redirection when running/testing your program:

```
./myass1 < argyle-00050.tsv > test1-myout.txt
```

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

```
diff test1-myout.txt test1-out.txt
```

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

Remember:

your code might work well on the supplied data sets, but fail on some other...

Compiler warnings: -2.0

COM
P100
02.W
orksh
op A

overly complex algorithmic approach, -1.0;

Examples of overly complicated:

sort the data when not actually required

too many levels of nested loops/if

COM
P100
02.W
orksh
op A

duplicate code segments: turn similar segments into a function

When?

- having a block of ≥ 2 lines appears in more than one place?
- copying a code segment to another place and change a bit?

Solutions to avoid:

- Build a loop
- Form a new function for the duplicated part
- ...

COM
P100
02.W
orksh
op A

ASS1 Q&A:

COMP20005
.Workshop.A

LAB: do Assignment 1 OR exercises in C07

Notes:

For assignment 1, you can use VS/gcc (preferable), or grok.

There is an additional tool in grok: the Makefile. You can also copy that Makefile into your VS directory, make a change to “program”, and try.

Assignment 1: A reasonable way to start with VS/gcc

	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 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 VS <code>editor</code> to do your assignment	
9	<code>gcc -Wall -o ass1 ass1.c</code>	<i>compile the program</i>
10	<code>./ass1 < argyle-00050.tsv > test1-myout.txt</code>	<i>run program with redirection</i>
11	<code>diff test1-myout.txt test1-out.txt</code>	<i>check if your output is the same as the expected</i>

Additional Slides

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.*

*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 :example 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.