NOTEBOOK - DATA STRUCTURES

CSC 240 Data Structures Lyon College Fall 2024

Marcus Birkenkrahe

October 2, 2024

Contents

1 Week 0 - Why You Are Here

Image: Why I like programming in C.

This week we met for our first session to get (re)acquainted with one another, begin to learn about data structures, and understand what the course is about.

1.1 Why is the maximum number for a `char` type `127` and what does that mean?

It means that the computer can represent 127 characters, including lower- and upper-case alphanumeric characters like `'a'` and `'A'`, symbols like `+`, punctuation like `!`, and special characters like newline, carriage return etc.

The maximum value is a constant stored in `limits.h`:

```
printf("%d\n", SCHAR_MAX);
```

```
127
```

`127` is the maximum because the ASCII standard historically uses 7 bits - with 7 bits, each bit has two values so you can represent 2^7=128 different values, from 0 to 127.

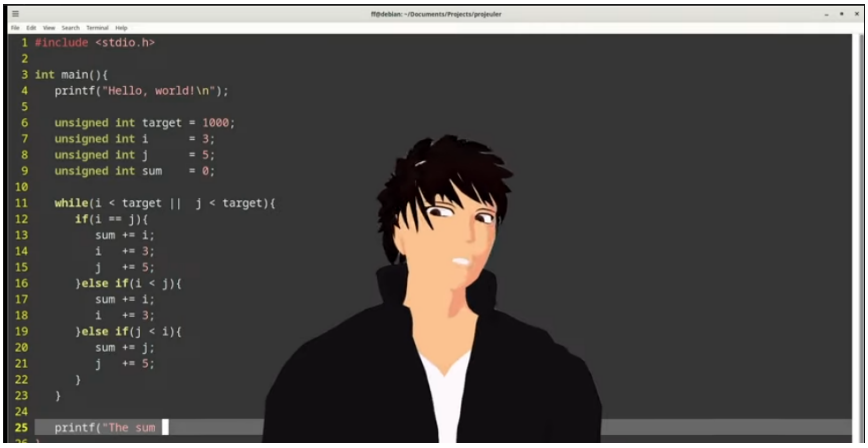
To print the ASCII value of a character, print it with the `%d` format specifier reserved for `int` (integer) values:

```
printf("%d %d %d\n",'a','A','\\'); // the `\` must be escaped to be
// printed as a character
```

```
97 65 92
```

Later, an 8-th bit was added to extend the character set to $2^8 = 256$ so that special symbols (like German umlauts ü, or the German ß) could be represented.

2 Week 1 - What Data Structures Are About



2.1 Syllabus: About Using AI to write code for you or debug your code

I'm feeling quite strongly about this - here are my views, which I also attached to the syllabus. The short version: "Don't do it."

2.2 Review questions (week 1)

1. What's the point of studying data structures?

   > To know the most efficient ways of storing, organizing and accessing data to solve a given computational problem.

2. What is a reason to use C in a course on data structures?

C is small and basic, and does not have many layers of abstractions (aka complex concepts) so that you can see and use data structures more directly through memory allocation and de-allocation.

3. Do you remember any of the differences between C, C++, and C#?

   For example:

   - C allows you to manage your computer's memory directly, and was developed in the 1970s.
   - C++ is an Object-Oriented extension of C,and was developed in the 1980.
   - C# was developed by Microsoft for commercial applications, and was developed in the early 2000s.

4. What is the computing infrastructure that we use in this course? And what does each component deliver?

   (a) GitHub - Course materials repository
   (b) Linux - Operating System
   (c) Emacs - Editor to create, document, and run source code
   (d) Canvas - Grades, assignments, tests

5. What does "computing on the edge" refer to and do you have an example?

   "Edge computing" refers to processing data close to where it is generated, such as sensors, IoT devices, or machines, instead of sending the data to a distant data center for processing. Examples: Autonomous vehicles, programming in space, and augmented reality.

6. How should you study for this course?

   (a) Code every day, create small examples
   (b) Review lecture notes and lectures on GitHub
   (c) Seek help when you need it and don't wait

7. What are C's primitive (or built-in) data types?

   - Integer (`int`)
   - Floating-point (`float` or `double`)

- Character (`char`)
- Void (`void`)

8. What does it mean that the maximum value for a `char` data type is `127`?

   It means that the computer can represent 127 characters, including lower- and upper-case alphanumeric characters like `'a'` and `'A'`, symbols like `+`, punctuation like `!`, and special characters like newline, carriage return etc.

   `127` is the maximum because the ASCII standard historically uses 7 bits - with 7 bits, each bit has two values so you can represent `2^7=128` different values, from 0 to 127.

9. What will this code print?

```
int i = 10000000000;
printf("%d\n",i);
```

```
1410065408
```

   Explain the result!

```
#include <limits.h>
printf("%d\n",INT_MAX); // Max integer: 2,147,483,647
// All 31 bits besides the sign bit are '1' = 2^31 - 1
// Any number above leads to overflow with ill results
```

```
2147483647
```

   In Python:

```
print(2**(31)-1); # 2,147,483,647
```

```
2147483647
```

10. When you see `%zu` in a C program, what do you expect?

```
const size_t INT = sizeof(int);
printf("An integer is stored in %zu bytes.\n", INT);
const size_t CHAR = sizeof(char);
printf("A character is stored in %zu bytes.\n", CHAR);
```

```
An integer is stored in 4 bytes.
A character is stored in 1 bytes.
```

2.3 Orientation II: Course Content (cont'd) & Development Tools

Figure 1: Still Life With a Volume of Wither's Emblemes by Edward Collier (1696)

- ☒ Derived data types
- ☒ Data structures
- ☒ Linux (with practice in Google Cloud Shell)
- ☒ Emacs (with practice & assignment)
- ☒ GitHub (with assignment)

2.4 Assignments (Details in Canvas) by next week (Tuesday)

The first one of these is similar to our practice in class (in Google Cloud Shell), while the second one involves many more steps and requires more independence.

1. Solve any one of the 10 programming entry problems and submit your solution as an Emacs Org-mode file, including:

   (a) A file header:

   ```
   #+TITLE: [give it a title]
   #+AUTHOR: [your name] (pledged)
   #+SUBTITLE: CSC 240 - Data Structures with C++ - Lyon College, Fall'24
   #+STARTUP: overview hideblocks indent
   #+PROPERTY: header-args:C :main yes :includes <stdio.h> :results output
   ```

   (b) An Org-mode headline with the program name, e.g. `* COOL PROGRAM`

   (c) A short description of what the program does

   (d) A code block

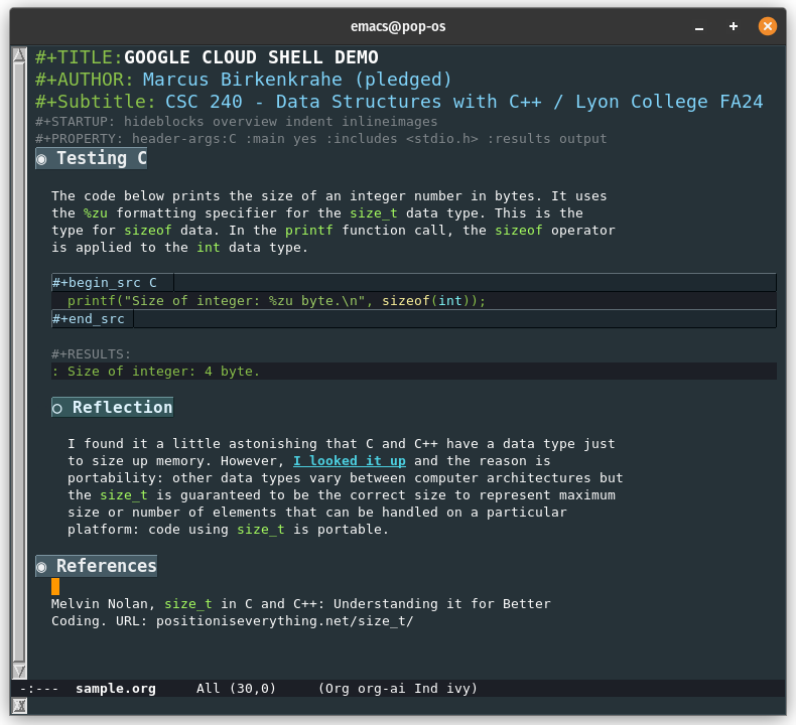
   (e) Output as requested

   (f) A short text reflecting on your experience - especially any difficulties you had during solving the exercise.

   To download the file from Google Cloud Shell, `"Open editor"` at the top, then open the explorer (`"Home directory"`), right-click on the file name and `download` it to your PC.

   If your Emacs skills are rusty, you may need to complete the Emacs tutorial (perhaps for the second time). See also the two videos that I made, linked at the start of the tutorial.

   The screenshot shows a sample file that is built exactly like the Org-mode file that you should submit (except with different code of course):



2. Complete the Hello World Project in GitHub and share a screenshot of your public GitHub repo with the hello-world repository.

   You find detailed step-wise instructions here at the end of these lecture notes in GitHub.

2.5 Review questions:

1. Examples for digital-to-analog, and for analog-to-digital conversion? Why is this relevant to data structures?

   - Digital-to-analog: Player piano
   - Analog-to-digital: Voice recording
   - Relevance: Data structures are integral to the efficient storage, processing, compression, transmission, and conversion of digital data derived from analog signals. From basic arrays that store sample data to complex trees and graphs used in compression and error detection, they ensure that the digitization process is accurate, efficient, and reliable.

2. Explain this (full sentence): `const float PI = 3.14;`

   Statement: Assign the value 3.14 to a constant floating-point variable `PI`

3. Explain this (full sentence): `#define PI 3.14`

   Preprocessor directive: Replace the expression `PI` everywhere by the floating-point value `3.14`

4. What is Google Cloud Shell?

   Google Cloud Shell is a command-line application that gives you access to a virtual Ubuntu 22.04 LTS Linux distribution.

5. What does the `gdb` debugger allow you to do?

   The `gdb` program allows you to step through your program as it runs, and get information about variables and functions.

6. What do you need to do to use `gdb`?

   You need to compile the source code with `gcc` and the `-g` flag, for example: `gcc main.c -o main -g` generates a debuggable executable `main`. Now you can start the debugger with `gdb main`.

2.6 Introduction to Data Structures (Lecture) I (handout)

3 Week 2 - The Concert in the Egg Begins

See on YouTube: The music in the painting, "Toutes les nuits que sans vous je me couche" (Every night that I go to bed without you) by Thomas Crecquillan (1549).

☒ Setup: Linux server

☒ Review: Intro to data structures

☒ Array basics

☐ Quiz 2 is live (some missed Quiz 1)

☐ Graded: first two assignments

Figure 2: The Concert in the Egg ca. 1550 AD

☐ Shared: sample solutions for C++ and for C

☐ Review: Array basics (with practice)

☐ Array as data structure (code along lecture)

3.1 Linux server has arrived

- You should have received your VM server address and password
- Start "Remote Desktop Protocol" on your (Windows) PC
- Enter the server name (e.g. `cslinux01.lyon.edu`) `> =Connect`
- Login the `Xorg` session with username `firstname.lastname`, pw
- **Do not shut down or log out of the VM but only close the window**
- Currently access only from Lyon 104 but other subnets will be added.
- IT will give me admin access and set VMs to auto-boot soon.

3.2 Review: Introduction to Data Structures I

1. Define "data structure"

   A data structures is a way of organizing and storing data in a computer using a programming language.

2. Do different programming languages have different data structures?

   Yes, different programming languages have different data structures. Some are built in, others have to be user-defined.

3. Name at least three different data structures!

   (a) Arrays, vectors, matrices
   (b) Lists, Dictionaries
   (c) Structures, classes
   (d) Enumerations, strings
   (e) Data frames, tables

4. Define "algorithm".

A step-by-step procedure to solve a problem with or without a computer, for example searching, sorting, printing.

5. What is the most common data structure in C, Python, R?

   (a) C: array
   (b) Python: list
   (c) R: vector

3.3 Introduction to Data Structures (Lecture) II

3.4 Array basics (review)

3.5 On the Programming Assignments

- This term, I am implementing a new rubric for grading your programming assignments. Check it out in the syllabus.
- The core message: complete the assignments on time and to the letter to get 100% of the points. Don't, and you lose points.
- For this first assignment, you can fix your mistakes and resubmit for full points. Only a handful completed all of the simple tasks.
- Make sure to read the full assignment and satisfy it to the letter. In this case, there was a complete sample solution available (GitHub or raw org), and all you had to do was imitate it.

3.6 Review: Intro to data structures / array basics

We'll combine this review with some light coding. Useful also for the 2nd quiz where I've used some of these examples.

- Create an .org file
- Add at the top:

  ```
  #+property: header-args:C :main yes :includes <stdio.h> :results output
  ```

- Run this line with C-c C-c
- Test it by creating a code block with: <s <TAB>
- In the code block write `printf("Hello, beautiful!");`

Figure 3: Portrait de femme au col d'hermine (Olga) - Picasso (1923)

- Run this thing: C-c C-c

1. What distinguishes the choice of different data structures?

   - Performance (speed, for example when searching through data)
   - Standardization (adherence to coding and data standards)
   - Portability (ability to run code anywhere)

2. You declare an array `A` in `main`. Where is this array stored in memory?

   In the stack, because it is a local variable that disappears as soon as the `main` function is finished (when `return 0` is reached).

   Code example:

   ```
   int main(void)
   {
     int A[5]; // array declaration
     return 0;
   }
   ```

3. Which parts of the memory are read-only?

   The memory sections for `const` "variables", and for machine code.

   Code example:

   ```
   const float pi = 3.1459; // stored in read-only memory
   pi++;  // pi = pi + 1 - generates error
   ```

4. What is the memory "heap"?

   The heap is memory reserved for "dynamically allocated" variables - variables whose memory is determined during run-time rather than compile-time. Such memory is allocated using the `malloc` function.

   Code example:

```
int *A = (int*)malloc(5 * sizeof(int)); // Allocate memory for 5 integers
// You can now assign values to A[0] through A[4]
free(A); // Free the allocated memory
```

5. What are the stages of solving a computational problem?

   (a) Understanding the problem
   (b) Identify solution (pseudocode)
   (c) Identify data structures
   (d) Implement solution
   (e) Check solution (rinse & repeat if necessary)

6. Define "Abstract Data Type" and give at least one example

   An abstract data type is a data structure together with basic operations (creation, deletion, insertion, extraction etc.)
   Examples:

   (a) user-defined classes (`dog`), methods `doTricks`, `sleep`
   (b) built-in arrays with element insertion, extraction
   (c) linked lists with item insertion, removal

7. How would you declare and initialize an integer array of five elements with zero?

```
// Explicit initialization
int A[5]; A[0]=A[1]=A[2]=A[3]=A[4]=0;
// Implicit initialization
int B[5] = {0};
// Loop initialization
int i; int C[5];
for (i = 0; i < 5; i++)
  C[i] = 0;
// Check results
int j;
for (j=0;j<5;j++) {
  printf("A[%d]: %d ", j, A[j]);
  printf("B[%d]: %d ", j, B[j]);
  printf("C[%d]: %d \n", j, C[j]);
 }
```

```
A[0]: 0 B[0]: 0 C[0]: 0
A[1]: 0 B[1]: 0 C[1]: 0
A[2]: 0 B[2]: 0 C[2]: 0
A[3]: 0 B[3]: 0 C[3]: 0
A[4]: 0 B[4]: 0 C[4]: 0
```

8. What do you get when you print undeclared array elements?

   You get 'undefined behavior'. Example: change the upper limit of the `Check results` loop from `5` to `10`.

```
// Explicit initialization
int A[5]; A[0]=A[1]=A[2]=A[3]=A[4]=0;
// Implicit initialization
int B[5] = {0};
// Loop initialization
int i; int C[5];
for (i = 0; i < 5; i++)
  C[i] = 0;
// Check results
int j;
for (j=0;j<7;j++) {
  printf("A[%d]: %d ", j, A[j]);
  printf("B[%d]: %d ", j, B[j]);
  printf("C[%d]: %d \n", j, C[j]);
 }
```

```
A[0]: 0 B[0]: 0 C[0]: 0
A[1]: 0 B[1]: 0 C[1]: 0
A[2]: 0 B[2]: 0 C[2]: 0
A[3]: 0 B[3]: 0 C[3]: 0
A[4]: 0 B[4]: 0 C[4]: 0
A[5]: 0 B[5]: 0 C[5]: 0
A[6]: 653063017 B[6]: -1075053569 C[6]: 146550272
```

9. When is the length of an array determined?

   When the length of an array is determined depends on type of array: if it is a regular array, it is determined at compile

time and cannot be changed when the program is run. If it is a variable-length array, or if it is a dynamically allocated array, its length is determined when the program is run ("at run-time").

10. When would it be useful to determine the length of an array when the program is running?

    For example when the size of the data is user-driven: if a user specifies how many numbers they want to input, the array size must be determined based on their input.

    Code example for a variable-length based array:

```
int n; // user-defined array length
scanf("%d",&n); // get the length n from keyboard
int A[n]; // Variable-Length Array
int i;
for(i=0;i<n;i++) {
  A[i]=i*i;  // assign value
  printf("%d ",A[i]); // print value
 }
```

```
0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484 529
```

    Run tangled source code file:

```
gcc main.c -o main
echo 10 | ./main
```

4 Week 3 - Array Basics

- ☒ Quiz 2 to be completed (Fri 6 Sep)
- ☒ Two new programming assignments (Tue 10 Sep)
- ☒ If you're stumped - contact me

Figure 4: M C Escher (geese)

4.1 Reviewing the Array Basics Review

1. If `A` is an array, what is `&A`?

   If `A` is an array, then `&A` is the address of `A[0]`, the first element of `A`.

```
int A[2] = {100,200}; // declare & initialize array of two elements
printf("%p\n%p", &A, &A[0]); // print address-of-A and address-of-A[0]
```

```
0x7ffdc44160a0
0x7ffdc44160a0
```

2. How is the memory of an array organized?

   The memory of an array is contiguous in memory, that is that the memory cells that hold array values are placed next to one another (this is the basis of **pointer arithmetic**). For a 2-dim array (aka matrix), the elements are stored in row-wise fashion.

```
int A[2][2] = {{1,2},{3,4}};
printf("%p %p %p %p\n",       // print addresses as hexadecimals
  &A[0][0], &A[0][1],
  &A[1][0], &A[1][1]);
printf("%d %d %d %d\n",       // print addresses as decimals
  &A[0][0], &A[0][1],
  &A[1][0], &A[1][1]);
```

```
0x7ffcabaf7190 0x7ffcabaf7194 0x7ffcabaf7198 0x7ffcabaf719c
-1414565488 -1414565484 -1414565480 -1414565476
```

3. What's pointer arithmetic?

   Arithmetic with memory positions. If `p` is an integer pointer, then `*(p + 1)` moves the pointer `p` by `1` integer (4 bytes), e.g. from memory position 100 to 104, because `sizeof(int)` is 4 bytes.

4. What's a variable-length array?

   An array whose length is determined at run-time. It cannot be initialized in the source code, only declared, and its length comes from the user when the program runs.

5. What does `:main no` mean as a header argument?

   It does not add `int main(void) { ... return 0; }` to the source code.

6. What does `:includes <limits.h>` mean as a header argument?

   It adds `#include <limits.h>` to the C source code at the top.

7. In Linux, how can you find out how much memory is available?

```
free -h  # give me the free memory for humans
```

```
               total        used        free      shared  buff/cache   available
Mem:            62Gi       6.8Gi        50Gi       107Mi       5.1Gi        51Gi
Swap:           15Gi          0B        15Gi
```

8. What is `bash`? What does it do?

   `bash(1)` is a Linux shell program. It allows you to run other programs like `gcc` or `echo` or `free`. 'Run X on the shell/in the terminal' means 'let `bash` run it'. Some commands, like `cd`

   Example:

```
gcc --version
```

9. What does "piping input into `main`" mean? Example?

   > It's a way of passing output from one program to another program. Example: in the command `echo 10 | ./main`, the number 10 is passed to a program called `main`. If `main` cannot use the number, it is simply ignored by `main`.

   An example that only works if you have a `main.c` program that takes the number `10` as a keyboard input:

   ```
   gcc main.c -o main
   echo 10 | ./main
   ```

10. What's wrong with this header argument?

    ```
    #+PROPERTY: header-args:C :main yes :includes <stdio.h> results: output
    ```

    Answer: The `results` argument needs a colon in front of it:

    ```
    #+PROPERTY: header-args:C :main yes :includes <stdio.h> :results output
    ```

4.2 Arrays as data structures (handout)

5 Week 4 - VLAs, Macros, Arrays

`../img/escher.gif`

- ☒ Emacs tip of the day: creating more than one window (C-x 5 2)
- ☒ Video + solution "Sample Array Operations" assignment
- ☒ 2 more programming assignments for arrays
- ☒ Continue with the "reverse array elements" program (code along)
- ☒ Quiz 3 is live
- ☒ Learn different ways to compile and run a source code file
- ☒ Learn Variable-Length Arrays (VLAs) for dynamic allocation
- ☒ Learn parameter macros

Figure 5: M C Escher (birds)

5.1 Programming assignments and sample solutions

Please let me know how to make these videos better (if you have an idea, and if you don't, just give me some general feedback).

Use my sample solution to create your own solution if you like and submit it late for at most 50% of the points, but make sure that you don't copy and paste but write all code by hand and understand it.

6 Week 5 - Diligence and Debugging

- ☒ Assignment 3 + 4 review
- ☒ Assignment 5 preview
- ☒ Review (and quiz 4 preview)
- ☒ Using the `gdb` debugger
- ☒ Three simple practice exercises
- ☐ Functions & Recursion

6.1 Assignment 3 review - A sermon, code, questions, and some answers

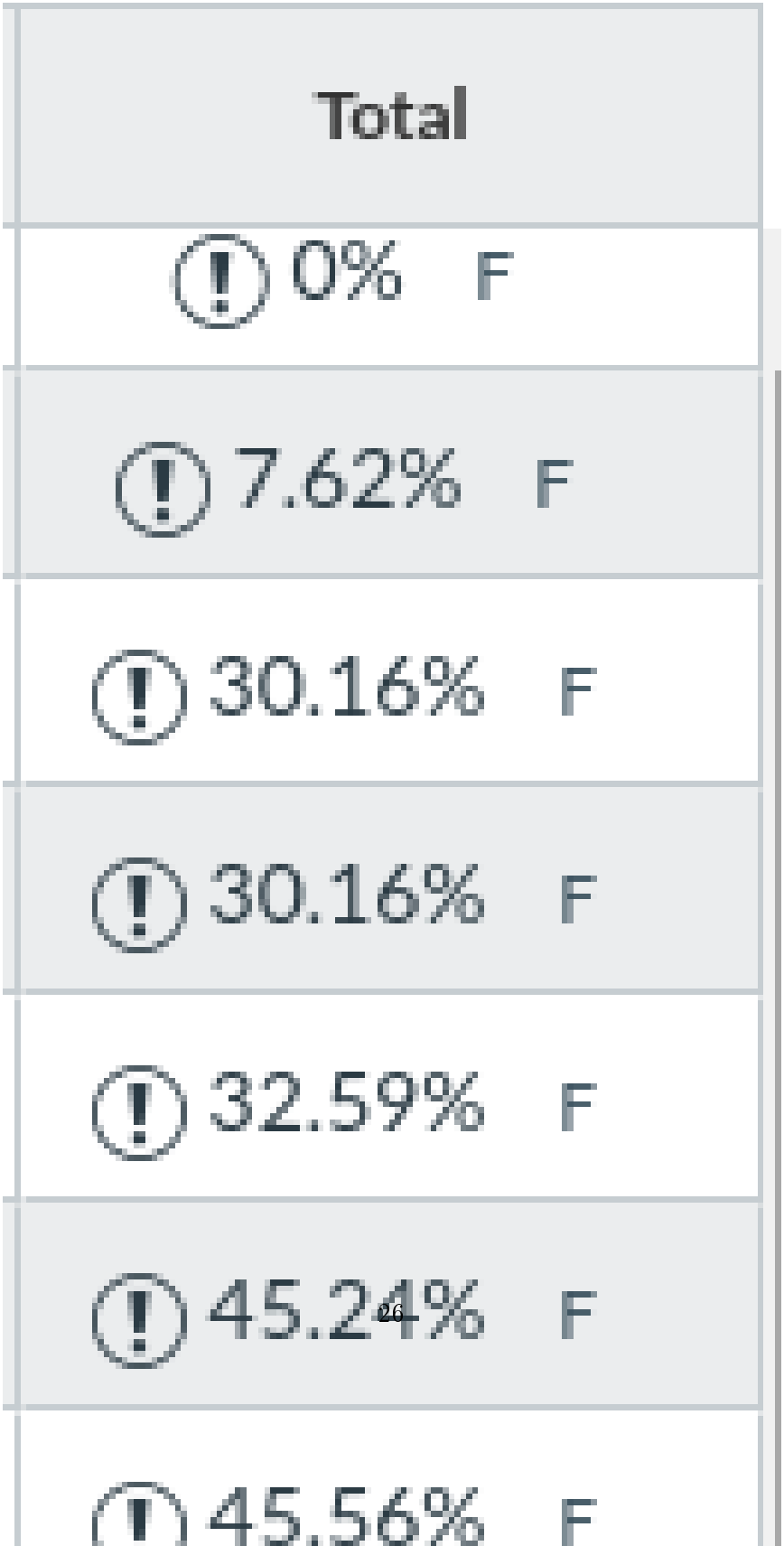
Submission

1. Org-mode file with complete meta data: `TITLE`, `AUTHOR` (with pledge), `SUBTITLE` (course information), `PROPERTY` (C code blocks) and `STARTUP` (for file startup in Emacs). Compare previous exercise.
2. Include a short description on how you tackled the problem, for example by explaining your steps (outside of the source code block).
3. Your source code should be accompanied by minimal comments to help you understand it better later. Your source code block should have the additional header argument `:tangle main.c` so that you can tangle the file with `M-x org-babel-tangle` and run it separately.
4. Your file must pass the test case, and input and output must be part of your submission.

For this assignment, I will waive the 50% rule for late submissions. If you submit a correct version within 1-2 weeks you can still get full points.

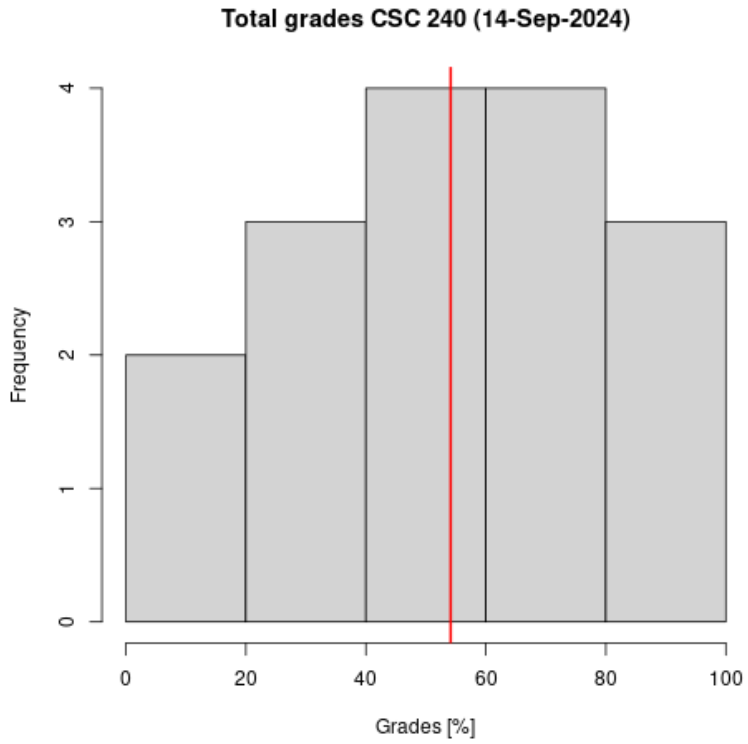
The complete problem & sample solution file is on GitHub: tinyurl.com/simple-array-operations-org - the video is on YouTube: `http://www.youtube.com/@LiterateProgramming`

6.1.1 A look at a sad,sad gradebook (snapshot, Sat 14-Sep)



1. Data science people: Plot it using R!

```
grades <- c(0.,7.62,30.16,30.16,32.59,45.24,45.56,53.02,
    59.26,62.22,63.17,65.4,76.19,98.33,98.41,98.89)
hist(grades, xlab="Grades [%]", main="Total grades CSC 240 (14-Sep-2024)");
abline(v=mean(grades),col="red",lwd=2)
```



6.1.2 Did anyone do well?

- Only one student (Austin) received a grade for a complete & correct submission.
- Everyone else either did not submit (0 points) or received feedback and was asked to resubmit within 1-2 weeks for full points.
- If you did not submit (and have no accommodations), you get at most 50% (so it's worth submitting even if you have very little).

- If you have accommodations, you've got an extra week (you might have to remind me).

6.1.3 Why can't we just submit the code?

You know the answer: I want you to engage not just with solving the problem, which is often very simple, but with the infrastructure, with the class material, and with your own problem solving abilities (or lack thereof). The raw code is something AI can produce already.

Another reason: Your code is buggy. Do you really mean for me to run your code, fix your errors, etc. Would that irritate or interest you if you were me? I believe I can spend my time a lot more useful than that. Time-wise, checking a submission & fixing & giving feedback takes no less than 15 minutes, which would be four hours per programming assignment for this class only, or 8 hours for two weekly assignment = one whole work day.

6.1.4 What areas of improvement were there this time?

1. Submission contains only the code and some accidental output.
2. The test case, which must be reproduced to the letter, is missing.
3. The reflection comes at the end and appears perfunctory.
4. The solution path is missing (usually at the start).
5. The document contains no structural elements (like headlines).
6. The code shows no indentation, and is hard to read (use C-M-\)
7. The `main` and `include` commands are already in the header but they are repeated.
8. The program is missing a header - not explicitly asked for but important: do you really want to have to read the code/comments to find out what this program does?
9. The shell code block is missing the `:results output` header.
10. Using different loop variables though there is only one type of loop (over the elements of the array).

11. The Org-mode file meta data at the top of the file are malformatted: for example '#+title' is just text - the keyword is `#+title:`, and '#+' on its own achieves nothing.

12. The documentation is not to be hidden after C comments '//' or '#+' or in a separate text file but in (ideally ahead of) the code.

All of this is perfectly clear in any of my lecture of practice files.

6.1.5 What happens if I always submit correct code but don't respect the instructions?

You know the answer: If I don't get a resubmission, you will be judged according to the rubric. Data structures is not an introductory course for people who want to see if programming is for them. It's a serious, difficult collection of topics that you need to master on your way towards becoming a computer or data science **professional**.

6.2 Which rubric?

Component	Weight	Excellent	Good	Satisfactory	Needs Improvement	Unsatisfactory
Participation and Attendance	0%	Consistently attends and actively participates in all classes.	Attends most classes and participates in discussions.	Attends classes but participation is minimal.	Frequently absent and rarely participates.	Rarely attends classes and does not participate.
Programming assignments	50%	Completes all assignments on time with high accuracy (90-100%).	Completes most assignments on time with good accuracy (80-89%).	Completes assignments but with some inaccuracies or delays (70-79%).	Frequently late or incomplete assignments with several inaccuracies (60-69%).	Rarely completes assignments and shows minimal understanding (0-59%).
Tests	25%	Demonstrates thorough understanding and application of concepts (90-100%).	Shows good understanding with minor errors (80-89%).	Displays basic understanding with some errors (70-79%).	Limited understanding with several errors (60-69%).	Minimal understanding and many errors (0-59%).
Final Exam (Optional)	25%	Demonstrates comprehensive understanding and application of course concepts (90-100%).	Shows strong understanding with minor errors (80-89%).	Displays adequate understanding with some errors (70-79%).	Limited understanding with several errors (60-69%).	Minimal understanding and many errors (0-59%).

6.2.1 Is there no other way?

Not at Lyon. You can probably create a brilliant career for yourself studying hard and with great discipline, on your own, with mostly free resources, spending time rather than money - I've

done it myself, but currently you're on a different path, and why not make the best of it?

6.2.2 What if I just didn't have enough time but wanted to submit what I had?

That's perfectly fine if you respond to my feedback within a reasonable amount of time (1-2 weeks) and resubmit with fixes. I am easy in that I will still give you full points for applying yourself - but just trying is no longer enough.

6.2.3 Why do students not respect instructions for submission?

- This is what ChatGPT has to say what the resons could be why some, not all students systematically ignore my instructions:

  There are several potential reasons why students might systematically ignore submission instructions:

  1. **Lack of Clarity or Understanding**: Students may not fully understand the instructions due to unclear language or insufficient examples. If instructions are ambiguous or overly complex, students may misinterpret them.
  2. **Overwhelming Details**: When submission instructions contain too many steps or technical details, students may feel overwhelmed and skip parts they consider less important.
  3. **Lack of Emphasis on Consequences**: If students don't perceive clear consequences for failing to follow submission guidelines, they may not see the value in adhering to them strictly.
  4. **Time Management Issues**: In a rush to meet deadlines, students may prioritize completing the task itself over ensuring the submission follows the format exactly.
  5. **Lack of Familiarity with Tools**: Students unfamiliar with tools like Org-mode or Emacs may struggle with those aspects of the submission process, especially if they haven't had enough practice or guidance.

6. **Technical Difficulties**: Issues such as trouble with software setup, tangling files, or file submission platforms might lead students to avoid following the specific instructions.
7. **Perception of Redundancy**: Students may see submission instructions as bureaucratic or unnecessary if they believe the core task (solving the problem) is more important than the format in which it's submitted.
8. **Disregard for Attention to Detail**: Some students may not yet appreciate the importance of attention to detail, particularly in fields like computer science, where strict adherence to instructions is often essential.
9. **Inadequate Feedback**: If students previously ignored submission instructions without receiving specific feedback about that, they may not realize it's a problem.
10. **Language Barriers**: For students for whom the language of instruction is not their first language, submission instructions might be harder to grasp, particularly if there's jargon or technical language involved.

Understanding these reasons can help instructors adjust their approach—whether by simplifying instructions, providing more training on tools, or emphasizing the importance of submission formatting.

Short summary:

ISSUE	ACTION
Lack of clarity	Ask! Let me know if anything is unclear
Overwhelming detail	Translate instructions for yourself
Lack of consequences	You must resubmit until you have fixed your submission
Time management	Start early, program a little daily, ask for help
Unfamiliar tools	Go through the Emacs tutorial. Code along my videos.
Technical issues	Practice more and/or install tools on your laptop
Format unimportant	Just believe me when I say it's important
No attention to detail	Switch off distractions and focus on the details
Inadequate feedback	You should have had message(s) from me about this
Language barrier	Again: ask me, or ask other students, and learn

Short response:

1. Suck it up!
2. It's for your own good!
3. No pain, no gain!

6.2.4 What else will you do to help us?

- I will make a video of my sample solution, which you can use to improve your game, identify your mistakes, and help me make better videos.
- I will continue to give you detailed feedback so that you can fix your errors. I usually "reassign" the assignment without giving 0 points. In this way your grade does not suffer but if I don't get a resubmission within 1-2 weeks, I will give you partial points (50%).
- Talk to me if you're unhappy about this and we can try to find a solution that satisfies both my and your needs!

6.2.5 Any questions?

You must learn to ask questions, ideally in the class: most problems that I've encountered are shared among students. Help others and they will help you!

6.3 Assignment 4 review

Submission

1. Org-mode file with complete meta data: `TITLE`, `AUTHOR` (with pledge), `SUBTITLE` (course information), `PROPERTY` (C code blocks) and `STARTUP` (for file startup in Emacs). Compare previous exercise.
2. Include a short description on how you tackled the problem, for example by explaining your steps (outside of the source code block).
3. Your source code should be accompanied by minimal comments to help you understand it better later. Your source code block should have the additional header argument `:tangle main.c` so that you can tangle the file with `M-x org-babel-tangle` and run it separately.
4. Your file must pass the test case, and input and output must be part of your submission.

For this assignment, I will waive the 50% rule for late submissions. If you submit a correct version within 1-2 weeks you can still get full points.

The complete problem & sample solution file is on GitHub: tinyurl.com/array-temperature-analysis-org - a short video (v1 only) is on YouTube: `http://www.youtube.com/@LiterateProgramming`

Nothing new - almost everybody made the same mistakes (3/9 submissions got full points). I noticed that many of you seem to submit 5 minutes before the deadline. That's not leaving yourself enough time for something that requires a lot of diligence.

6.4 Review: VLA, Macros & Quiz 4 Preview

1. How would you compute the course average using VLAs in C! Here are the grades for you to copy: tinyurl.com/grades-fall24

```
echo 7.62 30.16 30.16 32.59 45.24 45.56 53.02 59.26\
62.22 63.17 65.4 76.19 98.33 98.41 98.89 > ../data/grades
cat ../data/grades
```

```
7.62 30.16 30.16 32.59 45.24 45.56 53.02 59.26 62.22 63.17 65.4 76.19 98.33 98.41
```

This is really easy in R:

```
df <- read.csv("https://tinyurl.com/grades-fall24",sep=" ",header=FALSE)
df
mean(read.csv("https://tinyurl.com/grades-fall24",sep=" ",header=FALSE))
```

```
V1    V2    V3    V4    V5    V6    V7    V8    V9   V10  V11   V12   V13
1 7.62 30.16 30.16 32.59 45.24 45.56 53.02 59.26 62.22 63.17 65.4 76.19 98.33
    V14   V15
1 98.41 98.89
[1] NA
Warning message:
In mean.default(read.csv("https://tinyurl.com/grades-fall24", sep = " ",  :
  argument is not numeric or logical: returning NA
```

```
int i, n;
float grades[n], sum=0.;
scanf("%d",&n);
for (int i=0;i<n;i++) {
  scanf("%f",&grades[i]);
  printf("%g ",grades[i]);
  sum+=grades[i];
 }
printf("\nAverage = %g\n",sum/n);
```

```
1.4013e-45 0 0 0 1.4013e-45 0 2.23002e+13 4.25266e-41 2.23054e+13 4.25266e-41 2.23
Average = -nan
```

```
gcc main.c -o main && echo "16 0. 7.62 30.16 30.16 32.59 45.24 45.56
53.02 59.26 62.22 63.17 65.4 76.19 98.33 98.41 98.89" | ./main
```

You can also get this from the URL with `wget`:

```
echo 16 > grades2
wget -q -O - https://tinyurl.com/grades-fall24 >> grades2
cat grades2
```

```
16
7.62 30.16 30.16 32.59 45.24 45.56 53.02 59.26 62.22 63.17 65.4 76.19 98.33 98.41
```

Now you can redirect the file `grades2` into the executable `./main`

```
gcc main.c -o main && ./main < grades2
```

Incidentally, you can do this also in the Org-mode code block using the header argument `:cmdline < grades2`, and then you don't have to tangle the source code:

```
int i, n;
float grades[n], sum=0.;
scanf("%d",&n);
for (int i=0;i<n;i++) {
  scanf("%f",&grades[i]);
  printf("%g ",grades[i]);
  sum+=grades[i];
 }
printf("\nAverage = %g\n",sum/n);
```

```
7.62 30.16 30.16 32.59 45.24 45.56 53.02 59.26 62.22 63.17 65.4 76.19 98.33 98.41
Average = 54.1387
```

In future assignments with user input, you can pick your poison - `bash` code block input with pipe command, or `cmdline` file input.

2. What constraints are place on VLA declaration?

   Variable-length arrays get their length at run-time so they cannot be initialized at compile-time.

3. In an array `a[10]`, what is the difference between `sizeof(a[0])` and `sizeof(a[10])`, and how would you show this?

```
int a[10]={0};
printf("Size of a[0] = %zu\nSize of a[10] = %zu\n",
  sizeof(a[0]),       sizeof(a[10]));
```

```
Size of a[0] = 4
Size of a[10] = 4
```

4. How would you convert the following selection statement into a parameter macro?

```
if (x < y)
   x
else
   y
```

```
#define MIN(x,y) ((x)<(y)?(x):(y))

int x = 100, y = 200;
printf("min(%d,%d) = %d", x,y,MIN(x,y));
```

```
min(100,200) = 100
```

5. What if you were tired of writing `sizeof(a)/sizeof(a[10])` - could you write it as a parameter macro?

```
#define LEN(a) sizeof(a)/sizeof(a[10])

int a[10];
printf("Length of array: %zu\n", LEN(a));
```

```
Length of array: 10
```

6. What if you don't want to use `0` and `1` for `false` and `true`?

   > If you want to use `true` and `false`, `#include <stdbool.h>`.
   > Now you can declare a `bool` data type.

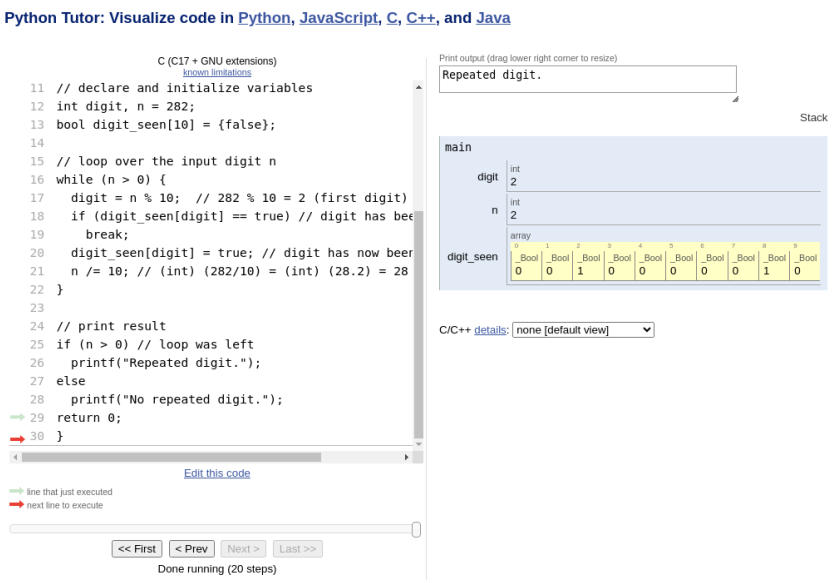
   Example:

```
#include <stdbool.h>

bool beauty = true;
bool ugly = false;

printf("If beautiful equations are %s,\nthen ugly equations are %s.\n",
  beauty ? "true" : "false",
  ugly   ? "true" : "false");


If beautiful equations are true,
then ugly equations are false.
```

7. When using AI to help you solve an assignment, what must you do?

   > When using AI assistance in any way, I suggest you mention
   > in your solution that, and how, you made use of it (and if it
   > helped you).

6.5 Using the C debugger `gdb`

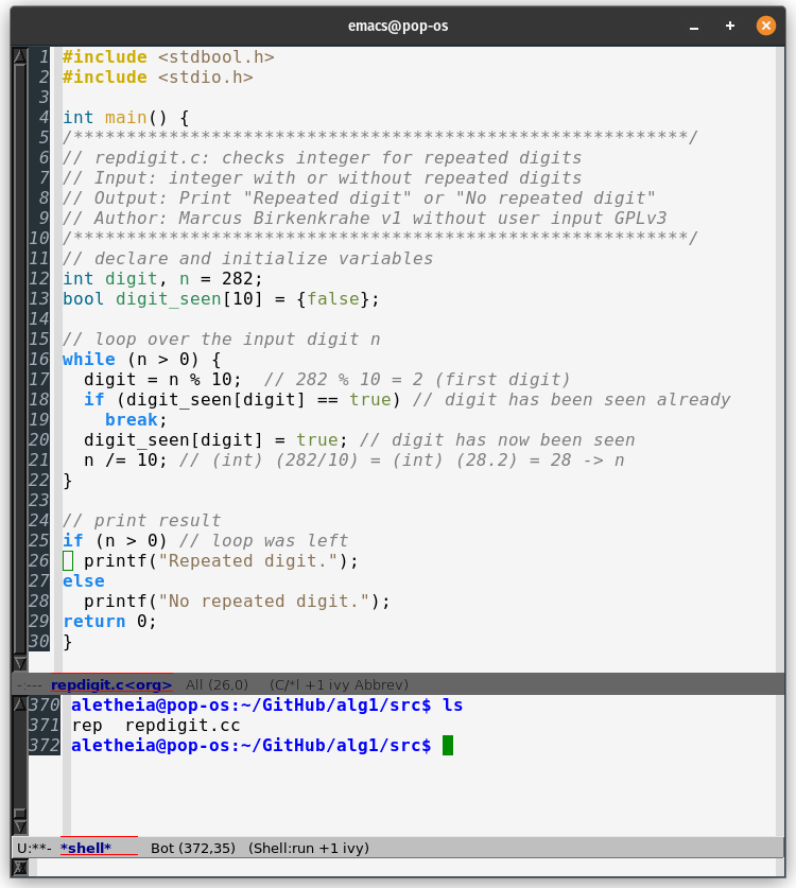


- Get the raw file from tinyurl.com/repdigit-c on the shell (`M-x shell`

```
wget -O repdigit.c tinyurl.com/repdigit-c
ls -l repdigit.c
```

```
-rw-rw-r-- 1 aletheia aletheia 885 Oct  2 20:50 repdigit.c
```

- Open the source code file with `M-x global-linum-mode`
- Split the screen into shell and source code file:



- Compile the file with the `-g` flag:

```
gcc repdigit.c -o rep -g
ls -l rep
```

```
-rwxrwxr-x 1 aletheia aletheia 17384 Oct  2 20:50 rep
```

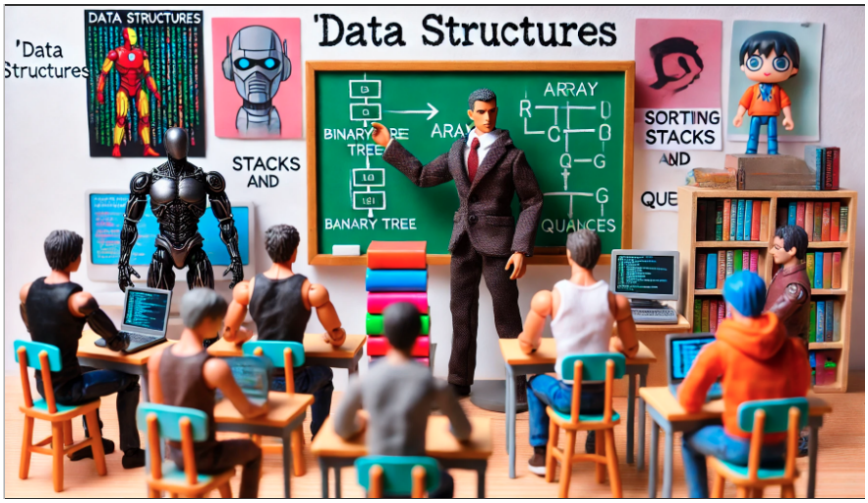
- Run it through the `gdb` debugger with: `gdb rep`
- Commands to try in `gdb`:

```
(gdb) run
(gdb) help
(gdb) info locals
```

```
(gdb) break 23
(gdb) step
(gdb) print n
(gdb) continue
```

- Compare with `pythontutor.com`:
  1. Choose `C`
  2. Paste `repdigit.c` into editor
  3. Visualize execution
  4. Step through program with `Next`

6.6 Three practice exercises



Solve these exercise in class and upload your results for bonus points to Canvas:

- Boolean array declaration
- Array of Fibonacci numbers
- Initialize and print matrix

  Solutions to the exercises are in GitHub in the PDF directory.

7 Week 6 - Function Prototypes and Parameters

Image: Four monks by Claudio Rinaldi (1852-1909). (Dorotheum, Munich)

Topics:

☒ Warm-up program

☒ Function prototypes [assignment 6]

☒ Arguments vs. Parameters

☒ Passing arrays (by reference [assignment 7]

☐ Compound literals

☐ Return and exit

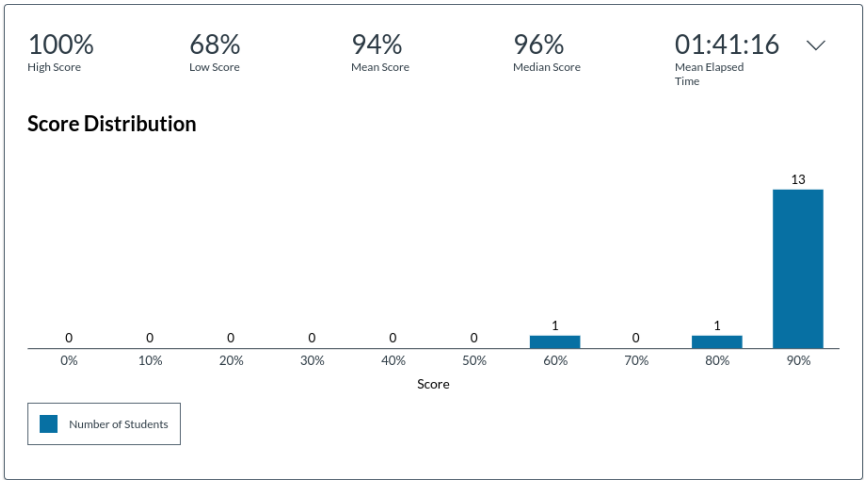
☐ Recursion

☐ Quicksort algorithm

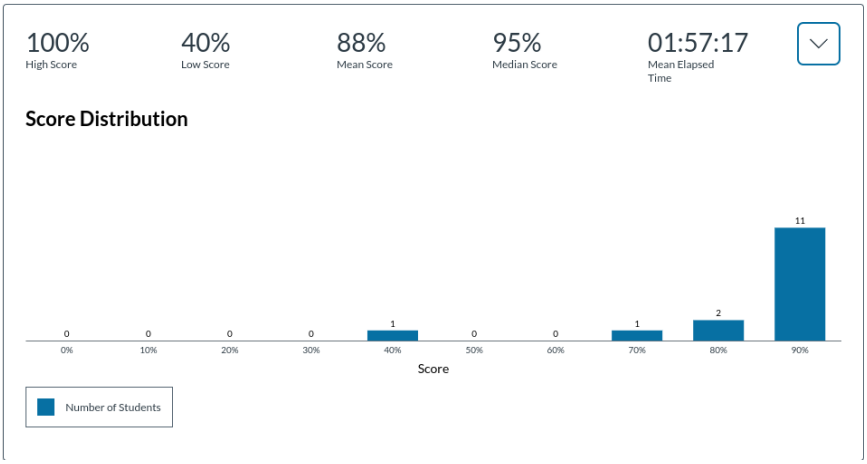
Every topic will generate 1 home programming assignment!
Only 2 assignments per week will be mandatory, the others: bonus!

- Turn warmup program into program with 3 functions
- Write function headers for functions seen in class

7.1 Results: Quiz 3 + 4



Quiz 3



Quiz 4

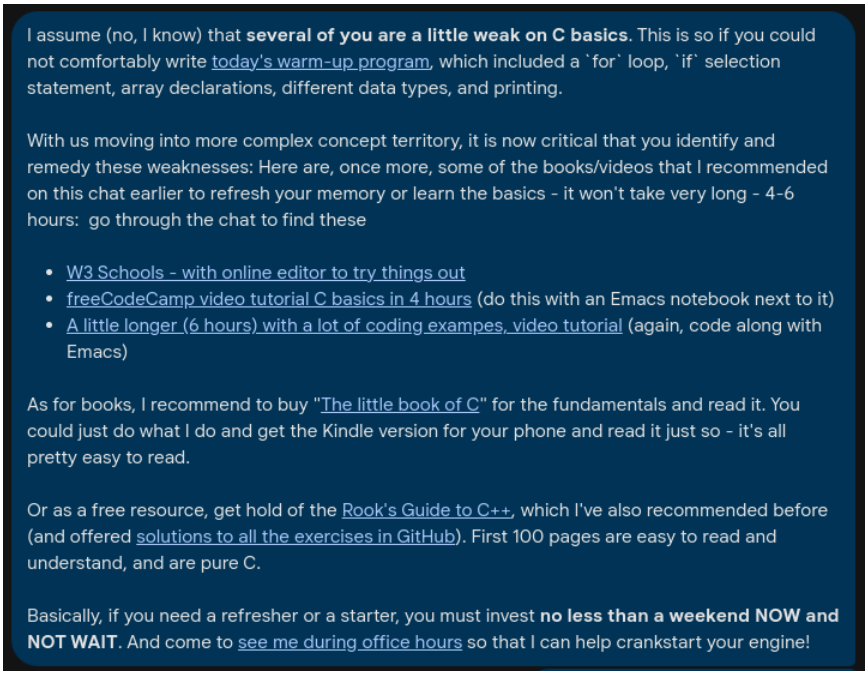
- Good: Almost everybody turns in the quiz on time!
- Good: Consistent performance (one outlier only)
- OK: Average time 12-17 minutes (one extreme outlier)

Figure 6: Detail from Raphael, The School of Athens (1511)

7.2 Results: Programming assignment

- No update for the last two assignments yet: **If you haven't resubmitted yet and are at a loss: contact me!** (You got till Friday)
- Some have mended their submission ways as discussed
- Some have not (why? Time? Reassignment = rehabilitation)
- More than 40% have not submitted at all (why? Time?)
- **Submit what you have before the deadline to get a 2nd chance**!
- If you like watch my 30 minute solution video (new style) - 5 views

7.3 Function prototypes, arguments vs. parameters



I assume (no, I know) that **several of you are a little weak on C basics**. This is so if you could not comfortably write today's warm-up program, which included a `for` loop, `if` selection statement, array declarations, different data types, and printing.

With us moving into more complex concept territory, it is now critical that you identify and remedy these weaknesses: Here are, once more, some of the books/videos that I recommended on this chat earlier to refresh your memory or learn the basics - it won't take very long - 4-6 hours: go through the chat to find these

- W3 Schools - with online editor to try things out
- freeCodeCamp video tutorial C basics in 4 hours (do this with an Emacs notebook next to it)
- A little longer (6 hours) with a lot of coding exampes, video tutorial (again, code along with Emacs)

As for books, I recommend to buy "The little book of C" for the fundamentals and read it. You could just do what I do and get the Kindle version for your phone and read it just so - it's all pretty easy to read.

Or as a free resource, get hold of the Rook's Guide to C++, which I've also recommended before (and offered solutions to all the exercises in GitHub). First 100 pages are easy to read and understand, and are pure C.

Basically, if you need a refresher or a starter, you must invest **no less than a weekend NOW and NOT WAIT**. And come to see me during office hours so that I can help crankstart your engine!

YOU MUST ADDRESS YOUR C BASICS WEAKNESSES NOW

("... or forever hold your peace").

7.4 Review (Function prototypes)

1. What's the purpose of a function prototype?

   The purpose of a function prototype is to declare a function to the compiler by declaring the header (return type, name, parameters) without the implementation (the function body).

2. What's a function definition?

   A function definition is a fully implemented function, including a header (return type, name, parameters), and a body (code to be executed when the function is called).

3. Where can you define a function in a C program?

- If you declared its prototype before `main`, the function can be defined anywhere.
- If you did not declare a prototype, the function must be defined before or inside the `main` function.

4. The function is defined as `void f(int a, int b)`. Can the function prototype be declared like this?

```
void f(int, int);
```

   Yes. The prototype declaration does not need to contain parameters, only parameter data types.

5. Can a `void` function have a `return` command?

   No. If it's a `void` function that means that it returns nothing. A `return 0;` command will compile but generate a warning.

```
void hello(void)
{
  printf("hello");
  return 0;
}
```

6. How should you call a function with a `void` parameter list?

   If it has a `void` or empty `()` parameter list, it is called without arguments.

```
// function definition
void hello(void)
{
  printf("hello");
}
// main program
int main (void)
{
  hello();
  return 0;
}
```

```
hello
```

7. Will the following code compile?

```
// prototype
void f(int a);
// main program
int main (void)
{
  float b;
  f(b);
  return 0;
}
// function definition
void f(int a) {}
```

8. What does "passing arguments by value" mean?

   "Passing arguments by value" means that arguments after a function call are copied so that the original arguments are not changed.

9. What do you need to change when your code block has a `main`?

   You need to set the header argument `:main` to the value `no`.

10. How can you "break" a function without violating Syntax rules?

    - Mismatched `return` type between declaration and `return` argument
    - Mismatched `return` type between declaration and definition
    - Missing `return` command though a `return` type is declared
    - Argument mismatch
    - Modifying `const` parameters inside the function

8 Week 7 - Functions, Recursion and Pointers

Announcements:

☒ **New!** 12-minute podcast on "C Arrays" (see chat) - let me know what you think! I've already turned "C Functions" into a podcast - fun review!

☐ **Poll!** Who has got **WSL** (Windows Subsystem Linux) on his Windows PC? Enrol my help to do this (same environment as in class)!

☐ **New!** 6-minutes podcast on "C functions (see chat) - let me know what you think! (Based on a 46p. script)

☐ Two bonus programs available right now (complete by Oct-7).

Next topics:

☒ Passing by reference

☐ Multi-dimensional arrays

☐ Compound literals

☐ Return and exit

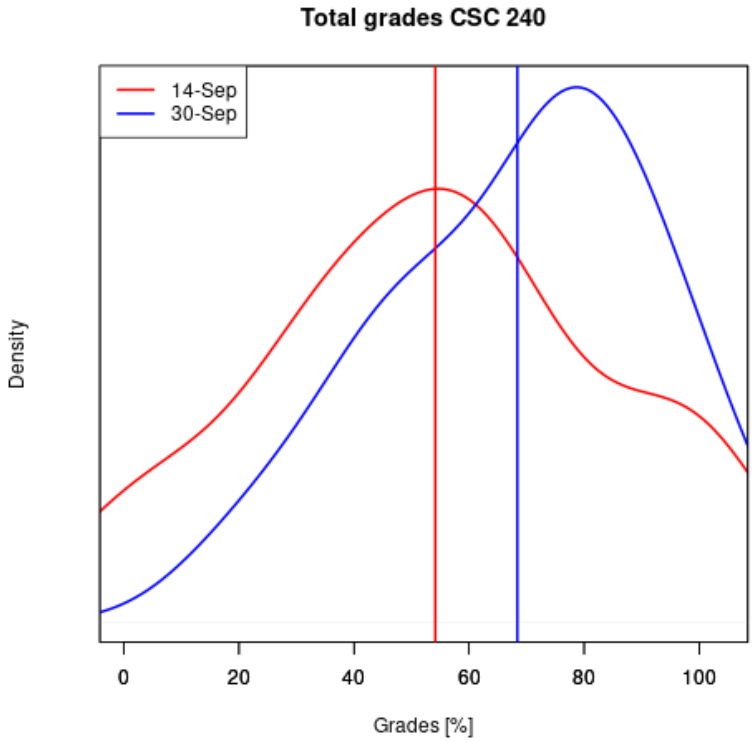
☐ Recursion

☐ Quicksort algorithm

8.1 DONE Programming assignments are getting better!

- R code chunk (if you like how this looks, get into data science!)

```
grades1 <- c(0.,7.62,30.16,30.16,32.59,45.24,45.56,53.02,
       59.26,62.22,63.17,65.4,76.19,98.33,98.41,98.89)
grades2 <- c(20.56,35.42,42.78,48.75,52.78,57.5,69.86,70.42,
       75.,76.11,81.53,83.19,86.39,91.25,98.47,104.29)

## Get the common x and y limits
xlim <- range(c(grades1, grades2))
ylim <- range(c(density(grades1)$y, density(grades2)$y))

## First plot with specified xlim and ylim
plot(density(grades1), lwd=2, col="red",
     xlab="", ylab="", main="", xlim=xlim, ylim=ylim, yaxt="n")

## Overlay the second plot with the same xlim and ylim
par(new=TRUE)
plot(density(grades2), lwd=2, col="blue",
     xlab="Grades [%]", main="Total grades CSC 240",
     xlim=xlim, ylim=ylim, yaxt="n")

## Add mean lines for grades1
abline(v=mean(grades1), col="red", lwd=2)

## Add mean lines for grades2
abline(v=mean(grades2), col="blue", lwd=2)

## add a legend
legend("topleft", legend=c("14-Sep", "30-Sep"),
 col=c("red", "blue"), lwd=2)
```



- Many of you still have not submitted assignment 2 (= **low hanging fruit**).
- Some of you have not submitted the last 2-3 assignments (**see me!!!**).
- Some of you are still struggling with **sticking to instructions**.
- From now on I will grade what you submit **as I see it**.
- Late submissions will get **at most 50%** of the available points.
- **Late submissions for mid-term must be submitted by 12 pm on 8 Oct**
- You can always send me your solution for comment before submitting.
- Use "Hurkle-Durkle-Day" weekend to **improve your grade & skills**.
- Extended deadline for assignments 6 + 7 is **Mon 7 October (11:59pm)**
- I will video my sample solutions & create a podcast on them.

8.2 DONE Review (Passing by value, passing arrays)

1. What should be included in a function documentation header?

   Name, purpose, return type, parameters, edge cases/use.

   Example:

   ```
   // Name: power
   // Purpose: Compute power of an integer
   // Returns: integer = n-th power of integer x
   // Params: integer (base) x, integer (power) n
   ```

2. What does "passing an array by reference" mean?

   Array arguments are inherently passed by reference.
   Instead of passing a copy of the array, a pointer to the first element of the array is passed to the function.

3. Do changes made to the array elements inside the function affect the original array in the calling function?

   Yes, any changes made to the array elements inside the function affect the original array:

   (a) When an array is passed to a function, it "decays" into a pointer to its first element - only the address of that element is received.
   (b) Modifications to the array elements change the original array, since no copy was made and the pointer operates on the original.

4. How does the function call to pass an array `a[10]` look like?

   ```
   int a[10]; // array declaration
   f(a,10);  // function call
   ```

5. Show this with a short example including `main` and a function!

   (a) Prototye `void` function `foo` with array parameter
   (b) In `main`: Define 1-dimensional array `bar[2]` and init to `{0}`
   (c) In `main`: Print array elements

(d) In `main`: Call function on array

(e) In (`void`) function: change elements in array to `{1}`

(f) In main: Print array elements

Solution:

```
void foo(int [], int); // 1

int main(void)
{
  int bar[2]={0}; // 2
  printf("%d %d\n",bar[0],bar[1]); // 3
  foo(bar,2); // 4
  printf("%d %d\n",bar[0],bar[1]); // 5
  return 0;
}

void foo(int a[], int n)
{
  a[0] = a[1] = 1;  // 4
}


0 0
1 1
```

8.3 TODO Review (Passing by value vs. reference)

1. In the `decompose` program, we split a `float` into an `int` and another `float` - which format specifiers are used to print these?

```
float x = 3.14159;
float frac_part;
int int_part;

// split x
int_part = (int) x;
frac_part = x - int_part;

// print results
printf("x = %g, i: %d, d: %g\n", x, int_part, frac_part);
```

```
x = 3.14159, i: 3, d: 0.14159
```

2. What if we wanted to split a `double` floating-point instead?

```
double x = 3.141592653589793;
double frac_part;
long int_part;

// split x
int_part = (long) x;
frac_part = x - int_part;

// print results
printf("x = %.15f, i: %ld, d: %.15f\n", x, int_part, frac_part);
```

```
x = 3.141592653589793, i: 3, d: 0.141592653589793
```

3. If `*p` is a pointer to `&i`, the address of the `int` variable `i`, what are the following variables, and are they even allowed?

```
&(*p)
*(&i)
*(&(*p))
&(*(&i))
```

```
int i = 100; // integer
int *p = &i; // pointer to integer

// integer, address-of integer, pointer
printf("i        = %d, &i       = %d, p = %d\n", i, &i, p);
// dereferenced pointer, address-of dereferenced pointer
printf("*p       = %d, &(*p)    = %d\n", *p, &(*p));
// deferenced reference/address-of integer (value)
printf("*(&i)    = %d\n", *(&i));
// dereferenced address-of dereferenced pointer (value),
// address-of dereferenced address-of integer (address
printf("*(&(*p)) = %d, &(*(&i)) = %d\n",*(&(*p)),&(*(&i)));
```

```
i        = 100, &i       = -993424932, p = -993424932
*p       = 100, &(*p)    = -993424932
*(&i)    = 100
*(&(*p)) = 100, &(*(&i)) = -993424932
```

4. What could `**p` be if `*p` is a pointer to `&i`? Is that allowed?

   `*(*p)` is a pointer to a pointer (aka "double pointer"): it contains the memory address of the pointer (which in turn points at the memory address of `i`.
   Double pointers give you control over both `p` and `i` at different levels.

5. Change where `p` points to, and the value of `i` using double pointers:

   (a) Start with `int i = 100; int* p = &i;`
   (b) Define pointer `pp` that points to `p`
   (c) Print `*p` = *(&i) = i
   (d) Print `*pp` = *(&p) = *(&(&i)) = &i
   (e) Print `**pp` = *(*pp)= *(&i) = i
   (f) Add 100 to `**p` and print it
   (g) Define `int j = 300`
   (h) Redirect `pp` to `j`: **pp = *(&p) = p = &j
   (i) Print `*p` = *(&j) = j

```
int i = 100;
int* p = &i; // p points to i (address)
int** pp = &p; // pp points to p (address)

printf("i    = %d\n",i); // integer value
printf("p    = %d\n",p); // pointer to i
printf("*p   = %d\n",*p); // dereference p = value of i (value)
printf("*pp  = %d\n",*pp); // dereference pp = value of p (address)
printf("**pp = %d\n",**pp); // dereference *pp = value of *p (value) = i

(**pp) += 100; // add 100 to **pp = i
printf("**p  = %d, i = %d\n",**pp,i); // dereference *pp = value of *p (value)
```

```
int j = 300;
(*pp) = &j; // pp points to p which points to j
printf("p now points to: %d\n", *p);


i    = 100
p    = -1365510320
*p   = 100
*pp  = -1365510320
**pp = 100
**p  = 200, i = 200
p now points to: 300
```

6. Coding challenge:

   (a) 6 generates the output: 0 0

   (b) Copy the following code in 6.

   (c) Copy the code block to a new block 6.

   (d) Modify the function to pass-by-reference:

   - The function call passes a reference (address)
   - The function parameter becomes a pointer (pointing to `n`)
   - The function body uses pointers

   (e) 6 generates the output: 0 1

   Increment a number with a pass-by-value function:

```
void increment(int num)
{
  num++;
}
int main(void)
{
  int n = 0;

  printf("%d\n", n);
  increment(n);
  printf("%d\n", n);

  return 0;
}
```

```
0
0
```

Increment a number with a pass-by-reference function:

```
void increment(int* num)
{
  (*num)++;
}
int main(void)
{
  int n = 0;

  printf("%d\n", n);
  increment(&n);
  printf("%d\n", n);

  return 0;
}
```

```
0
1
```

7. Prove 'algebraically' that *p = i if p = &i - what is *p?

   `*p = *(&i) = *&(i) = i since *& = 1`

   If p is a pointer to i, then the dereferenced p is the value that p points to, i.

8. Prove 'algebraically' that **pp = *p if pp = &p - what is **pp?

   `**pp = **(pp) = **(&p) = *(*& p) = *(p) = *p`

   If pp is a pointer to p, then the double pointer **pp is the value that p points to.

9. Could you modify 6 to increment a 1-dim array?

```
void increment2(int num[], int n)
{
  num[0]++;
  num[1]++;
}
int main(void)
{
  int n[2] = {0};

  printf("%d %d\n", n[0], n[1]);
  increment2(n,2);
  printf("%d %d\n", n[0], n[1]);

  return 0;
}
```

```
0 0
1 1
```

10. In the previous example, why are the array values changed in `main`?

    The array values in `main` are changed because arrays are always passed by reference: when passed, the array decays into a pointer to its first element, which points at the original, not at a copy. Consequently, when changing array values in the function, they are changed everywhere.