CC Agenda

CSC 100 - Spring 2022

Table of Contents

1 README

This file contains the agenda overview (what I had planned), the objectives (what we managed to do) and (much of the) content of each taught session of the course. I want to avoid splitting the content up over many files - so that you have to navigate as little as possible (like a book)!

The companion file to this file, less structured and with the captain's log, is the notes.org file.

2 Welcome to the course - w1s1 (01/12/22)

2.1 Welcome

- Introduction to the course & the lecturer
- Homework assignment: GitHub Hello World
- What's next?

2.2 Entry survey (Google Forms)

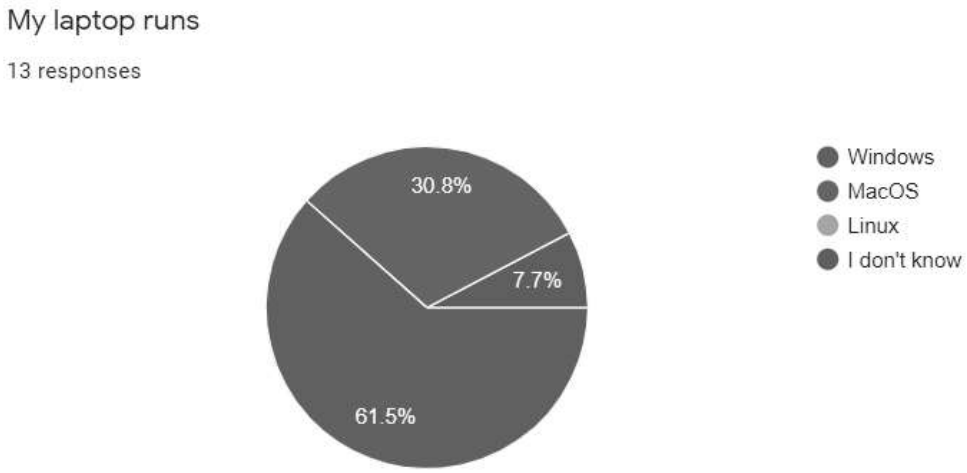
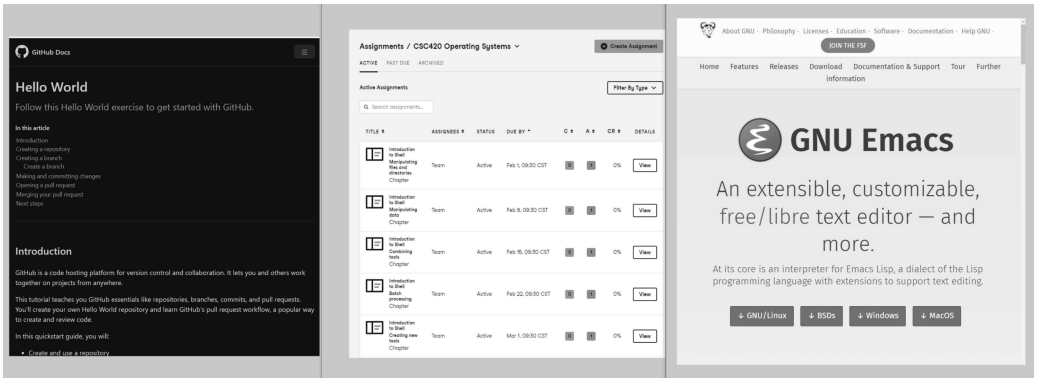


Figure 2: What's your operating system? (Spring 2022 survey)

2.3 Introduction to the course & the lecturer

- PhD theoretical particle physics / WWW development
- C/C++ since ca. 1990 (created multigrid library)
- Professor, Business Informatics @Berlin Univ
- Visiting Assoc Prof for Data Science @Lyon (2021-23)
- Syllabus for this course (Schoology)

2.4 Homework assignments week 1 (14-Jan-2022)



- **GitHub Hello World Exercise (Info: FAQ) - by Friday 14-Jan!**

2.4.1 GitHub

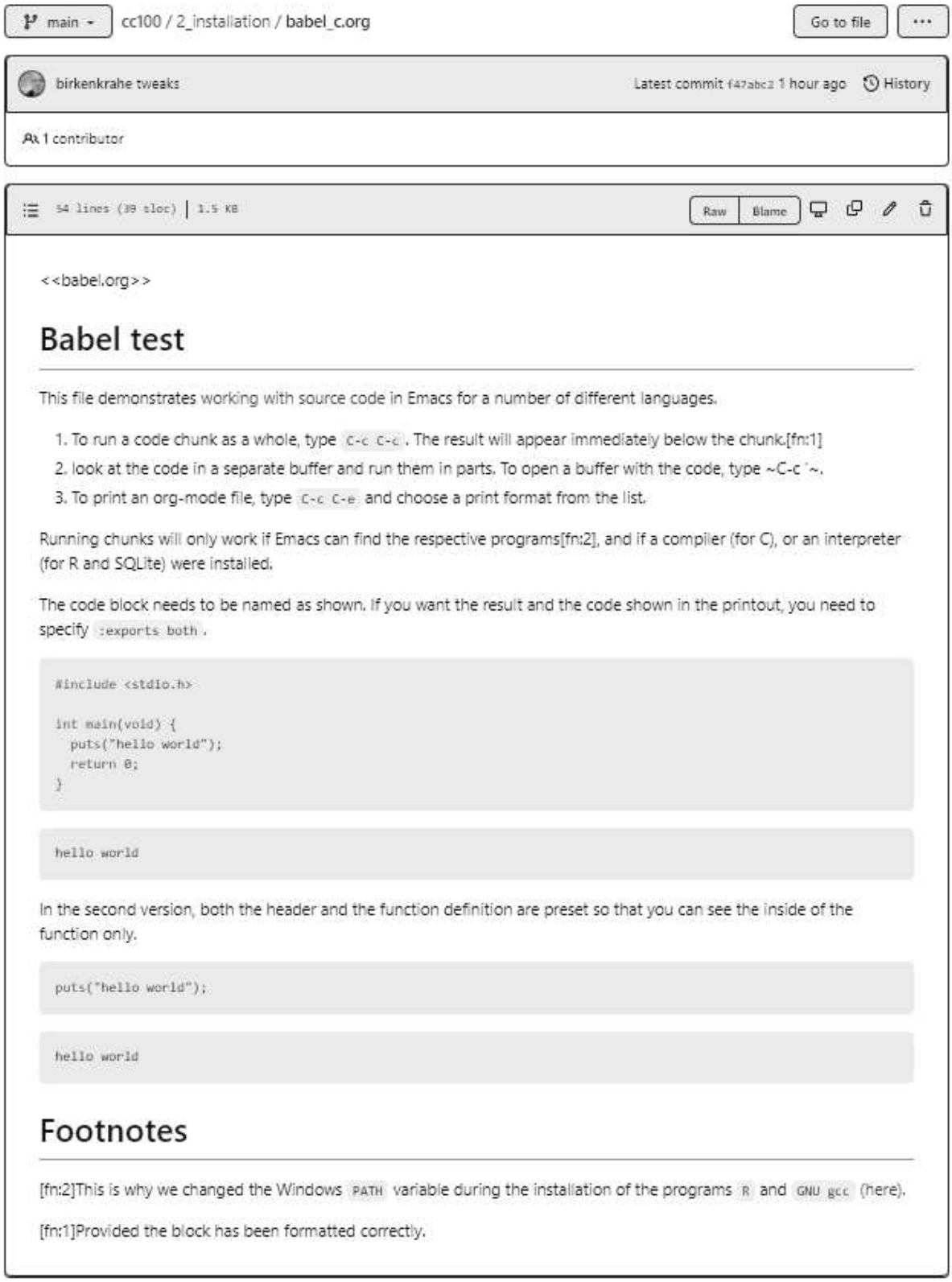
1. What is it?
    - Software development platform (like GitLab, BitBucket, SourceForge, etc.)
    - Built around Git by Linus Torvalds
    - Bought by Microsoft in 2018 (like OpenAI - home of GPT3)
    - 77 mio users (developers) + 200+ mio software projects
    - AI support (e.g. GitHub Copilot - AI-enabled code generator)

Watch: "What is GitHub?" (GitHub, 2016)

Gif: "So long binder of requirements" Source: GitHub

2. Why are we using it?

Image: Org-mode file in GitHub



main ▾ cc100 / 2_installation / babel_c.org Go to file ···

birkenkrahe tweaks Latest commit f47abc2 1 hour ago History

1 contributor

54 lines (39 sloc) | 1.5 KB Raw Blame

<<babel.org>>

Babel test

This file demonstrates working with source code in Emacs for a number of different languages.

1. To run a code chunk as a whole, type `C-c C-c`. The result will appear immediately below the chunk.[fn:1]
2. look at the code in a separate buffer and run them in parts. To open a buffer with the code, type ~C-c '~.
3. To print an org-mode file, type `C-c C-e` and choose a print format from the list.

Running chunks will only work if Emacs can find the respective programs[fn:2], and if a compiler (for C), or an interpreter (for R and SQLite) were installed.

The code block needs to be named as shown. If you want the result and the code shown in the printout, you need to specify `:exports both`.

```
#include <stdio.h>

int main(void) {
  puts("hello world");
  return 0;
}
```

```
hello world
```

In the second version, both the header and the function definition are preset so that you can see the inside of the function only.

```
puts("hello world");
```

```
hello world
```

Footnotes

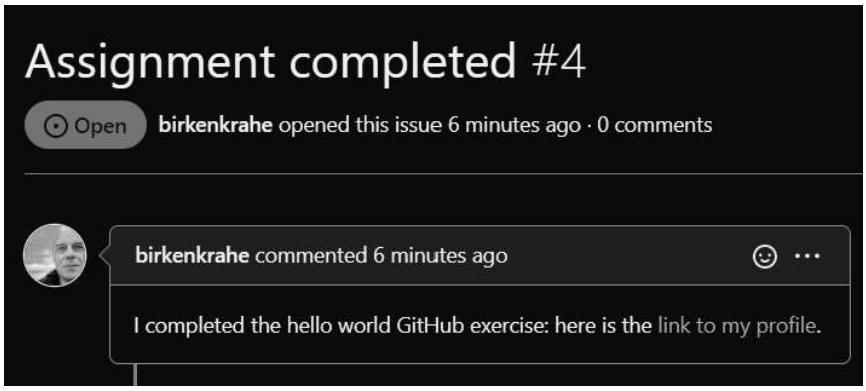
[fn:2]This is why we changed the Windows `PATH` variable during the installation of the programs `R` and `GNU gcc` (here).

[fn:1]Provided the block has been formatted correctly.

- It's free
- To host course materials
- Upload assignments (esp. Emacs Org-files)
- Discussion
- Wiki for collaboration
- Complements Schoology

3. What will you have to do?
    - Sign up with GitHub - use Lyon Email
    - Pick an available username **using your own first and last name**
    - Complete the "Hello World" exercise (FAQ)
    - Create an issue from the `cc100` repository like in the example below (except from your account instead of mine).

Image: Issue "Assignment completed"



If you do have a GitHub account already, do the exercise anyway using your existing account (it takes 10 min)! Make sure you let me know what your user name is so that I can add you to my repo.

4. What else can you do?
    - You can fork the `cc100` repository
    - You can watch the `cc100` repository - and set `Notifications` to `Participating and @mentions` so that you see my comments (see image below).

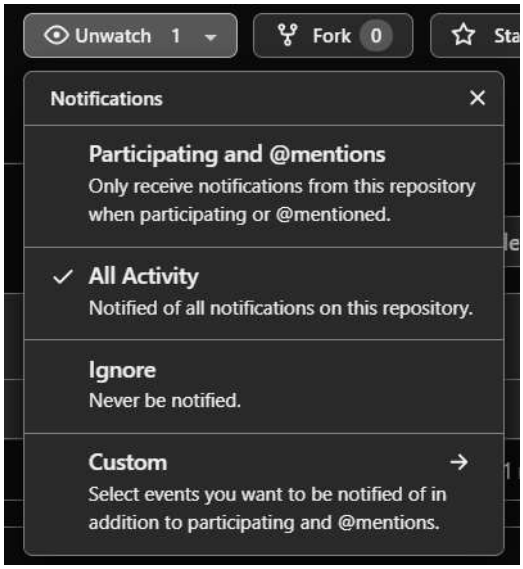
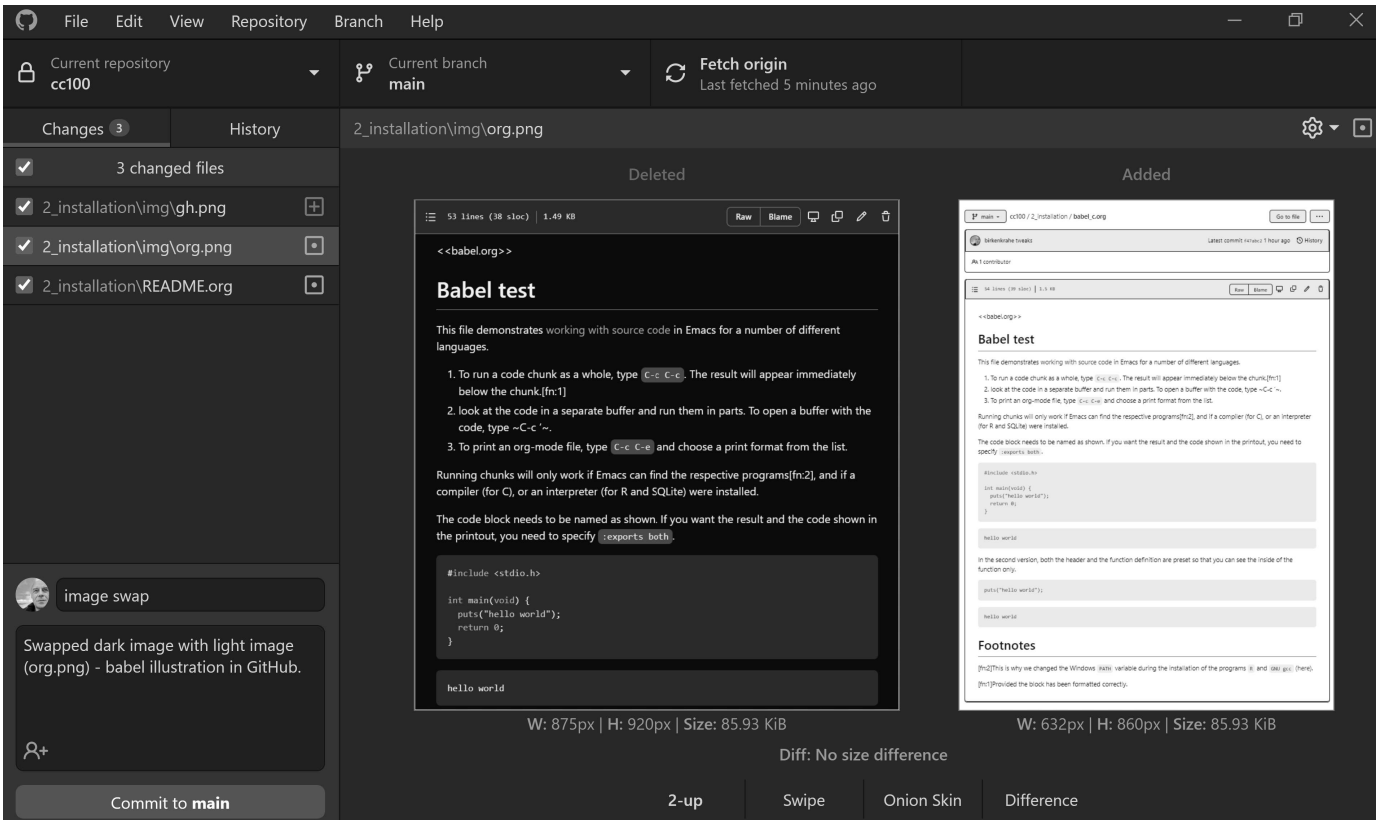


Image: Notifications settings when watching a repository

    - You can submit issues from the repository (e.g. if you notice mistakes or if you want extra information, or to share a link)
    - You can participate in discussions (sometimes I will make you)
    - You can add to the wiki (e.g. comments and links to interesting resources)
    - You can use it as a platform for projects or coding
    - You can download the desktop client to manage repos on your PC (see image below).

Image: GitHub desktop client commit



2.5 What's next?

- See schedule (GitHub)
- Watch online lecture on "Systems" (to be done)
- Later: online summary (`notes.org` in GitHub)
- Sometimes: diary notes (`diary.org` in GitHub)
- Class on Friday 14-Jan will be online!
- Hope to see you at school next Monday!

3 Git, GitHub, History of C - w1s2 (01/14/22)

3.1 Overview

HOW	WHAT
Review	GitHub Hello World exercise (see FAQ)
Lecture	Introduction to C
Practice	Install C compiler (see FAQ)
	Set PATH environment variable
	Test C compiler

3.2 Objectives

- [X] Review the basics of Git and GitHub
- [X] Understand what C is, and why you learn it
- [ ] Install the GNU C and C++ compiler (`gcc`)
- [ ] Set `PATH` environment variable under Windows
- [ ] Test the C compiler

4 Introduction to C and C++ - w2s3 (01/19/22)

4.1 I'm back

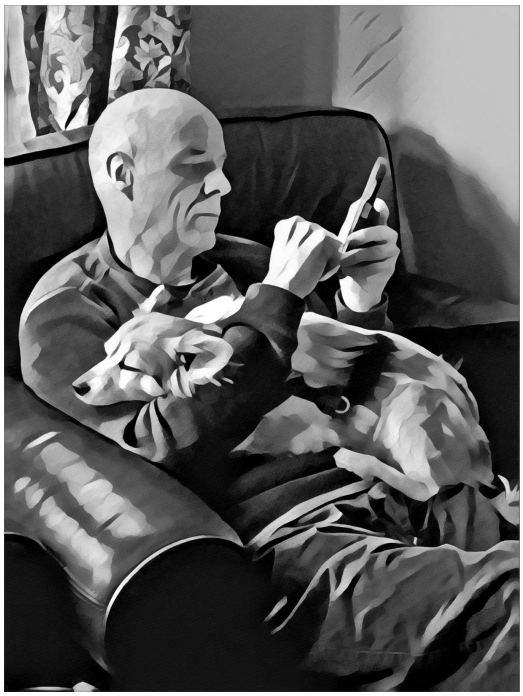


Figure 9: "I'm back, baby."

4.2 Overview

HOW	WHAT	TIME

HOW	WHAT	TIME
Review	Quiz on last week's content	15'
	Quiz: feedback discussion	5'
Lecture	Introduction to C (cont'd)(gh)	10'
	Installation of tools (gh)	
Practice	Install C compiler[1] (see FAQ)	
	Set PATH environment variable	
	Test C compiler	

gh = GitHub link

4.3 Objectives

- [x] Review last week & discuss & get feedback on quiz 1
- [x] Complete introduction to C (and C++)
- [ ] Understand installation process (philosophy)
- [ ] Install the GNU C and C++ compiler (`gcc`)
- [ ] Set `PATH` environment variable under Windows
- [ ] Test the C compiler

5 Install C compiler, set PATH - w2s4 (01/21/22)

5.1 Overview

HOW	WHAT	TIME
Lecture	Installation of tools (gh)	
Practice	Install C compiler[2] (see FAQ)	
	Test C compiler `gcc --version`	
	GNU Emacs tutorial in class (gh)	
HOME	Set PATH environment variable	

gh = GitHub link

5.2 Objectives

- [x] Understand installation process (philosophy)
- [x] Install the GNU C and C++ compiler (`gcc`)
- [x] Set `PATH` environment variable under Windows
- [x] Test the C compiler
- [x] GNU Emacs tutorial

6 Install GNU Emacs - w3s5 (01/24/22)

6.1 Overview

HOW	WHAT
Practice	Emacs training script
	See also video playlist

HOW	WHAT
Assignment	Set `.emacs` init file in your home directory

6.2 Objectives

- [X] Work through short tutorial for GNU Emacs
- [X] Explain Emacs assignment

7 Create GNU Emacs Org-mode file - w3s6 (01/26/22)

7.1 Overview

HOW	WHAT
Practice	Set `.emacs` init file in your home directory
Demo	Creating Emacs Org-mode with C code and run it
Assignment	Create Emacs Org-mode file (GitHub)

7.2 Objectives

- [X] Understand Emacs initialization with `.emacs`
- [X] Learn how to create an Org-mode file
- [ ] Run a C program inside Emacs

8 GNU Emacs initialization file - w3s7 (01/28/22)

1. We continue where we left it on Wednesday (w3s6)
2. Fixing the .emacs problem (FAQ)
3. Finish Org-mode assignment (GitHub)
4. Submit results to GitHub as issue (ZIP) - by 11.59PM tonight

9 GCC Lab session - w4s8 (01/31/22)

Figure 10: Teaching Emacs on Dagobah

We will hold a special lab session tomorrow, Monday 31 January 11-11.50 AM, to sort out any issues related to Emacs or GCC. Bring your own PC to the session, or work on a lab desktop. I will spend the time going round to make sure that you can

- Install/ open / use the Emacs editor

- Create, run and tangle Org-mode files
- Install / use the C compiler GCC
- Understand the recent program assignments

The necessary steps are also demonstrated in this tutorial video playlist.

We will continue with our regular program on Wednesday, 2nd February at 11 AM - a short quiz will be available before.

For those who know or can do all of this already: here's a second challenge (with solution) to practice while I sort others out.

10 Structure of a C program - w4s9 (02/02/22)

10.1 Overview

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Write and execute `pun.c`	
Assignment	Write a checkmarks program	Friday, 4 Feb, 11AM

10.2 Objectives

- [X] Understand the basic structure of a C program
- [X] Write a simple, complete C program (`pun.c`)
- [X] Submit simple assignment for Friday 4 Feb 11 AM

10.3 Assignment: checkmark program

Submit program and output as an Org-mode file. It should look like this (code block is folded):

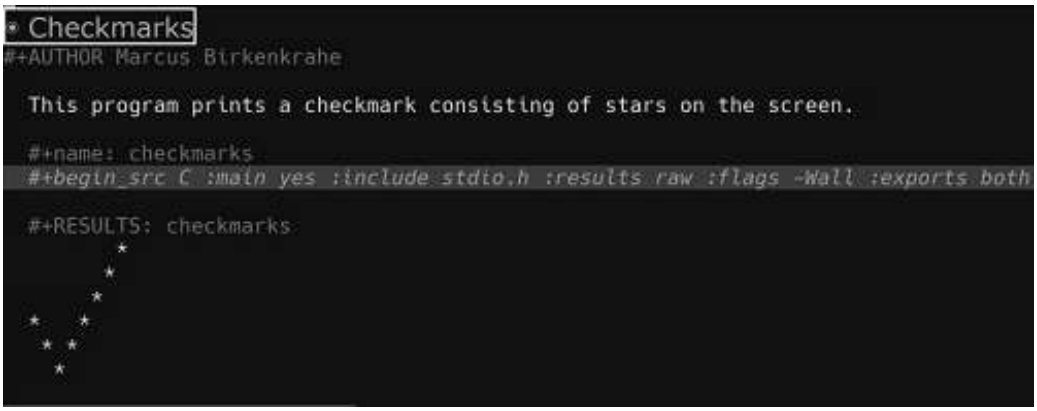


Figure 11: Checkmarks solution (code block folded)

11 Variable type declarations and assignments - w5s10 (02/07/22)

11.1 Overview

HOW	WHAT	WHEN
Review	Structure of a program	

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Computing the weight of a box	
Test 1	10 from Quiz 1-3 + 10 new MPCs	Friday, 11 Feb, 11AM

11.2 Objectives

- [x] Understand the framework for the first test
- [x] Understand variable, data types and type declarations
- [x] Understand variable assignments
- [ ] Understand printing formats
- [ ] Write a program with variable declarations and assignments
- [ ] Understand printing formats

12 Formatting printout - w5s11 (02/09/22)

12.1 Overview

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Interactive notebook: `printf` formatting	
Test 1	10 from Quiz 1-3 + 10 new MPCs	Friday, 11 Feb, 11AM

12.2 Objectives

- [x] Understand printing formats
- [x] Write a program with variable declarations and assignments

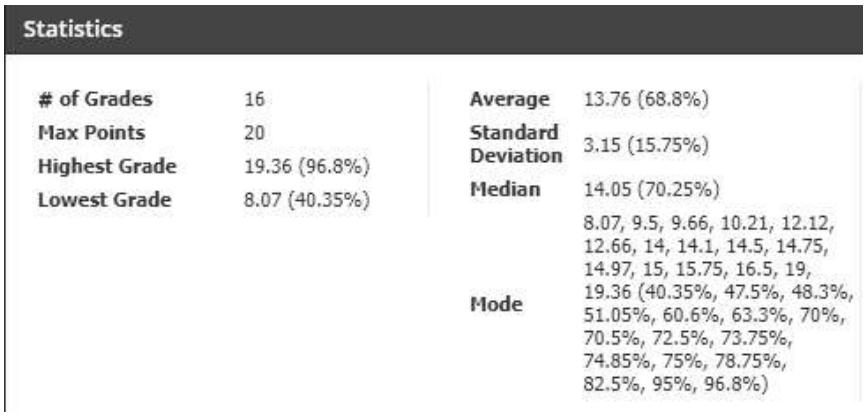
13 Test review - w6s12 (02/14/22)

13.1 Objectives

- [x] Understand test results
- [x] Know what to do different next time
- [x] Discuss selected questions and answers

13.2 Test results - stats and plots

- The results are nothing to write home about - though > 50% means that the class passed (on average).



Statistics

# of Grades	16	Average	13.76 (68.8%)
Max Points	20	Standard Deviation	3.15 (15.75%)
Highest Grade	19.36 (96.8%)	Median	14.05 (70.25%)
Lowest Grade	8.07 (40.35%)	Mode	8.07, 9.5, 9.66, 10.21, 12.12, 12.66, 14, 14.1, 14.5, 14.75, 14.97, 15, 15.75, 16.5, 19, 19.36 (40.35%, 47.5%, 48.3%, 51.05%, 60.6%, 63.3%, 70%, 70.5%, 72.5%, 73.75%, 74.85%, 75%, 78.75%, 82.5%, 95%, 96.8%)

Figure 12: Test 1 results (Schoology)

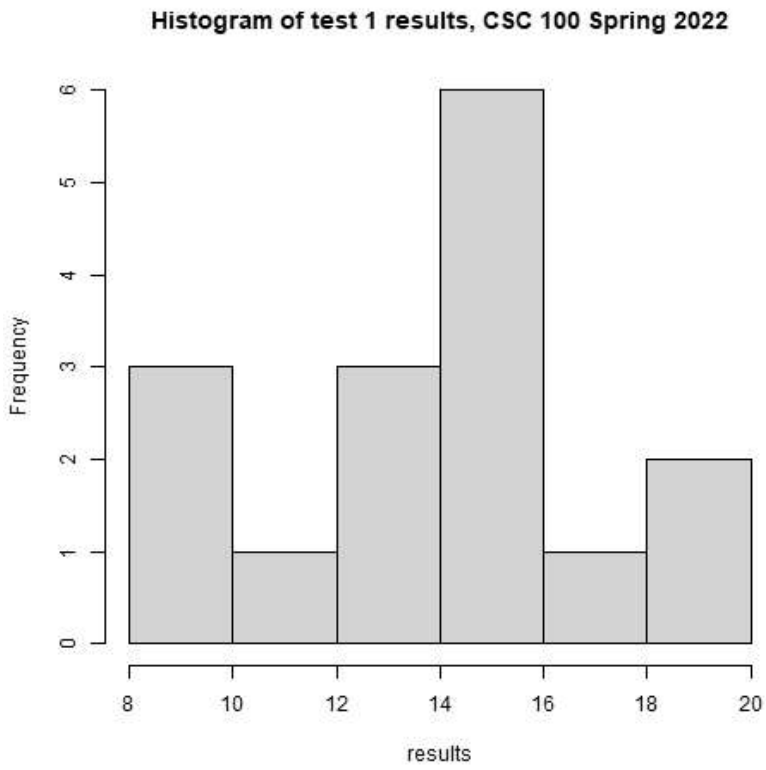
- I am an obsessive fact-checker. When checking the stats with R, I find slightly different results:

```
results <- c(8.07,9.5,9.66,10.21,12.12,12.66,14,14.1,
             14.75,14.75,14.97,15,15.75,17,19,19.36)
sd(results)
summary(results)
```

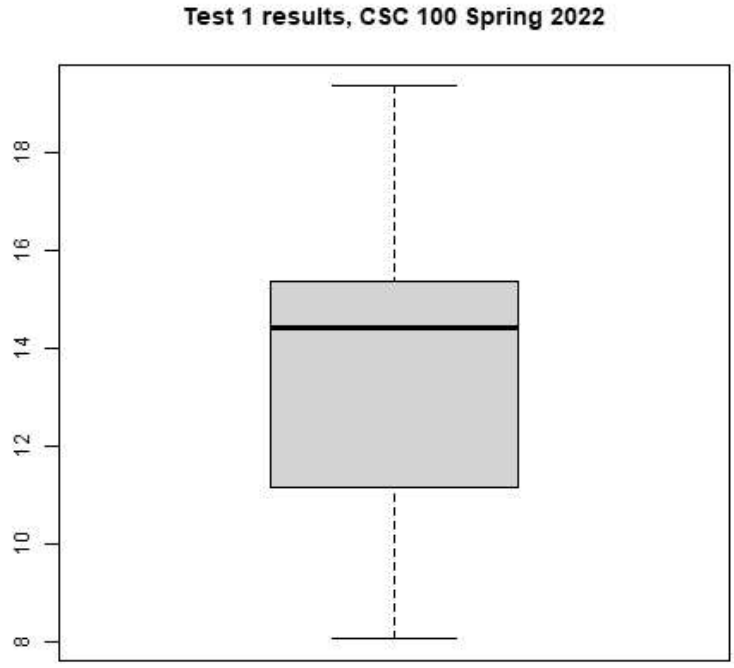
```
[1] 3.288485
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.07   11.64   14.43   13.81   15.19   19.36
```

- Let's make some plots: histogram, boxplot and density plot. I'dd like the histogram and the density plot (a smoothened histogram) to peak more to the right, and for the boxplot to be smaller and higher up.

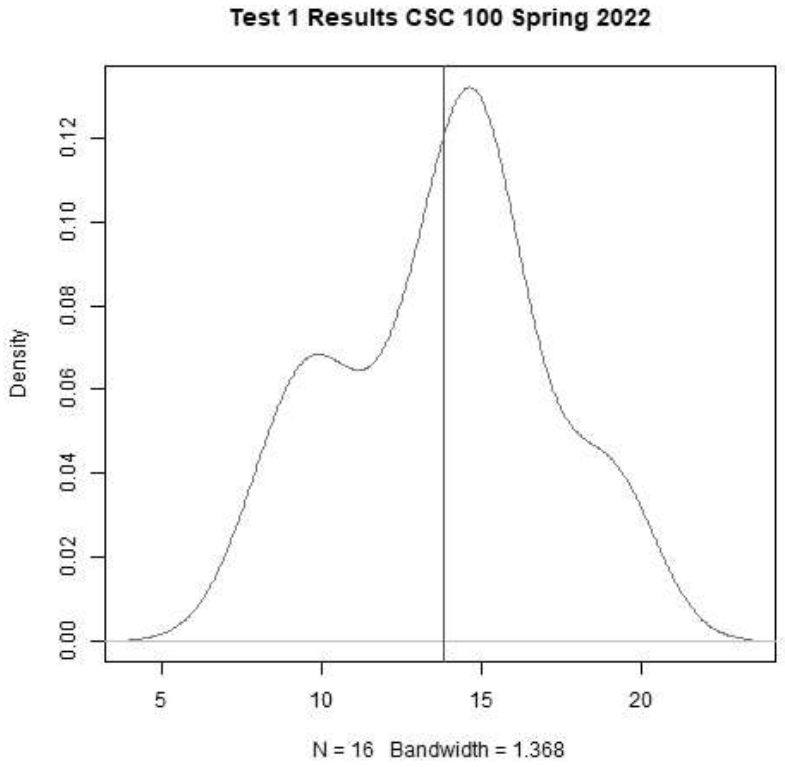
```
hist(results, main="Histogram of test 1 results, CSC 100 Spring 2022")
```




```
boxplot(results, main="Test 1 results, CSC 100 Spring 2022")
```



```
ave <- mean(results)
d <- density(results)
plot(d, col="steelblue",main="Test 1 Results CSC 100 Spring 2022")
abline(v=ave,col="red")
```



13.3 Analysis - feedback and action points

- Test 1 can now be played an unlimited number of times. I have added feedback to all new questions.
- There will not be another paper-based test: the results weren't much better than in other courses, and test preparation and grading are excruciating if partial credit is given.
- What surprised me most was that many of you did not use the available time.
- See also: "I can teach it to you but I cannot learn it for you"
- Questions:
  - How did you study for this test?
  - If you didn't perform well, what will you change?
  - What can I do to help you help yourself?
- CHANGES TO BE APPLIED BY BIRKENKRAHE (FUTURE QUIZ/TESTS):
  - Fewer choices for the multiple choice
  - Announce if a question has > 1 answer (and how many)

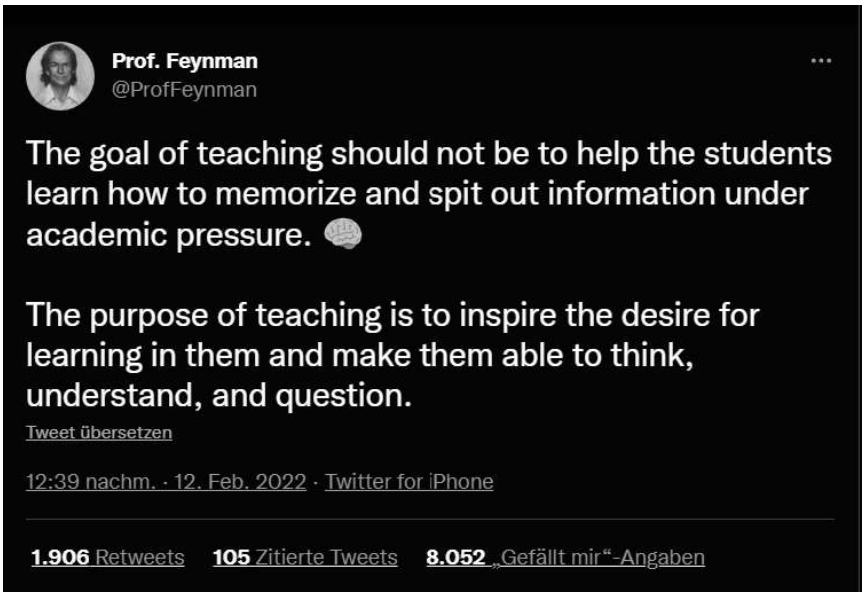


Figure 16: Feynman (via Twitter)

- Post-mortem on a couple of questions
  - comment format (last question)
  - `printf` format

```
/*     one line
 *     multi line
 */
float f = 4.0f;
printf("hello %.pf", f);
/* one line
   or multi line
*/
```

13.4 Test questions and answers

- We go through all questions together
- Everybody can contribute an answer
- Write down questions and ask them now!

13.5 This week (6)

- Test review
- Program assignment
- Defining constants
- Naming identifiers
- Reading input

13.6 Program assignment (until Monday 21-Feb, 11AM)

Compute the volume of a sphere

- Write a program that computes the volume of a sphere with a 10-meter radius, using the formula $v = 4/3 \times \pi r^3$.
- Write the fraction $4/3$ as `4.0f/3.0f`. (Try writing it as 4/3 and see what happens.)
- Remember that C does not have an exponentiation operator, so you need to write r^3 as `r*r*r`.
- Upload your solution program as a `.c` file or as a `.org` file to Schoology not later than 11 AM on Monday, February 21st. Make sure that your program actually runs without errors!

- Be prepared to present your solution in class.

14 Formatting, C constants - w6s13 (02/16/22)

14.1 Objectives

- [X] Review: be the compiler / function challenge
- [X] Understand different ways of defining C constants
- [ ] Practice constant definitions
- [ ] Understand naming conventions for C programs

14.2 Warming up: be the compiler!

The output of the program below should be:

```
: The tax on 200.00 is 35.00, so the grandtotal is 235.00.
```

but instead the output is

```
: The tax on -1093221452 is 0, so the grandtotal is 1080623104.
```

Can you fix the program 1?.

```
float subtotal;
int tax;
float grandtotal;
float taxrate;

taxrate = 0.175;
subtotal = 200;
tax = subtotal * taxrate;
grandtotal = subtotal + tax;

printf("The tax on %d is %d, so the grandtotal is %d\n",
       subtotal, tax, grandtotal);
```

14.3 Programming challenge: hello world function

- Things to remember:
  - You need to declare stuff before you can use it in C.
  - The function definition looks just like `main`
- Challenge: Write a function `hello()` that prints out `hello world` when called.
- You can do this in Emacs in one code block, which contains the definition and the function call.
- This challenge carries extra credit! Submit Org-file via email.
- Solution next week! (This implies that you send it to me before the first class next week.)

15 C constants, naming - w6s15 (02/18/22)

15.1 News

- Invitation for an online data science seminar with the data scientist of Stone Ward, Little Rock/Chicago, Matthew Ward: join us at 3 pm via Google Meet (see Schoology) - incl INTERNSHIPS

- Great tips for getting better with Emacs quickly in this post. See also my Emacs for Beginners YouTube playlist.

  > I started as a rank newbie 14 days ago, and I've learned enough to "get started", kind of grok emacs beginner fundamentals and simple navigation, and I feel VERY good about my progress. I wanted to share how I got to this point. (Note that I happened to have the time to read/watch/practice a few hours a day. If I only had 30 min per day, it would have taken me a month or two to get to this level.)

15.2 Objectives

- [X] Learn more ways of defining C constants
- [X] Practice constant definitions
- [X] Understand naming conventions for C program

15.3 Warming up: be the compiler!

- What's wrong with this program? (It won't even compile.)

```
#include <stdio.h>
#define CONST_ 100;

int main(void) {
  int value = CONST_, newValue;

  newValue = value + 1;

  printf("%d %d %d\n", CONST_, value, newValue);
  return 0;
}
```

  - Answer in the class notes file `notes.org`.

15.4 Next week

- Quiz 4 (play me once, then play me often!)
- Review of the volume/hello programming challenges
- A new programming challenge!
- Reading input with `scanf` (read me, baby!)
- Good program layout
- Arithmetic operators (math, yay!)

16 Program assignment review, scanf - w7s16 (02/21/22)

16.1 News

- Script is always in GitHub (use it for your class post mortem)
- Literate Programming: Empower Your Writing with Emacs Org-Mode

16.2 Objectives

- [X] Review programming assignment "volume"
- [X] Review programming challenge "hello world function"
- [X] Understand reading input (from the command line)
- [ ] Understand how in-class assignments work
- [ ] New programming exercise "phone number conversion"
- [ ] Complete two in-class assignments
- [ ] See how reading input works in Emacs Org-mode

16.3 Warming up: be the compiler!

- The following program should print out `3.0` but it does not even compile! Find all mistakes, fix them and run the program in Emacs!

```
float int1=1.f, int2=2.f, int;
printf("%.2f\n"; int = int 1 + int 2);
```

- SOLUTION: see class notes.

16.4 How do class assignments work?

- In-class assignments are **10%** of your total grade
- They are labeled **class assignments** in the Schoology gradebook
- You get the points if you attend and participate **actively**
- If you check your phone instead, you're **not** active
- If you could not attend, submit **late**

16.5 Programming assignment review

- I left detailed comments for most of you: use them or ask me! Use them by modifying your submission accordingly.
- You can find examples and solutions in the class notes later.
- Future submissions will be graded not just for effort but for accuracy (this includes following the instructions). However, you can (and should) always re-submit an improved version.
- Computer science is a craft like any other: (1) you keep at it until you got it right, (2) you don't give up easily, and (3) you ask for help when you're at the end of your wits.
- Please submit an Org-mode file if you have one so that I don't have to check your results line-by-line but can just run it.

- Org-mode meta data were often not correct (file will still run)

```
#+AUTHOR: name

#+name: code
#+begin_src C
...
#+end_src
```

- A couple of programs looked awfully similar (down to errors):
  - There is no need to bend the rules: this is not a rat race!
  - If in doubt about "what is cheating", check the syllabus.
  - Add `#+HONOR: pledged` to the top of your Org-mode submissions
- You can find online REPLs (Read-Eval-Print-Loops) like this one to try things out if you're flummoxed by Emacs Org-mode (at this point, you shouldn't be but Windows can be toxic…)

16.6 Challenge review

- User-defined functions are where programming really begins
- See class notes for challenge solution

17 Reading input with Emacs, tangling code - w7s17 (02/23/22)

17.1 Objectives

- [X] New programming exercise "phone number conversion"
- [X] How to use `scanf` inside Emacs (and other input)
- [X] Understand reading input (from the command line)

17.2 How to use `scanf` inside Emacs with redirection

- Emacs can fetch arguments via the header arguments `:cmdline` (as input redirected from a file), or via `:var` through assignment.

```
float i,j;
scanf("%f %f\n",&i, &j);
printf("input was: %f %f\n",i,j);
```

- [X] If your neighbor does not get along, help him/her without being asked explicitly! (Well done everyone!)
- [X]

  Create a file called `input` in the current working directory. It contains only the numbers `1.` and `2.`:

```
1. 2.
```

- [X] Create and run the code block 1 in Emacs in a Org-mode file. Make sure that you have `:main yes` and `:include <stdio.h>` either on the code block header line or in the `#+PROPERTY:` meta data of your Emacs Org-mode file.
- [X] Create and run the code block 1 in Emacs. Make sure that you have `:main yes` and `:include <stdio.h>` either on the code block header line or in the `#+PROPERTY:` meta data of your Emacs Org-mode file.
- [X] Tangle the code block and look at the file `cmd.c`. To tangle, use the key sequence `C-c C-v t`.
- [X] Open a CMD terminal (or `M-x shell` inside Emacs), and compile the program `cmd.c`. Call the executable `cmd`.
- [X]

  Run the program on the command line like this:

```
cmd < input
```

- Shell "redirection" (directing the content of `input` into `cmd`) is an important shell process. It uses both `stdin` and `stdout`. See GNU bash manual for details.

17.3 Programming assignment (until ~~Monday, 28-Feb~~ Tuesday, 1-Mar, 11 AM)

- Write a program that prompts the user to enter a telephone number in the form `(xxx) xxx-xxxx`, and then displays the number in the form `xxx.xxx.xxxx`.
- Example input/output of the first program, `phone1.c`:

```
Enter phone number [(xxx) xxx-xxxx]: (870) 456-7890
You entered 870.456.7890
```

- Write another program that asks for the input format in the form `xxx\xxx\xxxx`, and then displays the number in the form `(xxx)xxx-xxx`.
- Example input/output of the second program, `phone2.c`:

```
Enter phone number [xxx\xxx\xxxx]: 870\456\7890
You entered (870) 456-7890
```

- Submit one Emacs Org-mode file `phone.org` with both programs in it as code blocks that can be **tangled** as `phone1.c` and `phone2.c`, resp.
- The header information of your Org-mode file should look like this:

```
#+TITLE: Phone number conversion
#+AUTHOR: [your name]
#+HONOR: pledged
```

- Tip: some characters, like `\` are protected because they are part of the file `PATH`. If you want to use them, you have to "escape" them with an extra `\`, like the newline character `\n`. So to print or to scan the character `\`, you use `\\`.

```
printf("hi there\n");              // string output
printf(" \"hi there\" \n");        // escaped \" will print
printf("This is a slash: \ \n");   // slash will not print
printf("This is a slash: \\ \n");  // escaped slash\\  will print
```

18 Lab session: scanf.org, reading input - w8s18 (02/28/22)

18.1 Objectives

- [X] Review: quiz questions 4 + 5 (complete them/ask)
- [X] Be the compiler: `scanf` formatting

- [ ] Review: compile and run C on the command line
- [ ] Understand tangling code once and for all

18.2 Be the compiler: `scanf` formatting

- Download `reading_input.zip` from GitHub (cc100/practice)
- The ZIP file contains an interactive notebook `scanf.org` and several prepared input files
- Work through the notebook at your own pace
- If you cannot finish in class, finish it at home!
- Help thy neighbour and/or ask me for help!

18.3 Compile and run C in the shell

- [ ]

  Create a minimum C file, e.g. a hello world program `hw.c`- either in Emacs or in any other editor you like:

```
#include <stdio.h>

int main() {
  puts("hello world");
}
```

- [ ] Open a terminal (or `M-x shell` or `M-x eshell` in Emacs).
- [ ] Convince yourself that the file can be found with `DIR hw.c`
- [ ] Compile the file with `gcc -o hello hw.c`
- [ ] The `-o hello` means "name the executable `hello`"
- [ ] Run the file by entering the name of the executable
- [ ] As you can see via `dir hello*`, the executable is actually called `hello.exe` in Windows.

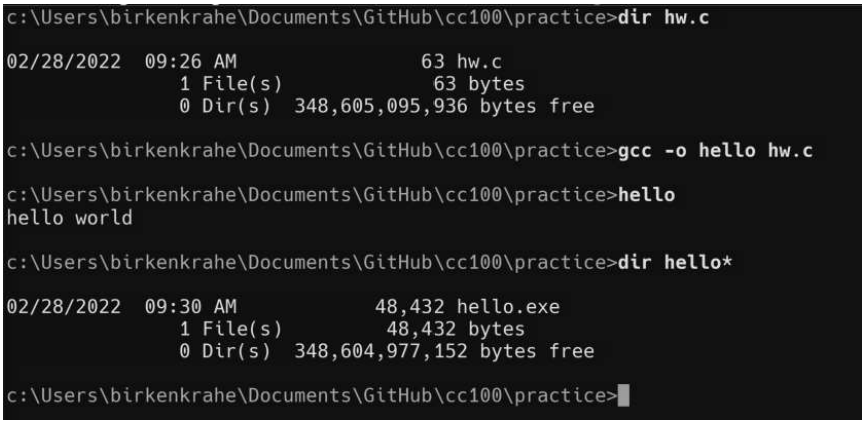


Figure 17: Compile and run C program on the shell

18.4 2 Tips for the programming assignment

1. (02/25/2022) If you want to test and run your file **inside** Emacs, which saves a lot of time, just add this to the header arguments (after the `C`):

```
:cmdline < input
```

   where `input` is a file that contains the phone number as requested for input, `(xxx) xxx-xxxx` or `xxx\xxx\xxxx`. Now `C-c C-c` will run and execute the program. Alternatively, you have to tangle the code block, compile and run it by hand on the command line.

2. (02/27/2022) the formatting in `scanf()` needs to match the input pattern. E.g. if the input is given as

```
foo==++//bar
```

(with `foo` and `bar` decleared as `int` variables) then only the following command will pick the input up properly:

```
scanf("%d==++//%d", &foo, &bar);
```

irrespective of the output, which can be formatted in any way we like. `scanf()` only looks for two integers separated by all that junk between the numbers.

Example: the file `inputfile` contains only: `444==++//555`. We want to only print out `444` and `555`.

```
int a,b;
scanf("%d==++//%d", &a, &b);
printf("%d %d", a,  b);
```

19 Compiling, LitProg, Program Layout - w8s19 (03/04/22)

19.1 News

- `[X]` Mid-term grade speech / Improve grade with a project (FAQ)
- `[X]` Program assignment **for extra credit**: `PDF` (GitHub)
- `[X]` Practice material/notebooks now in GDrive (`reading_input`)
- `[X]` PDFs of the quizzes with solutions now in GitHub

19.2 Improve your grade with a project

If you want to improve your grade, you can talk to me about doing a small, independent research project leading to a writeup in the form of a notebook, or a short (10-15 min) presentation. The topic must be related to the topic of this course.

19.3 Objectives

- `[X]` Understand how to compile and run C in the shell
- `[X]` Understand literate programming once and for all
- `[X]` Understand what good program layout means (practice)

19.4 Compile and run C in the shell

- Three shells[3] are at your disposal: Microsoft shell (MacOS terminal), Emacs eshell, Emacs shell (= terminal) = CMD line:
  - Start Windows CMDline from the the Windows Search (`CMD`)
  - Start Emacs shell (Windows CMDline in Emacs): `M-x shell`
  - Start Emacs eshell with `M-x eshell`
- Files you run in the terminal: Windows `*.exe` files - executable in binary format = machine code
- `[X]`

  Create a minimum C file, e.g. a hello world program `hw.c`- either in Emacs or in any other editor you like:

```
#include <stdio.h>

int main() {
  puts("hello world");
}
```

```
hello world
```

  You should see the output:

```
#+RESULTS: helloworld
: hello world
```

- [X] Tangle the code block above with `C-c C-v t` (if you have more than one code block in an Org file, use `C-u C-c C-v t`)
- [X] Open a terminal (or `M-x shell` or `M-x eshell` in Emacs).
- [X] Convince yourself that the file can be found with `DIR hw.c`
- [X] Compile the file with `gcc -o hello hw.c`
- [X] The `-o hello` means "name the executable `hello`"
- [X] Run the file by entering the name of the executable
- [X] As you can see via `dir hello*`, the executable is actually called `hello.exe` in Windows.

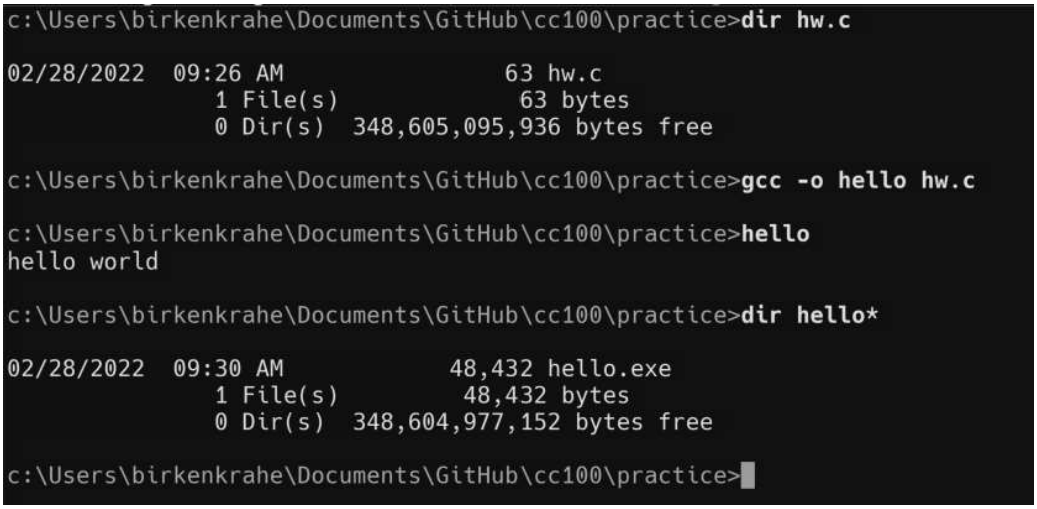


Figure 18: Compile and run C program on the shell

- [X] The general structure of the compile command `gcc` is

```
gcc -o [outfile] [infile]
```

19.5 Understand tangling code once and for all

19.5.1 Concepts

Concept review: make sure that you can answer these questions:[4]

What is Emacs?
What is Org-mode?
What does "tangle" C code mean?
What does "weaving" documentation mean?

19.5.2 Guided tour

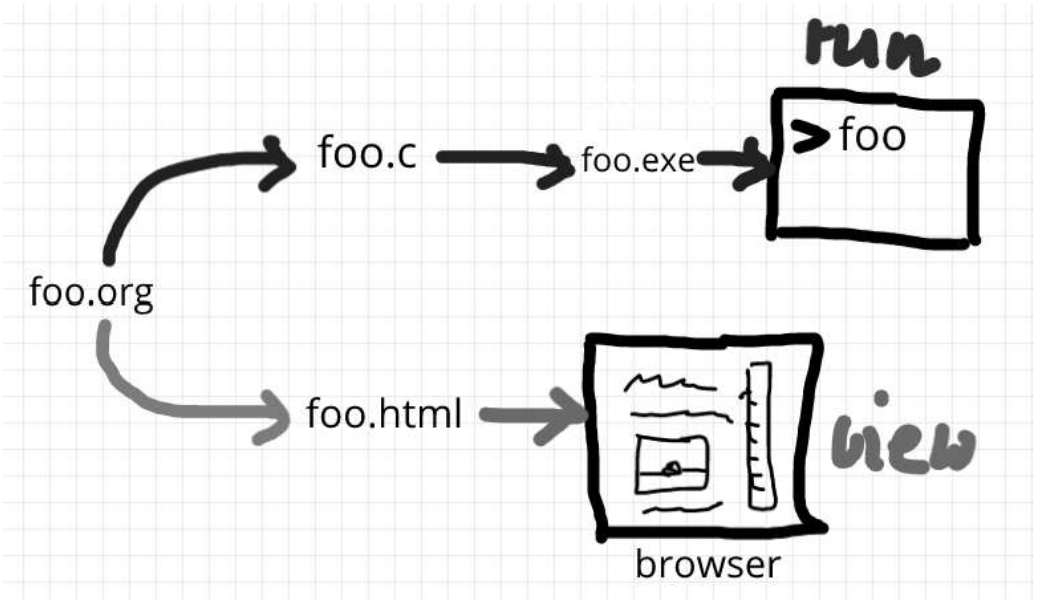
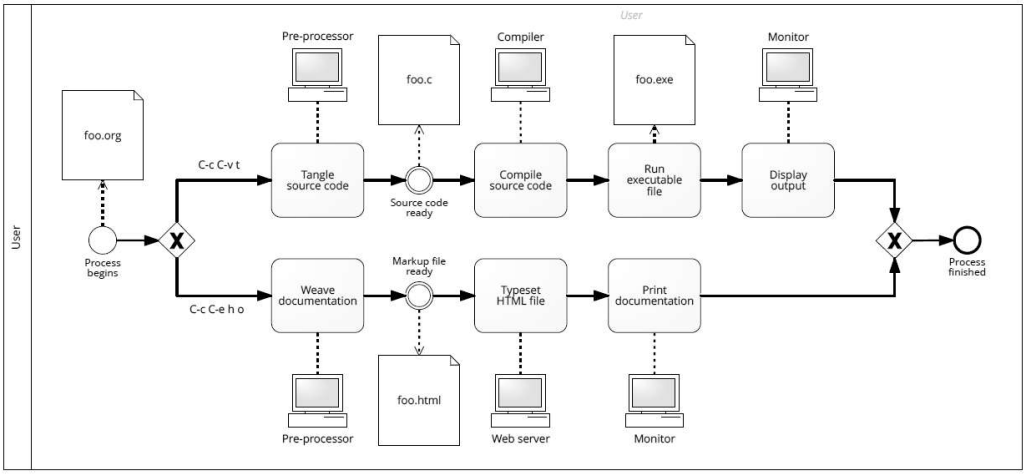


Figure 19: Short explanation of literate programming



Tangle foo.org in Emacs: C-c C-v t Weave HTML page in browser: C-c C-e h o
Compile foo.c in shell: gcc foo.c Run executable foo.exe: $ foo

Figure 20: Detailed description of literate programming for C

```
* The following statement reads an ~int~ value and stores it in the
  variable ~i~. To run it, you have to tangle the code block (with
  ~C-u C-c C-v t~), compile and run it on the command line - not on
  the simulated ~eshell~ or the Emacs ~shell~, but on the Windows
  CMD terminal command line[fn:14].
  #+name: iscan
  #+begin_src C :tangle iscan.c :includes <stdio.h> :main yes :comments both
    int i;
    puts("Enter an integer!");
    scanf("%d", &i);
    printf("You entered %d\n", i);
  #+end_src
1 U\---  README.org     14% (674,17)  Git:main  (Org)
```

Figure 21: Code block and comment in Org-mode

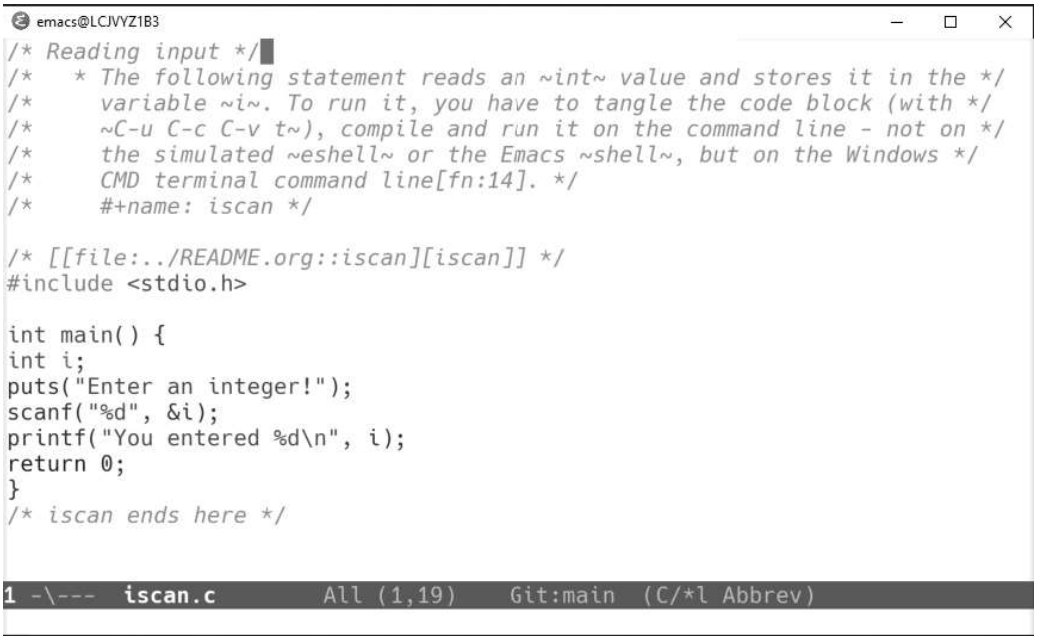


Figure 22: Tangled source code C file with comment

C Basics

CSC100 Introduction to programming in C/C++

- The following statement reads an `int` value and stores it in the variable `i`. To run it, you have to tangle the code block (with `C-u C-c C-v t`), compile and run it on the command line - not on the simulated `eshell` or the Emacs `shell`, but on the Windows CMD terminal command line[1].

```
int i;
puts("Enter an integer!");
scanf("%d", &i);
printf("You entered %d\n", i);
```

Footnotes:

[1] You could try and run it in Emacs. Can you explain the result?

Author: Marcus Birkenkrahe
Created: 2022-02-25 Fri 14:05

Figure 23: Documentation woven as HTML file in browser

```
#+name: iscan
#+begin_src C :tangle iscan.c :cmdline < input :results output
  int i;
  puts("Enter an integer!");
  scanf("%d", &i);
  printf("You entered %d\n", i);
#+end_src

#+RESULTS: iscan
: Enter an integer!
: You entered 1000
```

Figure 24: Executable C code block in Org-mode with results

19.6 Phone assignment solutions

- See GitHub folder assignments/

19.7 Program layout

- See GitHub folder 3_basics/

19.8 Next

- Solution to the `layout.c` challenge
- More on `scanf` and `printf` formatting (DIY notebook)
- Arithmetic expressions and operators (lecture)
- Lab session (in-class assignment notebook)
- Program assignment no. 6

20 Printf conversion - w9s20 (03/07/22)

20.1 To do

- [ ] Waking up in AppData/Roaming? Change your Emacs HOME now.
- [ ] Warming up: layout solution / NoppeLayout / NoppeLayoutModified
- [ ] Getting more deeply into formatting I/O with printf and scanf

20.2 Program Layout practice exercise

- Don't be stingy with text when it comes to documenting your work
- Documenting now will save you lots of time later

- Make use of these examples for the next program assignment

  See program examples NoppeLayout and NoppeLayoutModified.

20.3 Formatting I/O - printf

- Interactive notebook to code along (GDrive) - `io_printf_nb.org`

21 Scanf conversion - w9s21 (03/09/22)

21.1 Preparations for test 2 (Monday, 14-Mar)

- Test 2 will only cover questions from quiz 4-6 + new questions.

- You can find quiz 4-6 with solutions + feedback as PDF
- I will create an update of content Org files (in pdf/)

21.2 Warming up: `tprintf.c`

- Remember conversion specifiers m.pX
- New: a negative m left aligns a number: 123 as %-d => |123 |
- [ ]

  Guess the output before running 1

  - how many spaces m are reserved?
  - how many decimal places after the point?
  - is the number left or right aligned?

```
int i;
float x;

i = 40;
x = 839.21f;

printf("|%d|%5d|%-5d|%5.3d|\n", i, i, i, i);
printf("|%10.3f|%10.3e|%-10g|\n", x, x, x);
```

21.3 Formatting I/O - scanf

- Interactive notebook to code along (GDrive) - `io_scanf.org`
- Download the ZIP file and unpack it anywhere. All the files for the session are inside.

21.4 Programming assignment 6:

- Write a program `div3` that reads in three floating point numbers, adds the first two and divides the total by the third number. Test with these data:

```
(1610 + 2004) / 2365
```

- Bonus program for extra credit points: change the program to `div4`. This program should read in four floating point numbers. Add the first two together, then add the second two together. Divide your first sum by your second sum. Test with these data:

```
(1610 + 2004) / (2005 + 360)
```

- Submit in Schoology as an Org file with your name and `#+HONOR: pledged` in the header. The code blocks should be run and the `#+Result:` should show the correct answer.
- To get full points, submit the Org file with some comments outside of the code blocks - explaining what you do, what the code is about, or anything else really that you want to share[5]. Use NoppeLayoutModified.org as an inspiration.
- A comment could look this is: I absolutely, and unreservedly LOVE this class!
- This exercise originally came from Joyce (2018).

22 Pgm 6, Operators, Pseudocode - w10s23 (03/16/22)

22.1 Agenda

- [X] Review: programming assignment 6 & Harry's HTML
- [X] Outlook: C++ Basics (planned video series)
- [X] C conditions - operators/pseudocode/flowcharts

22.2 Harry's HTML

- Fancy looking HTML

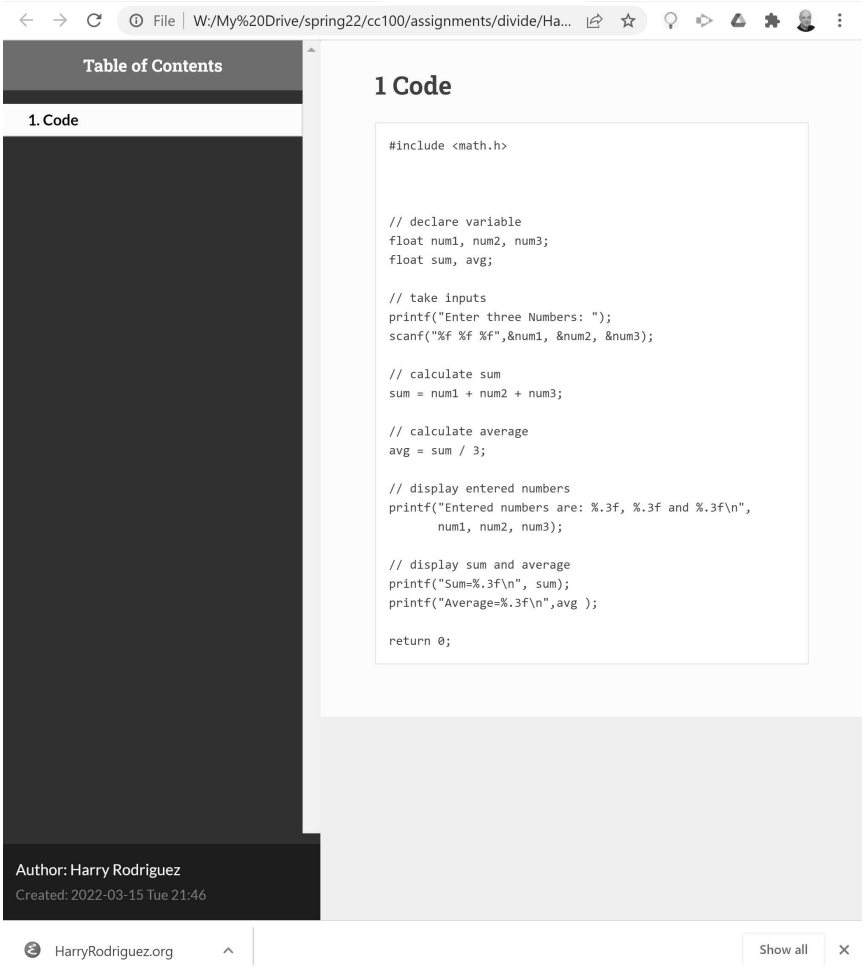


Figure 25: Harry's submission rendered from Org-mode as HTML file

- Harry's code

```
#+HTML_HEAD: <link rel="stylesheet" type="text/css" href="style1.css" />
#+HTML_HEAD_EXTRA: <link rel="alternate stylesheet" type="text/css" href="style2.css" />
#+OPTIONS: html-style:nil
#+SETUPFILE: https://fniessen.github.io/org-html-themes/org/theme-readtheorg.setup
```

22.3 Conditions/Pseudocode

See README.orgin `4_conditions/` GitHub

23 Character conversion, Flowcharts, BPMN, if structures - w10s24 (03/18/22)

23.1 Warming up with pseudocode

- Turn the following sentence into pseudocode:

  "On March 19, spring break starts."

- To do this, you need a condition check
- The condition check requires a variable and an operator
- You do not necessarily need an alternative

- Sample solution

```
if date == March 19
   start spring break
```

23.2 Flowcharts and IF statements

See README.orgin `4_conditions/` GitHub

23.3 Program assignment 7 - deadline Friday April 1

Link to assignment in Schoology

23.3.1 Problem

- Modify the code from class ("Battle by numbers" in GitHub) to use characters as menu choices instead of numbers.
- To do this, you need to change from an integer data type to a character data type.
- Both `scanf` functions and `if` structure have to be changed to accommodate this change.
- The conversion specification for characters is `%c`.
- Note that character variables should be defined and used with single quotes, e.g. `response = 'a'` or `response == 'a'`.

23.3.2 Character conversion

- Create an input file with one character `'a'` in it.

```
echo 'a' > input
```

- Declare a variable `response`, scan the character from the input file `input`, and print it.

```
char response = '\0';
scanf("%c", &response);
printf("The character input was: %c\n", response);
```

```
The character input was: a
```

```
The character input was: a
```

- To check the value of a `character` variable, use logical operators like `==`. E.g. if `char r = 'y'`, then `r == 'y'` is `true`, and `r != 'y'` is `false`.

23.3.3 Submission

- Submit the code as a C source code file.
- Make sure that the C code compiles and runs properly before submitting.
- Late submissions will not be accepted for this assignment.
- Extra credit (5 pts) for submitting a commented Org mode file (with the usual meta data in the header, and the results of a trial run).
- Extra credit (5 pts) for submitting an Org mode file with a BPMN diagram of the algorithm included (send both files to me via email).

24 Booleans, operator precedence - w11s25 (03/28/22)

24.1 Term plan

The plan for the rest of the term:

- week 11 - `switch` and `break` statements

- week 12 - iteration statements (`for`, `while do` loops)
- week 13 - structured programming with functions
- week 14 - grouping variables with arrays
- week 15 - memory manipulation with pointers
- week 16 - text manipulation with string functions
- Includes some do-it-yourself reading (covered in weekly quiz 7-10)
- Weekly programming assignments 8-12 (submit C or Org versions)
- Final exam (TBC) on Monday May 9, 10:30-12:30 hrs in Lyon 104.

24.2 Review and warming up: if statements

- Though C has many operators, it has few statements: they fall in three categories, depending on how they affect the order in which statements are executed: selection, iteration and jump statements.

24.2.1 Selection statements

- `if` and `switch` statements allow a program to select a particular execution path from a set of alternatives.
- What is the output of this code?

```
if (0) printf("1"); else printf("0");
```

  The following if statements have the same output:

```
if (5 < 2) printf("1"); else printf("0\n");

if (!(2 * 2) && 1) printf("1"); else printf("0\n");

if (!'A') printf("1"); else printf("0\n");
```

- Which alternative is chosen depends on the value of the expression in parentheses:

EXPRESSION	BOOLEAN	WHY
5 < 2	FALSE (0)	5 is smaller than 2
!(2 * 2) && 1	FALSE (0)	!4 is 0
!'A'	FALSE (0)	A is converted to non-zero integer

24.2.2 The "dangling else" problem

- The following example illustrates the importance of braces: to which of the two statements does the `else` clause belong?

```
if (y != 0)
  if (x != 0)
    result = x / y;
  else
    printf("Error: y is equal to 0\n");
```

- If `y` is non-zero, the second if clause is entered, and if `x` is also non-zero, the variable `result` is computed.
- The rule in C is that an `else` clause belongs to the nearest `if` statement that has not already been paired with another `else`. This is what Emacs does, too.
- Alas: if y is non-zero and x is zero, we are told that y is 0. The print statement should be aligned with the first if, not with the second.
- To make the clause part of the outer statement, use braces:

```
if (y != 0) {
  if (x != 0)
    result = x / y;
 } else
  printf("Error: y is equal to 0\n");
```

24.2.3 Code template, pseudocode and BPMN model

The key to "critical thinking" is to be able to look at a problem from different sides without prejudice and with perseverance.

Remember: if all you have is a hammer, everything looks like nail.

1. Code template for if statements (=standard):

   if (expression) statement else statement

2. Pseudocode (=subjective) for if statements

```
if (true)
   do one thing
else
   do another thing
```

3. BPMN model

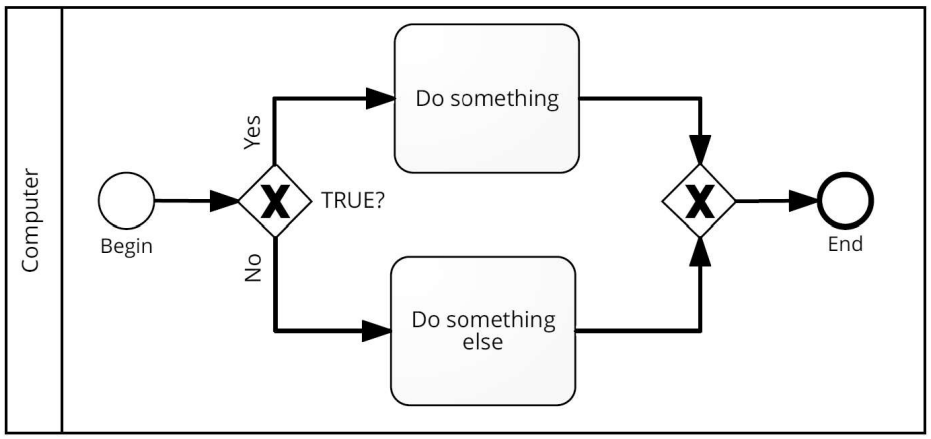


Figure 26: BPMN model of if else

24.3 Boolean algebra - do it yourself!

The text for this section is on GitHub.

- [ ] What algebra is really about
- [ ] Truth tables
- [ ] Compound logical expressions

24.4 Operator precedence - do it yourself!

- [ ] increment/decrement operators
- [ ] order of precedence table
- [ ] parenthesizing compound expressions

24.5 In-class: compound if structures and input validation

- [ ] && and || operators
- [ ] Checking for lower- and upper case
- [ ] Checking a range of numbers

With practice notebook `compound_if.org` in GDrive.

25 Compound if structures, input validation - w11s26 (04/01/22)

In this session, we mix lecture and interactive exercises for compound if structures (i.e. conditions with more than one logical condition and logical operators).

- [ ] Download the updated practice file config_if.org from GDrive

26 Switch, case, break, pgm 7, pgm 8 - w11s27 (04/01/22)

26.1 Warming up: compound if conditions, i < j < k

- [ ]

  Problem last time: I used the wrong operator!

  - To test if i in [m,n), use (m <= i && i < n)
  - To test if i is NOT in [m,n), use (i < m || n >= i)

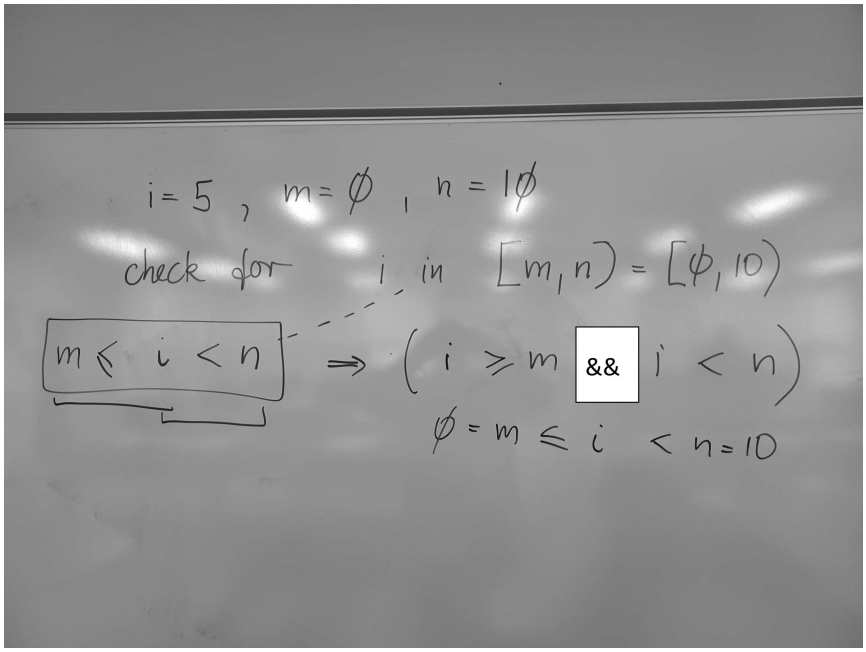


Figure 27: whiteboard screenshot (corrected)

- [ ] Warming up exercise: how would you test if
  1. j=1 is in the interval (-5,5] (-5 < j && j <= 5): j=1 => TRUE (1)
  2. k=100 is NOT in the interval (1,100) (k <= 1 || 100 <= k)
  3. l=2 is NOT in the interval [0.5,0.9]
  4. what data type does l have to be?

- Let's check your answers!

```
int j=1;
int k=100;
float l=0.1;
```

```
if ( -5 < j && j <= 5 ) printf("%d is in (5,-5]\n", j);
if ( k <= 1 || k >= 100 ) printf("%d is NOT in (1,100)\n",k);
if ( l < 0.5 || l > 0.9 ) printf("%g is NOT in [0.5,0.9]\n",l);
```

```
1 is in (5,-5]
100 is NOT in (1,100)
0.1 is NOT in [0.5,0.9]
```

- Tricky: i < j < k (see practice file)

26.2 Programming assignment 7 (review, deadline 01-Apr)

Here is the template, the code in 1 for number input from the file created in 1 first, then the solution for character input in 1, including the bonus solution (inline image of the BPMN model for the algorithm).

26.2.1 Battle by numbers

The code in 1 below reads input from a file `idrink` (the file contains the number `1`, one of the two possible responses to the battle question). You can change the input to `2` to get the other response.

```
echo "1" > idrink
```

```
int response = 0; // initialize response

puts("\n\tIn-Battle Healing\n\n1:\tDrink health potion\n\n2:\tResume battle\n");

printf("\nEnter your selection: ");
scanf("%d", &response);

if (response == 1)
  printf("\nYou entered \"%d\":\nDrinking health potion!\n", response);

if (response == 2)
  printf("\nYou entered \"%d\":\nResuming battle!\n", response);
```

26.2.2 Battle with character

The code in 1 below reads input from a file `cdrink` (the file contains the character `a`, one of the two possible responses to the battle question). You can change the input to `b` to get the other response.

```
echo "b" > cdrink
```

```
char cresponse = '\0'; // initialize response

puts("\n\tIn-Battle Healing\n\na:\tDrink health potion\n\nb:\tResume battle\n");

printf("\nEnter your selection: ");
scanf("%c", &cresponse);

if (cresponse == 'a')
  printf("\nYou entered \"%c\":\nDrinking health potion!\n", cresponse);

if (cresponse == 'b')
  printf("\nYou entered \"%c\":\nResuming battle!\n", cresponse);
```

26.3 Programming assignment 8 (deadline 08-Apr)

26.3.1 Problem

- Using the `switch` statement or cascaded `if` statements, write a program that converts a numerical grade into a letter grade.
- Example run:

```
Enter numerical grade: 84
Letter grade: B
```

- Use the following grading scale:

Numerical grade	Letter grade
90-100	A
80-89	B
70-79	C
60-69	D
0-59	F

- Print an error message if the grade is larger than 100 or less than 0.
- *Hint:* You can break the grade into two digits, then use a `switch` statement to test the ten's digit.

26.3.2 Tip: how to analyse a programming problem

- A programming solution requires identifying
  - problem (given in the text - is it clear?)
  - constants (which values do not change?)
  - variables (which values do change?)
  - statements (what needs computing?)
- For any but trivial problems, spending time on gathering and structuring this information before beginning to code will save you lots of debugging time

27 Iteration: while and do loops - w12s28-30 (04/06-08/22)

- [X] `while` , `do` and `for` statements
- [ ] Exiting loops with `continue`, `goto`, and `Null`
- [X] Practice workbooks in GDrive
- [X] Add `(setq-default org-hide-emphasis-markers t)` to `.emacs`
- [X] Review program assignment 7 (`if` and `switch`)
- [X] New program assignment 8

27.1 Emacs tip: hide emphatic characters

To **not** see the emphatic characters like ~ or * or / in the Org file text, run the following code chunk (or put the code in your `/.emacs` file): if successful, you should see `"t"` in the minibuffer.

```
(setq-default org-hide-emphasis-markers t)
```

If you don't put it in your `/.emacs` file, the command will only work for the current Emacs session.

27.2 Emacs tip: change theme to "Leuven"

- In Emacs, open the theme selector with `M-x custom-themes`
- Select `Leuven` and click on `Save theme settings`
- Your code blocks are now more clearly visible

28 Review pgm 8, `for` practice, exiting loops - w13s31 (04/11/22)

- [x] Last quiz no. 9 - review on Apr-25, Test 3 on Apr-27 to Apr-28 (24 hrs) - unless I find time to make a last (ungraded) quiz…
- [x] The final exam on May 9 will contain 10 new questions from the topics of the next few weeks (but we will discuss them in class)
- [x] Review of programming assignment 8 (grades)
- [x] Review of `for` statement practice assignment
- [x] Exiting loops with `break`, `continue`, `goto`

29 One-dimensional arrays - w13s32 (04/13/22)

- [ ]

  Book recommendation for the holidays! "How to solve it" by George Pólya (1990 - see also application example).

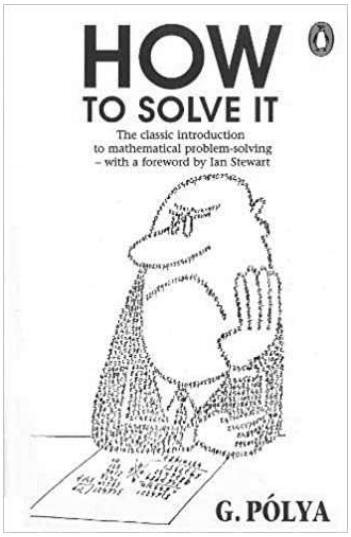


Figure 28: Perennial classic: Pólya's HOW TO SOLVE IT

- [ ] Change of plan: there will be **2 more quizzes** before the final exam, and one more before the Test 3, but no new questions on either Test 3 or final exam, so you can use all tests to prepare 100%.
- [ ] As promised there will be some [optional] **C++ training material** available after the end of term: it will consist of a series of notebooks introducing the C++ equivalents of the topics that we covered in this course.
- [ ] If you are going to take **data structures** (CSC 240) next, you should probably take a stab at this material. You can (and should!) contact me at any time if you have ouble with it.
- [ ] Today: "arrays" script in GitHub and Emacs workbook in GDrive.

30 Multidimensional arrays, sizeof - w14s33 (04/20/22)

30.1 Housekeeping

- [x] Play **quiz 7-9** - review next week
- [x] **Test 3** will be online only Thu 28-Apr 4pm to Fri 29-Apr 4pm
- [x] Final exam will cover all quizzes, all tests only
- [x] C++ overview planned for the last week (not in final exam)

30.2 Warming up with 1-dim arrays

- [ ]

  An array can be initialized with a single value. Complete the statements to print out the first and the last element of an array of 5 elements.

```
int iArray[5] = {0};
printf("first: %d, last: %d", iArray[0], iArray[4]);
```

- [ ]

  You can also initialize an array with a `for` loop. Find all mistakes in the following code block, which should initialize the array `iArray[5]`.

```
int iArray[5];

for (int i = 0; i < 5; i++ ) {
  iArray[i] = 0;
 }
```

- Solution:

```
int iArray[5];

for (int i = 0; i < 5; i++ ) {
  iArray[i] = 0;
  printf("%d %d\n",i, iArray[i]);
 }
```

30.3 Topics

- [ ] Multidimensional arrays: definition, declaration, initialization
- [ ] Size of arrays and `sizeof` operator
- [ ] Practice notebook `array2.org` in GDrive
- [ ] Solutions in GitHub (pdf/)
- [ ] Script in GitHub (7_{arrays}/)

30.4 Next: functions > pointers > strings > C++

31 Functions - w14s34 (04/22/22)

- [ ] Programming assignment 10: Fibonacci sequence (by V. Noppe)

32 Review Quiz 7-9 - w15s35 (04/25/22)

- FREE PIZZA + ORGANIC CHEMISTRY!

  > Interesting Topics in Organic Chemistry Seminars - On April 27th at noon in Derby 16. Pizza will be provided. The short seminars will be conducted by CHM 220 students and the content is geared to freshmen-sophomore and/or a general audience. (Prof. Narawathne)

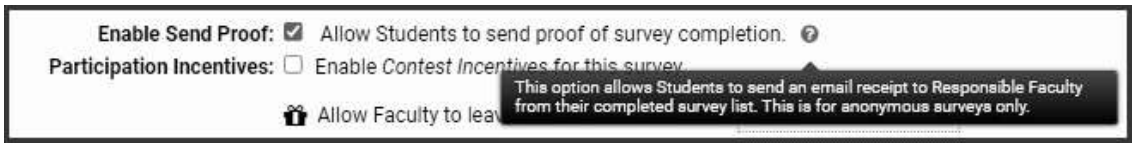
- [ ] Who is interested? (headcount = 0)
- **Test 3 will run from Thursday 6 pm to Saturday 8 am:**
  - You have 30 minutes for 15 questions
  - **You cannot resume your submission**
  - Let me know if there are any difficulties during the test (cell (501) 422 4725 or email)

33 Pointers - w15s36 04/27/22

- [X]

  Evaluate this course! Help Lyon decide if I stay or if I go!

  Extra credit for evaluating! - Don't forget to click the box.



- [x] Topic: pointers - C's memory fingers.
  - See `pointers_practice.org` in GDrive
  - See script in `9_pointers/` in GitHub
- [x] Quiz 10 (w/e) will cover: arrays, functions, and pointers
- [ ]

  Pgm 11 (w/e, Org-file, BONUS only) will cover pointers

Figure 30: Another pointer (Photo: N Spehner on Unsplash)

34 C++ basics - w16s37 (05/02/22)

34.1 Housekeeping

- [x] Quiz 10 is live (unlimited play from today)
- [x] No more program assignments, alas (ran out of time)
- [x] Fibonacci program assignments: corrections by mid-week
- [x] **Any outstanding work must be finished by Monday May 9**
- [x] Wednesday: final exam questions, summary & outlook
- [x] I will go through the individual questions with you

34.2 C++ basics

- [x] Overview / history
- [x] Basics of the language
- [x] Object orientation

34.3 Final exam scheduled

The final exam will be **in room 104 of the Lyon building** (our usual classroom) on **Monday, May 9 from 8-10 AM** (see exam schedule). You will complete the exam in Schoology.

The exam will consist of 40 questions drawn at random from the pool of quiz and test questions. Completing the various quizzes and tests until you've reached 100%, and revisiting your past tests should enable you to pass this exam with flying colors!

If you are a senior, you need to take the exam earlier - I'm going to offer additional slots on Friday, 6 May. Please reach out to me if this applies to you, or if you want to write the exam on Friday for any other reason.

Figure 31: sunset

35 Last Rites / final exam review - w16s38 (05/04/22)

- [ ] Object orientation (part 2)
- [ ] Your final exam questions
- [ ] Going through the exam questions

35.1 Evaluation

35.1.1 What I would change (probably) - video proto-script

- I have **really** missed student projects in this course. Perhaps it strikes you as odd to expect absolute beginners to do projects but I find it both interesting and instructive.

- Example (group or individual) projects could cover any of the many aspects of C programming that we did not cover for lack of time. Including:

    - Use of C in cybersecurity
    - Use of C in Internet of Things (IoT) and embedded computing
    - Popularity of C: why is this old horse still so attractive?
    - Comparing C with other languages (not C++)
    - C and the UNIX operating system (or Linux)
    - Debugging Cx
    - Quantum Computer Language (very similar to C)
    - The true history of the C programming language
    - GCC - the C compiler and compiling source code
    - Analyzing C programs
    - Using C for Artificial Intelligence and Machine Learning

  I have books or articles on every single one of these topics!

- I would force everyone to complete the (1 hr) Emacs tutorial **in class**. To the very end, I had the impression that 1/2 the class was not really mastering this tool and spent too much time trying to do simple things (like changing buffers, opening, saving and writing files, finding directories, etc.)
- My learning: I need to spend more effort trying to get you to take your **tools** seriously. Without mastery of your tools you are nothing in any discipline but this is especially true for programming. Many tools (like Emacs, liked editors, compilers, IDEs) don't look like tools because clever software engineers have created an environment for you where the food falls from the tree straight into your mouth most of the time. But that's not the real world! In the real world you have to **learn** to use a hammer to drive a nail into wood without hurting yourself!
- This last insight goes for all of my courses this year but especially for the introductory programming course.

35.1.2 Take 10 minutes to evaluate now

- [ ] **Please take 10 minutes to complete the course evaluation now**
- [ ]

  If possible, make specific suggestions on what (not) to do!

36 Final exam (PROCTORED) - w17s39 (05/09/22)

The final exam will be in room 104 of the Lyon building (our usual classroom) on Monday, May 9 from 8-10 AM (see exam schedule).

The exam will consist of 40 questions drawn at random from the pool of quiz and test questions. Completing the various quizzes and tests until you've reached 100%, and revisiting your past tests should enable you to pass this exam with flying colors!

If you are a senior, you need to take the exam earlier - I'm going to offer additional slots on Friday, 6 May. **Please reach out to me** if this applies to you, or if you want to write the exam on Friday for any other reason. (Schoology event)

37 References

- For some of the programming projects, see: King (2011). C Programming. W Norton & Co.
- Joyce (2018). Numerics in C. Springer Apress.

37.1 Image references

- Pointer photo by Nathalie Spehner on Unsplash

Footnotes:

[1] This requires system admin privileges, which you only have on your own PC. In the computer lab, I have such principles, and as soon as I managed to install our tools, you can also use them on the lab equipment.

[2] I managed to install GCC on the lab computers and run it inside GNU Emacs. This is something that you should do at home with your own computer. I'm going to demonstrate the process in class and I will also make a short video showing how to do it (for Windows 10).

[3] In other Operating Systems but Windows, there is a fourth shell, the terminal (`M-x terminal`), and in Windows, you also have PowerShell. All of these are ways of interacting with the OS.

[4] GNU Emacs is an extensible text editor. Org-mode is a plain text environment in GNU Emacs for keeping notes, authoring documents, computational notebooks, and more. Tangling code means extracting (human readable) source code from an Org-mode file in order to run it. Weaving documentation means extracting a document from an Org-mode file in order to read it.

[5] It's a notebook, so you could for example write a paragraph reflecting on how easy or difficult you found this. Or you could add code blocks that didn't work and comment on why not.

Author: Marcus Birkenkrahe
Created: 2022-05-03 Tue 21:27