Introduction to C

Tutorial 1: Introduction
Course Staff

Lecturer: Dr. Tal Tabachnik
Email: tabtal@campus.technion.ac.il

TA: Dr. Ron Adar
Email: adaron@technion.ac.il
Course Website

http://webcourse.cs.technion.ac.il/234126/

Homework

Course News

Helpful material, Past Exams, FAQ, and more

Lectures and Tutorials
Important messages

Must sign up!
Homework

Automatic extension of 3 days (without bonus)

1-2 weeks (with 5 pt. bonus)

Some of the assignments will have dry (printed) parts

Electronic Submission

Weight: 15%

Required for everyone!

Where to submit dry HW?

מבוא לשפת סי - אנגלית

Automatic extension of 3 days (without bonus)
Homework Submission

HW0
HW0.pdf 899 KB
expected.txt 22 Bytes
Due date: 9/11/2017, 23:55
Expected publication date: 22/10/2017
Electronic Submit
Submission in singles only.
Submission must be a ZIP file.
Submission in a ZIP file must exclusively contain the following files:

hw01.c
output.txt
student.txt

Note that the listed directories with trailing / (if exist) mark that you can add to them any files you wish (not only those listed above).

234126 - Introduction to Computer Language C
Winter 2017-2018
Assignments - Electronic Submit - HW0 -.eng

Follow the steps below to electronically submit your assignment.

1. Submission in singles only.
2. Prepare the file you wish to submit. You may submit only one file. Submission must be a ZIP file.
3. Click the Proceed button below:

(Proceed)
Save the confirmation code after submitting homework!
Course textbook

Primary book is:
"ABC" or A Book on C
by Kelley and Pohl

C development book
The C Programming Language
by Kernighan and Ritchie
Not recommended for beginners
Important to know!

Copying
Whoever copies will get a grade of 0 and may be subject to the Disciplinary Committee
Course Goals

• Understand the basic building blocks of a computer
• What is a programming language?
• How to write programs in C
• What is an algorithm?
• Why should you learn how to program?
Computer structure

USB Ports

Mouse

Screen

Keyboard
Computer Structure from the inside

(Bus - Communication channel among components)

Input

CPU

Memory

Output

Other memory devices
• Every computer requires an operating system, from traditional computers to cellular phones.
• An operating system is a **software program** just like any other.
• It supplies different types of services to the user.
  – Makes working with files, input and output devices.
  – Running of many programs simultaneously.
  – Calculator, browser, other small programs, etc.
• In addition, the operating system provides services that are required to run the programs that you wrote.
  – Memory allocation
  – Creating Input and output
  – Easy work with files
  – Interaction with other programs
  – And much more…
• In a computer, information is stored as bits
• Each file is a sequence of bits
• A computer has hundreds of thousands of files
• How is it possible to organize them?
Files

- File name is made up of a name and extension.

  - Ex: my_file.doc

- The extension signifies the content that is in the file.
  - .doc, .txt → word document / text files
  - .gif, .bmp, .jpg, .png → images
  - .avi, .mpg → movies
  - .wav, .mp3 → audio
  - .exe → Programs

A program is actually a series of instructions in machine code/language.

עברית לשפת סי - אנגלית

Page 16
File organization

- Files are stored on disks
- Every disk’s name consists of a name followed by a colon (Ex. C:)
- Within the disks, files are stored in partitions
- **Partitions** are sometimes called **directories** and **folders**.
- Every partition can have any number of files and other partitions.
File organization

At the top of the tree is "primary partition", or "root partition" of the disc.

The file structure resembles a tree

This is called a hierarchical organization.
Paths

• Every file in a directory must have a unique name. Files can have the same name if they are in different folders.

• Therefore to deal with a specific file, it is not enough to only specify the file name.

• To deal with this problem, we specify the path to the file we want.

The path for ex1.c:

C:\home\progs\ex1.c

– Separate steps in the path with “\”
– What is the path for the other ex1.c?
The C language & Code::Blocks
- Translation program is called a compiler
- Translation process is called compilation
Writing a C program

- Write the program in the C language, and save it in a "c" file.
- Compilation outputs an "exe" file, which contains machine code.
- Run the "exe" file

A language that we understand

Language that the computer understands
What is Code::Blocks?

• Free, Open source development environment.
  – Editor – for writing program
  – Compiler
  – Debugger – tools for finding mistakes
• Supports many programming languages, including C
• As in any industry, there are many C standards
  – We will use the ANSI-C standard
• There are versions for Windows, OS X, and Linux
Installation of Code::Blocks

Download Code::Blocks from the “Course Material” section in the course’s website.
Working with Code::Blocks
Writing a new program

File → New → Project → 234114 Course Homework
Writing your first program

• Title your project according to the assignment number. i.e. “HW0”
Writing your first program

Inside the code blocks editor, text is highlighted to help you.

Every color has significance. We’ll cover the details later.

As the course progresses, we’ll understand all of the components of this program.
Running the program

• Press on the F9 or the button to compile and run.
• Use Ctrl+F11 to rebuild a project from scratch, if you want to ensure that all files were compiled.
• Code::Blocks will then compile the program. If there are no compilation errors, Code::Blocks will create an .exe of your program and run it.

Output of the program appears in the window that opens:
Compilation errors

• Just like any language, you can have errors writing in C.
• If there is an error, then compilation will fail. The program will not run and you will get the following message.

• Clicking the error will bring you to the line number in the code

Important: Fix errors in order from top to bottom!
Opening an existing program

• You can save a program that you wrote and continue it at a later time. (File → Save File)
• To open it, open the project file (.cbp) and **not** the .c file.
• This can be done by accessing the .cbp file directly, or (File → Open...)
Running a program
Running of the program

• We saw that you can run a program through Code::Blocks

• You can also run a program by double clicking on the .exe file in Windows.
  – Drawback: Output window closes before we can see it.

• Another way to run the program - Windows Command Prompt.
  – In other operating systems, this is called the “Terminal”
Opening Command Prompt

- Windows XP
  Start → Run → “cmd” → Enter
- Windows 7
  Start → “cmd” → Enter
- Windows 10
  Search tool → “cmd” → Enter
Working with Command Prompt

• Enter DOS commands
  – They are executed immediately
• If the window is too small, or text not readable, etc., change the properties of the window.
Upon opening the command prompt, the system waits for commands.

Before each command, the system tells us the current directory (where the command will execute)

- This is the file path from the root directory to the current directory
Moving between folders

- You can change the current folder using the command `cd name`
- `cd` is an abbreviation for change directory. You can change directories to any within the current folder.
Errors

• What happens if we type \texttt{cd name} for a directory that does not exist?

• We get an \texttt{error message}.
• It’s not a disaster – we can simply try again and get it right.
Moving to a parent directory

- The folder named “..” (two dots) is a special directory.
- It represents the directory that holds the current one.
  - This refers to the “parent” directory
- Example: we can use this as part of our “cd” command.

- Using “cd ..” we will eventually arrive to the root directory.
- What will happen if we execute “cd ..” from there?
Uppercase or lowercase?

• You can type CD, cd, CD, Cd, etc. Windows will accept any form.
• We will see, however, that sometimes this makes a big difference.
• In C, for example, the following two are different.
Running a program

• To run a program, type the filename and press enter.

• You can also type the name without .exe
• Pressing Tab will attempt to complete the file name.

Code::Blocks keeps the .exe within the project folder that was selected during project creation.
• Every program can receive data from the user as **input**.
• Every program can show data to a user as **output**.
• Keyboard and screen are defaults for input and output (I/O)
Input and output Redirection

• You can simulate the input / output by using files.
• With every execution, you can print the output into a text file, instead of the screen.

Example:

```
hw0q1.exe > my_output.txt
```

How is this done?
Why is this useful?

• The main use (in this course) of reading input from files and sending the output to the files will be the homework.

• You will occasionally get files of input with which you will run your program.

• You will be able to check that your output matches sample output.

• This also saves the time it would take to enter in a lot of data.
Creating a ZIP file for submission

• You will put all of the required files in a ZIP archive
• If you do not have a program that does this, you can download 7zip.
• How is it done?
  – Copy all of the files for submission to a folder and create the archive
  – Upload the zip file to site, for the corresponding assignment.
Creation of a zip file
Summary: How to do the assignments.

1. Think about the problem on paper (Optional)
2. Open Code::Blocks.
3. Open a new project.
4. Write the program and compile (Ctrl+F11 or F9).
5. Open the command prompt.
6. Change the current directory with “cd”.
7. If necessary, write input file(s) with notepad.
8. Run the program with redirection for the input file(s), verifying correct expected output.
9. Zip the files in the project folder in an archive called <student_id>.zip
10. Upload file to the course website.