

Programming in C

Jan Faigl

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Course Organization
B0B36PRG – Programming in C


Overview of the Lecture

- Part 1 – Course Organization
 - Organization
 - Course Goals
 - Means of Achieving the Course Goals
 - Evaluation and Exam
 - Communication
 - Tools and Academic Network Services




Part I Part 1 – Course Organization

Course and Lecturer

B3B36PRG – Programming in C

- Course web page <https://cw.fel.cvut.cz/wiki/courses/b3b36prg>
- Submission of the homeworks – **BRUTE** Upload System <https://cw.felk.cvut.cz/brute> and individually during the labs.
- Lecturer:
 - prof. Ing. **Jan Faigl**, Ph.D. 
 - Department of Computer Science – <http://cs.fel.cvut.cz>
 - Artificial Intelligence Center (AIC) <http://aic.fel.cvut.cz>
 - Center for Robotics and Autonomous Systems (CRAS) <http://robotics.fel.cvut.cz>
 - Computational Robotics Laboratory (ComRob) <http://comrob.fel.cvut.cz>

Teachers

- RNDr. **Ingrid Nagyová**, Ph.D. 
- MSc. **Yuliia Prokop**, Ph.D. 
- Ing. **Martin Zoula** 


Course Organization and Evaluation

- B3B36PRG – Programming in C; Completion: Z,ZK; Credits: 6
Z – ungraded assessment, ZK – exam
- 1 ECTS credit is about 25–30 hours per semester, six credits is about **180 hours per semester**
 - Contact part (lecture and labs): 3 hours per week, i.e., 42 hours in the total
 - Exam including preparation: *10 hours*
 - Home preparation (first **book reading** and followed by homeworks) approx **9 hours per week** *Median load*
- Ongoing work during the semester**
 - Homeworks *mandatory, optional, and bonus parts*
 - Semestral project** – multi-thread computational applications.
- Exam test and implementation exam – verification of the acquired knowledge and skills from the teaching part of the semester. *An independent work with the computer in the lab (class room).*
- Attendance to labs, submission of homeworks, and semestral project.
- Consultation** - If you do not know, or spent too much time with the homework, consult with the instructor/lecturer.
 - Maximize the contact time during labs and lectures, ask questions, and discuss.**





Lectures – Spring Semester Academic Year 2024/2024

- Schedule for the academic year 2023/2024. <https://intranet.fel.cvut.cz/cz/education/harmonogram.html>
- Lectures:
 - Dejvice, Lecture Hall No. T2:D3-209, Tuesday, 16:15-17:45.
- 14 teaching weeks - (19.2.–26.5.2024); 13 weeks in practice.
 - National holiday – 01.04.2024 (Monday).
 - National holiday – 01.05.2024 (Wednesday).
 - National holiday – 08.05.2024 (Wednesday).
 - Rector's day – 14.05.2023 (Tuesday).
 - Thursday 09.05.2024 – classes as on Wednesday (odd teaching week).

Resources and Literature

- Textbook**
 - „C Programming: A Modern Approach“ (King, 2008) 
 - C Programming: A Modern Approach, 2nd Edition, *K. N. King*, W. W. Norton & Company, 2008, ISBN 860-1406428577 *The main course textbook*
- During the first weeks, take your time and read the book!**
The first homework deadline is in 18.3.2023.
- Lectures – support for the textbook, slides, comments, and **your notes**.
Demonstration source codes are provided as a part of the lecture materials!
- Laboratory exercises – gain practical skills by doing homeworks (yourself).

Further Books

-  **Programming in C, 4th Edition**, *Stephen G. Kochan*, Addison-Wesley, 2014, ISBN 978-0321776419
-  **21st Century C: C Tips from the New School**, *Ben Klemens*, O'Reilly Media, 2012, ISBN 978-1449327149
-  **The C Programming Language, 2nd Edition (ANSI C)**, *Brian W. Kernighan, Dennis M. Ritchie*, Prentice Hall, 1988 (1st edition – 1978)
-  **Advanced Programming in the UNIX Environment**, 3rd edition, *W. Richard Stevens, Stephen A. Rago* Addison-Wesley, 2013, ISBN 978-0-321-63773-4

Further Resources

-  **The C++ Programming Language, 4th Edition (C++11)**, *Bjarne Stroustrup*, Addison-Wesley, 2013, ISBN 978-0321563842
-  **Introduction to Algorithms, 3rd Edition**, *Cormen, Leiserson, Rivest, and Stein*, The MIT Press, 2009, ISBN 978-0262033848
-  **Algorithms, 4th Edition**, *Robert Sedgewick, Kevin Wayne*, Addison-Wesley, 2011, ISBN 978-0321573513

The Assignment Principle

- Writing a program to assign values to variables *a* and *b* and then assigning variable *b* to *a*.
Assigning a value to a variable
- ```

1 int a = 10;
2 int b = 20;
3
4 a = b;

```
- What are the values of the variables *a* and *b*?
 

|                   |                   |
|-------------------|-------------------|
| a. a = 20, b = 0  | f. a = 30, b = 0  |
| b. a = 20, b = 20 | g. a = 10, b = 30 |
| c. a = 0, b = 10  | h. a = 0, b = 30  |
| d. a = 10, b = 10 | i. a = 10, b = 20 |
| e. a = 30, b = 20 | j. a = 20, b = 10 |

*Program actually "only" moves and modifies numeric values in memory based on defined conditions!*

### Overview of the Lectures

- Course information, Introduction to C programming *K. N. King: chapters 1, 2, and 3*
- Writing your program in C, control structures (loops), expressions *K. N. King: chapters 4, 5, 6, and 20*
- Data types, arrays, pointer, memory storage classes, function call *K. N. King: chapters 7, 8, 9, 10, 11, and 18*
- Data types: arrays, strings, and pointers *K. N. King: chapters 8, 11, 12, 13, and 17*
- Data types: Struct, Union, Enum, Bit fields. Preprocessor and Large Programs
- Input/Output – reading/writing from/to files and other communication channels, Standard C library – selected functions *K. N. King: chapters 10, 14, 15, 16, and 20*
- Parallel and multi-thread programming – methods and synchronizations primitives *K. N. King: chapters 21, 22, 23, 24, 26, and 27*
- Multi-thread application models, POSIX threads and C11 threads
- C programming language wrap up, examples such as linked lists
- Accuracy and Speed of Calculation*
- ANSI C, C99, C11 and differences between C and C++* Introduction to C++.
- Quick introduction to C++
- Reserve (Rector's day)*
- Resource Ownership in C++*

All supporting materials for the lectures are available at <https://cw.fel.cvut.cz/wiki/courses/b3b36prg/start>  
Read slides, **textbook**, or even watch the recorded lectures before the lecture contact time!

### Course Goals

- Master** (yourself) programming skills. *Labs, homeworks, exam*
- Acquire** knowledge of C programming language
- Acquire experience** of C programming to use it efficiently *Your own experience!*
- Gain experience** to read, write, and understand small C programs
- Acquire** programming habits to write
  - easy to read and understandable source codes
  - reusable programs
- Experience** programming with
  - Workstation/desktop computers – using services of operating system  
*E.g., system calls, read/write files, input and outputs*
  - Multithreaded applications
  - Embedded applications – STM32F446 Nucleo

### Program is a "Recipe"

- Program is "recipe" – a sequence of steps (calculations) describing the process of solving a problem.**
  - Programming is the ability to **independently**
    - Create programs;**
    - Decompose** problems into smaller units;
    - build larger programs from **subparts** to solve a complex problem.
- B3B36PRG – is an opportunity to learn and gain these skills.

### Homework and Other tasks

- Independent work to gain practical experience.
- Assignment at the lectures and defined submission date. All assignments are defined.
- Submission of homework through BRUTE. <https://cw.felk.cvut.cz/brute>
- Uploading the archive with the necessary source files.
- Verify the correctness of the implementation with automated tests.
- Penalties for exceeding the number of uploads. **Submit correct codes, not "only" code that passes tests!**
- Plagiarism detection *The aim of solving the problems is to get your own experience!*
- Tasks are designed to be achievable. Plan and keep track of time, consult early.
- Independent work and mastery of techniques and knowledge is the key to successful completion of the course. *Continuous work and problem solving!*
- If you do not understand something, **ask!** *If you make mistakes you learn, if you do not make mistakes you already know!*

### Teaching Programming

„Separating Programming Sheep from Non-Programming Goats“  
<http://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats>  
<http://www.eis.mdx.ac.uk/research/PhDArea/saaed/paper1.pdf>

- Effective methods of teaching programming have been sought since the early days of computers. *More than 50 years.*
- Yet, it seems that every basic programming course is difficult and about 30%–60% of students fail it for the first attempt. a *Success rate in the PRGA is much higher.*

|                                             |
|---------------------------------------------|
| 2022/2023: 73% (97% of awarded credits, 72) |
| 2021/2022: 60% (97% of awarded credits, 75) |
| 2020/2021: 60% (95% of awarded credits, 97) |
| 2019/2020: 73% (97% of awarded credits, 91) |

- The basic concept is to understand the principle of assigning a value to a variable!**

*It mainly about understanding the memory representation and access to it, which is very direct in C.*

### Teaching Programming in B3B36PRG

- Our aim is to build your experience and develop your programming skills.
  - Programming vs. algorithmization;
  - Programming is the "craft" of how to implement an algorithm correctly.
  - Functional is not enough - the program must be correct too!** *Expected input vs. what the user can input.*
- The learning load is therefore spread over the course of the semester.
  - Practice assignments and homework deadlines.
- Systematic development of programming skills throughout the semester is essential. *Typically, there is time at the beginning of the semester to understand the principles (reading the textbook!)*
- Without knowing the constructs and basic commands, you cannot program effectively.
- Know and know how to use (not "stick"). *Dependence on whisperer or Co-pilot!*
  - Starting with relatively simple tasks to learn programming constructs and how to organize source code. *Code clarity and the ability to navigate code efficiently!*
  - The assignments can always be implemented based on the topics covered the lectures/labs.** *Solutions with more advanced constructs may be more elegant(shorter), but may not provide the necessary insight.*
  - In the first lectures we cover the necessary knowledge, which is further deepened.
    - Exercises complement the lectures and give more space for practical learning.
- You can choose a practical way of absorbing programming knowledge from examples, which is suitable to complement **theoretical preparation from textbook(s).**

### Homeworks

- 1+7 homeworks - seven for the workstation. <https://cw.fel.cvut.cz/wiki/courses/b3b36prg/hw/start>
- HW 00 – **Testing** (1 point) 1 h
  - HW 01 – **ASCII Art** (2 points) 3 h
- Coding style penalization – up to -100% from the gain points.**
- HW 02 – **Prime Factorization** (2 points + 4 points bonus) **Coding style** 4 h + 4 h (bonus)
  - HW 03 – **Caesar Cipher** (2 points + 2 points bonus) **Coding style** 3 h + 3 h (bonus)
  - HW 04 – **Text Search** (2 points + 3 points optional) 5 h
  - HW 05 – **Matrix Calculator** (2 points + 3 points optional + 4 points bonus) **Coding style!** 6 h + 5 h (bonus)
  - HW 06 – **Circular Buffer** (2 points + 2 points optional) 5 h
  - HW 07 – **Linked List Queue with Priorities** (2 pts + 2 pts optional) 7 h
- All homeworks must be submitted to award an ungraded assessment *Total about 42–47 hours. Late submission is penalized!*
  - Coding style needs to be learned, penalization is to motivate you thinking about it and learn the craft of coding. *If you improve over the semester, penalization can be compensated at the end.*

## Semestral Project

- A combination of control and computational applications with multithreading, communication, and user interaction.
  - <https://cv.fel.cvut.cz/wiki/courses/b3b36prg/semestral-project/start>
- Mandatory task can be awarded up to **20 points**.
- Bonus part can be awarded for additional **10 points**.
  - Up to 30 points in the total for the semestral project.
- Minimum required points: 10!**
  - Deadline – best before 17.05.2024.
  - Further updates and additional points might be possible!
  - Deadline – 19.05.2024.
- Expected required time to finish the semestral project is about 30–50 hours.

## Course Evaluation

| Points              | Maximum Points | Required Points | Minimum Points |
|---------------------|----------------|-----------------|----------------|
| Homeworks and labs  | 40             | 25              |                |
| Semester project    | 30             | 10              |                |
| Exam test           | 20             |                 | 10             |
| Implementation exam | 20             |                 | 10             |
| Total               | 110 points     | 35 points is F! |                |

- 25 points** from the homeworks and 10 points from the semestral project are required for awarding ungraded assessment.
- The course can be passed with **ungraded assessment and exam**.
- All homeworks must be submitted and they have to pass the mandatory assessment.**

## Computers and Development Tools

- Computer labs - network boot.
  - Sync your files using, e.g., ownCloud, gdrive, ssh, ftp.
  - You have to set your password via <https://felk.cvut.cz> – rooms of Dept. of Computer Science.
  - You need the access for implementation exam.
- Compilers **gcc** or **clang**.
  - <https://gcc.gnu.org> or <http://clang.llvm.org>
- Project building **make** (GNU make).
  - Examples of usage on lectures and labs.
- Text editor – **gedit**, **atom**, **sublime**, **vim**.
  - <https://atom.io/>, <http://www.sublimetext.com/>
  - <http://www.root.cz/clanky/textovy-editor-vim-jako-ide>
- Visual Studio Code** – code – great for editing and terminal based compilation.
- C/C++ development environments – **WARNING: Do Not Use An IDE** at the beginning, to become familiar with the syntax.
  - <http://c.learncodethehardway.org/book/ex0.html>
  - Visual Studio Code**; CLion – <https://www.jetbrains.com/clion/>; Code::Blocks, CodeLite, NetBeans (C/C++), Eclipse-CDT.
- Embedded development for the Nucleo**.
  - ARMmbed – <https://os.mbed.com/platforms/ST-Nucleo-F446RE/>
  - <https://studio.keil.arm.com/>
  - System Workbench for STM32 (based on Eclipse); direct cross-compiling using makefiles.

## Homework Assignment – BRUTE

- BRUTE** – Bundle for Reservation, Uploading, Testing and Evaluation
  - Formal check – compiling the program.
  - Functionality and correctness testing – **checking output for a given input**.
    - Public inputs and corresponding outputs / non-public inputs.
  - Test the program yourself before uploading it.
    - Using the available inputs and outputs.
    - Creating your own inputs and debugging the program.
    - Creating inputs **with the included input generator**.
    - Verifying the output **with the attached test or reference program**.
  - Understanding the code and checking possible states.
    - For each line, you should be able to answer why it is there and what it does!**
    - For **each function or input retrieval** from the user, parse the possible input values or **function return values!**
      - If the input or return value is critical in terms of functionality, **check the input and/or the appropriate action**, e.g., output a message and exit the program.
      - For example, the expected input is a number and the user enters something else.

## Grading Scale

| Grade | Points | Mark | Evaluation   |
|-------|--------|------|--------------|
| A     | ≥ 90   | 1    | Excellent    |
| B     | 80–89  | 1,5  | Very Good    |
| C     | 70–79  | 2    | Good         |
| D     | 60–69  | 2,5  | Satisfactory |
| E     | 50–59  | 3    | Sufficient   |
| F     | <50    | 4    | Fail         |

- Expected results
  - Timely submission of all homework with required and optional assignments (**35 points**).
  - Semestral project (**20 points**) and bonus assignments (**5–10 points**).
  - Exam test (**15+ points**). 15 and more points is respectable result!
  - Exam implementation (**20 points**).
  - 95+ points** and more (A – Excellent) – with small imperfection.
  - 76 points** (C – Good) for 20% loss .

76 and more points represents a solid background for further development of your programming skills.

## Services – Academic Network, FEE, CTU

- <http://www.fel.cvut.cz/cz/user-info/index.html>
- Cloud storage ownCloud – <https://owncloud.cesnet.cz>
- Sending large files – <https://filesender.cesnet.cz>
- Schedule, deadlines – FEL Portal, <https://portal.fel.cvut.cz>
- FEL Google Account** – access to Google Apps for Education
  - See <http://google-apps.fel.cvut.cz/>
- Gitlab FEL – <https://gitlab.fel.cvut.cz/>
- Information resources (IEEE Xplore, ACM, Science Direct, Springer Link)
  - <https://dialog.cvut.cz>
- Academic and campus software license
  - <https://download.cvut.cz>
- National Super Computing Grid Infrastructure – MetaCentrum
  - <http://www.metacentrum.cz/cs/index.html>

## Tasks and BRUTE

- Tasks are not just about submitting an implementation that passes the BRUTE tests.
  - The goal is not to submit tasks in BRUTE, it to verify the program functionality.
  - BRUTE is a tool to continuously check progress and gain knowledge.
  - The goal is to learn to **independently program** functional programs correctly.
- Tasks are all about gaining **gradual experience** with specific constructs.
  - All of the task assignments have been implemented many times, and even generative AI can do it.
  - In this course you have the opportunity to understand C programming through your own implementation of assignments. **The task successful submission is a means to reach the goal, not the goal itself.**
- Tasks are very similar in relative difficulty. It is important to solve the tasks independently and to learn the sub-skills.
  - Absolutely, the tasks get progressively more and more difficult!
- Rather than struggling too long by your own, ask (on Discord), for practice or **consultation**.
- Tasks HW01–HW03 and HW05 are checked for correctness and clarity of code.
  - Focused on consistency, readability, and **modularity** (splitting into functions).
  - In terms of training and learning, try to split even a seemingly trivial program into multiple functions.
  - The motivation is not to spend too much time implementing without significant progress.

## Communicating Any Issues Related to the Course

- Ask the lab teacher or the lecturer.
- Use e-mail for communication.
  - Use your **faculty e-mail**.
  - Put **PRG** or **B3B36PRG** to the subject of your message.
  - Send copy (Cc) to lecturer/teacher.
- Discord channel**.