Methodology and Programming Techniques

Department of Telecommunications, iet

dr inż. Jarosław Bułat

kwant@agh.edu.pl

Outline

- course regulation
- scope of course
- basic concepts
- ways of representing information
- what is a computer - Turinga machine
- my first program in C++
- GIT
- where to find more information?
- MOOC

course regulation

- Lecture (28h) + Laboratory classes (28h)
- Lecture is a practical introduction to Lab (!)
- Lab is obligatory (compulsory)
- Classes teacher: Artur Kos
- Lab consists of making practical exercises
  - Linux is preferred
  - own laptops are allowed (**BYOD**)
  - primary language: C++, (Python, Matlab in the summer semester)
- Lab grade is issued on the basis of: final test (50 pts) + class test (5*10 pts)
  - during final test **you can use** notes, books, Internet, etc...
  - during final test **you cannot** communicate with others (eg. forums, FB,...)
- Final grade: floor((lab grade + exam grade)/2) ex: {4.5, 4} -> 4
  - 0th exam date only if 4.5+ from Lab

course regulation

questions?

Motivation :-)

- » IT is a vast field of knowledge
- » you can not learn it "chronologically"
- » part of the information you have to accept "on faith", will be clarified later
- » do not be discouraged if you do not understand something
- » if you can program, come to the lecture:
  - – you will know a different point of view
  - – you will correct me if I make a mistake :-)

AGH

Do not understand something?

ask a question !!!

Do not understand something?

ask a question !!!

now (during lecture)
later (e-mail, office-hour)
ask lecturer or lab teacher
ask your friends

Scope of course

- Applied computer science: **you will be able to write a program that solves a specific problem**
- It is large variation of your knowledge/experience (possible projects)
- Course scope **does NOT include**: information theory, operating system construction, build compilers, etc ...
- Scope of the course **includes**:
  - practical programming skills (C++, python)
  - basic programming techniques and methods
  - basic algorithms (implementation ability)
  - analysis,debugging code
  - teamwork elements (git, coding standards, etc ...)

Basic concepts - **computer science**

- Computer science is the study of the theory, experimentation, and engineering that form the basis for the design and use of computers (wikipedia)
- Selected topics:
  - information theory
  - algorithm (creation and analysis of algorithms)
  - programming languages (design)
  - computer hardware (design, construction)
  - **computer programming** (algorithm implementation in selected programming language on computer hardware)
  - building software systems and hardware
  - software engineering (computer science + management)
  - network administration
  - computer graphics, simulation, CPU architecture, AI, webmastering,

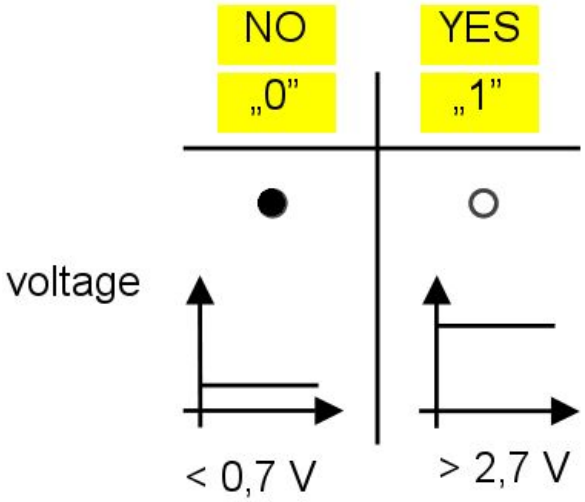
Basic concepts - **information**

in the context of informatics, used in an objective sense

- » object property (not necessarily physical)
  - "no. shoe" (my), file size, color, ...
- » highlighted state of the object (indicated state, from the set of possible)
  - wall switch, ON or OFF
- » Information can be considered in the context of:
  - communication (communication, communication speed)
  - structure (object layout or/and properties)

binary number system

» most of computers are digital
» binary system is the most common representation of information in c.



sequences of bits: ●○○○●●●○

- numbers:
fixed-point
floating -point
data for computations (input/output)
addresses

- characters:
alphanumerical (letters, digits, ...)
control (space, backspace, ...)

- processor instructions

Numeral system

x={0,1}

x	xx	xxx	xxxx
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

MSB LSB
0111001010001101

MSB = Most Significant Bit
LSB = Least Significant Bit

Numeral system

x	xx	xxx	xxxx
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

x={0,1}

no. of bits	no. of states (comb.)
x	=2
xx	=2*2
xxx	=2*2*2
xxxx	=2*2*2*2

Numeral system

x	xx	xxx	xxxx
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

x={0,1}

no. of bits	no. of states (comb.)
x	=2
xx	=2*2
xxx	=2*2*2
xxxx	=2*2*2*2
xx... (n)	$=2^n$
bits	**$=2^n$ states**

Numeral system

x	xx	xxx	xxxx	x={0,1}	no. of bits	no. of states (comb.)
0	00	000	0000			
1	01	001	0001		x	=2
	10	010	0010		xx	=2*2
	11	011	0011		xxx	=2*2*2
		100	0100		xxxx	=2*2*2*2
		101	0101			
		110	0110		xx... (n)	$=2^n$
		111	0111		**bits**	**$=2^n$ states**
			1000		n=1	=2
			1001		n=2	=4
			1010		n=3	=8
			1011		n=4	=16
			1100		n=8	=256
			1101		n=16	=65536
			1110		n=32	=4294967296
			1111		n=64	=18446744073709551616

Numeral system

x	xx	xxx	xxxx	x={0,1}	no. of bits	no. of states (comb.)
0	00	000	0000			
1	01	001	0001		x	=2
	10	010	0010		xx	=2*2
	11	011	0011		xxx	=2*2*2
		100	0100		xxxx	=2*2*2*2
		101	0101			
		110	0110		xx... (n)	$=2^n$
		111	0111		**bits**	**$=2^n$ states**
			1000		n=1	=2
			1001		n=2	=4
			1010		n=3	=8
			1011		n=4	=16
			1100		n=8	=256
			1101		n=16	=65536
			1110		n=32	=4294967296
			1111		n=64	=18446744073709551616

$n=\log_2$(no. of states)

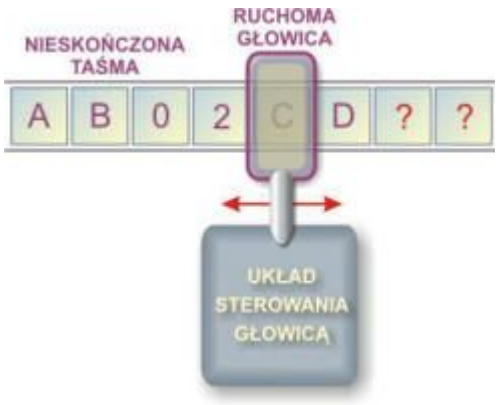
bin-dec-hex conversion

next week

How is the computer built?

Turing machine

- A mathematical model of computation that defines an abstract machine, simulate algorithm's logic
  - infinite memory tape
  - reading/writing head
- **tape**: memory
- **head**: I/O device
- **control unit**: CPU
- It is used to prove theorems
- Modern computer programs can be performed by Turing machine

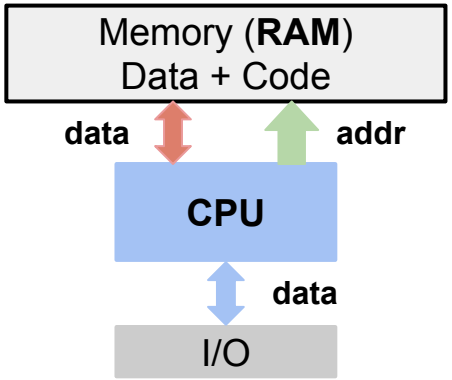


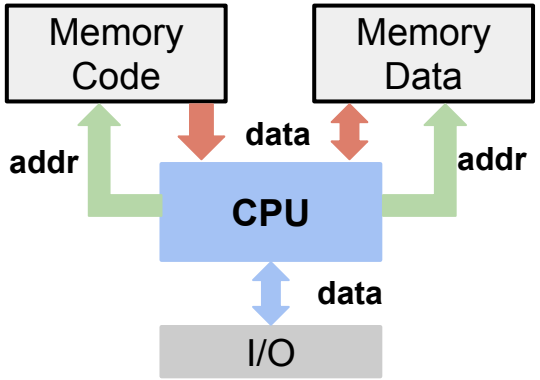
http://eduinf.waw.pl/inf/prg/003_mt/0001.php

computer architecture

- **Computer**
  - machine (electronic/digital) designed to data processing
  - **programmable** (universal)
- Basic computer components:
  - CPU
  - RAM
  - I/O
- architecture:
  - von Neumann
  - Harvard
  - mixed

von Neumann vs Harvard





- Single RAM, one buse - **cheap**
- PC, servers, general computing

- Two memory, two busses: parallel access to data and instruction (**faster**)
- Program code is protected against changes
- DSP, uC (short programs)

How do I "order" the computer to do something?

in a computer language...

Computer programming

- **programming -** the process of designing, creating (writing), testing, and maintaining the source code of programs
- **A programming language** consists of rules (grammar) + instructions (words)
- **Source code** is the program written in a programming language
- programming (coding, developing) is one of the stage of program creation (software engineering)

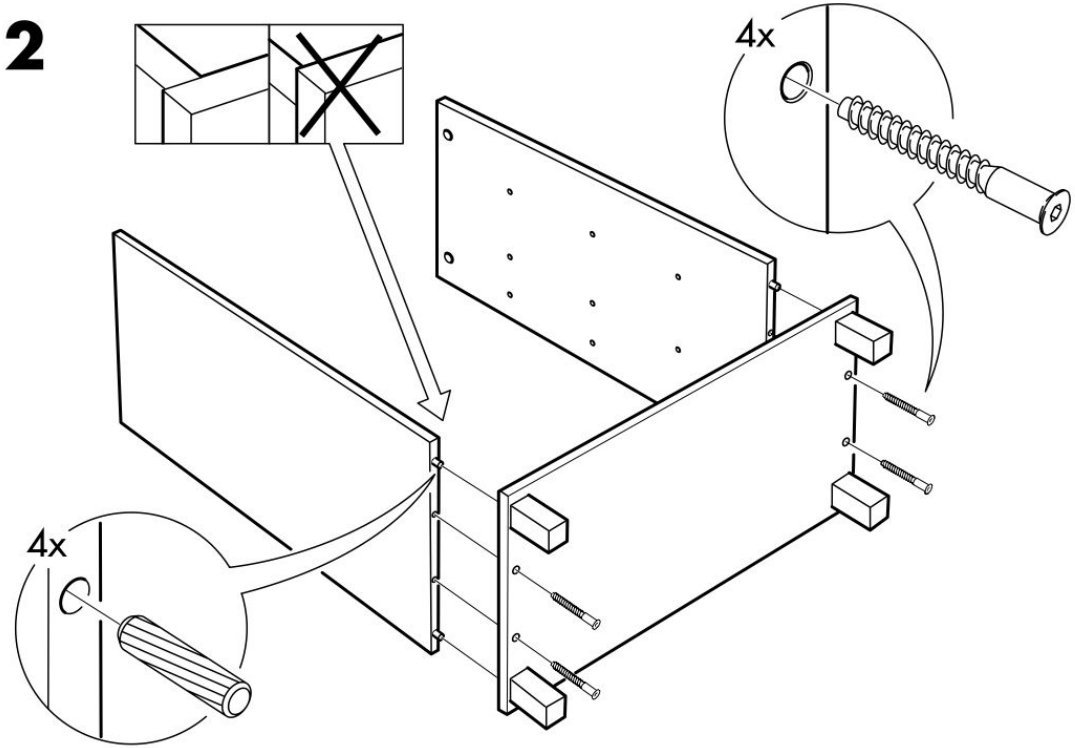
A programing language

- **A programming language** – a formal language that specifies a set of instructions that can be used to create computer program
- The programming language: syntax + instructions
  - It is a formal language = clear rules
  - stored in the form of instructions, keywords comply with the rules of syntax
  - the order of writing instruction is important

Classification of computer languages

- Abstraction level:
  - high-level languages (C/C++, C #, Java, Python, ...)
  - low-level language (Assembler, gC - GPU)
- Implementation/execution:
  - compiled (C/C ++, Java*)
  - interpreted (JavaScript, Python, PHP, Perl, Matlab)
- Basic paradigms:
  - **imperative** (instructions that execute on the data to achieve a goal)
  - Functional (how to submit expressions to achieve the target)
  - Descriptive (for what state the form of inputs and the system will be achieved goal)
  - logical (proof of the theorem which is the expected result?)
  - ...

imperative “program”



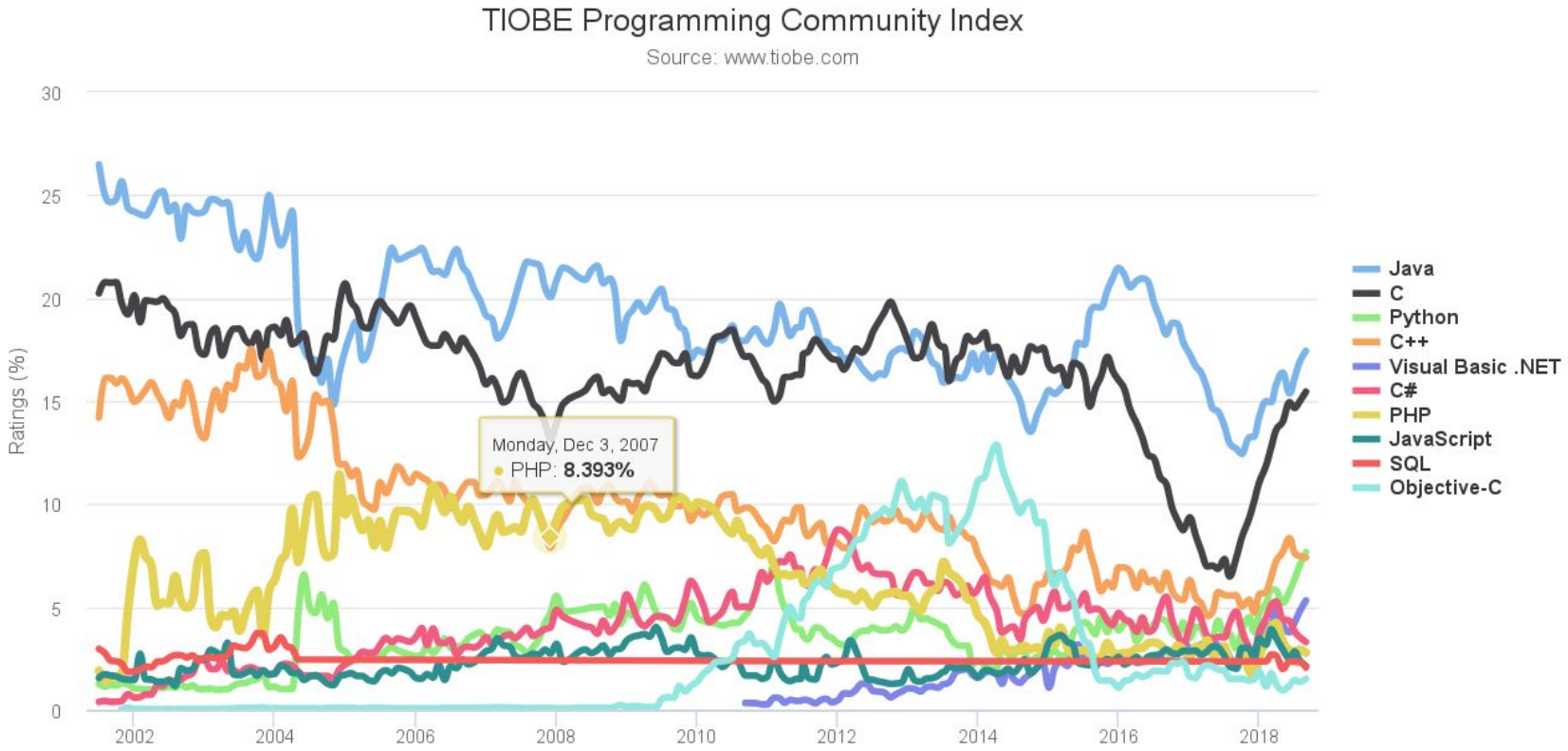
Programming models

- » linear (Basic)
- » procedural
- » structural (Pascal, C)
- » **object** (C++, Java)
- » functional (Haskel, LISP)
- » states (PLC controllers)
- » declarative
- » logical
- » ***: parallelism, security, speed of development, time-to-market :-/

Programming languages

- TIOBE Index (September 2018)
  - Java 17.4% (+4.7%)
  - C 15.4% (+8.1%)
  - Python 7.6% (+4.7%)
  - C++ 7.4% (+1.8%)
  - VB.NET 5.3% (+3.3%)
  - C# 3.3% (-1.5%)
  - PHP 2.8% (+0.5%)
- www.tobie.com "TIOBE index is not about the best programming language or the language in which most lines of code have been written.
  The TIOBE Programming Community index is an indicator of the **popularity**"

Programming languages



My first program
C++

```cpp
#include <iostream>

int main(){
    std::cout << "Hello world" << std::endl;
}
```

```
 ~/D/P/lab_02_fistCPP> g++ ex1.cpp  -o ex1
 ~/D/P/lab_02_fistCPP> ls -al
razem 40
drwxrwxr-x 2 kwant kwant 4096 paź  7 18:33 ./
drwxrwxr-x 5 kwant kwant 4096 paź  7 18:29 ../
-rwxrwxr-x 1 kwant kwant 9216 paź  7 18:33 ex1*
-rw-rw-r-- 1 kwant kwant   76 paź  7 18:29 ex1.cpp
 ~/D/P/lab_02_fistCPP> ./ex1
Hello world
 ~/D/P/lab_02_fistCPP>
```

```
#include <iostream>
int main(){
        std::cout << "Hello world" << std::endl;
}
```

- **#include <...>** - **preprocessing** directive
- **#include <iostream>** - include **header file** “iostream”, with I/O library **declaration**
- **int main()** - main function, automatically executed
- **{...}** - braces: define a block of code, scope
- **std::cout** - stdout: standard **output stream**
- **<<** - stream operator (inserts in the stream)
- **“Hello world”** - **literal constants** (sequence of characters)
- **;** - end of statement marker

```cpp
#include <iostream>

int main(){
    std::cout << "Hello world" << std::endl
}
```

```
~/D/P/lab_02_fistCPP [1]> g++ ex2.cpp
ex2.cpp: In function ‘int main()’:
ex2.cpp:5:1: error: expected ‘;’ before ‘}’ token
 }
 ^
```

» The compiler tells you where the error is, sometimes very accurately

» In this example error occurred in another location (another line)

```
#include <iostream>

int main()
    std::cout << "Hello world" << std::endl;
}
```

```
~/D/P/lab_02_fistCPP [1]> g++ ex2.cpp
ex2.cpp:4:2: error: expected initializer before ‘std’
  std::cout << "Hello world\n";
  ^
ex2.cpp:5:1: error: expected declaration before ‘}’ token
 }
 ^
```

- The compiler tries to compile the entire code
- One error can trigger a cascade of errors
  - read errors chronologically, correct first and compile again

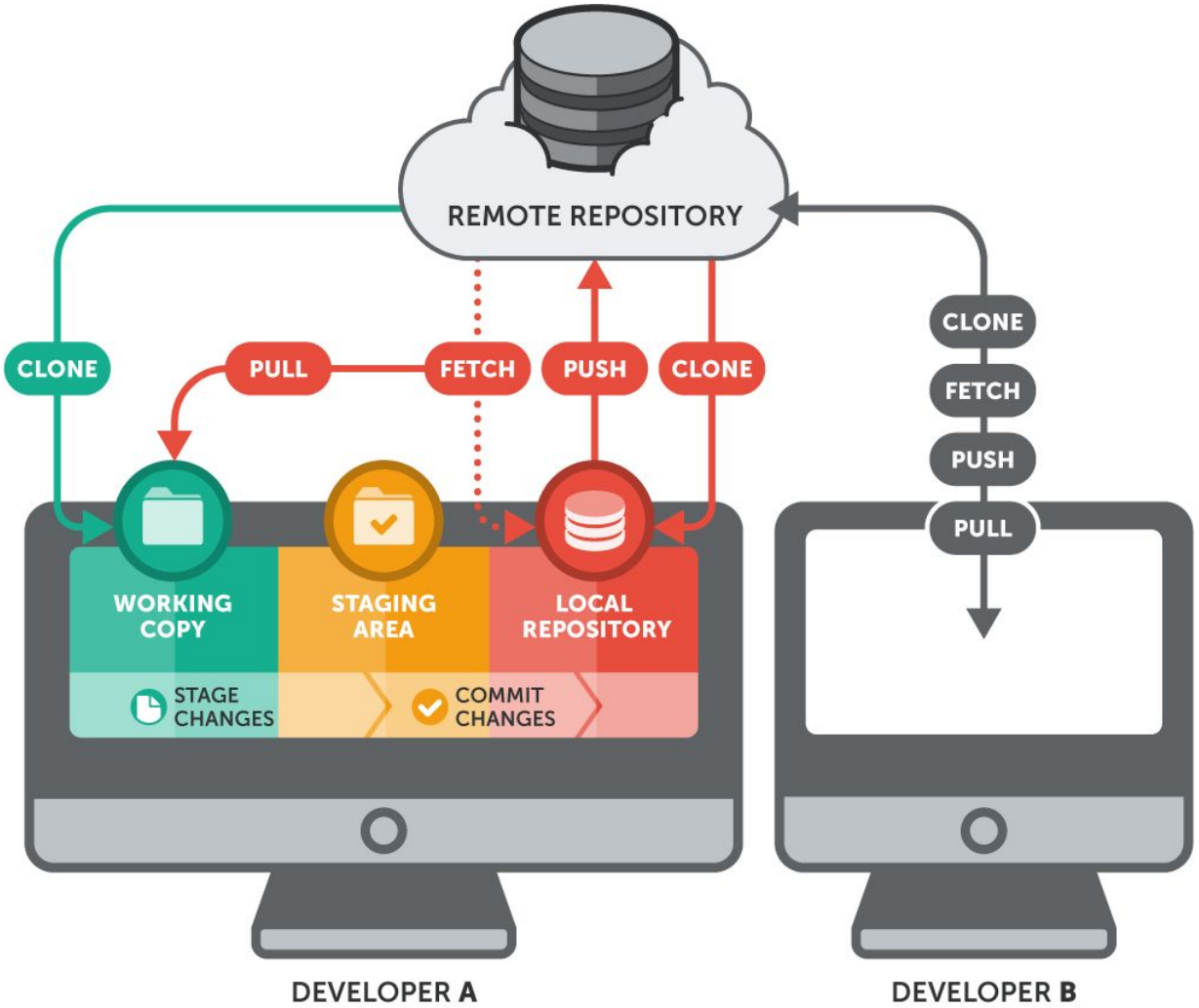
Where should I store *.cc? in Git

Version-control system

- Where to store *.cc? **On you SDD, in folder :-)**
  - how to send it to your colleague (developer)?
  - how to effectively share with many devs?
  - how to store many versions?
- VCS - Version Control System
  - **CSV** (Concurrent Versions System)
  - **SVN** (Subversion)
  - **GIT**

Version-control system

- server (remote) store all versions of files of all developers
- locally a server “mirror” (on disk)
- functions:
  - version-control
  - keep history (who create/change file)
  - conflict resolving (merge)
  - possibility to go back to any version



http://gdibtv.github.io/gdi-core-git-github/images/basic-remote-workflow.png

Literature

- **The Internet***
- any other code (decent!)
- https://stackoverflow.com C/C++, algorithms, systems, conf., …
- https://www.wikibooks.org C/C++ (good PL/EN)
- **Bjarne Stroustrup**, *Język C++*, (assume knowledge of C, PL/EN)
- Jerzy Grębosz, *Symfonia C++*, (from C to C++, popular)
- Stephen Prata, *Language* C++, (PL/EN)

- ~~Bruce Eckel, Thinking in C++~~

MOOC /muːk/

https://coursera.org
(Stanford, Princeton, …)

https://udacity.com
(Georgia IofT, Google, Facebook, Nvidia, …)

https://sololearn.com
(Android app.)

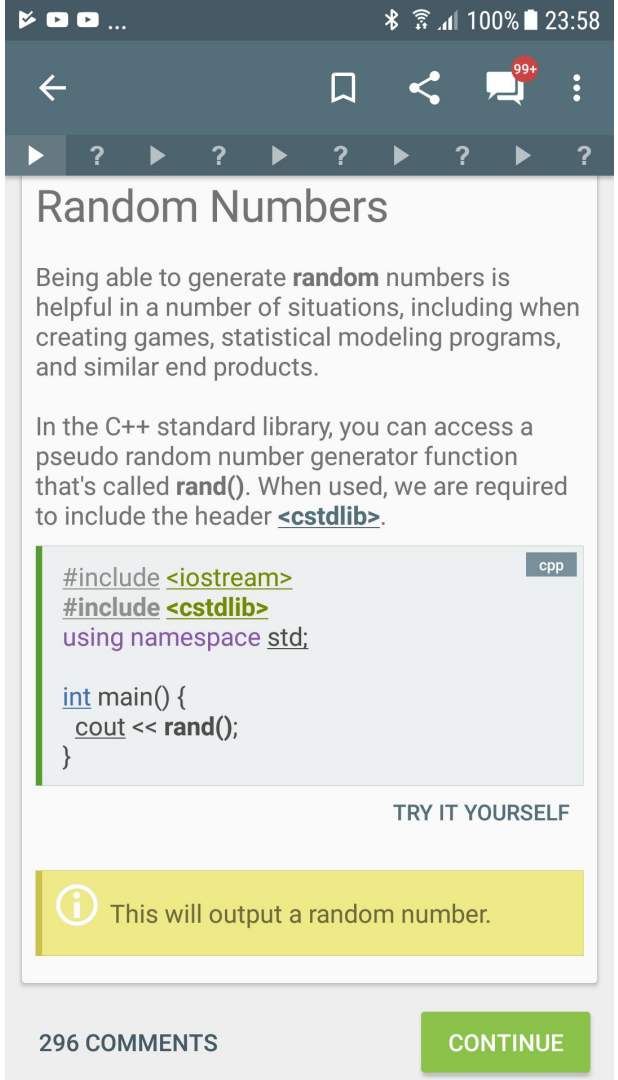
SoloLearn

“Learn to code for FREE!
Anytime and Anywhere, on Any Device“

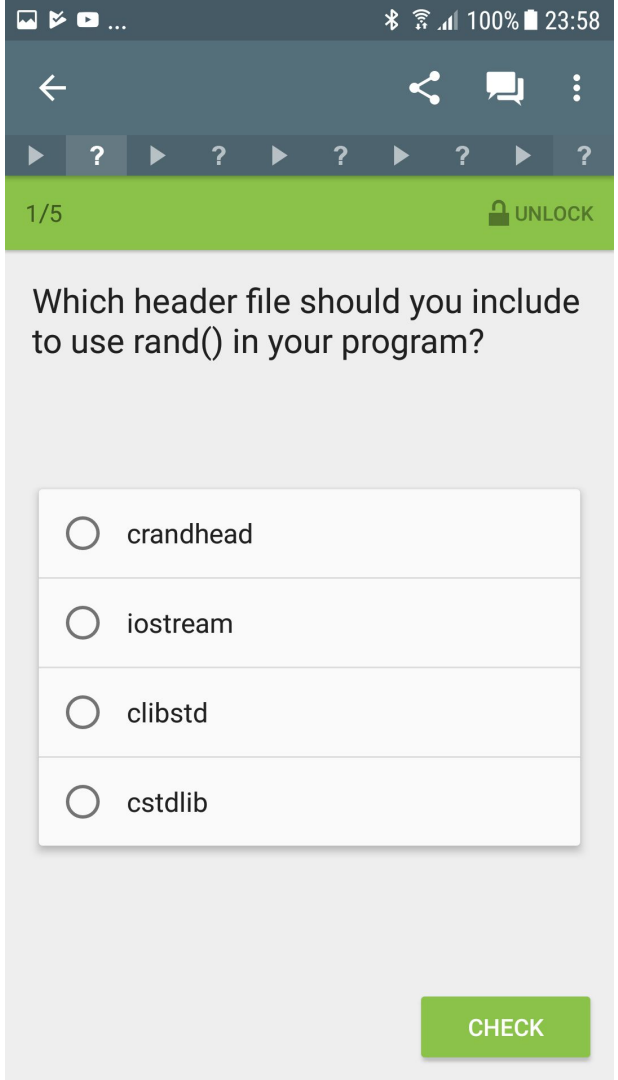
- Short lessons with verification-test (1-2 minutes)
- Basic knowledge about language and programming rules
- You will not learn how to program but you will learn the basics + many useful information, you will refresh your knowledge
- Gamification, rankings, diplomas, medals, challanges, etc ...
- Various languages (C++, Python, HTML, etc…)
- Boring for "advanced" - the entire C ++ course in a few hours
- C++: errors in "references" and poorly explained "exceptions"
- Overall: cool and recommend

SoloLearn

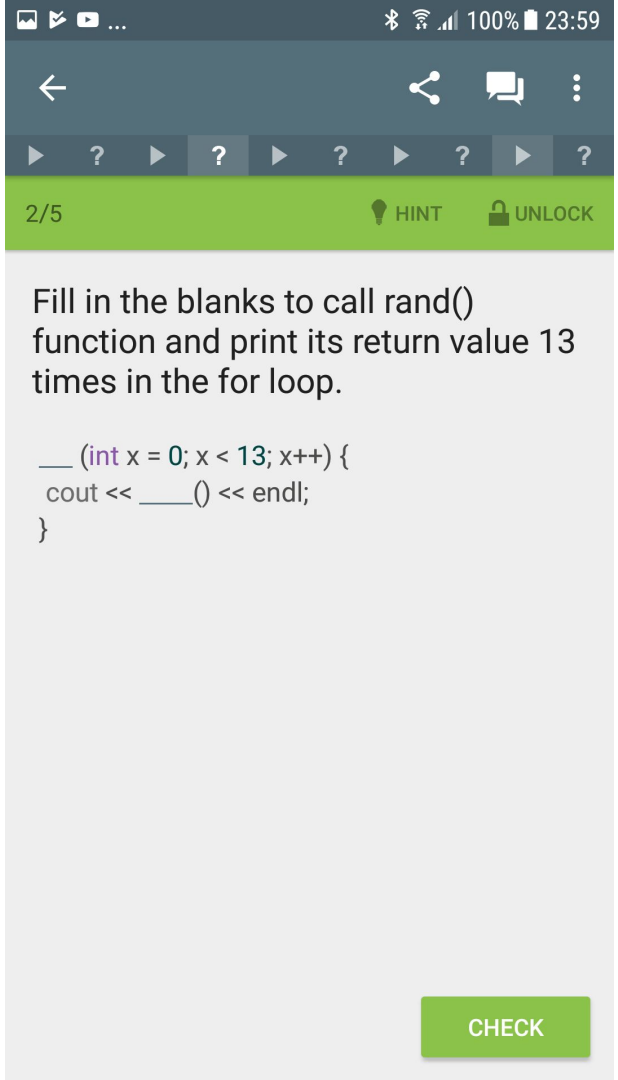
» One lesson - 30-60 seconds



SoloLearn



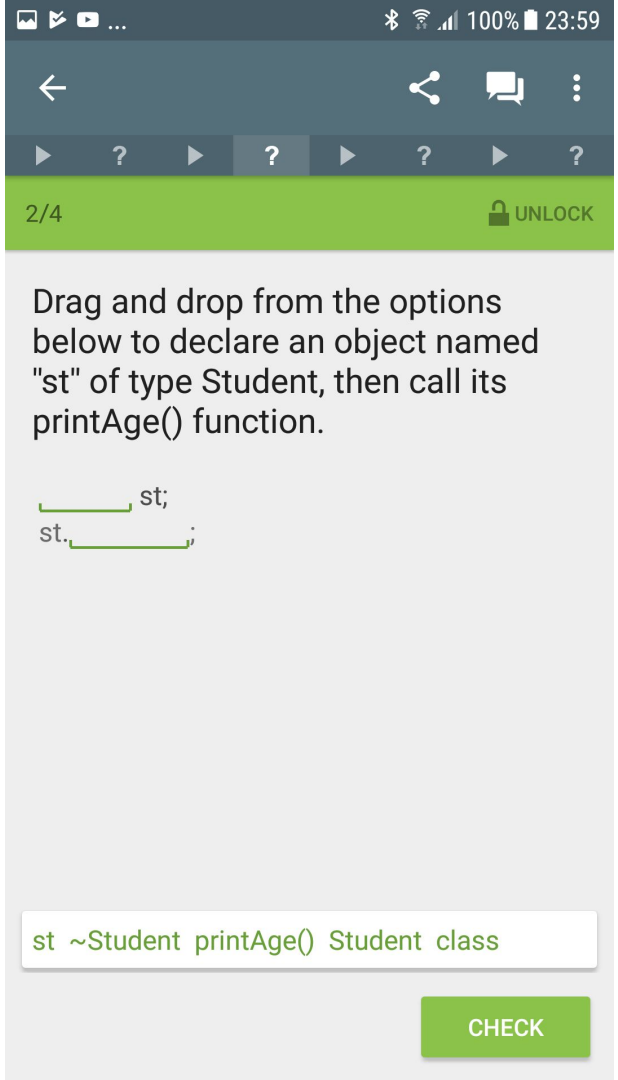
- One lesson - 30-60 seconds
- After lesson, question:
  - one of ….



SoloLearn

- One lesson - 30-60 seconds
- After lesson, question:
  - one of ….
  - complete code (it’s worth having virtual keyboard like: CodeBoard, Hacker’s)

SoloLearn



- One lesson - 30-60 seconds
- After lesson, question:
  - one of ….
  - complete code (it’s worth having virtual keyboard like: CodeBoard, Hacker’s)
  - drag&drop

SoloLearn

“Learn to code for FREE!
Anytime and Anywhere, on Any Device“

- Main advantage: **Anytime and Anywhere**
- Instead of tweeting/snapchatting while waiting at the bust-stop, you can solve 2-3 short programming tasks
-
- Shallow: it will not teach you how to program it !!!
- Useful: you will learn the basic syntax and techniques
- Challenge, Forum, **Andorid**/**iOS**/WWW

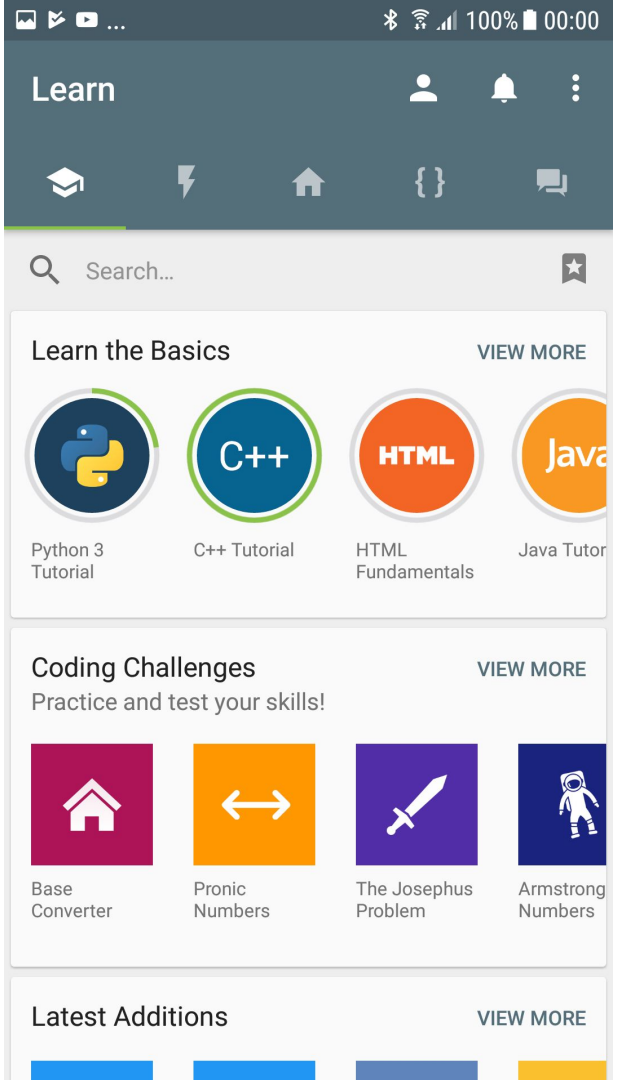
SoloLearn

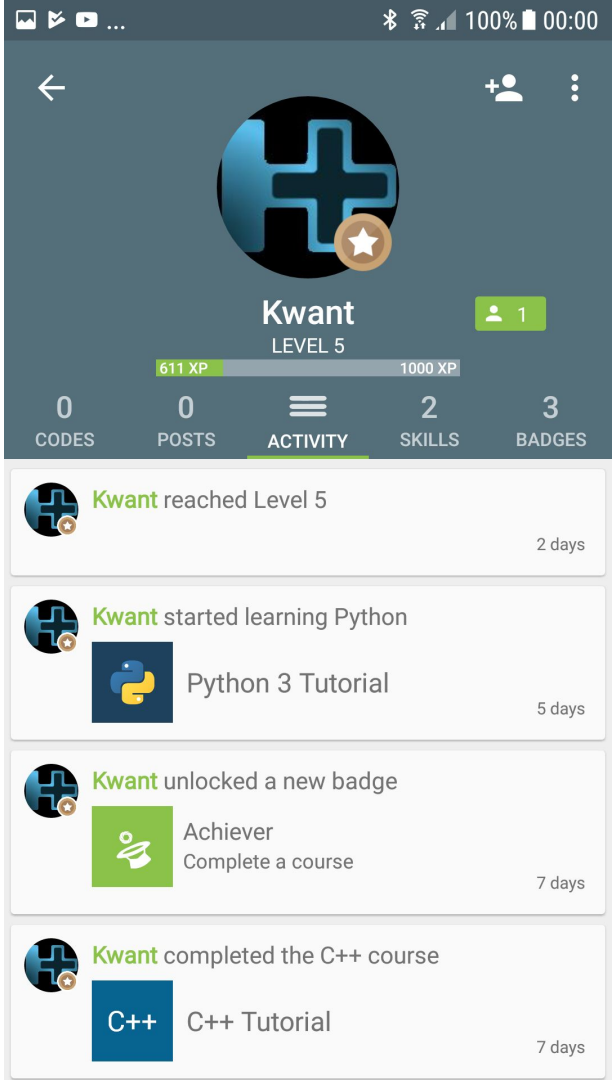
Instead of wasting your time on the Facebook, remember the syntax of the language or do two "challenge"

In the queue you can get two experience points ;-)

AGH

www.agh.edu.pl





homework :-)

- find a computer (could be a PC), check how much it has:
  - memory (RAM)
  - storage (HDD/SSD)
  - how CPU is fast in terms of FLOPS
  - how fast CPU can communicate with:
    - HDD
    - RAM
    - L1
    - L2
    - L3

Appendix

information vs entropy

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$H(X)$ - **entropy from set of samples** X
n - number of events
p_i - probability of event occurrence x_i

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$H(X)$ - **entropy from set of samples** X
n - number of events
p_i - probability of event occurrence x_i

x_i

0
7
0
2
0
2
0
7
4
2

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$H(X)$ - **entropy from set of samples** X
n - number of events
p_i - probability of event occurrence x_i

x_i

0
7
0
2
0
2
0
7
4
2

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$H(X)$ - **entropy from set of samples** X
n - number of events
p_i - probability of event occurrence x_i

x_i

0
7
0
2
0
2
0
7
4
2

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$H(X)$ - **entropy from set of samples** X
n - number of events
p_i - probability of event occurrence x_i

x_i

0
7
0
2
0
2
0
7
4
2

Basic concepts - **information**

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

p_i	x
0.4	0
0.3	7
0.2	0
0.1	2
	0
	2
	0
	7
	4
	2

Basic concepts - **information**

p_i	x
0.4	0
0.3	7
0.2	0
0.1	2
	0
	2
	0
	7
	4
	2

Quantities of information:

$$I_i = \log_r \frac{1}{p_i} = -\log_r p_i$$

I_i - **is the amount of information received when an event occurs** x_i
p_i - probability of event occurrence x_i
r - base of logarighm

Entropy (average amount of info.):

$$H(X) = -\sum_{i=1}^{n} p_i \log_r p_i$$

$$-(0.1\log_2(0.1) + 0.2\log_2(0.2) + 0.3\log_2(0.3) + 0.4\log_2(0.4)) = 1.846\ldots$$

= 2 bits

Thank you!