Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Департамент программной инженерии

Рабочая программа дисциплины
«Основы и методология программирования»
(преподается на английском языке)

для образовательной программы «Программа двух дипломов НИУ ВШЭ и Лондонского университета “Прикладной анализ данных”»
направления подготовки 01.03.02 «Прикладная математика и информатика»
уровень бакалавр

Разработчики программы:
Яворский Р. Э., к. ф.-м. н., mailto:ryavorsky@hse.ru
Шершаков С. А., mailto:sshershakov@hse.ru

Одобрена на заседании департамента _____ «____»_____ 2017 г.
Руководитель департамента _____ _____

Утверждена Академическим советом образовательной программы
«____»_____ 2018 г., № протокола _____

Академический руководитель образовательной программы
Вознесенская Т. В. _____

Москва 2018

1 Course Description

Course title: Introduction to Programming
Authors of the program: Rostislav Yavorskiy, *PhD* (responsible lecturer)
Sergey A. Shershakov, *MSc* (responsible lecturer)

1.1 General Information and Field of Application

The training course "Introduction to Programming" is offered to students of Bachelor Program "HSE and University of London Double Degree Programme in Data Science and Business Analytics" (area code *01.03.02*) at the Faculty of Computer Science of the National Research University — Higher School of Economics (HSE). The course is classified as an *compulsory subject* (*Б.Пр.Б* unit / base module, *Б.Пр* – Major disciplines of 2018–2019 academic year working curriculum); it is a two-module course (semester A quartile 1 and semester B quartile 3). The duration of the course amounts to *144* class periods (both lectures and practical trainings) divided into *72* contact hours and *72* practical training hours. In addition, *198* academic hours are set aside for self-studying of students. Total number of course hours is equal to *342* (144 + 198).

The course is divided into two logical parts, which do not basically depend on each other. *The first part* is given during semester A quartile 1 under responsibility of Dr. Rostislav Yavorskiy. The duration of the first part amounts to *56* lecture hours. *The second part* is given during semester B quartile 3 under responsibility of Lect. Sergey Shershakov. The duration of the first part amounts to *88* lecture hours.

The syllabus is prepared for teachers responsible for the course (closely related disciplines), teaching assistants, students enrolled in the course as well as experts and statutory bodies carrying out assigned or regular accreditations.

1.2 Pre-requisites

1.2.1 Part I

As a matter of fact, good English and Math are usually enough to enter the program, so no preliminary knowledge in data science or specific programming skills are required, because some of the students may have zero experience in these.

1.2.2 Part II

Successful completion of the first part of the course is the sole prerequisite for being enrolled for the second part.

1.3 Abstract

1.3.1 Part I

The first part of the course is intended to be taught during the first module (quartile) of the program, so it could be treated as introduction into the subject of "Applied data analysis" for students who just entered the program.

The course comprises 28academic hours of lectures and 28academic hours of seminars. The lectures and seminars are mostly independent. Lectures are focused on general topics of software engineering, data science, software development project management, etc. The seminars are targeted at teaching basics of Python programming language. Much attention is given to development of learning skills, that is why the grading formula encourages different kind of self-study, collaboration and team work.

1.3.2 Part II

The *second part* of the course is intended to be taught during the third module (quartile) of the program. It is dedicated to the base features of the C++ programming language and C++ Standard Library (STL). The part covers all necessary topics that are needed to start developing a modern CLI[1]-applications in C++14.

The course comprises *44* academic hours of lectures and *44* academic hours of practical classes. They are closely inter-related. The lectures are primarily intended to introduce new topics, whereas the practical classes are intended for solving specific problems by coding programs in C++.

2 Learning Objectives

During the course "Introduction to Programming" the participants will:

- study approaches and toolkits for fast development of modern Python applications;
- perform the following analytical tasks: prepare weekly blog posts in the online community; stream video to present solution of a coding task in Python; prepare online video presentation of a Python library;
- take part in a team project on development of online analytical service with Django or Flask;
- study approaches and toolchains for the development of modern C++-applications;
- practice application debugging and deployment with respect to various platforms and toolchains.

3 Learning Outcomes

Students who complete this course successfully will learn or acquire:

(Technical skills)

- basic concepts and methods of software development;
- skills in Python programming to formalize and solve simple development tasks;
- analyze a problem to be implemented in the form of an C++-application;
- design app architecture with respect to problem decomposition and known limitations;
- select the most appropriate toolset for app development;
- create a testbed environment for essential testing of the application.

(Soft skills)

- improve team-working skills with using collaborative working tools;
- improve presentation skills;
- improve skills on writing reports and technical documentation, including rapidly changing documentation with using wiki and other specific tools;
- improve self- and peer-review skills.

[1] CLI — Command-line interface.

4 Course Plan

4.1 Part I: semester A quartile 1

4.1.1 Lectures

#	Topic of the Lecture	Acad.hours
1	Overview of the Python Tutorial. Input/Output. Standard functions and data structures.	4
2	Object Oriented Programming. The key principles. HTML DOM (Document Object Model). Python libraries for DOM processing.	4
3	Client-server architecture. Specifics of front-end and back-end development.	4
4	Basics of software development project management. Work breakdown structure. Team member responsibility matrix. Persona description. Agile development.	4
5	Introduction to data science. Data exchange formats (CSV, JSON, XML). Architecture of a typical analytical service. Data cleaning. Exploratory analytics.	4
6	Team projects on development of prototype of online analytical service.	8
	Total:	**28**

4.1.2 Practical Training

#	Topic of the Seminar	Acad.hours
1	Introduction to Python: numbers, strings, lists; input/output; `if` statement, `for` statement; Python interpreter and development environments.	4
2	Exercises on basic data types and operators.	4
3	Exercises on using `list` data type.	4
4	Exercises on using `string` data type.	4
5	Exercises on using `dict` data type.	4
6	Exercises on using the Python standard library.	8
	Total:	**28**

96 academic hours are set aside for self-studying of students.

4.2 Part II: semester B quartile 3

#	Topic Name	Course Hours, Total	Audience Hours		Self Study
			Lectures	Practical Studies	
1	Introduction to C++ (Sec. 7.1.1)	9	2	2	5
2	Data types and Objects (Sec. 7.1.2)	13	3	3	7
3	Control flow statements (Sec. 7.1.3)	17	4	4	9
4	Expressions and Operators (Sec. 7.1.4)	9	2	2	5
5	Functions and procedures (Sec. 7.1.5)	13	3	3	7
6	C++ Memory model(Sec. 7.1.6)	17	4	4	9
7	Classes (Sec. 7.1.7)	17	4	4	9
8	Strings (Sec. 7.1.8)	17	4	4	9
9	Standard library (STL) (Secs. 7.1.9, 7.1.10)	26	6	6	14
10	Operator overloading (Sec. 7.1.11)	9	2	2	5
11	Templates (Sec. 7.1.12)	9	2	2	5
12	Exceptions and Move semantics (Secs. 7.1.13, 7.1.14)	13	3	3	7
13	Inheritance (Sec. 7.1.15)	13	3	3	7
14	Object-Oriented Analysis and Design (Sec. 7.1.16)	8	2	2	4
15	*Optional*: C++ plus Python (Sec. 7.1.17)				
	Total:	190	44	44	102

Notes:

1. Each sequential number above corresponds to a separate theme, whereas a theme can span over one or more lectures and/or practical classes.

5 Reading List

5.1 Part I

1. *The Python Tutorial*. URL: https://docs.python.org/3/tutorial/index.html.
2. Alexander Petkov. *How to explain object-oriented programming concepts to a 6-year-old*. URL: https://medium.freecodecamp.org/object-oriented-programming-concepts-21bb035f7260.
3. *Introducing JSON*. URL: https://www.json.org/.
4. *W3Schools XML tutorial*. URL: https://www.w3schools.com/xml/.
5. Cathy O'Neil and Rachel Schutt. *Doing data science: Straight talk from the frontline*. O'Reilly Media, Inc., 2013.
6. *12 Principles Behind the Agile Manifesto*. URL: https://www.agilealliance.org/agile101/.
7. Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK)*. 2004.
8. Rikke Dam and Teo Siang. *Personas – A Simple Introduction*. URL: https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them.

5.2 Part II

Books

1. Stephen Prata. *C++ Primer Plus*. 6th ed. Addison-Wesley Professional, 2011.
2. Stanley Lippman, Josée Lajoie, and Barbara Moo. *C++ Primer*. 5th ed. Addison-Wesley Professional, 2012.
3. Stanley Lippman. *Essential C++*. 1st ed. Addison-Wesley Professional, 1999.
4. Bruce Eckel. *Thinking in C++*. 1st ed. Prentice Hall Ptr, 1995.
5. Scott Meyers. *Effective Modern C++*. O'Reilly Media, 2014.
6. Nicolai Josuttis. *The C++ Standard Library — A Tutorial and Reference*. Addison Wesley Longman, 2012.
7. Jacek Galowicz. *C++17 STL Cookbook: Discover the latest enhancements to functional programming and lambda expressions*. Packt Publishing, 2017.
8. Bjarne Stroustrup. *Programming: Principles and Practice Using C++*. 1st ed. Addison-Wesley Professional, 2008.
9. Matthew Wilson. *Extended STL, Volume 1: Collections and Iterators*. Addison-Wesley Professional, 2007.
10. Andrei Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. 1st ed. Addison-Wesley Professional, 2001.

6 Grading System

The course grade is based on both ongoing assessment and final examination. Every module ends up with an final exam. The grade for the exam together with a cumulative grade represent a final grade for the module. The ultimate grade G for the whole course is calculated as:

$$G = 0.7 \cdot \min(P_1, P_2) + 0.3 \cdot \max(P_1, P_2), \tag{1}$$

where P_1 is a *first part* final grade, and P_2 is a *second part* final grade.

Grade G is rounded (up or down) to an integer number of points before entering them into records. P_1 and P_2 are also rounded in (1). The conversion of rounded 10-point scaled results to 5-point scaled ones is performed according to Table 1.

Table 1: Correspondence of ten-point to five-point marks

Ten-point scale [10]	**Five-point scale [5]**
1 — unsatisfactory 2 — very bad 3 — bad	Unsatisfactory — 2
4 — satisfactory 5 — quite satisfactory	Satisfactory — 3
6 — good 7 — very good	Good — 4
8 — nearly excellent 9 — excellent 10 — brilliant	Excellent — 5

6.1 Part I Grading Details

The final grade of the *first part* is computed by the following formula:

$$P_1 = 0.2 \cdot E_1 + 0.4 \cdot CS + 0.4 \cdot AS = 0.2 \cdot E_1 + 0.8 \cdot (0.5 \cdot CS + 0.5 \cdot AS), \tag{2}$$

where E_1 is a grade for the *first part exam*, which takes place at the end of the quartile 1 (semester A), CS is a score for *coding skills* and AS is a score for *analytical skills*.

Due to HSE rules the cumulative score P_1 must be a whole number.

The *first part final exam* will consist of several coding tasks and open-ended questions.

Coding skills are assessed during seminars of the first part (50 %) and the intermediate tests of the first part (another 50 %).

The maximal score for the analytical skills is 10, which could be accumulated from the following tasks:

- weekly blog posts in the online community (0.5 score points for each),
- streaming video to present solution of a coding task in Python (2 points),
- online video presentation of a Python library (3 points),
- team project on development of online analytical service with Django or Flask (5 points).

6.2 Part II Grading Details

The *final grade* P_2 for the *second part* is calculated as follows:

$$P_2 = 0.4 \cdot E_2 + 0.6 \cdot OA, \tag{3}$$

where E_2 is a grade of the *second part exam*, which takes place at the end of the quartile 3 (semester B), *OA* is an *ongoing assessment* grade (both 10-point scale). The ongoing assessment *OA* measures participant's performance throughout all classes and involves various types of activities (see Sect. 6.3).

The *final exam for the second part* as well as the intermediate tests are given in the form of a written test (paper- *or* computer-based, *subject to further clarification*). One (10-point scale) grade is given for the exam.

6.3 Ongoing Assessment

The ongoing assessment grade is accumulated throughout all the classes and is related to a participant's activity. An ongoing control structure is individual for every class.

During the classes, there are some activities available for students to be involved in. They include (but are not limited by) writing code and developing applications, evaluating practical problems, solving tests, answering questions and so on. Every activity is evaluated and grants some points (*RP*) to participants.

As a result of every class, a certain maximum number of *RP* can be earned (RP_{max}). *RP*s are accumulated during a module and give a resulting grade as maximum **10 points** (10-point scale) to *OA* only if every class brings a maximum number of points (RP_{max}) for a given participant.

Finally, some kinds of out-of-class activities can be accounted for as a part of ongoing assessment. *Peer review work*, *preparing and reporting one of a course-related topics* are examples of such activities.

Ongoing activities grades can be significantly decreased (down to 0 points) if a participant does not attend classes or attends them sporadically even though all other formal requirements for earning points are met.

6.3.1 Regular tests

Students' skills in programming are tested using automated testing. This way, a student is assigned an individual task, prepares it by using a personal computer and, then, submits it by using a special service, such as Yandex.Contest or a repository-based tool. The specific solution is subject to further clarification.

The individual home-based task submissions are to be further reassessed through in-class tests or examinations.

For any two corresponding submissions, one for home work and one for class work, graded as H and C respectively, the resulting grade R is calculated as follows:

$$R = 0.8 \cdot \min(H, C) + 0.2 \cdot \max(H, C). \tag{4}$$

7 Course Content

7.1 Part 2: Programming in C++

7.1.1 Introduction to C++

Introduction to C++. C++ program structure. Statements. Programs and modules. Toolchains and building C++ Programs.

7.1.2 Data types and Objects

Datatypes and Objects. Primitive, composite and user datatypes. Scope of variables (objects). Constants. Basic input/output. Introduction to streams.

7.1.3 Control flow statements

Control flow statements: conditional (`if`, `switch`); loops (`while`, `do..while`, `for`).

7.1.4 Expressions and operations

Expressions. Operations and operators. Operator precedence. Logical and Bitwise operations.

7.1.5 Functions and procedures

Procedural decomposition. Functions and procedures. Formal and Actual Parameters of a Function. Function return value. Function signature. Passing parameters to a function *by value* and *by reference*. Function overloading. Operator overloading.

7.1.6 C++ memory model

Memory model of a C++ program. Addresses and pointers. Difference between pointer and reference. Stack- and heap-memory. Object life-cycle.

7.1.7 Classes

Classes overview. Class members: methods and fields. Access modifiers: `public`, `protected`, `private`. Classes vs. structures. The (`.`) "dot" and (`->`) "arrow" operators. Constructors. Destructor. Safe array. Copy constructor. Copy operation. *Copy-and-swap* idiom. Rule of three.

7.1.8 Strings

Strings in C++. Null-terminated string. `std:string` class. Unicode supporting. String streams. Approaches to deal with strings.

7.1.9 Standard library overview

Standard library (STL) overview. Containers and adapters. Algorithms. Iterators. "`for`-each" loop for iterating collections/containers. `stl::vector<T>` and C-style arrays. Dynamic resizing of a `vector`.

7.1.10 Standard library components

Sequence containers: `vector`, `list`, `dequeue`*. Associative containers: `map`, `set`, `unordered_map`, `unordered_set`. Adapters: `stack`, `priority_queue`. Main `std` algorithms.

7.1.11 Operator overloading

More on operator overloading. Bitwise operations. `std::bitset<N>` class.

7.1.12 Templates

Templates: classes and functions. Header-only approach. `typename` and inner types. Templates and duck-typing. Concepts. `std` template framework. Lambda functions.

7.1.13 Exceptions

Exceptions. *RAII* idiom. Smart pointers.

7.1.14 Move semantics

Move semantics. *Rvalue* reference. Rule of five.

7.1.15 Inheritance

The Three Pillars of Object-Oriented Programming: *Encapsulation*, *Inheritance*, *Polymorphism*. Single inheritance. Virtual and pure-virtual methods. Abstract classes and interface classes.

7.1.16 Object-Oriented Analysis and Design

Introduction to Object-Oriented Analysis and Design. Class relationships.

7.1.17 C++ plus Python

Optional: binding c++ and python together.

8 Special Equipment and Software Support (if required)

Students are highly reccommended to use their own laptops with pre-installed and configured software, if possible. The computers in computer classes are also suitable for performing programming tasks. The exact set of software needed for the courses will be listed in an associated educational service, such as LMS or wiki.

References

[1] *12 Principles Behind the Agile Manifesto*. URL: https://www.agilealliance.org/agile101/.

[2] Andrei Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. 1st ed. Addison-Wesley Professional, 2001.

[3] Rikke Dam and Teo Siang. *Personas – A Simple Introduction*. URL: https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them.

[4] Bruce Eckel. *Thinking in C++*. 1st ed. Prentice Hall Ptr, 1995.

[5] Jacek Galowicz. *C++17 STL Cookbook: Discover the latest enhancements to functional programming and lambda expressions*. Packt Publishing, 2017.

[6] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK)*. 2004.

[7] *Introducing JSON*. URL: https://www.json.org/.

[8] Nicolai Josuttis. *The C++ Standard Library — A Tutorial and Reference*. Addison Wesley Longman, 2012.

[9] Stanley Lippman. *Essential C++*. 1st ed. Addison-Wesley Professional, 1999.

[10] Stanley Lippman, Josée Lajoie, and Barbara Moo. *C++ Primer*. 5th ed. Addison-Wesley Professional, 2012.

[11] Scott Meyers. *Effective Modern C++*. O'Reilly Media, 2014.

[12] Cathy O'Neil and Rachel Schutt. *Doing data science: Straight talk from the frontline*. O'Reilly Media, Inc., 2013.

[13] Alexander Petkov. *How to explain object-oriented programming concepts to a 6-year-old*. URL: https://medium.freecodecamp.org/object-oriented-programming-concepts-21bb035f7260.

[14] Stephen Prata. *C++ Primer Plus*. 6th ed. Addison-Wesley Professional, 2011.

[15] Bjarne Stroustrup. *Programming: Principles and Practice Using C++*. 1st ed. Addison-Wesley Professional, 2008.

[16] *The Python Tutorial*. URL: https://docs.python.org/3/tutorial/index.html.

[17] *W3Schools XML tutorial*. URL: https://www.w3schools.com/xml/.

[18] Matthew Wilson. *Extended STL, Volume 1: Collections and Iterators*. Addison-Wesley Professional, 2007.

Author of the program: _____ Rostislav Yavorskiy

Author of the program: _____ Sergey Shershakov

Ред. 0.1 / 31.08.2018 г.