ISIJ curriculum

Vladimir M. KIRYUKHIN, Marina S. TSVETKOVA

Headings. Abstract.
Part 1 shows the topics of Informatics course for General education. This is a General competence in Informatics for every Junior. “Informatics” as an academic subject imparts a culture of information science and algorithms on a student; the ability to format and structure information, knowledge and experience using a variety of methods of data representation in relation to a particular task (lists, graphs, arrays, tables, schemes, graphics, diagrams and hierarchical structures) and using relevant programs of data processing; the idea of a computer as a universal information-processing device; the idea of basic concepts which they study: information, algorithms, models, and their properties. It develops algorithmic thinking necessary for professional activity in modern society; explains how concepts and constructs of informatics are applied in the real world, about the role of information technology and automated devices in people’s lives, as well as in industry and scientific research; skills and abilities for safe and appropriate use of computers and internet networks, and the ability to observe the norms of communication, and operate ethically and within the law.
Part 2 shows the curriculum of Olympiad Informatics, shows input skills (start) and key planning skills (finish) at the Junior Olympiad in Informatics in process of teaching for 2-3 years.

Part 3 shows a template of the individual plan of the Junior for 2-3 years for his movement to the medal result at the IOI, taking into account the passage of the stage of participation in the Junior Olympics - EJOI and IAITI in Bulgaria , Shumen.

Part 1. Secondary School Informatics Curriculum

Introduction

General informatic skills for all juniors informatic

Every child 12-16 years (5-9 grades) studies the subject Informatics at school. A child motivated by Informatics studies Olympiad Informatics in addition to the school course. It is very important to help him in this no later than 13-15 years of age.
Curriculum Olympiad in Informatics for juniors, children in 5-9 grades, includes the main topics for the IOI syllabus and collections of Olympiad tasks in different complexity levels (tasks of the national Olympiads, Junior Olympiads tasks, tasks, IOI) . Junior should own this curriculum at the level of concepts and fundamental skills and techniques based on the experience of participating in the national Olympiad in Informatics. These competencies Informatics Olympiad, junior develops with the coach , but the important independent work with resources Olympiad of Informatics in the public domain (Massive open online courses, online contests, collections of tasks for independent decisions in an open system of competition). Each coach should be aware of these resources and help the student build his or her individual plan.
IOI syllabus and collections of Olympiad tasks in different complexity levels (tasks of the national Olympiads, Junior Olympiads tasks, tasks IOI) . The student who is keen on Olympiad Informatics should know this curriculum at the level of concepts and basic skills and techniques based on the experience of participation in the national Olympiad in Informatics. These competencies Olympiad Informatics child develops with a coach , but it is important to work independently with the resources of Olympiad Informatics in the public domain (Mass open online courses, online Olympiads, collections of problems for self-solution).Each coach should be aware of these resources and help the student build his or her individual plan.

The coach must show the Junior level of the complexity of his future achievements at the start and help the Junior to build your personal training plan to prepare for international competitions up to 15 years (and it allows you to make ISIJ), and become the Navigator of the movement for 2-3 years to the result based on the curriculum of Olympiad of Informatics with use of e-courses, daily mini training sessions on the tasks at IOI and discussions with critiques of the solutions and the analysis of algorithmic problems on the topics of the syllabus IOI

Topic "Informatics and Information Processes"

Section "**Introduction**"

- 5th–6th grade

  Information is one of the main general concepts in modern science.

  There are different elements to the word "information": information in the form of data, which can be processed by an automated system, and information in the form of knowledge intended for human interpretation.

  Types of information and data: texts, numbers, graphics and sound.

- 7th–9th grade

  The history of the development of informatics as a science, and the work of Russian scientists. The concept of cybernetics. The ability to describe continuous objects and processes using discrete mathematics.

  Information processes – processes connected with storage, conversion and transfer of data. Examples of information processes in technical and social areas, and in nature.

Section "**The computer as a universal data-processing device**"

- 5th–6th grade

  Variety of computers. Information carriers used in ICT. Historical and future developments.

  Hygiene, ergonomics and technical conditions of using ICT equipment.

- 7th–9th grade

  Computer architecture: processor, RAM, external NVRAM, input/output devices; their specifications.

Computers embedded in technical devices and production complexes. The robotic of production, additive technologies (3D-printers).

Software and hardware in a computer's operation.

Data volumes and access speeds characteristic of different types of carriers.

Information carriers in nature.

Safety techniques and rules of computer use.

Economic, legal and ethical aspects of ICT use.

History and trends in computer development, improving computer characteristics.

Supercomputers.

Physical limitations on computer characteristics.

Parallel computing.

Topic "The Mathematical Basis of Informatics"

Section “**Coding**”

- 7th–9th grade

  Symbols. The alphabet is a finite quantity of symbols. Text is a finite sequence of symbols in a given alphabet. The quantity of different texts of a certain length in a certain alphabet.

  Variety of languages and alphabets. Natural and formal languages.

  Coding symbols of one alphabet using code words in another alphabet; code tables, decoding.

Section “**Numeral systems**”

- 5th–6th grade

  Positional and non-positional numeral systems. Examples of representing numbers in positional numeral systems. Octal and hexadecimal numeral systems. Converting natural numbers from the decimal system to octal and hexadecimal systems and back again.

- 7th–9th grade

  The base of a numeral system. The alphabet (plurality of digits) of a numeral system. Quantity of digits used in a numeral system with a particular base. Short and long forms of written numbers in positional numeral systems.

  The binary numeral system, writing whole numbers within the range of 0 to 1024. Converting natural numbers from the decimal system to binary and binary to decimal.

  Converting natural numbers from the binary system to the octal and hexadecimal system and back again.

  Arithmetic in numeral systems.

Section “**The binary alphabet**”

- 7th–9th grade

  The representation of data in a computer as texts in a binary alphabet.

  Binary code with a fixed codeword length. Code width – length of codeword. Examples of binary code with widths of 8, 16, 32.

  Units of measurement for binary text length: bit, byte, kilobyte, etc. The quantity of information contained in a message.

  Kolmogorov’s approach to determining the quantity of information.

  The dependence of the quantity of combinations of code on code width. *ASCII code*. Coding the Cyrillic alphabet. Examples of coding letters in national alphabets. Standard Unicode. *Code tables with an alphabet other than binary.*

  Distortion of code during transfer. Error-correcting codes. The capability of unique decoding for codes of various codeword length.

Section “**Discretising**”

- 7th–9th grade

  Measurement and discretising. General digital representations of audiovisual and other continuous data.

  Coding colours. Colour models. RGB and CMYK models. *HSB and CMY models*. Depth of coding. Acquaintance with raster and vector graphics.

  Coding sound. Width and frequency of a recording. Quantity of recording channels.

Evaluating quantitative parameters connected with representation and storage of image and sound files.

Section "**Elements of combinatorics, set theory and mathematical logic**"

- 5th–6th grade

  Sets. Specific quantities of elements in sets obtained from two or three base sets using union, intersection and addition operations.

  Expressions. Simple and complex expressions. Euler-Venn diagrams. Logical values of expressions. Logical expressions. Logical operations: "and" (conjunction, logical multiplication), "or" (disjunction, logical addition), "is not" (logical negation).

- 7th–9th grade

  Calculating the quantity of variants: formulae for multiplication and addition of variant quantity. Quantity of texts of a given length in a given alphabet.

  Logical expressions. Rules of recording logical expressions. Logical operation priorities.

  Truth tables. Building truth tables for logical expressions.

  Logical sequential operations (material conditional) and logical equality operations. Properties of logical operations. Laws of algebraic logic. Use of truth tables to prove laws of algebraic logic.

  Acquaintance with the logic foundations of a computer. The cell base of a computer. The history of cell bases. Logic cell circuits and their physical (electronic) realisation.

Section "**Lists, graphs and trees**"

- 7th–9th grade

  Lists. First element, final element, previous element, next element. Insertion, deletion and substitution of elements.

  Graphs. Vertices, edges and paths. Directed and undirected graphs. The initial vertex (source) and final vertex (terminal) in a directed graph. Length (weight) of an edge and path. The concept of shortest path. Adjacency matrices of a graph (with edge lengths).

  Trees. Roots, leaves and vertices (nodes). Earlier vertex, later vertex. Subtrees. Peripherals. *Binary trees. Genealogical trees.*

Section "**Executors and algorithms. Executor control**"

- 5th–6th grade

  States, possible circumstances and system of executive commands; order commands and enquiry commands; executor refusal. The need formally to describe an executor. Manual executor control.

  Executor control algorithms. A computer is an automatic device able to control executors to carry out commands according to a previously-assembled program. Computer control of an executor.

  Computer control of a self-propelled robot. Compiling algorithms and programs to control an executor on a computer (robot, turtle etc.).

- 7th–9th grade

  Signals. Feedback. Examples: a computer and an executor under their control (including a robot); a computer receiving a signal from a digital sensor during observations and experiments and controlling actual (including moving) devices.

Examples of robotised systems (a system controlling movement in a transport system, a welding line in a car factory, an automated home heating system, an autonomous transport control system etc.).

Section “**Robotics**”

- 5th–6th grade

  Robotics is the science of developing and using automated technical systems. Autonomous robots and automated complexes. Microcontrollers. Signals. Feedback: receiving signals from digital sensors (touch, range, light, noise etc.).

  Autonomous moving robots. Executor devices and sensors. Robot command system. Robot design. Robot modelling: command executor and control device. Manual and computer control of robots.

  Example of a learning environment for developing control programs for moving robots. Control algorithms for moving robots. Realisation of “obstacle- avoidance”, “follow-the-line” and other algorithms.

  Analysis of robot-activity algorithms. Robot mechanism testing, debugging of robot-control program. Impact of measurement and calculation mistakes on the fulfilment of robot-control algorithms.

Topic “Algorithms and Elements of Programming”

Section “**Algorithms**”

- 5th–6th grade

  Verbal description of algorithms. Describing algorithms using flowcharts. Distinguishing verbal descriptions of algorithms from descriptions in formal algorithmic language.

  Algorithmic constructs.

  Sequence constructs. Linear algorithms. Limitations of linear algorithms: the inability to foresee the dependence of a sequence of actions to be fulfilled on source data.

  Selection constructs. Conditional operator: full and non-full forms.

  Fulfilment and non-fulfilments of a condition (true and false expressions). Simple and compound conditions. Compound condition recording.

  Repeat constructs: cycles with a set number of repetitions, with a condition for fulfilment, with a cycle variable. *Checking of fulfilment conditions of a cycle before and after the cycle body: post- and preconditions of the cycle. Cycle invariants.*

  Recording algorithmic constructs in a selected programming language.

- 7th–9th grade

  An algorithmic language (programming language) is a formal language for writing algorithms. A program is an algorithm written in a specific algorithmic language.

  Examples of written commands of selection, repeat and other constructs in different algorithmic languages.

  Programming systems. Means of creating and executing a program.

Section “**Development of algorithms and programs**”

- 7th–9th grade

Concepts of a program's development stages: compiling a program's requirements, selecting an algorithm and realising it in the form of a program in a chosen algorithmic language, debugging the program using a chosen programming system, and testing.

The simplest techniques of dialogue debugging programs (selecting a breakpoint, executing it step-by-step, revising the values of the variables, post-mortem debugging).

Acquaintance with program documentation. *Compiling descriptions of programs by sample.*

Examples of program development.

Operators.

Representations of data structures.

Constants and variables. Variable: name and value. Types of variables: integer, real, symbolic, string, and logic. Tabular values (arrays). Single-dimensional arrays. *Two-dimensional arrays.*

Subprograms.

Examples of data processing problems:

- ○ Identifying the smallest and largest number from two, three or four supplied numbers.
- ○ Identifying all roots of a quadratic equation.
- ○ Populating a number array in relation to a formula or numerical input path.
- ○ Identifying the sum of elements in a given finite number sequence or array.
- ○ Identifying the smallest (largest) element in an array.

Acquaintance with algorithms to solve these tasks. Realising these algorithms in a chosen programming environment.

Acquaintance with arrangements of more complex data-processing problems and the algorithms of their solutions: array sorting, element-wise operations with arrays; processing of integers represented in decimal and binary numeric systems, identifying the largest common denominator (Euclid's algorithm) etc.

Section "**Analysis of algorithms**"

- 7th–9th grade

Complexity of an algorithm: time to execute the program and number of operations performed, memory usage; reliance on the size of source data. Examples of short programs process a small volume of data over many steps; examples of short programs processing a large volume of data.

Determining possible algorithm results in a given set of input data; determining possible input data leading to this result. Examples of describing objects and processes using a selection of digital characteristics, as well as interdependencies of said characteristics expressed by means of a formula.

Section "**Mathematical modelling**"

- 5th–6th grade

Concept of mathematical models.

Problems solved using mathematical (computer) modelling. Distinguishing mathematical models from full-scale models and verbal (literal) descriptions of objects.

- 7th–9th grade

  Using computers while working with mathematical models. Computer experiments.

  Examples of mathematical (computer) model usage when solving technical problems. Representation of a modelling cycle: building a mathematical model, its realisation on a computer, simple example checking (testing), carrying out computer experiments, analysing the results, refining models.

Topic "Using Computer Programs and Services"

Section "**File system**"

- 5th–6th grade

  Principles of building file systems. Catalogues (directories). Fundamental operations for working with files: creating, editing, copying, moving, deleting. Types of files.

- 7th–9th grade

  Typical sizes of different types of files: a page of text, a whole book, a minute-long video, a one-and-a-half-hour film, a file of astronomical observations data, a temporary information file when mathematically modelling complex physical processes, etc.

  File manager.

  File system search.

  Archiving and dearchiving.

  Overview of programs. Data compression concepts. Work with archivers.

Section "**Preparing texts**"

- 5th–6th grade

  Text documents and their structural elements (page, paragraph, line, word, character).

  Methods of textual input using a scanner, textual recognition software, speech recognition software. Machine translation.

  Spellcheckers and dictionaries.

- 7th–9th grade

  Word processor – an instrument for creating, editing and formatting texts. The properties of a page, paragraph, character. Style formatting.

  Including lists, tables and graphics in a text document. Including diagrams, formulae, page numbers, headers and footers, links etc. in a text document. *Track changes.*

  The concept of a system of information, library and publishing standards. Business correspondence, academic publications, collaborative work. Abstract and footnotes.

Section "**Computer presentation**"

- 5th–6th grade

  Preparing computer presentations.

- 7th–9th grade

  Including audiovisual objects in the presentation. Using templates. Display controls of a presentation.

Section “**Graphic editing**”

- 5th–6th grade

  Acquaintance with graphic editing.

  Inserting images using different digital devices (digital cameras and microscopes, video-cameras, scanners etc.).

- 7th–9th grade

  Raster and vector graphics.

  Editing graphic objects: altering size, compressing an image, cropping, rotating, inverting, working with particular areas (highlight, copy and fill colour), altering colour, brightness and contrast. *Acquaintance with photo processing. Geometrical and stylistic transformations.*

  Means of computer projecting. Blueprints and working to them. Base operations: highlighting, joining, geometrical transformation of fragments and components. Diagrams, plans and maps.

Section “**Electronic (dynamic) tables**”

- 7th–9th grade

  Electronic (dynamic) tables. Formulae using absolute, relative and mixed addressing; transforming formulae when copying. Highlighting table range and ordering (sorting) its elements; building graphs and diagrams.

Section “**Databases. Managing requests**”

- 7th–9th grade

  Databases. A table as a representation of relationships. Building a data search request in a ready-prepared database. *Links between tables.*

Section “**Work in the information space. Information and communication technologies**”

- 5th–6th grade

  Computer networks. The internet. Internet addresses. Domain name system. Websites.

  Types of internet network activity. Internet services: e-mail, reference services (maps, planners etc.), search engines, software update services etc.

- 7th–9th grade

  Means and methods of information searches. Constructing enquiries; browsers. Computer encyclopaedias and dictionaries. Computer maps and other reference systems.

  Network data storage. *More data in nature and technology (genomic data, results of physical experiments, internet data, in particular data on social networking sites). Technologies of processing and storing it.*

  Techniques to improve internet safety. Computer viruses and other malware; defending against them.

  Problems of authenticating obtained information. Electronic signatures, certified sites and documents. Methods of individual and collective packaging of information on the internet. Interaction on computer networks: e-mail, chat, forums, teleconferences etc.

Social communication on the internet. Personal information, and means of protecting it. Organisation of personal information space on a network.

Basic trends in the future development of ICT.

Standards in the field of informatics and ICT. *Standardisation and standards in the field of informatics and ICT before the computer age (numerals, alphabets of national languages etc.) and in the computer age (programming language, addressing, internet networks etc.)*

Conclusion

Informatics as an academic subject imparts a culture of information science and algorithms on a student; the ability to format and structure information, knowledge and experience using a variety of methods of data representation in relation to a particular task (lists, graphs, arrays, tables, schemes, graphics, diagrams and hierarchical structures) and using relevant programs of data processing; the idea of a computer as a universal information-processing device; the idea of basic concepts which they study: information, algorithms, models, and their properties; it develops algorithmic thinking necessary for professional activity in modern society; explains how concepts and constructs of informatics are applied in the real world, about the role of information technology and automated devices in people's lives, as well as in industry and scientific research; skills and abilities for safe and appropriate use of computers and internet networks, and the ability to observe the norms of communication, and operate ethically and within the law.

The subject's place is determined by academic results and reflected in personal, interdisciplinary and subject-specific results.

Personal results of mastering the basic educational programme of secondary general education should reflect the ability to engage in communication in social and collaborative processes of educational, socially-beneficial, academic-research, creative and other activities.

Interdisciplinary results of mastering the basic educational programme of secondary general education should reflect the formation and development of competence when using information and communication technologies (hereon "ICT capabilities").

The twofold inclusion of informatics into the school curriculum is reflected in the two vectors of academic performance:

- The development of algorithmic and logical thought processes and of mathematical models, including:
    - The creation of a culture of information science and algorithms.
    - The basic concepts studied: information, algorithms, models and their properties.
    - The development of algorithmic thinking seen as necessary for professional activity in modern society.

  - The development of the ability to compile and write an algorithm for a specific operator.
  - An understanding of algorithmic constructs, logical values and operations.
  - A knowledge of one of the programming languages and fundamental algorithmic structures – linear, conditional and cyclic.
  - The ability to format and structure information, select a method of data representation in relation to a particular task – tables, schemes, graphics or diagrams using relevant programs of data processing.
  - The mastering of simple methods of presenting and analysing statistical data; the representation of statistical regularity in the real world and of various methods of their study, the depiction of the simplest probability models; the development of the ability to extract information represented in tables, diagrams and graphics, describing and analysing numerical data arrays using suitable statistical characteristics and using an understanding of the probability properties of environmental factors when making decisions.
- The representation of fundamental informational processes in real-life situations and on a computer:
  - The representation of a computer as a universal information-processing device.
  - The development of basic skills and abilities for using computer devices.
  - The development of abilities to apply the studied concepts, results and methods for solving practical tasks and tasks in related disciplines by using a computer.
  - The obtaining of skills and abilities for safe and appropriate use of computers and the internet, and the ability to observe the norms of communication, operate ethically and within the law.

The informatics course at secondary school is augmented with electronic training materials on the websites, and also electronic textbooks and manuals on course.

The course extends additional lessons according to the choice of school students, including on olympiad informatics (Kiryukhin and Tsvetkova, 2014). Content of preparation on olympiad informatics and the technique of such preparation has been presented earlier (Kiryukhin, 2007; Kiryukhin, 2010; Kiryukhin and Tsvetkova, 2011).

References

Ershov, A.P., Monahov, V.M., Beshenkov, S.A. (Eds.). (1985). *Fundamentals of the Computer Science: Trial Textbook for the Secondary Schools* (in two parts). Prosveschenie, Moscow. (In Russian). http://lib.mexmat.ru/books/65541

Kiryukhin, V. (2007). The modern contents of the Russian national Olympiads in informatics. Olympiads in Informatics, 1, 90–104.

Kiryukhin, V. (2010). Mutual influence of the National educational standard and Olympiad in Informatics contents. *Olympiads in Informatics*, 4, 15–29.

Kiryukhin, V., Tsvetkova, M. (2011). Preparing for the IOI trough developmental teaching. *Olympiads in Informatics*, 5, 44–57.

Kiryukhin. V., Tsvetkova, M. (2014). *Informatics. Programs of Extracurricular Activities of the School Students Studying on Preparation for the All-Russian Olympiad in Informatics: Grades 5-11. Methodological Manual for a Teacher*. BINOM, Laboratory of knowledge. (In Russian).

Part 2 The curriculum of Olympiad Informatics

Introduction

Start - skills for participants Olympiads and planned Finish- skills in Olympiad Informatics for juniors

Preparation of juniors and coaches in informatics olympiads on the one hand should be based on the state school educational standard on informatics and on the other hand should take into account the modern competition tasks contents of national (Kiryukhin, 2008; 2009) and international (Verhoeff *et al.*, 2006; Kiryukhin and Okulov, 2007) informatics olympiads. The analysis of the currently enforced state school educational standard on informatics in Russia (Kiryukhin, 2010) has suggested a complex curriculum of preparation of informatics teachers on based on their professional ICT competence.

Taking into account the approach described in the paper (Kiryukhin, 2007) defining the contents of the Olympiad on Informatics we will select the following key parts of the complex curriculum of preparation of juniors and coaches in informatics olympiads:

1. Mathematical Fundamentals of Informatics.
2. Developing and Analyzing of Algorithms.
3. Programming Fundamentals.
4. Computer Literacy.
5. Operating Systems.
6. Basis of Programming Technology.
7. Fundamental Methods of Calculations and Modeling.
8. Introduction to Network Technologies.

Taking this into account, the complex curriculum presented below for the preparation in informatics olympiads consists of eight sections, each of which reveal sub-topics. Each topic in turn contains, in more detail, the didactic units revealing key knowledge and skills that allow for each student to draw up an individual trajectory of preparation for the Olympiad in Informatics.

Each didactic unit has a certain level of complexity to which readers will be given guidance. In particular, three levels of complexity are selected, marked as follows:

• didactic unit without “*” – means that it concerns initial level of complexity (the **start level for juniors**), and the knowledge of these didactic units allows pupils to take part in school and municipal stages of the Russian Olympiad in Informatics;

• didactic unit with “*” –means a level of preparation sufficient for successful performance at regional stages of the National Olympiad in Informatics (the **planning level of complexity for juniors** / EJOI), provides practical and conceptual level of requirements to the participant of informatics olympiad, allows intelligently to come to solution of competition tasks and provides with it possibility technologically to present the own ideas;

• didactic unit with “**” – means the **finish planning level of complexity for IOI**; additional studying of these didactic units forms key abilities for students for solving competition tasks, opens before the olympiad participant the possibility to show the creative potential and to get solutions of enough complex competition tasks which are offered at a final stage of the Russian Olympiads in Informatics and the IOI, allows the filling of student portfolio achievements with diplomas of winners or prize-winners of the final stages of the Russia Olympiad in informatics and the IOI.

Taking into account the above, the mentioned complex curriculum of preparation on informatics olympiads will look like the following.

Topic 1. Mathematical fundamentals of informatics

1.1. Functions, relations and sets
1.1.1. *Functions, inverse functions, composition*
1.1.2. *Relations (reflexivity, symmetry, transitivity, equivalence, lexicographical order)*
1.1.3. *Sets (Venn diagrams, complements, Cartesian products)*
1.1.4. *Well-ordered sets* *
1.1.5. *Cardinability and countability* **
1.2. Basic geometry
1.2.1. *Point, line, segment, vector, angle*
1.2.2. *Cartesian coordinate system in Euclidean space*
1.2.3. *Euclidean distance*
1.2.4. *Cross-product and scalar product on the plane*
1.2.5. *Triangle, rectangle, polygon*
1.2.6. *Convex polygons*
1.2.7. *Trigonometric functions and formulae* *
1.2.8. *Voronoi diagram and Delaunay triangulation* **
1.3. Basic logic
1.3.1. *Logic variables, operations, expressions*
1.3.2. *Truth tables*

1.3.3. *Boolean functions*
1.3.4. *Universal and existential quantification*
1.3.5. *Ways to describe Boolean functions*
1.3.6. *Transformation of logical expressions*
1.3.7. *Normal forms (conjunctive and disjunctive)* *
1.3.8. *Minimizing Boolean functions* *
1.3.9. *Axiomatic of propositional logic* *
1.3.10. *Predicate logic* *
1.4. Basics of counting
1.4.1. *Counting arguments:*
• *Sums and product rule*
• *Arithmetic and geometric progressions*
• *Fibonacci numbers*
• *Inclusion-exclusion principle* *
1.4.2. *Recurrent relations*
1.4.3. *Matrices and matrix operations* *
1.4.4. *Fast multiplication of numbers or matrices* **
1.5. Proof techniques
1.5.1. *Direct proofs*
1.5.2. *Drawer principle*
1.5.3. *Proof by counterexample*
1.5.4. *Proof by contraposition*
1.5.5. *Proof by contradiction*
1.5.6. *Mathematical induction*
1.5.7. *The structure of formal proofs* *
1.6. Basics of theory of number
1.6.1. *Prime numbers, Fundamental Theorem of Arithmetic*
1.6.2. *Division with remainder*
1.6.3. *Greatest common divisor*
1.6.4. *Co-primes*
1.6.5. *Divisibility. Residue ring* *
1.6.6. *Chinese remainder theorem* *
1.6.7. *Primitive roots and discrete logarithms* **

1.7. Basics of algebra
1.7.1. *Polynomials and operations with them. Solving quadratic equations. Viet theorem*
1.7.2. *The general case of Viet theorem. Symmetric polynomials* *
1.7.3. *The concept of group* **
1.7.4. *Groups properties* **
1.7.5. *Normal subgroups* **
1.7.6. *Theorems about homomorphism and isomorphism* **

1.7.7. *Using group theory to solve combinatorial problems* **
1.8. Basics of combinatorial calculus
1.8.1. *Permutations, arrangements and combinations:*
• *Basic definitions*
• *Pascal's rule*
• *Binomial theorem*
1.8.2. *Grey codes: subsets, combinations, permutations* *
1.8.3. *Inverse tables* *
1.8.4. *Sets partitioning. Stirling numbers* *
1.8.5. *Parentheses sequences* *
1.8.6. *Connection between parentheses sequences and other combinatorial objects (binary and rooted trees, triangulations etc.)* **
1.8.7. *Estimating numbers of combinatorial objects. Stirling's formula. Corollaries.* **
1.9. Graph theory
1.9.1. *Kinds of graphs*
1.9.2. *Paths and connectivity*
1.9.3. *Operations with graphs*
1.9.4. *Trees*
1.9.5. *Spanning trees*
1.9.6. *Graph coloring*
1.9.7. *Eulerian and Hamiltonian graphs*
1.9.8. *Graph covering and independent sets* *
1.9.9. *Planar graphs and graph layout* *
1.9.10. *Biconnectivity. Bridges, articulation points* *
1.9.11. *Connection between DAGs and order relations. Transitive closure* *
1.9.12. *Bipartite graphs* *
1.9.13. *Flows and networks* *
1.9.14. *Planning network graph* *
1.9.15. *k-connectivity* **
1.10. Basics of probability theory
1.10.1. *Probability and expectation. Axiomatic of probability theory* *
1.10.2. *Total probability lemma and Bayes formula. Conditional probability and expectation* **
1.10.3. *Generating random objects for testing* **
1.10.4. *Approximate optimizing methods* **
1.11. Basics of games theory
1.11.1. *Game, the result of the game*
1.11.2. *Basic games and strategies*
1.11.3. *Sprague-Grundy function* *
1.11.4. *Matrix games* **

1.12. Linear programming
1.12.1. *Linear programming problems. Geometrical interpretation* *

1.12.2. *Basic methods of solving linear programming problems: simplex method, table interpretation* **
1.12.3. *Duality* **
1.12.4. *Examples of linear programming problems: maximum flow, assignment problem, shortest path* **
1.12.5. *Discrete linear programming* **
1.13. Basics of mathematical analysis
1.13.1. *Derivative and integral. Evaluating the area* *
1.13.2. *Green formula* **
1.14. Automatons and grammars
1.14.1. *Finite automaton* **
1.14.2. *Connection between finite automatons and grammars* **
1.14.3. *Using finite automatons* **
1.14.4. *Normal forms* **

Topic 2. Developing and analyzing algorithms

2.1. Algorithms and their properties
2.1.1. *The concept of algorithm*
2.1.2. *Properties of algorithms*
2.1.3. *Non-formal notation of algorithms*
2.2. Data structures
2.2.1. *Basic data structures*
2.2.2. *Sets*
2.2.3. *Sequences*
2.2.4. *Lists*
2.2.5. *Non-directed graphs*
2.2.6. *Directed graphs*
2.2.7. *Trees*
2.2.8. *Heap and RMQ/RSQ tree**
2.2.9. *Fenwick trees and their* n*-dimensional implementation* *
2.2.10. *Balanced trees* *
2.2.11. *Hash tables and associative arrays* *
2.2.12. *Trie* **
2.2.13. *Suffix tree* **
2.3. Basic algorithmic analysis
2.3.1. *Big O notation*
2.3.2. *Complexity sets*
2.3.3. *Asymptotic analysis of upper and average complexity bounds*
2.3.4. *Balance between memory and run-time complexity* *
2.3.5. *Using recurrent relations to analyze recurrent algorithms* *

2.3.6. *NP-complexity* **
2.3.7. *Computability* **
2.3.8. *Universal algorithms and self-applicability problem* **
2.3.9. *Matroid theory* **
2.4. Algorithmic strategies
2.4.1. *Brute force*
2.4.2. *Greedy algorithms*
2.4.3. *Divide et impere* *
2.4.4. *Backtracking* *
2.4.5. *Heuristics* *
2.4.6. *Branch-and-bound method* **
2.4.7. *Simulated annealing* **
2.4.8. *Four Russians speed-up* **

2.5. Recursion
2.5.1. *The concept of recursion*
2.5.2. *Recursive mathematical functions*
2.5.3. *Simple recursive functions*
2.5.4. *Implementing recursion*
2.5.5. *Divide et impere **
2.5.6. *Backtracking **
2.6. Basic computational algorithms
2.6.1. *Simple numeric algorithms*
2.6.2. *Classical combinatorial algorithms*
2.6.3. *Subset algorithms: generating all, generating next and previous, obtaining number and obtaining by number*
2.6.4. *Combinations and permutations algorithms: generating all, generating next and previous, obtaining number and obtaining by number*
2.6.5. *Linear and binary search*
2.6.6. *Quadratic sorting algorithms (selection, insertion)*
2.6.7. *Counting sorting*
2.6.8. *O(N log N) sorting (Quicksort, heap sort, merge sort) **
2.6.9. *Digital sort **
2.6.10. *Obtaining word number in lexicographically ordered set of its letters permutations **
2.6.11. *Arbitrary-size arithmetic **
2.7. *Numeric algorithms*
2.7.1. *Integer factorization*
2.7.2. *Eratosthenes sieve*
2.7.3. *Euclid's algorithm*
2.7.4. *Extended Euclid's algorithm. Implementing without division**
2.7.5. *Solving linear modular equations using Euclid's algorithm **
2.7.6. *Effective implementation of Eratosthenes sieve (O(n)) **

2.7.7. *Gaussian method and matrix inversions ***
2.7.8. *Fast power calculations. RSA ***
2.7.9. *Discrete logarithms ***
2.7.10. *Roots modulo n ***
2.7.11. *Effective primality test ***
2.7.12. *Fast factoring algorithms. Rho heuristics ***
2.7.13. *Berlekamp's algorithm ***
2.8. String processing
2.8.1. *Search for substring. Naïve method*
2.8.2. *Linear substring algorithms (Knuth–Morris–Pratt, Z-function) **
2.8.3. *Cyclic strings **
2.8.4. *Editorial distance and optimal alignment **
2.8.5. *Boyer–Moore algorithm ***
2.8.6. *Aho–Corasick algorithm ***
2.8.7. *Building suffix trees ***
2.8.8. *Digital suffix sorting ***
2.8.9. *Prime strings, string factorization ***
2.8.10. *Building suffix automatons ***
2.9. Graph algorithms
2.9.1. *Breadth- and depth-first search*
2.9.2. *Implementation methods of BFS (with queue or without)*
2.9.3. *Connectivity test*
2.9.4. *Shortest path problem in weighed graphs*

2.9.5. *Topological sort, strong connectivity and order diagram **
2.9.6. *Negative cycles – criterion, search algorithm **
2.9.7. *Time synchronization and linear inequality system **
2.9.8. *Eulerian cycle search (incl. lexicographically minimal) **
2.9.9. *Transitive closure **
2.9.10. *Weighted minimal spanning trees **
2.9.11. *Search for 2-connectivity components, bridges and articulation points using DFS **
2.9.12. *Bipartite matching and minimum vertex cover search **
2.9.13. *Searching for maximal flow ***
2.9.14. *Searching for maximal flow of minimum cost ***
2.9.15. *Assignment problem – Hungarian method. Connection between Hungarian method, minimal-cost maximal-flow and Dijkstra algorithm ***
2.9.16. *Fast algorithms for maximal-flow problem:* $O(N3)$ **
2.10. Dynamic programming
2.10.1. *The concept of dynamic programming. Recursive and linear implementation.*
2.10.2. *Monotone-direction problems*
2.10.3. *Backpack problem*

2.10.4. *Eliminating extra parameters **
2.10.5. *Building the way of solution using dynamic table **
2.10.6. *Common scheme of dynamic programming **
2.10.7. *Subset and profile dynamic programming **
2.10.8. *Broken-profile dynamic programming **
2.11. Game theory algorithms
2.11.1. *Dynamic programming and brute-force. DAG games **
2.11.2. *Retro-analysis. Effective implementation ***
2.11.3. *Fast Boolean evaluation. Using it in game analysis ***
2.11.4. *Position scoring. Alpha-beta pruning ***
2.12. Geometrical algorithms
2.12.1. *Test for coincidence of points, rays, lines and segments*
2.12.2. *Storing methods for points, lines and segments*
2.12.3. *Calculating distance between objects on the plane **
2.12.4. *Intersecting segments on the plane **
2.12.5. *Calculating polygon area using vertex coordinates. Pick's formula **
2.12.6. *Convex hull algorithms **
2.12.7. *Circles on the plane. Intersecting circles and other objects **
2.12.8. *Determining if the point is inside polygon **
2.12.9. *Scanning line method **
2.12.10. *Half-plane method ***
2.12.11. *Boundary detour method ***
2.12.12. *Effective algorithm of searching for two nearest points ***
2.12.13. *Effective algorithm for Voronoi diagram ***

Topic 3. Programming basics

3.1. Programming languages
3.1.1. *Types of languages*
3.1.2. *Procedure languages*
3.1.3. *Syntax and semantics of high-level languages*
3.1.4. *Formal syntax description methods: Backus–Naur form**
3.1.5. *Object oriented languages **
3.2. Programming constructions
3.2.1. *Variables, types, expressions and assignments*
3.2.2. *Input/output basics*

3.2.3. *Conditions and cycles*
3.2.4. *Functions and parameters*
3.2.5. *Structured decomposition **
3.3. Variables and data types
3.3.1. *Data types as set of values and operations*
3.3.2. *Declaration properties (linking, visibility area, blocks and lifetime)*
3.3.3. *Type matching*

3.4. Data structure types
3.4.1. *Primitives*
3.4.2. *Arrays*
3.4.3. *Record*
3.4.4. *Strategies of selection of appropriate data structure*
3.4.5. *Data storage in memory **
3.4.6. *Static, automatic and dynamic memory allocation **
3.4.7. *Pointers and references **
3.4.8. *Linked structures **
3.4.9. *Methods of implementing stacks, queues and hash tables **
3.4.10. *Methods of implementing graphs and trees **
3.5. Abstraction mechanisms
3.5.1. *Procedures, functions and iterators as abstraction mechanisms*
3.5.2. *Parameterization mechanisms (references and values)*
3.5.3. *Units in programming languages*
3.5.4. *Parameterized types ***
3.6. Fundamental programming species
3.6.1. *Task solving strategies*
3.6.2. *Role of algorithms in problem-solving process*
3.6.3. *Strategies of implementing algorithms*
3.6.4. *Implementing recursion*
3.6.5. *Testing strategies **

Topic 4. IT facilities

4.1. Digital logic
4.1.1. *Logical schemes*
4.1.2. *Scales of notation*
4.1.3. *Computer arithmetic*
4.2. Data representation
4.2.1. *Bits, bytes and words*
4.2.2. *Numeric representation **
4.2.3. *Fixed and floating point **
4.2.4. *Signed numbers representation **
4.2.5. *Non-numeric data representation, character codes, graphic data **
4.2.6. *Arrays and records **
4.3. Computer engineering principles
4.3.1. *Von Neumann principle*
4.3.2. *Control block: decoding and executing instructions*
4.3.3. *Instruction set and types (data manipulation, control, input/output)*
4.3.4. *Instruction formats **
4.3.5. *Addressing methods **
4.3.6. *Procedure calls and return **

4.3.7. *Input/output and interrupts **
4.4. Memory
4.4.1. *Operations with memory*

4.4.2. *Memory hierarchy*
4.4.3. *Data encoding, compression and integrity **
4.4.4. *Cache **
4.5. Communications
4.5.1. *I/O basics*
4.5.2. *External memory, external devices*
4.5.3. *Network technologies*
4.5.4. *DMA **

Topic 5. Operating systems

5.1. Operating systems basics
5.1.1. *Role and tasks of operating systems*
5.1.2. *Typical operating system functions*
5.1.3. *Directories: contents and structure*
5.1.4. *Naming, search, access, backup*
5.2. Basic functions of operating systems
5.2.1. *Abstractions, processes and resources*
5.2.2. *Device organization*
5.2.3. *Protection, access and authentication*
5.3. Memory control
5.3.1. *Physical memory and memory-controlling hardware*
5.3.2. *Segment and paged memory model. Flat model **
5.3.3. *Caching **

Topic 6. Basics of programming technologies

6.1. Programming facilities and space
6.1.1. *Programming space*
6.1.2. *Testing facilities **
6.2. Software quality assurance
6.2.1. *Basics of testing, testing plan **
6.2.2. *White box and black box methods **
6.2.3. *Integrated testing, element testing, system testing, quality assurance **
6.2.4. *Stress-testing **

Topic 7. Calculation methods and modeling

7.1. Basics of computational mathematics
7.1.1. *Basic methods of computational mathematics*
• *Finding roots and values of function **
• *Finding perimeter, area and volume **

7.1.2. *Grid and step methods **
7.1.3. *Floating point arithmetic ***
7.1.4. *Precision loss, stability, convergence ***
7.2. Modeling basics
7.2.1. *Concepts of model and modeling*
7.2.2. *Basic model types*
7.2.3. *Components of computer model and description methods: input and output variables, state variables, entry/exit functions, time shift*
7.2.4. *Stages and specials of building computer model*
7.2.5. *Basic stages of using computer models during solving problems*

Topic 8. Network technologies

8.1. Networks and communications
8.1.1. *Network cards and network devices*
8.1.2. *Data transfer media*
8.1.3. *Network architecture*
8.1.4. *Passwords and access control mechanisms*

8.1.5. *Service quality: performances, restoring after fault**
8.2. Wireless networks
8.2.1. *Specific problems of wireless and mobile computing*
8.2.2. *Installing programs on wireless and mobile computers*
8.2.3. *Wireless networks and links*
It is necessary to notice, that for the preparation for informatics olympiads when using the described curriculum it is necessary to consider,
that studying is presented in topics and didactic units should be going from simple to difficult. In particular, at first it is necessary to master the first level of complexity, and then to go further. Moreover, if coaches do not put before themselves the task of preparation of students for a successful performance at final stage of the National Olympiad in Informatics or the IOI it is enough for such teachers or coaches to master only the first level of complexity. It is important as such informatics teachers or coaches should have more knowledge even if they only teach to the first level, but not all of them can be in a condition to master higher level of complexity.
Only the most prepared teachers and coaches can be promoted to the third level of complexity. Here it is important to remember what evaluation of level of any informatics teacher-coaches preparation can be done by means of their pupils' achievements at informatics olympiads. If pupils get medals at the IOI then their informatics teacher-coaches will make gold fund of coaches of the country which will be capable to prepare students for victories on the national and international informatics olympiads.

From juniors skills to IOI medals skills

The conclusions formulated in chapter 2 have allowed to allocate basic topics of computer science which determine the modern OI contents. In particular, as such topics the following have been chosen:
1. Mathematical Basis of Informatics.
2. Developing and Analyzing of Algorithms.
3. Programming Fundamentals.
4. Computer Literacy.
5. Operating Systems.
6. Basis of Programming Technology.
7. Fundamental Methods of Calculations and Modeling.
8. Introduction to Network Technologies.
The topic "Mathematical Basis of Informatics" substantially is connected with discrete structures and is a fundamental basis of computer science. It is especially important for participation in Olympiads in informatics as it is difficult to achieve success on competitions without good preparation in the field of set theory, logic, graph theory, combinatory theory and so on.
It is also important for the students to continue education in higher school. Moreover, escalating complexity of computer science methods influences the solution of practical professional problems. To solve this problems in the future for today's students are extremely necessary to have stable knowledge and skills in the different fields of mathematics especially discrete structures.
At successful familiarization of topic "Mathematical Basis of Informatics" secondary school students (juniors) should:
know/understand:
• fundamental notions of functions, relations and sets;
• permutations, arrangement and combinations;
• formal methods of propositional logic;
• basis of construction of recurrent expression;
• the basic proof techniques;

• basis of numbers theory;
be able:
• to carry out the operations connected with sets, functions and relations;
• to calculate permutations, arrangement and combinations of set and also to interpret their values in a context of a specific task;
• to solve typical recurrent expression;
• to carry out formal logic proofs and a logic reasoning for modelling algorithms;
• to determine what kind of the proof approaches for the solving of a specific target is better;
• to use the basic algorithms of theory of numbers.
use the above-mentioned knowledge and skills at solving of practical tasks.
The basic themes of the topic “Mathematical Basis of Informatics” are:
1. Relations, Functions and Sets.
2. Basic Geometry.
3. Basic Logic.
4. Basics of Counting.
5. Proof techniques.
6. Basis of Theory of Numbers.
7. Basics of Algebra.
8. Basics of Combinatorial Calculus.
9. Basis of Graph Theory.
10. Basis of Probability Theory.
11. Basics of Game Theory.
The topic “Developing and the Analyzing of Algorithms” is very important for Olympiads in informatics and determines a basis of productive activity of students, their creative selfexpression.
In this field of computer science participants of Olympiad in informatics have an opportunity to show the best creative qualities at the solution of competition tasks.
Importance of algorithms theory is difficult to overestimate. Actual value of any program or program system depends on two factors: applied algorithms and efficiency of their implementation. Therefore development of good algorithm has crucial importance for productivity of any program system. Besides studying of algorithms allows to penetrate more deeply into a current task and can prompt methods of the solution which are not dependent on the programming language, a paradigm of programming, hardware maintenance and other aspects of implementation.
Studying of algorithms theory helps to develop at student’s ability to choose the algorithm most suitable for the solution of the given task or to prove, that such algorithm does not exist. This ability should be based on knowledge of algorithms which are intended for the solution of the certain set of known problems, their understanding strong and weaknesses, applicability of various algorithms with an estimation of its efficiency.
At successful familiarization of topic “Developing and the Analyzing of Algorithms” secondary school students should:
know/understand:
• elements of theory of algorithms;
• basic data structures;
• basic notions of graph theory and graph properties;
• relate graphs and trees to data structures, algorithms and counting;
• properties inherent in ”good” algorithms;
• big O notation for the description of the amount of work done by an algorithm;
• determining the time and space complexity of simple algorithms;
• computing complexity of the basic algorithms of sorting, search and hashing;
• concept of recursion and the general recursively presented problem statement;

• hash function and its assignment;
• simple numerical algorithms;
• basic combinatory algorithms;
• basic algorithms of computing geometry;
• the most widespread algorithms of sorting;
• the most important algorithms of string processing;
• representations of graphs (adjacency list, adjacency matrix);
• fundamental algorithms for graphs: depth- and breadth-first traversals, shortestpath algorithms, transitive closure, minimum spanning tree;
• basic algorithmic strategy: brute-force algorithms, backtracking, ”greedy” algorithms, ”divide-and-conquer” algorithms and heuristic algorithms;
• basis of dynamic programming;
• elements of game theory;
be able:
• to choose suitable data structures for the solution of problems;
• to use the above-mentioned algorithms during solving of problems;
• to determine memory and run-time complexity of algorithms;
• to determine computing complexity of the basic algorithms of sorting, search and hashing;
• to use big O notation for the description of the amount of work done by an algorithm;
• to implement recursive functions and procedures;
use the above-mentioned knowledge and skills at solving of practical tasks.
The basic themes of the topic “Developing and the Analyzing of Algorithms” are:
1. Algorithms and their properties.
2. Data Structures.
3. Basic Algorithmic Analysis.
4. Algorithmic Strategy.
5. Recursion.
6. Fundamental Computational Algorithms.
7. Numeric Algorithms.
8. String Processing.
9. Graph Algorithms.
10. Dynamic Programming.
11. Algorithms of Game Theory.
12. Geometrical Algorithms.
The topic “Programming Fundamentals” and a high technological level of its possession are necessary conditions of successful performance of any students on Olympiads in informatics. To participate in Olympiad in informatics, each secondary school student should know and put into practice even one programming language C ++.

It is important to notice, that knowledge and skills in the field of programming, which are important for practice of programming, irrespective of an applied paradigm of programming. Therefore the given topic includes sections under fundamental concepts of programming, the basic structures of data and algorithms and also actually programming languages. Programming languages are the basic means of dialogue of the student and a computer during solving of competition tasks. Students should not simply be able to write the program in any one language, they should understand the various styles of programming inherent in different languages. The understanding of a variety of programming languages and various paradigms considerably facilitates fast study of new languages by them.
As a result of successful study of topic “Programming Fundamentals” secondary school students should

know/understand:

• basic structures of programming;
• concept of data type as sets of values and operations above them;
• basic data types;
• basic data structures: arrays, records, strings, stack, queues and hash tables;
• data presentation in memory;
• bases of input/output;
• operators, functions and parameter transmission;
• static, automatic and dynamic memory allotment;
• memory management during execution of the program;
• methods of realization of stacks, queues and hash tables;
• methods of realization graphs and trees;
• mechanism of parameter passing;
• features of realization of recursive solutions;
• useful strategy at program debugging;

be able:

• to analyze and explain behaviour of the simple programs including fundamental structures;
• to modify and expand the short programs using standard conditional and iterative operators and functions;
• to develop, realize, test and debug the program which to use all the most important structures of programming;
• to apply methods of structural (functional) decomposition to divide of the program into parts;
• to realize the basic data structures in high level language;
• to realize, test and debug recursive functions and procedures;

use the above-mentioned knowledge and skills at solving of practical tasks and confidently to program even on one of programming language permitted to use in RusOI (C++).

The basic themes of the topic “Programming Fundamentals” are:

1. Programming Languages.
2. Basic Programming Constructions.
3. Variables and Data Types.
4. Data Structure Types.
5. Mechanisms of Abstraction.
6. Fundamental Programming Species.

The topic “Computer Literacy” plays an important role in studying of computer science because a computer is the integral tool which students use during competition in informatics. Secondary school students should not perceive a computer as a black box executing the programs by means of unknown magic. All students during solving competition problems should know the main components of which the computer consists and understand how they operate, their main characteristics, productivity and interaction between them. The understanding of the computer and its organization allows also to write more effective programs.

At successful familiarization of topic “Computer Literacy” secondary school students should:

know/understand:

• logic variables, operations, expressions;
• scale of notations;
• formats of representation of numerical data;
• presentation of data with fixed and floating point and connection with accuracy;
• internal representation of non-numerical data;

• internal representation of symbols, strings, records and arrays;
• instruction representation at a machine level;
• basis of input/output;
• basic types of memory;
• bases of memory control
• access to data on hard disk;
be able:
• to convert numbers from one scale of notation in another;
• to use mathematical expressions for the description of functions of simple consecutive and combinational schemes;
• to transform numerical data from one format to another;
• to adjust the programmer's workbench for solution of competition tasks;
use the above-mentioned knowledge and skills at solving of practical tasks so that the students feels themselves confidently at work with a computer at solving of competition tasks.
The basic themes of the topic "Computer literacy" are:
1. Digital Logic.
2. Data Representation.
3. Computer Engineering Principles.
4. Memory.
5. Communications.
The topic "Operating systems" gives to students convenient abstraction of hardware maintenance of a computer. For many years operating systems and their abstraction became more and more difficult in comparison with usual applied programs. Nevertheless, for successful performance on Olympiad in informatics secondary school students should know:
• functions of modern operating systems;
• difference of primitive batch systems from complex multi-user operating systems;
• understanding of a logic level;
• page and segment organization;
• various ways of economy of memory;
• distinctions between the mechanisms used for communications with devices of a computer;
• advantages and shortcomings of direct memory access;
• requirements to recovery from failures.
Acquired knowledge of this topic should provide to students for the confident work with operating system at implementation of solution of competition tasks by means of a computer.
The basic themes of this topic are.
1. Operating Systems Basics.
2. Basic Functions of Operating Systems.
The topic "Basis of Programming Technology" is a part of software engineering, considering the application of corresponding knowledge and skills to effective implementation and operation of programs and program systems. Software engineering studies all phases of life cycle of program system: analysis of requirements, development of specifications, designing, implementation, testing, operation and support. Though on Olympiads in informatics there is no speech about development of program system, nevertheless, this topic plays a big role in solving of competition tasks and demands from students of quite certain knowledge and skills.
At successful familiarization of topic "Bases of Programming Technology" secondary school students should:
know/understand:

- purpose and structure of programming facilities and environments;
- role of program tools during development of the software;
- properties of designing of the ”good” software;
- basic methods of program testing and debugging;

be able:

- to choose and prove a set of programming facilities for support of development of the software;
- to use programming facilities and environments in development of software product;
- to compose tests;
- to test and debug of programs;
- to develop the program in the form of ready software product;

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of the topic “Bases of programming technology” are:

1. Programming facilities and environment.
2. Program Testing and Debugging.

The topic “Fundamental Methods of Calculations and Modeling” represents the area of computer science closely connected with calculus mathematics and numerical methods. As computers began capable to solve more and more challenges, similarly to computer science as a whole this area got the increasing value and importance. Moreover, by the end of the twentieth century scientific calculations have affirmed as the independent discipline having close connections with computer science, but, nevertheless, not identical with it.

In pure form methods of calculus mathematics and numerical methods are practically not used on the RusOI, exception is made only with methods of modeling. Nevertheless, the certain topics in this area play the important role in training of student for Olympiads in informatics. Their knowledge allows to approach more substantially to solving of competition tasks by students.

At successful familiarization of topic ““Fundamental Methods of Calculations and Modeling”” secondary school students should:

know/understand:

- notion of precision loss, computational stability, machine accuracy and error of calculus of approximations;
- sources of errors in calculus of approximations;
- basic algorithms of solving calculus mathematics tasks (calculation of value and roots of function, calculation of perimeter, square and volume, calculation of a point of crossing of two segments etc.);
- notions of model, modeling and simulation, basic types of models;
- specifications of computer model (input, output and state variables, state change functions, output function, time advance function) and ways of their description;
- basic phases and features of implementation and use of computer models;

be able:

- to calculate error estimation of calculus of approximations;
- to use basic methods of calculus mathematics at solving tasks;
- to formalize objects of modeling;
- to develop computer models of the elementary objects;

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of this topic are:

1. Basics of Computational Mathematics.
2. Modeling Basics.

The topic “Introduction to Network Technologies” is came into the OI contents in connection with last achievements in the field of networks and the telecommunications. On Olympiads in informatics it is necessary for students not only the nobility that such computer networks, but also directly to carry out competition tasks in local computer network environment and the corresponding software. Possession of network technologies includes both theoretical knowledge, and practical skills. Students have to get basis of this knowledge and skills to feel during competition more confidently.

As a result of successful study of topic "Introduction to Network Technologies" secondary school students should
know/understand:
• basic structure of network architecture;
• most important network standards;
• roles and the responsibility of clients and servers for various applications;
• problems of networks safety arising because of viruses and the attacks directed on initiation of refusals in service;
• basis of wireless computer networks;
be able:
• to customize programmer's workbench in client- server network;
• effectively to use a number of the widespread network applications, including webbrowsers and automated evaluating systems;
• to work with applications using mobile and wireless communications;
use the above-mentioned knowledge and skills for solving competition tasks in the environment which is installed for conducting IOI.
The basic themes of the topic "Introduction to Network Technologies" are:
1. Basis of Networks and Telecommunications.
2. Introduction to Wireless Networks.

Part 3 ISIJ: personal plan of Junior training in EJOI-IOI

Introduction

School Activities

The School is held as an international educational forum for national teams of countries participating in the international Olympiad in Informatics (IOI, EJOI, ets).
The team from a country consists of 2 to 6 participants and one team-coach. Age of participants is supposed 13, but to be not older than 16 years by the end of the year. The student can choose the level of training: advanced (A) or basic (B).

The ISIJ provides

- Classes focused on main Olympiads: EJOI (basic level- B) and IOI (advanced level-A), seminars. The content of classes of the school of Olympiad Informatics is based on the thematic sections of the International Olympiad in Informatics – "6.2 Algorithms and Complexity (AL)» https://ioinformatics.org/files/ioi-syllabus-2018.pdf
- Distance session of the ISIJ is in the autumn-winter months for students ISIJ http://isi-junior.com/klass/trainings.php. YandexContest are the main partners of the ISIJ.
- IT clubs on the use of IT in various fields of science and technology IT club is to form a space of formation of profile interests at the junction of science and information technology.
- All teams of ISIJ countries take part in the Cup in two categories: level A (medal advanced level) and level B (medal Basic level) .
- The School program provides Robot-club. The Junior teams of ISIJ countries (no more than 4 participants in the team) can take part in the Robot-competition.

Focus group of training - juniors, keen on Olympiad Informatics, participants of the national Olympiad in Informatics and coaches of the junior team from country.

Start - skills ISIJ

- knowledge of the subject of Olympiad tasks at the level of the National Olympiad,
- primary experience in solving simple problems of Olympiad Informatics (national Olympiad level, or levels of simple problems in theEJOI/ IOI collection. Select a simple task from the collection can be based on the statistics of its solutions to the full score, in the Statistics section on the website ioinformatics.org),
- introduction to the themes of Algorithms at the level without * (Part 2 see)
- availability of primary experience of participation in online Olympiads in Informatics
-ability to use the system of competitions, to send the decision for check, to test the decision
- primary C++programming skills
- ability to speed input on the keyboard

Planned personal Junior skills on ISIJ

The individual plan of the Olympiad training (for the period of 2-3 years) is focused on the independent work of the Junior with the resources of the Olympiad Informatics taking into account the specific tasks of IOI and syllabus IOI, and the following planned skills:

- knowledge of the subject matter of the ISIJ – curriculum (ioi Syllabus) at the level with * (see Part 2). The solution to a collection of EJOI/IOI-tasks for self-training
- ability to analyze the problem and determine the theme of the ISIJ – curriculum (Syllabus IOI) for the problems
- experience in solving problems of increased difficulty theme of the ISIJ – curriculum at the level with **(Part 2 see).,
- knowledge of the criteria of self-assessment of achievements based on solving problems of increased difficulty, the ability to conduct self-analysis on the results of training, to set tasks to eliminate the identified deficiencies
- the use of MOOC for development of the Olympiad in Informatics, Algorithms, data Structures, Programming,
- readiness to solve the tasks of summer ISIJ for daily training the practice on a full score
- ability to identify difficulties in solving the problem in the first approach to it, the ability to find and study the theoretical material and analysis of solutions on the basis of the first approach to eliminate deficiencies, the ability to bring the solution of the problem after the second/subsequent approaches to a full score
- the ability to build the personal training plan for the 2-3 years
- a daily 2 - hour lesson on the tasks, in a remote environment (for example, on the tasks ISIJ/EJOI https://contest.yandex.ru/ISIJ/, of IOI collection in the Yandex Context https://contest.yandex.ru/ioi/) , the ability to implement your plan without missing
- the ability to work out high-speed problem solving skills (for example, from the IOI collection) repeated solution of one problem after its solution and analysis for a full score not more than 1 hour.
- regular participation in a MOOC to improve the skill of programming in C++ (for example Cursere http://isi-junior.com/klass/trainings.php, Cisco Networking Academy http://isi-junior.com/klass/siscoacademy.php)
- regular participant in online tours IOI, online Olympiads: http://www.acsl.org/divisions.htm, http://train.usaco.org/usacogate, http://www.hsin.hr/coci/, http://www.hsin.hr/coci/
- elimination of difficulties in the English-speaking environment of the competition system and Olympiad problems
- the ability to analyze the text of the problem, the ability to analyze in the group solutions for subtasks and complete solutions
- fluency in systems of competitions in Informatics,
- the first experience of creativity to develop a task for the tour in a group with a coach, its full preparation for inclusion in the tour
- communication skills with colleagues

- understanding the value of active recreation and sports for the programmer, compliance with the time for recreation and creativity in the application environment
- knowledge in professions in the IT sector.