Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра иностранных языков

ИНОСТРАННЫЙ ЯЗЫК

Информационные технологии в управлении

Методические указания для самостоятельной работы студентов
по специальности 27.03.04

FOREIGN LANGUAGE

Information Technology in Management

САНКТ-ПЕТЕРБУРГ
2022

УДК 811.111 (073)

ИНОСТРАННЫЙ ЯЗЫК. Информационные технологии в управлении: Методические указания для самостоятельной работы студентов / Санкт-Петербургский горный университет. *Сост. Е.А. Кольцова. СПб, 2022. 32 с.*

Данные методические указания предназначены для самостоятельной работы студентов по дисциплине «Иностранный язык». Методические указания включают аутентичные тексты в соответствии с направлением подготовки. Тексты сопровождаются заданиями, целью которых является развитие умений чтения, перевода и поиска информации.

Методические указания предназначены для студентов, обучающихся по специальности 27.03.04 Управление в технических системах направления подготовки «Информационные технологии в управлении» и согласованы с программой по иностранному языку для студентов неязыковых вузов.

Научный редактор: доцент кафедры иностранных языков Горного университета, кандидат педагогических наук С.А.Бойко

Рецензент: доцент кафедры английской филологии и лингвокультурологии Санкт-Петербургского государственного университета, кандидат филологических наук, доцент О.В.Емельянова

ВВЕДЕНИЕ

Данные методические указания для самостоятельной работы по английскому языку предназначены для студентов магистров специальности 27.03.04 Управление в технических системах направления подготовки «Информационные технологии в управлении». Методические указания составлены в соответствии с учебной программой по дисциплине «Иностранный язык» для формирования иноязычной профессиональной компетенции будущих специалистов.

Предложенные методические материалы предназначены для самостоятельной работы студентов и состоят из пяти разделов, содержащих информацию о современных информационных технологиях, истории развития науки, языках программирования, типах систем и применении информационных технологий в различных сферах промышленности и управления. Каждый текст сопровождается комплексом послетекстовых заданий и упражнений, направленных на контроль понимания прочитанного материала, формирование умения ориентироваться в оригинальных текстах, отработку и закрепление изучаемого лексического материала, контроль навыков перевода.

Изучение предложенного материала имеет целью развитие и совершенствование навыков чтения и перевода оригинальной научной литературы по информационным технологиям и информатике, расширение словарного запаса, преодоление трудностей перевода и приобретение навыков устной и письменной коммуникации в сфере профессиональной деятельности.

UNIT 1 COMPUTER SCIENCE

TEXT 1.1 Introduction to the Computer Science

1 Read the introductory text about computer science and translate it.

Computer Science

Computer science, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information. The discipline of computer science includes the study of algorithms and data structures, computer and network design, modeling data and information processes, and artificial intelligence. Computer science draws some of its foundations from mathematics and engineering and therefore incorporates techniques from areas such as queueing theory, probability and statistics, and electronic circuit design. Computer science also makes heavy use of hypothesis testing and experimentation during the conceptualization, design, measurement, and refinement of new algorithms, information structures, and computer architectures.

Computer science is considered as part of a family of five separate yet interrelated disciplines: computer engineering, computer science, information systems, information technology, and software engineering. This family has come to be known collectively as the discipline of computing. These five disciplines are interrelated in the sense that computing is their object of study, but they are separate since each has its own research perspective and curricular focus. (Since 1991 the Association for Computing Machinery [ACM], the IEEE Computer Society [IEEE-CS], and the Association for Information Systems [AIS] have collaborated to develop and update the taxonomy of these five interrelated disciplines and the guidelines that educational institutions worldwide use for their undergraduate, graduate, and research programs.)

The major subfields of computer science include the traditional study of computer architecture, programming languages, and software development. However, they also include computational science (the use of algorithmic techniques for modeling scientific data), graphics and

visualization, human-computer interaction, databases and information systems, networks, and the social and professional issues that are unique to the practice of computer science. As may be evident, some of these subfields overlap in their activities with other modern fields, such as bioinformatics and computational chemistry. These overlaps are the consequence of a tendency among computer scientists to recognize and act upon their field's many interdisciplinary connections.

Computer science continues to have strong mathematical and engineering roots. Computer science bachelor's, master's, and doctoral degree programmes are routinely offered by postsecondary academic institutions, and these programs require students to complete appropriate mathematics and engineering courses, depending on their area of focus. For example, all undergraduate computer science majors must study discrete mathematics (logic, combinatorics, and elementary graph theory). Many programmes also require students to complete courses in calculus, statistics, numerical analysis, physics, and principles of engineering early in their studies.

Glossary

theoretical foundations
hardware and software
to process information
artificial intelligence
network design
queueing theory
incorporates techniques
probability and statistics
electronic circuit design
to make heavy use of measurement
refinement of new algorithms
software engineering
computer engineering
computer science
interrelated disciplines
curriculum
computational science
computer architecture
programming languages
software development

2 Give the English equivalents to the following terms and phrases.

Языки программирования, искусственный интеллект, теоретические основы, интенсивно использовать (широко применяться), аппаратное и программное обеспечение, обрабатывать информацию, информатика, разработка программного обеспечения, вероятность и статистика, сетевой дизайн, усовершенствование/доработка новых алгоритмов, теория массового обслуживания, конструкция электронной схемы, включать методы, измерение, вычислительная наука, программная инженерия, учебный план, компьютерная инженерия, взаимосвязанные дисциплины, компьютерная архитектура.

3 Answer the questions.

1. What is the subject of computer science? What aspects does it include?
2. What disciplines and fields of knowledge laid foundations for computer science?
3. What subfields constitute the computer science?

TEXT 1.2 University Degree in Computer Science

1 Read and translate the text about the university degree in Computer Science at Oxford University.

Degree in Computer Science

Computer science is about understanding computer systems and networks at a deep level. Computers and the programs they run are among the most complex products ever created; designing and using them effectively presents immense challenges. Facing these challenges is the aim of computer science as a practical discipline, and this leads to some fundamental questions:

- How can we capture in a precise way what we want a computer system to do?

- Can we mathematically prove that a computer system does what we want it to?
- How can computers help us to model and investigate complex systems like the Earth's climate, financial systems or our own bodies?
- What are the limits to computing? Will quantum computers extend those limits?

The theories that are now emerging to answer these kinds of questions can be immediately applied to design new computers, programs, networks and systems that are transforming science, business, culture and all other aspects of life.

Computer Science can be studied for three years (BA) or four years (Master of Computer Science). The fourth year allows the study of advanced topics and an in-depth research project. Students do not need to choose between the three-year and four-year options when applying to the course; all students apply for the four-year course, and then decide at the start of the third year whether they wish to continue to the fourth year (which is subject to achieving a 2:1 at the end of the third year).

The course concentrates on creating links between theory and practice. It covers a wide variety of software and hardware technologies and their applications. We are looking for students with strong mathematical ability, which you will develop into skills that can be used both for reasoning rigorously about the behaviour of programs and computer systems, and for applications such as scientific computing. You will also gain practical problem-solving and program design skills; the majority of subjects within the course are linked with practical work in our well-equipped laboratory.

Glossary

to understand smth. at a deep level
to present an immense challenge
to face the challenge
to model and investigate complex systems
emerging theories

to apply
to design (*new computers, programs, networks and systems*)
to transform (*science, business, culture and all other aspects of life*)
an in-depth research project
to gain practical skills
a well-equipped laboratory

2. Vocabulary training exercise
Paraphrase the sentences substituting the underlined parts with the words and phrases from the glossary to Text 1.1 and Text 1.2.

1. The university has a lot of laboratories with good equipment.
2. The control panel uses all the newest technology and is considered very modern.
3. The internship in such a company allows getting the skills necessary for future work.
4. We gather and distribute information about new disease threats around the world.
5. New theories started to appear to deal with some important issues in quantum computing.
6. New malware (malicious software) is a great problem nowadays.
7. To solve this problem new approaches and systems should be created and used.
8. A careful and detailed examination of the system was carried out.
9. This elective course can help you understand the subject better.
10. New computers and technology can change all spheres and aspects of life.

3. Watch the video telling about the degree in computer science at Oxford University and answer the questions.

Video link: https://www.youtube.com/watch?v=JG-mHQX8eZw&t=2s

1. How has the subject changed over times?
2. What is the 'heart' of the study programme and why?

3. What is the essence of the programme? What is studied?
4. What are the opportunities for the projects?
5. What classes and how many classes do they have a week? What are the benefits of individual tailored approach?
6. What is the fundamental difference between self-study or short courses in the sphere of computer science and the university degree in computer science?
7. What is the most important thing when picking the students for top universities like Oxford? How do the top universities cherry-pick their students?
8. How do the universities support their students?
9. What is the main goal of the education provided?

4 Follow-up task: presentation.

Prepare a 3-minute talk on the degree in computer science or information technology offered by Saint Petersburg Mining University. Compare the key principles and courses run at your university with those taught in top world universities.

You can use the website Top Universities https://www.topuniversities.com/university-rankings/university-subject-rankings to choose a university ranked among top 10 in the world in the respective discipline(s).

UNIT 2 HISTORY OF COMPUTER SCIENCE

TEXT 2.1 Development of computer science

1 Read the text below and find answers to the questions after the text.

Development of computer science (Part 1)

Computer science emerged as an independent discipline in the early 1960s, although the electronic digital computer that is the object of its study was invented some two decades earlier. The roots of computer science lie primarily in the related fields of mathematics, electrical engineering, physics, and management information systems.

Mathematics is the source of two key concepts in the development of the computer — the idea that all information can be represented as sequences of zeros and ones and the abstract notion of a "stored program." In the binary number system, numbers are represented by a sequence of the binary digits 0 and 1 in the same way that numbers in the familiar decimal system are represented using the digits 0 through 9. The relative ease with which two states (e.g., high and low voltage) can be realized in electrical and electronic devices led naturally to the binary digit, or bit, becoming the basic unit of data storage and transmission in a computer system.

Electrical engineering provides the basics of circuit design — namely, the idea that electrical impulses input to a circuit can be combined using Boolean algebra to produce arbitrary outputs. The Boolean algebra developed in the 19th century supplied a formalism for designing a circuit with binary input values of zeros and ones (false or true, respectively, in the terminology of logic) to yield any desired combination of zeros and ones as output. The invention of the transistor and the miniaturization of circuits, along with the invention of electronic, magnetic, and optical media for the storage and transmission of information, resulted from advances in electrical engineering and physics.

Management information systems, originally called data processing systems, provided early ideas from which various computer

science concepts such as sorting, searching, databases, information retrieval, and graphical user interfaces evolved. Large corporations housed computers that stored information that was central to the activities of running a business — payroll, accounting, inventory management, production control, shipping, and receiving.

Theoretical work on computability, which began in the 1930s, provided the needed extension of these advances to the design of whole machines; a milestone was the 1936 specification of the Turing machine (a theoretical computational model that carries out instructions represented as a series of zeros and ones) by the British mathematician Alan Turing and his proof of the model's computational power. Another breakthrough was the concept of the stored-program computer, usually credited to Hungarian American mathematician John von Neumann. These are the origins of the computer science field that later became known as architecture and organization.

In the 1950s, most computer users worked either in scientific research labs or in large corporations. The former group used computers to help them make complex mathematical calculations (e.g., missile trajectories), while the latter group used computers to manage large amounts of corporate data (e.g., payrolls and inventories). Both groups quickly learned that writing programs in the machine language of zeros and ones was not practical or reliable. This discovery led to the development of assembly language in the early 1950s, which allows programmers to use symbols for instructions (e.g., ADD for addition) and variables (e.g., X). Another program, known as an assembler, translated these symbolic programs into an equivalent binary program whose steps the computer could carry out, or "execute."

Other system software elements known as linking loaders were developed to combine pieces of assembled code and load them into the computer's memory, where they could be executed. The concept of linking separate pieces of code was important, since it allowed "libraries" of programs for carrying out common tasks to be reused. This was a first step in the development of the computer science field called software engineering.

Glossary

stored program	output
binary number system	information retrieval
binary digit	computational power
data storage	to carry out instructions
a sequence of	variable
data transmission	assembly language
the basics of circuit design	assembler
input value	to execute a program
Boolean algebra	linking loader
Boolean operators (and, not, or)	software engineering
management information systems	to load into the computer's memory

2 Answer the following questions:

1. What fields of knowledge became the basis for computer science?
2. What were the two basic concepts that mathematics brought to computer science?
3. How did electrical engineering contribute to the development of computer science?
4. What is a management information system? What key concepts did management information systems provide?
5. When did the theoretical work on computability begin? What was a most important event in the history of computer science?
6. What is the Turing machine?
7. How did John von Neumann contribute to the development of computer science?
8. Who were the computer users of 1950s and what did they use computers for?
9. What is the assembly language and why did it appear?
10. Are the terms 'an assembly language' and 'an assembler' similar or different?

11. What function did linking loaders perform? Why were they so important?

3 Read the text and find synonyms to the following words and word combinations:

to appear, main concepts, an important discovery, an improvement or development in something.

TEXT 2.2 Further Development of Computer Science

1 Continue reading the text on the history of computer science and translate it into Russian.

Development of Computer Science (Part 2)

Later in the 1950s, assembly language was found to be so cumbersome that the development of high-level languages (closer to natural languages) began to support easier, faster programming. FORTRAN emerged as the main high-level language for scientific programming, while COBOL became the main language for business programming. These languages carried with them the need for different software, called compilers, that translate high-level language programs into machine code. As programming languages became more powerful and abstract, building compilers that create high-quality machine code and that are efficient in terms of execution speed and storage consumption became a challenging computer science problem. The design and implementation of high-level languages is at the heart of the computer science field called programming languages.

Increasing use of computers in the early 1960s provided the impetus for the development of the first operating systems, which consisted of system-resident software that automatically handled input and output and the execution of programs called “jobs.” The demand for better computational techniques led to a resurgence of interest in

numerical methods and their analysis, an activity that expanded so widely that it became known as computational science.

The 1970s and ’80s saw the emergence of powerful computer graphics devices, both for scientific modeling and other visual activities. (Computerized graphical devices were introduced in the early 1950s with the display of crude images on paper plots and cathode-ray tube [CRT] screens.) Expensive hardware and the limited availability of software kept the field from growing until the early 1980s, when the computer memory required for bitmap graphics (in which an image is made up of small rectangular pixels) became more affordable. Bitmap technology, together with high-resolution display screens and the development of graphics standards that make software less machine-dependent, has led to the explosive growth of the field. Support for all these activities evolved into the field of computer science known as graphics and visual computing.

Closely related to this field is the design and analysis of systems that interact directly with users who are carrying out various computational tasks. These systems came into wide use during the 1980s and ’90s, when line-edited interactions with users were replaced by graphical user interfaces (GUIs). GUI design, which was pioneered by Xerox and was later picked up by Apple (Macintosh) and finally by Microsoft (Windows), is important because it constitutes what people see and do when they interact with a computing device. The design of appropriate user interfaces for all types of users has evolved into the computer science field known as human-computer interaction (HCI).

The field of computer architecture and organization has also evolved dramatically since the first stored-program computers were developed in the 1950s. So called time-sharing systems emerged in the 1960s to allow several users to run programs at the same time from different terminals that were hard-wired to the computer. The 1970s saw the development of the first wide-area computer networks (WANs) and protocols for transferring information at high speeds between computers separated by large distances. As these activities evolved, they coalesced into the computer science field called networking and communications. A major accomplishment of this field was the development of the Internet.

The idea that instructions, as well as data, could be stored in a computer's memory was critical to fundamental discoveries about the theoretical behaviour of algorithms. These discoveries were the origin of the computer science field known as algorithms and complexity. A key part of this field is the study and application of data structures that are appropriate to different applications. Data structures, along with the development of optimal algorithms for inserting, deleting, and locating data in such structures, are a major concern of computer scientists because they are so heavily used in computer software, most notably in compilers, operating systems, file systems, and search engines.

In the 1960s the invention of magnetic disk storage provided rapid access to data located at an arbitrary place on the disk. This invention led not only to more cleverly designed file systems but also to the development of database and information retrieval systems, which later became essential for storing, retrieving, and transmitting large amounts and wide varieties of data across the Internet. This field of computer science is known as information management.

Another long-term goal of computer science research is the creation of computing machines and robotic devices that can carry out tasks that are typically thought of as requiring human intelligence. Such tasks include moving, seeing, hearing, speaking, understanding natural language, thinking, and even exhibiting human emotions. The computer science field of intelligent systems, originally known as artificial intelligence (AI), actually predates the first electronic computers in the 1940s, although the term artificial intelligence was not coined until 1956.

Three developments in computing in the early part of the 21st century — mobile computing, client-server computing, and computer hacking — contributed to the emergence of three new fields in computer science: platform-based development, parallel and distributed computing, and security and information assurance. Platform-based development is the study of the special needs of mobile devices, their operating systems, and their applications. Parallel and distributed computing concerns the development of architectures and programming languages that support the development of algorithms whose components can run simultaneously and asynchronously (rather than sequentially), in order to

make better use of time and space. Security and information assurance deals with the design of computing systems and software that protects the integrity and security of data, as well as the privacy of individuals who are characterized by that data.

Finally, a particular concern of computer science throughout its history is the unique societal impact that accompanies computer science research and technological advancements. With the emergence of the Internet in the 1980s, for example, software developers needed to address important issues related to information security, personal privacy, and system reliability. In addition, the question of whether computer software constitutes intellectual property and the related question "Who owns it?" gave rise to a whole new legal area of licensing and licensing standards that applied to software and related artifacts. These concerns and others form the basis of social and professional issues of computer science, and they appear in almost all the other fields identified above.

So, to summarize, the discipline of computer science has evolved into the following 15 distinct fields: algorithms and complexity, architecture and organization, computational science, graphics and visual computing, human-computer interaction, information management, intelligent systems, networking and communication, operating systems, parallel and distributed computing, platform-based development, programming languages, security and information assurance, software engineering, social and professional issues.

Glossary

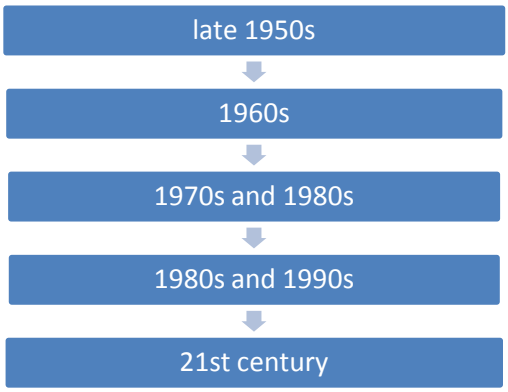
execution speed, at high speed
storage consumption
implementation
input and output
bitmap technology
bitmap graphics
high-resolution display screens
to carry out (various) computational tasks
human-computer interaction
to run a program

to store
to retrieve information
to transmit
artificial intelligence (AI)
to run simultaneously / asynchronously / sequentially

2 Give the English equivalents to the terms and phrases below:

Растровая графика, точечный рисунок; взаимодействие человека и машины; запустить программу; сохранять; мониторы высокого разрешения; скорость выполнения; выполнять различные вычислительные задачи на высокой скорости; извлекать информацию из памяти компьютера; расход памяти; искусственный интеллект; выполнение, осуществление; запускать одновременно или последовательно; ввод и вывод данных.

3 Summarise the chronology of computer science advancements using the scheme below.



4 Make a presentation on the key people who influenced computer science using the ideas below.

Who are the most well-known computer scientists?

The most influential computer scientists include **Alan Turing**, the World War II code breaker commonly regarded as the "father of modern computing"; **Tim Berners-Lee**, inventor of the World Wide Web; **John McCarthy**, inventor of the programming language LISP and artificial intelligence pioneer; and **Grace Hopper**, U.S. Navy officer and a key figure in the development of early computers such as the UNIVAC I as well as the development of the computer language compiler.

Key People

John von Neumann American mathematician	**Vannevar Bush** American engineer	**Alan Turing** British mathematician and logician	**Julian Assange** Australian computer programmer	**Steve Wozniak** American electronics engineer
Bill Gates American computer programmer, businessman, and philanthropist	**John Carmack** American computer-game designer	**Danny Hillis** American businessman	**Ahn Cheol-Soo** South Korean physician, educator, and entrepreneur	**Larry Ellison** American business executive
Douglas Engelbart American inventor	**Bill Joy** American software developer and entrepreneur	**Maurice Wilkes** British computer scientist	**Vinton Cerf** American computer scientist	**Ray Kurzweil** American computer scientist and futurist
Marvin Minsky American scientist	**Leonard Kleinrock** American computer scientist	**Paul Allen** American investor and philanthropist	**Nicholas Negroponte** American architect and computer scientist	**Edward Albert Feigenbaum** American computer scientist

UNIT 3 ALGORYTHMS AND COMPUTER LANGUAGES

TEXT 3.1 Algorithms and complexity

1 Read and translate the text.

Algorithms and complexity

An algorithm is a specific procedure for solving a well-defined computational problem. The development and analysis of algorithms is fundamental to all aspects of computer science: artificial intelligence, databases, graphics, networking, operating systems, security, and so on. Algorithm development is more than just programming. It requires an understanding of the alternatives available for solving a computational problem, including the hardware, networking, programming language, and performance constraints that accompany any particular solution. It also requires understanding what it means for an algorithm to be "correct" in the sense that it fully and efficiently solves the problem at hand.

An accompanying notion is the design of a particular data structure that enables an algorithm to run efficiently. The importance of data structures stems from the fact that the main memory of a computer (where the data is stored) is linear, consisting of a sequence of memory cells that are serially numbered 0, 1, 2,…. Thus, the simplest data structure is a linear array, in which adjacent elements are numbered with consecutive integer "indexes" and an element's value is accessed by its unique index. An array can be used, for example, to store a list of names, and efficient methods are needed to efficiently search for and retrieve a particular name from the array. For example, sorting the list into alphabetical order permits a so-called binary search technique to be used, in which the remainder of the list to be searched at each step is cut in half. This search technique is similar to searching a telephone book for a particular name. Knowing that the book is in alphabetical order allows one to turn quickly to a page that is close to the page containing the

desired name. Many algorithms have been developed for sorting and searching lists of data efficiently.

Although data items are stored consecutively in memory, they may be linked together by pointers (essentially, memory addresses stored with an item to indicate where the next item or items in the structure are found) so that the data can be organized in ways similar to those in which they will be accessed. The simplest such structure is called the linked list, in which noncontiguously stored items may be accessed in a pre-specified order by following the pointers from one item in the list to the next. The list may be circular, with the last item pointing to the first, or each element may have pointers in both directions to form a doubly linked list. Algorithms have been developed for efficiently manipulating such lists by searching for, inserting, and removing items.

Pointers also provide the ability to implement more complex data structures. A graph, for example, is a set of nodes (items) and links (known as edges) that connect pairs of items. Such a graph might represent a set of cities and the highways joining them, the layout of circuit elements and connecting wires on a memory chip, or the configuration of persons interacting via a social network. Typical graph algorithms include graph traversal strategies, such as how to follow the links from node to node (perhaps searching for a node with a particular property) in a way that each node is visited only once. A related problem is the determination of the shortest path between two given nodes on an arbitrary graph. A problem of practical interest in network algorithms, for instance, is to determine how many "broken" links can be tolerated before communications begin to fail. Similarly, in very-large-scale integration (VLSI) chip design it is important to know whether the graph representing a circuit is planar, that is, whether it can be drawn in two dimensions without any links crossing (wires touching).

The (computational) complexity of an algorithm is a measure of the amount of computing resources (time and space) that a particular algorithm consumes when it runs. Computer scientists use mathematical measures of complexity that allow them to predict, before writing the code, how fast an algorithm will run and how much memory it will

require. Such predictions are important guides for programmers implementing and selecting algorithms for real-world applications.

Computational complexity is a continuum, in that some algorithms require linear time (that is, the time required increases directly with the number of items or nodes in the list, graph, or network being processed), whereas others require quadratic or even exponential time to complete (that is, the time required increases with the number of items squared or with the exponential of that number). At the far end of this continuum lie the murky seas of intractable problems — those whose solutions cannot be efficiently implemented. For these problems, computer scientists seek to find heuristic algorithms that can almost solve the problem and run in a reasonable amount of time.

Further away still are those algorithmic problems that can be stated but are not solvable; that is, one can prove that no program can be written to solve the problem. A classic example of an unsolvable algorithmic problem is the halting problem, which states that no program can be written that can predict whether or not any other program halts after a finite number of steps. The unsolvability of the halting problem has immediate practical bearing on software development. For instance, it would be frivolous to try to develop a software tool that predicts whether another program being developed has an infinite loop in it (although having such a tool would be immensely beneficial).

Glossary

to solve a computational problem	layout
constraints	traversal
to run efficiently	graph traversal
a sequence of	node
linear array	arbitrary graph
integer	to determine
binary search technique	“broken” links
sort and search lists of data	very-large-scale integration
data item	planar
linked list	computational complexity
contiguous / noncontiguous	linear time

to implement	exponential
circuit elements	halting problem

2 Study the glossary and translate the following terms and word combinations into English.

Решать вычислительную задачу; элемент данных; упорядочивать и искать по спискам данных; работать эффективно; ограничивающее условие, уточнение; выполнять; целое число; последовательность, ряд; линейный массив; *матем.* определять, устанавливать, находить; метод двоичного поиска, дихотомического поиска; цепной список, список со связями, указателями; произвольный граф; смежный; перемещение, переход; компонент схемы, элемент схемы; компоновка, план; плоский, планарный; экспоненциальная, показательная; обход графа; разорванные связи; узел, точка пересечения; линейный, аналоговый; интеграция сверхвысокого уровня, СБИС; вычислительная сложность; зависимость; проблема остановки.

3 Discuss these questions with a partner.

1. What is an algorithm?
2. Why is particular data structure so important? Can you give some examples?
3. What is a linked list?
4. What are the main functions of pointers?
5. What is a computational complexity? How is the complexity of an algorithm measured?
6. What is an example of an unsolvable algorithmic problem? Can you give some other instances?

TEXT 3.2 Programming languages

1 Read and translate the text.

Programming languages

Programming languages are the languages with which a programmer implements a piece of software to run on a computer. The earliest programming languages were assembly languages, not far removed from the binary-encoded instructions directly executed by the computer. By the mid-1950s, programmers began to use higher-level languages.

Two of the first higher-level languages were FORTRAN (Formula Translator) and ALGOL (Algorithmic Language), which allowed programmers to write algebraic expressions and solve scientific computing problems. As learning to program became increasingly important in the 1960s, a stripped-down version of FORTRAN called BASIC (Beginner's All-Purpose Symbolic Instruction Code) was developed at Dartmouth College. BASIC quickly spread to other academic institutions, and by 1980 versions of BASIC for personal computers allowed even students at elementary schools to learn the fundamentals of programming. Also, in the mid-1950s, COBOL (Common Business-Oriented Language) was developed to support business programming applications that involved managing information stored in records and files.

The trend since then has been toward developing increasingly abstract languages, allowing the programmer to communicate with the machine at a level ever more remote from machine code. COBOL, FORTRAN, and their descendants (Pascal and C, for example) are known as imperative languages, since they specify as a sequence of explicit commands how the machine is to go about solving the problem at hand. These languages were also known as procedural languages, since they allowed programmers to develop and reuse procedures, subroutines, and functions to avoid reinventing basic tasks for every new application.

Other high-level languages are called functional languages, in that a program is viewed as a collection of (mathematical) functions and its semantics are very precisely defined. The best-known functional

language of this type is LISP (List Processing), which in the 1960s was the mainstay programming language for AI applications. Successors to LISP in the AI community include Scheme, Prolog, and C and C++. Scheme is similar to LISP except that it has a more formal mathematical definition. Prolog has been used largely for logic programming, and its applications include natural language understanding and expert systems such as MYCIN. Prolog is notably a so-called nonprocedural, or declarative, language in the sense that the programmer specifies what goals are to be accomplished but not how specific methods are to be applied to attain those goals. C and C++ have been used widely in robotics, an important application of AI research. An extension of logic programming is constraint logic programming, in which pattern matching is replaced by the more general operation of constraint satisfaction.

Another important development in programming languages through the 1980s was the addition of support for data encapsulation, which gave rise to object-oriented languages. The original object-oriented language was called Smalltalk, in which all programs were represented as collections of objects communicating with each other via message-passing. An object is a set of data together with the methods (functions) that can transform that data. Encapsulation refers to the fact that an object's data can be accessed only through these methods. Object-oriented programming has been very influential in computing. Languages for object-oriented programming include C++, Visual BASIC, and Java.

Java is unusual because its applications are translated not into a particular machine language but into an intermediate language called Java Bytecode, which runs on the Java Virtual Machine (JVM). Programs on the JVM can be executed on most contemporary computer platforms, including Intel-based systems, Apple Macintoshes, and various Android-based smartphones and tablets. Thus, Linux, iOS, Windows, and other operating systems can run Java programs, which makes Java ideal for creating distributed and Web-based applications. Residing on Web-based servers, Java programs may be downloaded and run in any standard Web browser to provide access to various services, such as a client interface to a game or entry to a database residing on a server.

At a still higher level of abstraction lie declarative and scripting languages, which are strictly interpreted languages and often drive applications running in Web browsers and mobile devices. Some declarative languages allow programmers to conveniently access and retrieve information from a database using "queries," which are declarations of what to do (rather than how to do it). A widely used database query language is SQL (Structured Query Language) and its variants (e.g., MySQL and SQLite). Associated with these declarative languages are those that describe the layout of a Web page on the user's screen. For example, HTML (HyperText Markup Language) supports the design of Web pages by specifying their structure and content. Gluing the Web page together with the database is the task of a scripting language (e.g., PHP), which is a vehicle for programmers to integrate declarative statements of HTML and MySQL with imperative actions that are required to effect an interaction between the user and the database. An example is an online book order with Amazon.com, where the user queries the database to find out what books are available and then initiates an order by pressing buttons and filling appropriate text areas with his or her ordering information. The software that underlies this activity includes HTML to describe the content of the Web page, MySQL to access the database according to the user's requests, and PHP to control the overall flow of the transaction.

Computer programs written in any language other than machine language must be either interpreted or translated into machine language ("compiled"). As suggested above, an interpreter is software that examines a computer program one instruction at a time and calls on code to execute the machine operations required by that instruction.

A compiler is software that translates an entire computer program into machine code that is saved for subsequent execution whenever desired. Much work has been done on making both the compilation process and the compiled code as efficient as possible. When a new language is developed, it is usually interpreted at first. If it later becomes popular, a compiler is developed for it, since compilation is more efficient than interpretation.

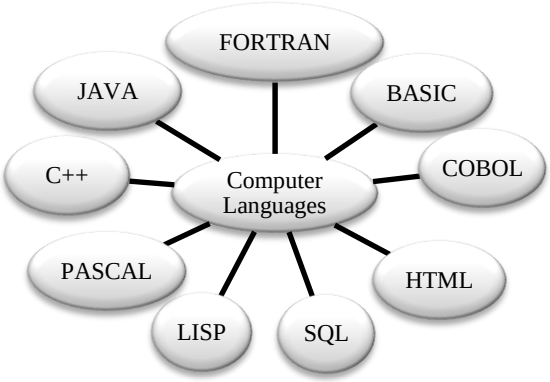
There is an intermediate approach, which is to compile code not into machine language but into an intermediate language (called a virtual machine) that is close enough to machine language that it is efficient to interpret, though not so close that it is tied to the machine language of a particular computer. It is this approach that provides the Java language with its computer platform independence via the JVM.

Glossary

programming languages	a sequence of commands
to implement	procedural languages
to run on a computer	functional languages
assembly languages	precisely
to execute	declarative languages
a higher-level language	scripting languages
to solve computing problems	to retrieve information
the fundamentals of programming	the layout of a Web page
application	to compile a program
imperative languages	an interpreter

2 Choose an extract of 850 symbols and translate it in written form.

3 Summarise the information in the text. Be ready to enlarge upon the computer languages features and application using the glossary.



UNIT 4 HARDWARE AND SOFTWARE

TEXT 4.1 Types and components of computer systems

1 Look at the terms in a bold type in the text. Give their Russian equivalents.

2 Read and translate the text.

What is inside a PC System?

Processing

The nerve centre of a PC is the **processor**, also called the CPU, or **central processing unit**. This is built into a single chip which executes program instructions and coordinates the activities that take place within the computer system. The chip itself is a small piece of silicon with a complex electrical circuit called an **integrated circuit**.

The processor consists of three main parts:

- The **control unit** examines the instructions in the user's program, interprets each instruction and causes the circuits and the rest of the components – monitor, disk drives, etc. – to execute the functions specified.
- The **arithmetic logic unit (ALU)** performs mathematical calculations (+, -, etc.) and logical operations (AND, OR, NOT).
- The **registers** are high-speed units of memory used to store and control data. One of the registers (the program counter, or PC) keeps track of the next instruction to be performed in the main memory. The other (the instruction register, or IR) holds the instruction that is being executed.

The power and performance of a computer is partly determined by the speed of its processor. A **system clock** sends out signals at fixed intervals to measure and synchronize the flow of data. **Clock speed** is measured in **gigahertz** (GHz). For example, a CPU running at 4 GHz 9four thousand million hertz, or cycles, per second) will enable your PC to handle the most demanding applications.

RAM and ROM

The programs and data which pass through the processor must be loaded into the main memory in order to be processed. Therefore, when the user runs a program, the CPU looks for it on the hard disk and transfers a copy into the RAM chips. **RAM (random access memory)** is volatile – that is, its information is lost when the computer is turned off. However, **ROM (read only memory)** is non-volatile, containing instructions and routines for the basic operations of the CPU. The **BIOS (basic input/output system)** uses ROM to control communication with **peripherals**.

RAM capacity can be expanded by adding extra chips, usually contained in small circuit boards called **dual in-line memory modules (DIMMS)**.

Buses and cards

The main **circuit board** inside your system is called the **motherboard** and contains the processor, the memory chips, expansion slots, and controllers for peripherals, connected by **buses** – electrical channels which allow devices inside the computer to communicate with each other. For example, the front side bus carries all data that passes from the CPU to other devices.

The size of a bus, called **bus width**, determines how much data can be transmitted. It can be compared to the number of lanes on a motorway – the larger the width, the more data can travel along the bus. For example, a 64-bit bus can transmit 64 bits of data.

Expansion slots allow users to install expansion cards, adding features like sound, memory and network capabilities.

3 Answer the following questions.

1. What is the main function of a computer processor?
2. What are the main parts of the CPU?
3. What does ALU stand for? What does it do?
4. What is the function of the system clock?
5. How much is one gigahertz? What is this unit used for in computers?
6. What type of memory is temporary? What is it used for?
7. What type of memory is permanent? What does it include?

8. How can RAM be increased?
9. What term is used to refer to the main printed circuit board?
10. What is a bus?
11. What is the benefit of having expansion slots?

4 Explain the way a PC system works using the terms from the text.

TEXT 4.2 Operating Systems

1 Read the text and be ready to discuss it.

Operating systems

An operating system is a specialized collection of software that stands between a computer's hardware architecture and its applications. It performs a number of fundamental activities such as file system management, process scheduling, memory allocation, network interfacing, and resource sharing among the computer's users. Operating systems have evolved in their complexity over time, beginning with the earliest computers in the 1960s.

With early computers, the user typed programs onto punched tape or cards, which were read into the computer, assembled or compiled, and run. The results were then transmitted to a printer or a magnetic tape. These early operating systems engaged in batch processing; i.e., handling sequences of jobs that are compiled and executed one at a time without intervention by the user. Accompanying each job in a batch were instructions to the operating system (OS) detailing the resources needed by the job, such as the amount of CPU time required, the files needed, and the storage devices on which the files resided. From these beginnings came the key concept of an operating system as a resource allocator. This role became more important with the rise of multiprogramming, in which several jobs reside in the computer simultaneously and share resources, for example, by being allocated fixed amounts of CPU time in turn. More sophisticated hardware allowed one job to be reading data while another wrote to a printer and still another performed computations. The

operating system thus managed these tasks in such a way that all the jobs were completed without interfering with one another.

The advent of time sharing, in which users enter commands and receive results directly at a terminal, added more tasks to the operating system. Processes known as terminal handlers were needed, along with mechanisms such as interrupts (to get the attention of the operating system to handle urgent tasks) and buffers (for temporary storage of data during input/output to make the transfer run more smoothly). Modern large computers interact with hundreds of users simultaneously, giving each one the perception of being the sole user.

Another area of operating system research is the design of virtual memory. Virtual memory is a scheme that gives users the illusion of working with a large block of contiguous memory space (perhaps even larger than real memory), when in actuality most of their work is on auxiliary storage (disk). Fixed-size blocks (pages) or variable-size blocks (segments) of the job are read into main memory as needed. Questions such as how much main memory space to allocate to users and which pages or segments should be returned to disk ("swapped out") to make room for incoming pages or segments must be addressed in order for the system to execute jobs efficiently.

The first commercially viable operating systems were developed by IBM in the 1960s and were called OS/360 and DOS/360. Unix was developed at Bell Laboratories in the early 1970s and since has spawned many variants, including Linux, Berkeley Unix, GNU, and Apple's iOS. Operating systems developed for the first personal computers in the 1980s included IBM's (and later Microsoft's) DOS, which evolved into various flavours of Windows. An important 21st-century development in operating systems was that they became increasingly machine-independent.

2 Answer the questions.

1. What is an operating system?
2. How did early operating systems work?
3. What is multiprogramming?
4. What is time sharing and how did it change the OS?

5. What is the function of virtual memory?
6. What is the key feature of modern operating systems?

3 Follow-up task: Presentation.

Choose any operating system mentioned in the last paragraph of Text 4.2. Make a 5-minute presentation covering the main aspects and features of the OS, its advantages and drawbacks.

REFERENCES

1. Encyclopedia Britannica. [Электронный ресурс]. URL: https://www.britannica.com/science/computer-science (дата обращения: 28.01.2022).
2. Oxford University [Электронный ресурс]. URL: https://www.ox.ac.uk/admissions/undergraduate/courses-listing/computer-science (дата обращения: 28.01.2022).
3. Top Universities [Электронный ресурс]. URL: https://www.topuniversities.com/university-rankings/university-subject-rankings (дата обращения: 28.01.2022).
4. Massachusetts Institute of Technology News [Электронный ресурс]. URL: https://news.mit.edu (дата обращения: 28.01.2022).
5. Esteras S.R. Infotech: English for computer users, 4th edition. Cambridge [England]: Cambridge University Press, 2014.
6. Wright V., Taylor D. ICT Coursebook. Revised Edition. Cambridge [England]: Cambridge University Press. [Электронный ресурс]. URL: http://www.cambridge.org/9781108698061(дата обращения: 28.01.2022).

CONTENTS