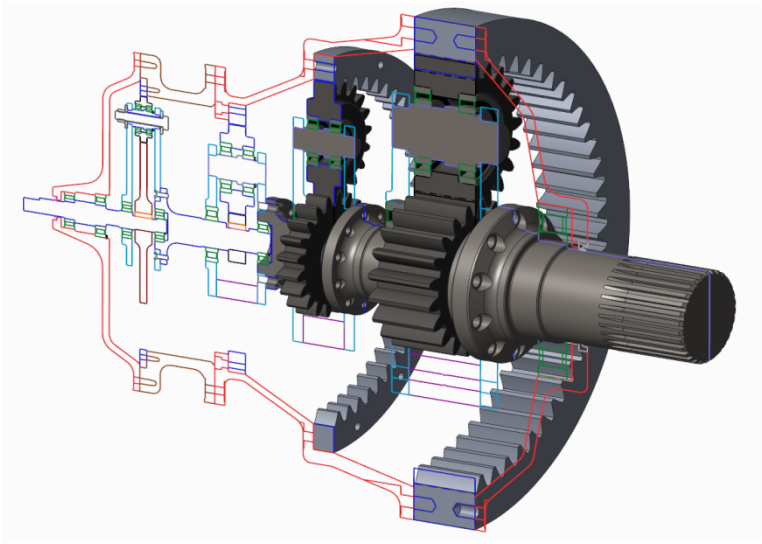
ENGLISH FOR IT ENGINEERS



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

КАЗПНСКИЙ ГОСУДАРСТВЕННЫЙ АРХИТЕКТУРНО-

СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ

Кафедра иностранных языков

English for IT Engineers

Методические указания для студентов бакалавров 1 курса дневного отделения направления 09.03.02 «Информационные системы и технологии»

Казань

2016

УДК 802.0:33
ББК 81.2Англ:69.003
Г94

English for IT Engineers: Методические указания для студентов бакалавров 1 курса дневного отделения направления 09.03.02 «Информационные системы и технологии» /Сост.: Абитов Р.Н, - Казань: Изд-во Казанского государственного архитектурно-строительного университета, 2016 – 24 с.

Печатается по решению Редакционно-издательского совета Казанского государственного архитектурно-строительного университета

Методические указания на английском языке «English for IT Engineers» составлены в соответствии с требованиями программы, определенной Государственным образовательным стандартом.

Методические указания для студентов бакалавров 1 курса дневного отделения направления 08.03.01 «Строительство», обучающихся на дневной форме обучения неязыковых вузов.

Основная цель данных методических указаний – выработать у студентов умение использовать языковой материал по теме в монологической и диалогической речи.

Рецензент
Кандидат технических наук,
зав. кафедры информационных технологий и систем автоматизированного проектирования КазГАСУ
Д.М. Кордончик

УДК 802.0:33
ББК 81.2Англ:69.003
Г94

UNIT 1

OPERATING SYSTEMS

1.1. Before you start

- **What is an operating system in your opinion?**
- **What operating systems (or operating system families) can you name?**

1.2. Read and remember following words and word combinations.

to take smth for granted - *воспринимать как само собой разумеющееся*

time consuming - *времязатратный*

punched cards - *перфокарты*

magnetictape - *магнитная лента*

store - *запоминать, хранить*

stacks - *пачки*

A Brief Introduction to the History of Operating Systems

Most ordinary computer users take their operating system for granted. The easiest way to understand what an operating system does is to take a close look at what computers were like before operating systems had been invented.

The earliest electronic computers did not have any operating system. If the user wanted to change what the computer was doing, the user had to open the back panel on the (then very large) computer, and change how the wires were connected. Changing what the computer did was very time consuming and required an expert.

Later, computer scientists decided to have the wires stay as they were, and feed instructions to the computer with punched cards (cards with holes that represented instructions) or magnetic tape. The computer would store the instructions in some

kind of memory. This way of operating a computer is called the von Neumann architecture.

Still, computers of the time generally only had enough memory to "remember" one program at a time. If the user wanted the computer to run a different program, the user had to wipe out the first program from memory and then load another program into memory.

Computer operators and computer scientists grew tired of carrying around large stacks of punch cards. They also wanted computers to run more than one program at a time. As years of work changed or replaced computers to have more memory, computer operators and computer scientists discovered that some computers could hold several programs in their memory. The computer user could then simply choose which program the user wanted to run. Running a computer this way requires a "boss" program that controls all the other programs, and asks the user what program the user wants to run. Such a boss program is called an operating system.

Having several programs in memory that can be run at any time makes some new problems. The operating system itself has to remember where the programs are at in memory. The operating system also has to prevent two programs from fighting over which one gets to use the processor.

Modern desktop or laptop computers need an operating system so that they can operate. They are usually sold with it already installed. Operating systems normally start up automatically when the user turns on the computer.

(Source: https://simple.wikipedia.org/wiki/Operating_system)

1.3. Answer the following questions.

1. Were there operating systems on the first computers?
2. What did the first computers look like?
3. In what way does the simplest operating system function?
4. What operating systems ans OS-families do you know?

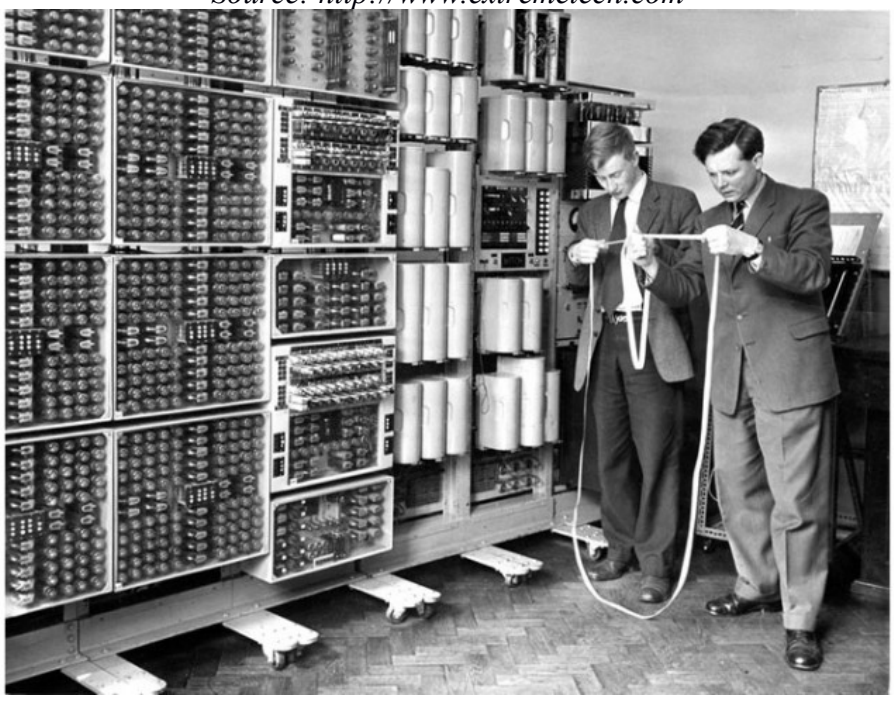
1.4. Say whether the following statements are true or false according to the text

#	Statements	True	False
1	Operating systems have existed since the emergence of the first computers		
2	Computer memory increase helped computer scientists to use two programs simultaneously which inevitably led to the emergence of operating systems		
3	The process governance is quite simple process in programmers point of view		
4	There are programs which can run without assistance of operating systems		

1.5. Give English equivalence to the following words and word combinations, mentioned in the text

Операционная система, перфокарта, приложение, программист, команды, фон-неймановская архитектура, память, стационарный персональный компьютер, ноутбук, запускать программу.

Source: http://www.extremetech.com



UNIT 2

CENTRAL PROCESSING UNITS

2.1. Before you start

- **What CPU-manufacturing companies can you name?**
- **What CPU's do you have on your PC?**

2.2. Read and remember following words and word combinations.

CPU	–	*Центральный процессор*
execute	–	*выполнять(программу)*
handle	–	*оперировать*
register	–	*регистр процессора*
chip	–	*полупроводниковый кристалл, чип*
cache	–	*кэш*
L1 cache	–	*кэш первого уровня*
L2 cache	–	*кэш второго уровня*
bus	–	*шина передачи данных*
pipeline	–	*конвеер процессора*

How does CPU work?

A central processing unit (CPU) is an important part of almost every computer. The CPU sends signals to control the other parts of the computer, almost like how a brain controls a body.

The CPU is an electronic machine that works on a list of things to do, called 'instructions'. A list of instructions that a CPU can run is a computer program.

The speed that a CPU works at is measured in hertz (Hz). Modern processors often run so fast that gigahertz (GHz) is used instead, which means a billion cycles per second.

The CPU market for desktop (home) computers is controlled by two companies: Intel and Advanced Micro Devices (usually shortened to AMD). There are other CPU manufacturers like ARM, IBM, VIA, MCST, ELVEES, SRISA, NTC Module, and Sun Microsystems, but their CPUs usually have more specific uses (for example in mobile phones, cars, game consoles, or in the military).

There are mainly two different types of CPUs used in modern desktop systems: 32-bit CPUs and 64-bit CPUs. The instructions in a 32-bit CPU are good at handling data that is 32 bits in size (most instructions "think" in 32 bits in a 32-bit CPU). Likewise, a 64-bit CPU is good at handling data that is 64 bits in size (and is often good at handling 32-bit data too). The size of data that a CPU handles best is often called the word size of the CPU. Many older CPUs from the 70s, 80s and early 90s (and some modern small CPUs) have an 8-bit or 16-bit word size.

When the CPU runs a computer program, it needs somewhere to store the data that the instructions operate on (the data that they read and write). These are called registers. A CPU usually has many registers.

Storing all data in registers would usually make the CPU too complicated (and very expensive). Therefore, registers usually only store the data that the CPU is working on "right now". The rest of the data used by the program is stored outside the CPU in RAM (memory) in separate chips.

When the CPU wants to read or write some data in RAM, it outputs an address to that data. The size of addresses is often the same as the word size: A 32-bit CPU uses 32-bit addresses, etc. (However, smaller CPUs, like 8-bit CPUs, often have an address size that is larger than the word size. A 64-bit processor might be able to handle up to 16 EB (16 exabytes, around 16 billion GB, or 16 billion billion bytes) of RAM.

On modern computers, RAM is much slower than registers, so accessing RAM slows down programs. To speed up memory accesses, a faster type of memory called a cache is often put between the RAM and the main parts of the CPU. The cache is usually a part of the CPU chip itself, and is much more expensive per byte than RAM. In multi-level caching, there are many caches, called the L1 cache, the L2

cache, and so on. The L1 cache is the fastest (and most expensive per byte) cache and is "closest" to the CPU. The L1 cache can often be viewed as a cache for the L2 cache, etc.

Buses are the "wires" used by the CPU to communicate with RAM and other components in the computer. Almost all CPUs have at least a data bus - used to read and write data - and an address bus - used to output addresses.

An instruction set (also called an ISA - Instruction Set Architecture) is a language understood directly by a particular CPU. They say how you tell the CPU to do different things, like loading data from memory into a register, or adding the values from two registers. Each instruction in an instruction set has an encoding, which is how the instruction is written as a sequence of bits.

Programs written in programming languages like C and C++ can't be run directly by the CPU. They must be translated into machine code before the CPU can run them. A program that translates assembly language into machine code is called an assembler.

Even very complicated programs can be made by combining many simple instructions like these. Many CPUs today can do more than 1 billion (1,000,000,000) instructions in a single second. In general, the more a CPU can do in a given time, the faster it is. Flops (Floating-point operations per second) and CPU clock speed (usually measured in gigahertz) are also ways to measure how much work a processor can do in a certain time.

(Source: http://simple.wikipedia.org/wiki/Central_processing_unit)

2.3. Answer the following questions.

1. Why is CPU often compared with a human brain?
2. How is CPU speed estimated?
3. What is the difference between 32 and 64 bit CPU's?
4. Why does CPU need registers?
5. Why does CPU need cache memory?

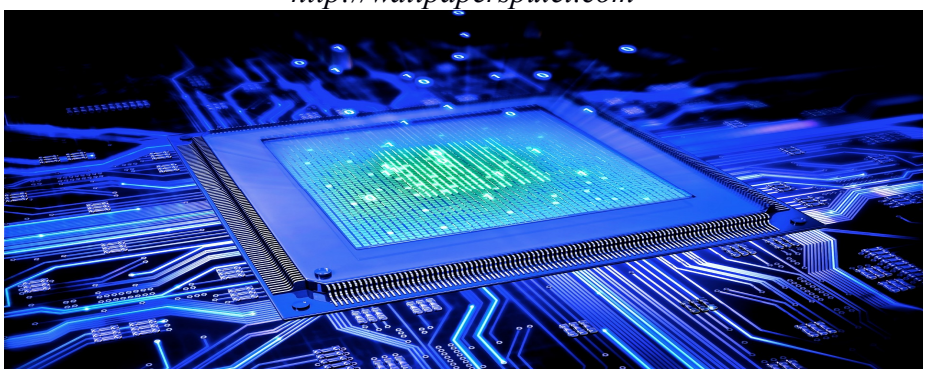
2.4. Say whether the following statements are true or false according to the text

#	Statements	True	False
1	There is only one monopoly manufacturing CPU's		
2	CPU registers are much faster than cache memory and Random Acces Memory.		
3	Programs written on C and C++ languages can run on computers right away		
4	CPU speed is measured with Hz		

2.5. Give English equivalence to the following words and word combinations, mentioned in the text

Центральный процессор, команды процессора, 32-битный процессор, 64-битный процессор, данные, регистр, полупроводниковая микросхема, Память произвольного доступа (оперативная память), адрес, замедлять, кэш, системная шина, шина данных, набор инструкций, ноль, единица, ассемблер, частота процессора, компьютерная периферия, модуль управления памятью,

Source:
http://wallpaperspixel.com



UNIT 3

PROGRAMMING LANGUAGE

3.1. Before you start

- **What is your favorite programming language?**
- **What is the difference between low-level and high-level programming languages?**

3.2. Read and remember following words and word combinations.

sourcecode	-	*исходный код (текст на языке программирования)*
compile	-	*компилировать (в машинный код процессора)*
assemblylanguage	-	*ассемблер (язык удобочитаемых команд машинного кода)*
bug	-	*баг, ошибка в программе*
declarativeprogramming	-	*декларативное программирование*
imperativeprogramming	-	*императивное программирование*
functionalprogramming	-	*функциональное программирование*
proceduralprogramming	-	*процедурное программирование*
stackbased	-	*стековый вид процедурного программирования*
object-orientedprogramming	-	*объектно-ориентированное программирование*
flow-orientedprogramming	-	*поточное программирование*
properties	-	*свойства объектов*
methods	-	*методы (процедуры объектов)*
classes	-	*классы (шаблоны объектов)*

Programming Languages and their Classification

A programming language is a type of written language that tells computers what to do. Programming languages are used to make all the computer programs and

computer software. A programming language is like a set of instructions that the computer follows to do something.

A programmer writes source code text in the programming language to create programs. Usually the programming language uses real words for some of the commands, so that the language is easier for a human to read. Many programming languages use punctuation just like a normal language. Many programs now are compiled. This means that the computer translates the source code into new languages such as assembly language or machine language, which are much faster and easier for the computer to read, but much harder for a person to read.

Computer programs must be written very carefully. If the programmer makes mistakes, or the program tries to do something the programmer did not design it to do, the program might then "crash" or stop working. When a program has a problem because of how the code was written, this is called a bug. A very small mistake can cause a very big problem. For example, forgetting a period or typing a plus sign instead of a minus sign can cause a bad bug.

Types of programming languages

There are many types of programming languages. Most programming languages do not follow one type alone, so it is difficult to assign a type for each language. The examples of each type are given in each section below because they are the best well-known examples of that type.

Declarative vs. Imperative programming

Declarative programming languages describe a "problem" but they usually do not say how the problem should be solved. The problem description uses logic, "solving" the problem often looks like automatically proving a system of logical axioms. Examples for such programming languages are Prolog, XSLT, LISP and SQL

Imperative programming languages describe a system of state changes. At the start, the program is in a certain state, and the computer is given steps to follow, in order to perform an action. Following the steps causes the program to "change state".

In general, declarative programming languages are safer and shorter. Imperative programming languages are more common, because they are easier to use.

Functional vs. Procedural

Functional programming looks at programming like a function in mathematics. The program receives input, some information, and uses this information to create output. It will not have a state in between, and it will also not change things that are not related to the computation.

Procedural programs are a set of steps or state changes.

Stack based

Stack based languages look at the some of the program's memory like a stack of cards. There are very few things that can be done with a stack. A data item can be put on the top of the stack. This operation is generally called "push". A data item can be removed from the top of the stack. This is called a "pop". You can look at the item at the top of the stack without removing it. This is called a "peek".

If a program is written as "push 5; push 3; add; pop;" it will put 5 on the top of the stack, put 3 on top of the 5, add the top two values (3 + 5 = 8), replace the 3 and 5 with the 8, and print the top (8). Examples for programming languages that are stack-based are the languages Postscript and Forth.

Object-oriented

Object-oriented programming languages place data and functions that change data into a single unit. This unit is called an "object". Objects can interact with each other, but one object can not change another object's data. This is usually called encapsulation or information hiding. Most modern programming languages are object-oriented. Examples of this are Java, C++, C# and other C languages.

Flow-oriented

Flow oriented programming sees programming as connecting different components. These components send messages back and forth. A single component can be part of different "programs", without the need to be changed internally.

Rules of programming

Every programming language has rules about what it can and can not do. These include:

Correct numbers (types of numbers, and how large or small the numbers can go)

Words (reserved words, case-sensitivity)

Limits on what the programming language can do

Most languages have official standards that define the rules of how to write the source code. Some programming languages have two or more standards. This can happen when a new standard replaces an old one. For example, the Perl 5 standard replaced Perl 4 in 1993. It can happen because two people made two standards at the same time. For example, there are several standards for APL.

Object-Oriented Programming

Object-Oriented Programming (sometimes shortened to OOP) is a form of programming where all parts of the program are objects. Objects are pieces of memory with the same structure that can be used again and again. A bank account, bitmap, or hero from a video game could all be objects within a program. Objects are made up of properties (pieces of information the object stores) and methods which are things the object can do. A Dog object might have properties like height and hairColor. Its methods might include bark() and wagTail().

All objects are created from templates called classes. You can think of a class as a mold from which objects are made. The class defines all the properties and methods that its objects will have. Objects created from a class are called instances of the class. A class can extend another class, which means that it takes all the properties and methods of the class but can add its own.

(Source: https://simple.wikipedia.org/wiki/Programming_languages)

3.3. Answer the following questions.

1. What do human languages and programming languages have in common?
2. Why must all the programs be written carefully?
3. Name the programming languages you know. Which one you started with?
4. Which programming language do you consider the hardest to master?
5. Give the main idea of object-oriented programming. Why is it so widespread in our days?

3.4. Say whether the following statements are true or false according to the text

#	Statements	True	False
1	All programming languages were initially created human-readable		
2	A mistake in a program code can eventually result in big problem		
3	Every programming language has its own strict rules		

3.5. Give English equivalence to the following words and word combinations, mentioned in the text

Исходный код, компиляция, язык ассемблера, баг, декларативное программирование, императивное программирование, смена состояния, функциональное программирование, процедурное программирование, стековое программирование, объектно-ориентированное программирование, чувствительность к регистру, зарезервированные операторы.

Source:http://pixabay.com

UNIT 4

COMPUTER AIDED DESIGN

4.1. Before you start

- **What does CAD acronym stand for?**
- **What was the name of the first CAD-system?**

4.2.Read and remember following word and word combinations.

Computer-AidedDesign / CAD	-	*Система автоматического проектирования / САПР*
decade	-	*десятилетие*
virtually	-	*фактически*
trace their lineage	-	*ведут свою родословную*
descriptive geometry	-	*начертательная геометрия*
drafting	-	*черчение*
drawings	-	*чертежи*
changed very little until after	-	*почти не менялись до*
MIT	-	*Массачусетский технологический институт.*
dozens	-	*десятки*
numerical control of machine tools	-	*проверка данных с помощью компьютера*
who are largely credited with setting the stage	-	*которые внесли большой вклад*
is credited	-	*считается*
presented his Ph.D. thesis	-	*защитил докторскую диссертацию*
CRT	-	*ЭЛТ монитор*

digitizer	-	*оцифровщик*
curveandsurfacemodeling	-	*моделирование поверхностей и кривых*
emergence	-	*появление*
show up	-	*появляться*
milestone	-	*веха*
solid geometry	-	*трехмерное моделирование*
kernels	-	*движки программ*
computations	-	*вычисления*
parametricandexplicitmodeling	-	*физическое и визуальное моделирование*

Evolution of Computer-Aided Design

While they may seem new to some, many of the computer-aided design programs we use today have been around for more than a decade, and virtually all trace their lineage to work begun more than 50 years ago.

Modern engineering design and drafting can be traced back to the development of descriptive geometry in the 16th and 17th centuries. Drafting methods improved with the introduction of drafting machines, but the creation of engineering drawings changed very little until after World War II.

During the war, considerable work was done in the development of real-time computing, particularly at MIT, and by the 1950s there were dozens of people working on numerical control of machine tools and automating engineering design. But it's the work of two people in particular—Patrick Hanratty and Ivan Sutherland—who are largely credited with setting the stage for what we know today as CAD.

Hanratty is widely credited as "the Father of CAD/CAM." In 1957, while working at GE, he developed PRONTO (Program for Numerical Tooling

Operations), the first commercial CNC programming system. Five years later, Sutherland presented his Ph.D. thesis at MIT titled “Sketchpad, A Man-Machine Graphical Communication System.” Among its features, the first graphical user interface, using a light pen to manipulate objects displayed on a CRT.

The 1960s brought other developments, including the first digitizer (from Autotrol) and DAC-1, the first production interactive graphics manufacturing system. By the end of the decade, a number of companies were founded to commercialize their CAD programs.

By the 1970s, research had moved from 2D to 3D. Major milestones included the work of Ken Versprille, whose invention of NURBS for his Ph.D. thesis formed the basis of modern 3D curve and surface modeling, and the development by Alan Grayer, Charles Lang, and Ian Braid of the PADL (Part and Assembly Description Language) solid modeler.

With the emergence of UNIX workstations in the early ’80s, commercial CAD systems like CATIA and others began showing up in aerospace, automotive, and other industries. But it was the introduction of the first IBM PC in 1981 that set the stage for the large-scale adoption of CAD. The following year, a group of programmers formed Autodesk, and in 1983 released AutoCAD, the first significant CAD program for the IBM PC. From then on, increasingly advanced drafting and engineering functionality became more affordable. But it was still largely 2D.

That changed in 1987 with the release of Pro/ENGINEER, a CAD program based on solid geometry and feature-based parametric techniques for defining parts and assemblies. It ran on UNIX workstations—PCs of the time were simply not powerful enough—but it was a game changer. The later years of the decade saw the release of several 3D modeling kernels, most notably ACIS and Parasolids, which would form the basis for other history-based parametric CAD programs.

By the 1990s, the PC was capable of the computations required by 3D CAD. In 1995, when the first issue of Desktop Engineering was published, SolidWorks was released. It was the first significant solid modeler for Windows. This was followed by Solid Edge, Inventor, and others. The decade also saw many of the original CAD

developers from the 1960s acquired by newer companies and a consolidation of the industry into four main players—Autodesk, Dassault Systems (which acquired SolidWorks in 1997), PTC, and UGS (now Siemens PLM)—along with a host of smaller developers.

(Source: http://www.pearltrees.com/jgggbdn/sw-fr/id13459155)

4.3. Answer the following questions.

1. When did the development of CAD-systems start?
2. What institution contributed most to the development of CAD programs?
3. Who are considered as the "fathers of CAD"?
4. When was the first version of AutoCAD released?
5. What major milestones of CAD development can you name?

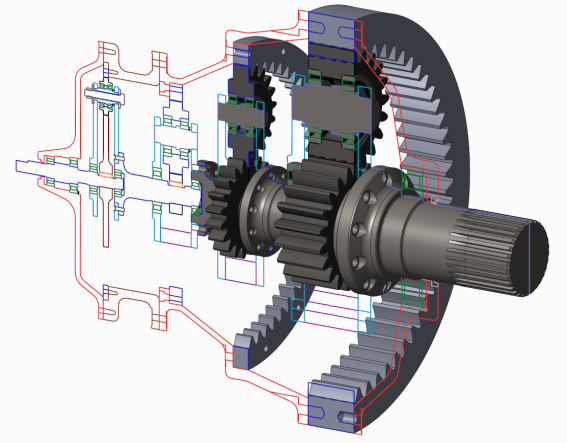
4.4. Say whether the following statements are true or false according to the text

#	Statements	True	False
1	CAD-systems have been there since the middle of the 20th century		
2	First realtime CAD-system was developed at MIT		
3	First CAD-systems were able to draw curves		
4	AutoCAD was the first CAD-system on PC		

4.5. Give English equivalence to the following words and word combinations, mentioned in the text

Практически, методы чертежа, автоматизация процесса проектирования, графический интерфейс пользователя, оцифровщик, была в большей степени двухмерной

Source:
http:/www.ptc.com



UNIT 5

BUILDING INFORMATION MODELING

5.1. Before you start.

- What does BIM acronym mean?
- Why is it considered to be the next step of CAD-systems?

5.2. Read and remember following word and word combinations.

collaboratively	-	*совместно*
don't be misled	-	*не путайте*
to be relevant	-	*относиться*
enormous gains in saving in cost and time	-	*существенно экономит время и деньги*
accuracy	-	*точность*
estimation	-	*оценка*
alterations	-	*изменения*
due to	-	*в связи с*
adopting	-	*применение*
representation	-	*представление*
facility	-	*здание*
life-cycle	-	*эксплуатация*
conception	-	*задумка*
demolition	-	*снос*
Pinningdown	-	*отвечая на заданный вопрос*
emphasis	-	*акцент, внимание*

implementation	-	*применение*
unified entity	-	*единое целое*
interacting	-	*взаимодействие*
complementing	-	*взаимодополнение*
feedback loops	-	*цикл взаимодействия*
eventually	-	*в конце концов*

WHAT IS BIM?

BIM is an acronym for Building Information Modeling, or Building Information Model. It describes the process of designing a building collaboratively using one coherent system of computer models rather than as separate sets of drawings. Don't be misled by the word 'building' – BIM is just as relevant to the civil engineering sector.

It offers enormous gains in saving in cost and time, much greater accuracy in estimation, and the avoidance of error, alterations and rework due to information loss. But adopting BIM involves much more than simply changing the software we use. To achieve all the benefits it offers, everyone in the architecture, engineering and construction industries will have to learn to work in fundamentally new ways. BIM is a whole new paradigm.

BIM is …

"A digital representation of physical and functional characteristics of a facility… and a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition."

Both technology and work processes are essential

Pinning down what BIM really means is easier said than done. The above definition comes from the National Institute of Building Sciences in the United States, but many alternatives have been suggested. Where they all agree is that BIM is the marriage of a technology and a set of work processes. Different definitions put the emphasis on either one or the other but there is widespread acceptance that they would be incomplete without both parts. It's important to note that one coherent system of models means exactly that – collaboration within disciplines isn't enough. Everyone involved must work together. BIM is by its nature multidisciplinary.

BIM is a Sociotechnical System

A sociotechnical system is the combination of man-made technology and the social and institutional consequences of its implementation in society. Like the telephone network, it is not just a collection of wires; it contains associated behaviours, social norms, certain kinds of relationships and cultural institutions.

BIM is a 'system' because it could be described as a unified entity consisting of many interacting parts, some physical, others not. It is 'sociotechnical' because it has social components, complementing the technical core like the leaves on a tree. The social parts influence the evolution of the technical core through feedback loops.

BIM is a multilayered system

At the technical core of BIM is the software that enables 3D modelling and information management. Extensive use of the software eventually leads to a more complete understanding of the technical core.

After technology come the work practices. Moving beyond the software comes the realisation that there is a lot more to BIM than the technical core. This becomes clearer as the technical core begins to shape social practices by expanding possibilities. At first, this means more intense collaboration between different disciplines. Eventually this leads to the creation of a whole new institutional and cultural environment.

(Source: http://www.wsp-pb.com/en/Who-we-are/In-the-media/News/2013/What-is-BIM/)

5.3. Answer the following questions.

1. What does BIM acronym stand for?
2. Give the approximate definition of BIM-technology.
3. Why does BIM technology save money and time?
4. Why is BIM-technology considered as a social network?
5. What disciplines are included into multidisciplinary set of BIM-technology?

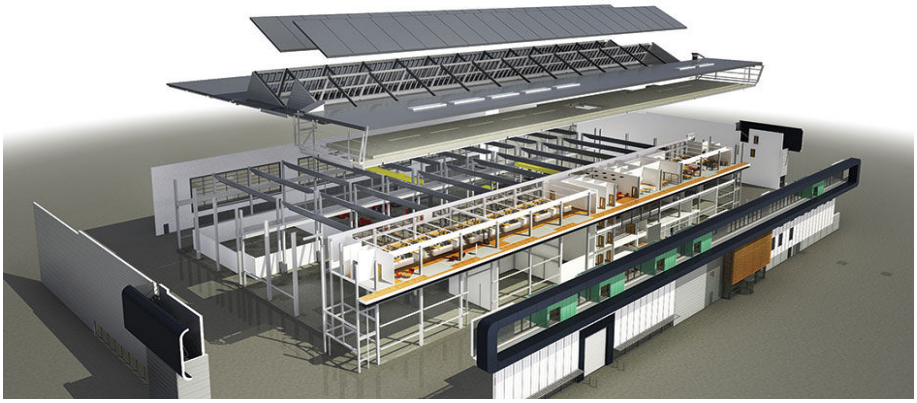
5.4. Say whether the following statements are true or false according to the text

#	Statements	True	False
1	BIM reduces time and cost while designing a project		
2	BIM offer a physical model of a building as well as representation		
3	All aspects in BIM are dealt with separately		

4.5. Give English equivalence to the following words and word combinations, mentioned in the text

Информационное моделирование проекта, экономия времени и денег, цифровое представление физических и функциональных характеристик, нехватка взаимодействия между разными специалистами.

Source:
http://www.archiexpo.com



English for IT Engineers

Методические указания для студентов бакалавров 1 курса дневного отделения направления 09.03.02 «Информационные системы и технологии»

Составитель: Абитов Р.Н.

Корректура автора

Издательство
Казанского государственного архитектурно-строительного университета

Подписано в печать 3.10.2016		Формат 60x84/16
Заказ №	Печать ризографическая	Усл.-печ л. 1,25
Тираж 70 экз	Бумага офсетная	Уч. изд. л. 1,25

Отпечатано в полиграфическом секторе
Издательства КГАСУ
420043, Казань, ул. Зеленая, д. 1.