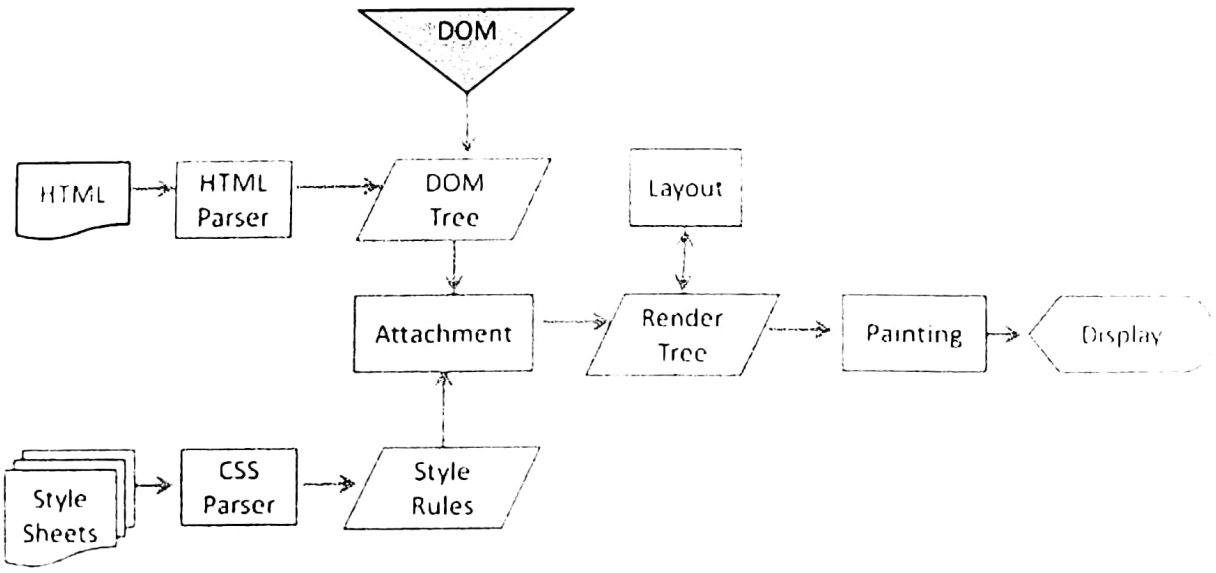
Function

The primary purpose of a web browser is to bring information resources to the user ("retrieval" or "fetching"), allowing them to view the information ("display", "rendering"), and then access other information ("navigation", "following links").

This process begins when the user inputs a Uniform Resource Locator (URL), for example http://geliyoo.com /, into the browser. The prefix of the URL, the Uniform Resource Identifier or URI, determines how the URL will be interpreted. The most commonly used kind of URI starts with http: and identifies a resource to be retrieved over the Hypertext Transfer Protocol (HTTP).[10] Many browsers also support a variety of other prefixes, such as https: for HTTPS, ftp: for the File Transfer Protocol, and file: for local files. Prefixes that the web browser cannot directly handle are often handed off to another application entirely. For example, mailto: URIs are usually passed to the user's default e-mail application, and news: URIs are passed to the user's default newsgroup reader.

In the case of http, https, file, and others, once the resource has been retrieved the web browser will display it. HTML and associated content (image files, formatting information such as CSS, etc.) is passed to the browser's layout engine to be transformed from markup to an interactive document, a process known as "rendering". Aside from HTML, web browsers can generally display any kind of content that can be part of a web page. Most browsers can display images, audio, video, and XML files, and often have plug-ins to support Flash applications and Java applets. Upon encountering a file of an unsupported type or a file that is set up to be downloaded rather than displayed, the browser prompts the user to save the file to disk.

Information resources may contain hyperlinks to other information resources. Each link contains the URI of a resource to go to. When a link is clicked, the browser navigates to the resource indicated by the link's target URI, and the process of bringing content to the user begins again.



UNIT 1. SOFTWARE USAGE

I. Make sure that you know the following words:

computer program, component ofcomputers, architecture, libraries and scripts, instructions, computer memory, machine language, individual processor, display of the computer system,programmers, operating system, peripherals, video games, compilers, databases, email filters

II. Learn the following terms:

- computersoftware - компьютерное программное обеспечение
- computerhardware - аппаратное обеспечение компьютера
- executable file - исполняемый файл
- binary values - двоичные значения
- storage location - ячейка памяти
- high-level programming languages – языки программирования высокого уровня
- low-level assembly language – низкоуровневый язык ассемблера
- general purpose computer - универсальная ЭВМ
- firmware - встроенное ПО
- device driver - драйвер устройства
- Graphical user interface - графический интерфейс пользователя
- To bundle with a computer – поставлять в комплекте с компьютером
- application software - прикладное ПО
- Office suite - офисный пакет, пакет приложений
- user-written software - ПО подготовленное пользователем
- To meet users needs - удовлетворять потребности пользователей
- spreadsheet template - шаблон электронной таблицы
- word processor - текстовый процессор
- to integrate into default - включать в пакет прикладных
- Application packages - программ по умолчанию

III. Read and translate the text:

Computer software, or simply **software** also known as computer programs, is the non-tangible component of computers. Computer software contrasts with computer hardware, which is the physical component of computers. Computer hardware and software require each other and neither can be realistically used without the other.

Computer software includes all computer programs regardless of their architecture; for example, executable files, libraries and scripts are computer software. Yet, it shares their mutual properties: software consists of clearly defined instructions that upon execution, instructs hardware to perform the tasks for which it is designed. Software is stored in computer memory and cannot be touched as a 3D model.

At the lowest level, executable code consists of machine language instructions specific to an individual processor – typically a central processing unit (CPU). A machine language consists of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. For example, an instruction may change the value stored in a particular storage location inside the computer – an effect that is not directly observable to the user. An instruction may also (indirectly) cause something to appear on a display of the computer system – a state change which should be visible to the user. The processor carries out the instructions in the order they are provided, unless it is instructed to "jump" to a different instruction, or interrupted.

Software is usually written in high-level programming languages that are easier and more efficient for humans to use (closer to natural language) than machine language. High-level languages are compiled or interpreted intomachinelanguage object code. Software may also be written in a low-level assembly language, essentially, a vaguely mnemonic representation of a machinelanguage using a natural language alphabet. Assembly language is converted into object code via an assembler.

Architecture.

Users often see things differently than programmers. People who use modern general purpose computers usually see three layers of software performing a variety of tasks: platform, application, and user software.

- Platform software: Platform software includes the firmware, device drivers, an operating system, and typically a graphical user interface which, in total, allow a user to interact with the computer and its peripherals (associated equipment). Platform software often comes bundled with the computer. On a PC one will usually have the ability to change the platform software.
- Application software: Application software or Applications are what most people think of when they think of software. Typical examples include office suites and video games. Application software is often purchased

separately from computer hardware. Sometimes applications are bundled withthe computer, but does not change the fact that they run as independent applications. Applications are usually independent programs from the operating system, though they are often tailored for specific platforms. Most users think of compilers, databases, and other "system software" as applications.

- User-written software: End-user development tailors systems to meet users' specific needs. User software includes spreadsheet templates and word processor templates. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is. Depending on how competently the user-written software has been integrated into default application packages, many users may not be aware of the distinction between the original packages, and what has been added by co-workers.

IV. Answer the following questions:

1. How could you define computer software?
2. What does computer software include? Give examples.
3. What does a machine language consist of?
4. What is the difference between high-level programming languages and machine language?
5. What layers of software are known to people who use modern general purpose computers?
6. What does platform software include?
7. What are typical examples of application software?
8. What does user-written software include?

V. Retell the text briefly using the new words and expressions from ex.II.

VI. Fill in the gaps with the words given below. Use the dictionary if necessary.

Software interfaces in practice

a) client, b) constants, c) functionality and stability, d) data types, e) exception, f) method signatures, g) implementation, h) interfaces, i) is only concerned, j) procedures, k) variables

A key principle of design is to prohibit access to all resources by default, allowing access only through well-defined entry points, i.e. (1) ... . Software interfaces provide access to computer resources (such as memory, CPU, storage, etc.) of the underlying computer system; direct access (i.e. not through well designed interfaces) to such resources by software can have major ramifications – sometimes disastrous ones – for (2) ... .

Interfaces between software components can provide: ... (3) ... (4) types of ...(5), ... (6) specifications and ... (7). Sometimes, public ... (8) are also defined as part of an interface.

The interface of a software module A is deliberately defined separately from the ... (9) of that module. The latter contains the actual code of the procedures and methods described in the interface, as well as other "private" variables, procedures, etc. Another software module B, for example the ... (10) to A that interacts with A is forced to do so only through the published interface. One practical advantage of this arrangement is that replacing the implementation of A by another implementation of the same interface is not relevant to B, which ... (11) with the specifications of the interface.

Grammar revision:

VII. Put the verbs in brackets into the right tense and voice.

The ways that web browser makers fund their development costs ... (change) over time. The first web browser, World Wide Web, ... (be) a research project. Netscape Navigator ... originally ... (sell) commercially, as was Opera; Netscape no longer exists and ... (replace) with the free Firefox, while Opera is now downloadable free of charge.

Internet Explorer, on the other hand, was from its first release always included with the Windows operating system (and furthermore was downloadable for no extra charge), and therefore it ... (fund) partly by the sales of Windows to computer manufacturers and direct to users. Internet Explorer also ... (use) to be available for the Mac. It is likely that releasing IE for the Mac was part of Microsoft's overall strategy to fight threats to its quasi-monopoly platform dominance – threats such as web standards and Java – by making some web developers, or at least their managers, ... (assume) that there was "no need" to develop for anything other than Internet Explorer (an assumption that later proved to be badly mistaken, with the rise of Firefox and Chrome). In this respect, IE may ... (contribute) to Windows and Microsoft applications sales in another way, through tricking organizations into inadvertent "lock-in" into the Microsoft platform.

Safari and Mobile Safari ... (be) likewise always included with OS X and iOS respectively, so, similarly, they were originally funded by sales of Apple computers and mobile devices, and ... (form) part of the overall Apple experience to customers.

VIII. Look through the text. Make a short summary of it:

Software vs Hardware

Computer software is a program that tells a computer what to do. These instructions might be internal commands, such as updating the system clock, or a response to external input received from the keyboard or mouse. Though there are

many different types of software made both with open source and proprietary standards, the programming mostly comes down to a few basic rules.

The fundamental difference between hardware and software is that the first is tangible while the second is not. Hardware is the machine itself and does all of the physical work, while software tells the various hardware components what to do and how to interact with each other. This makes it possible for computers to adapt to new tasks or to install new hardware. While hardware includes things like monitors, Central Processing Units (CPUs), keyboards, and mice; software includes things like word processing programs, operating systems, and games.

There are two main types of computer software: system and application. The first type is used just to run the hardware, while the second is used to do other things. The main types of system software are operating systems, like Windows™ OS X, or Linux; and drivers, which are programs that allow a computer to interact with other devices, like printers and video cards. There are many different types of application software, including games, media players, word processors, anti-virus programs, and applications for making new programs.

Hardware only understands the two basic concepts, on and off, which are represented as 1s and 0s in binary language. Software acts as the translator between human languages and binary, which makes it possible for the hardware to understand the instructions being fed into it. Programmers write commands called source code in programming languages that are similar to what someone might use in everyday speech. Another program called a compiler is then used to transform the source code commands into binary. The result is an executable computer program.

Programmers create either open source or proprietary computer software. The first type can be edited and adapted by users, while the second is protected and not intended to be edited by people outside the company that sells it. While open source programs are usually free, proprietary ones are licensed to distributors and must be paid for.

Both types generally have a comparable quality of programming, but fixes for bugs generally come faster for open source products than for proprietary ones. Also, some people prefer to use open source applications because they can feel a sense of ownership in the end project or feel that computer software companies create monopolies and want alternatives. Others prefer closed source programs because they are sometimes more stable or be less vulnerable to hackers, often come in suites, and come with customer service from the company.

IX. Translate the text in written form:

USER INTERFACES

Cheaper and more powerful personal computers are making it possible to perform processor-intensive tasks on the desktop. Breakthroughs in technology, such as speech recognition, are enabling new ways of interacting with computers. And the convergence of personal computers and consumer electronics devices is broadening the base of computer users and placing a new emphasis on ease of use.

Together, these developments will drive the industry in the next few years to build the first completely new interfaces.

True, it's unlikely that you'll be ready to toss out the keyboard and mouse any time soon. Indeed, a whole cottage industry – inspired by the hyperlinked design of the World Wide Web – has sprung up to improve today's graphical user interface. Companies are developing products that organize information graphically in more intuitive ways. XML-based formats enable users to view content, including local and network files, within a single browser interface. But it is more dramatic innovations such as speech recognition that are poised to shake up interface design.

Speech will become a major component of user interfaces, an applications will be completely redesigned to incorporate speech input. Palm-size and handheld PCs, with their cramped keyboards and basic handwriting recognition, will benefit from speech technology.

Though speech recognition may never be a complete replacement for other input devices, future interfaces will offer a combination of input types, a concept known as multimodal input. A mouse is a very efficient device for desktop navigation, for example, but not for changing the style of paragraph. By using both a mouse and speech input, a user can first point to the appropriate paragraph and then say to the computer, 'Make it bold'. Of course, multimodal interfaces will involve more than just traditional input devices and speech recognition. Eventually, most PCs will also have handwriting recognition, text to speech (TTS), the ability to recognize faces or gestures, and even the ability to observe their surroundings.