КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ МЕЖДУНАРОДНЫХ ОТНОШЕНИЙ,
ИСТОРИИ И ВОСТОКОВЕДЕНИЯ
Кафедра английского языка в сфере высоких технологий

Э.Х. Шамсутдинова

ENGLISH FOR PROGRAMMERS

Учебное пособие

Казань – 2016

УДК
ББК

Шамсутдинова Э.Х.
English for programmers: Учебное пособие/Э.Х.Шамсутдинова.-
Казань: Казанский университет, 2016. – 93 с.

Принято на заседании кафедры английского языка в сфере
высоких технологий
Протокол № 11 от 29 июня 2016 года

Рецензенты:
канд. фил. наук, доц. КФУ Д.Ф. Хакимзянова
канд. фил. наук, доц. КНИТУ им. А.Н.Туполева Е.В.Мусина

Данное учебное пособие по английскому языку предназначено для студентов, обучающихся по направлениям 01.03.02 «Прикладная математика и информатика» и 09.03.04 «Программная инженерия» и содержит материалы, дополняющие основной курс английского языка. Пособие может быть использовано как для аудиторной работы, так и для самостоятельной работы студентов.

Предисловие

Настоящее учебное пособие предназначено для занятий со студентами первого и второго курсов Института вычислительной математики Казанского (Приволжского) федерального университета, по 01.03.02 «Прикладная математика и информатика» и 09.03.04 «Программная инженерия». Пособие разработано с учетом требований государственного стандарта высшего профессионального образования и предназначено для студентов, продолжающих изучение английского языка на базе программы средней школы. Целью настоящего пособия является развитие навыков чтения аутентичных текстов, изучение и закрепление лексики, приобретение учащимися навыков правильного понимания и перевода оригинального текста по специальности, развитие навыков монологической и диалогической речи в сфере профессиональной коммуникации. Учебное пособие состоит из трех основных разделов, разделенных на два подраздела, глоссария, а также приложения, включающего в себя научные тексты для реферирования. В учебное пособие включены оригинальные тексты, опубликованные в зарубежных научных и научно-популярных изданиях и сайтах сети Интернет. Тексты снабжены упражнениями по изучению и закреплению лексики, развитию навыков речи. Все разделы по структуре идентичны, даны ясные формулировки заданий.

Content

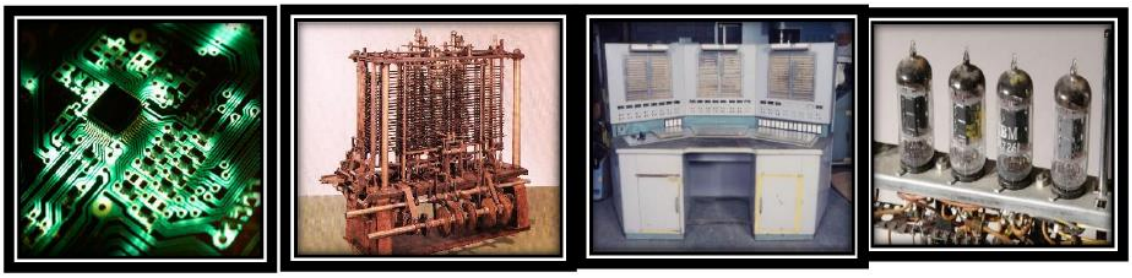
Unit 1 History of Computers

Lead-in

Match the words with pictures and discuss in pairs, using the table: Which of these terms do you know? Can you guess their role in evaluation of computers? Can you put these inventions in chronological order?

Express opinions *I think…; I believe…; In my opinion…; To my mind...; To me…*	***Agreeing*** *I couldn't agree more…; I quite agree with you…; That's true...; I partly agree…*
Asking for opinion *What do you think about…? What's your opinion…? Don't you agree…?*	***Disagreeing*** *I don't think so…; I don't agree…; Perhaps you are right, but on the other hand…; That's not quite the way I see it…; I see what you mean, but…*

a.Relay computer b. Analytical engine c. Abacus
d. Semiconductor chip e. Digital computer f. Hardware
g. Vacuum tube h. Software



1 2 3 4

5 6 7 8

Reading TEXT 1 A Brief History of the Computer

Application [ˌæplɪ'keɪʃ(ə)n]-приложение; применение

Software['sɔftweə]– программное обеспечение (ПО)

Programming language -язык программирования

Abacus['æbəkəs]- счеты \ мн.ч. abaci['æbəsaɪ], abacuses

To design[dɪ'zaɪn] - конструировать; разрабатывать (что-л. для какой-л. цели); предназначать

Device[dɪ'vaɪs]- устройство, приспособление; аппарат, машина, прибор

Input device['ɪnput]- устройство ввода

Storage['stɔ:rɪʤ]–хранение; память, запоминающее устройство

To process['prəuses]- обрабатывать (данные, информацию)

Processor['prəusesə]- процессор

Unit['ju:nɪt]- устройство; узел; блок; прибор; звено; элемент

Output device['autput] - устройство вывода

To utilize['ju:tɪlaɪz] - использовать, расходовать, употреблять

Punched card[pʌntʃt] - перфокарта; перфорированная карта

Circuit['sɜ:kɪt] - цепь, контур; схема

Calculation[ˌkælkju'leɪʃ(ə)n] - вычисление; подсчёт, расчёт

Analytical engine - аналитическая машина, устройство

Digital computer['dɪʤɪt(ə)l]- цифровой компьютер, цифровая вычислительная машина

Vacuum tube['vækju:m t(j)u:b]- вакуумная трубка

To crack a code[kræk] [kəud] - взломать код

Maintenance ['meɪnt(ə)nən(t)s] - содержание и техническое обслуживание, уход; текущий ремонт

To break (broke, broken) down - потерпеть неудачу; ломаться

Decimal digit['desɪm(ə)l 'dɪʤɪt]- десятичная цифра

Multiply ['mʌltɪplaɪ] – умножать

Multiplication[ˌmʌltɪplɪ'keɪʃ(ə)n] - умножение

To store[stɔ:] - запоминать, хранить (данные, информацию)

Relay computer['ri:leɪ] , [ri:'leɪ] - релейная вычислительная машина

To consume[kən'sju:m] – потреблять, расходовать

Watt[wɔt] - ватт

Overheating – перегревание

To perform a task [pə'fɔ:m] – выполнить задачу

To rewire [ˌriːˈwaɪə]- менять электропроводку, перемонтировать схему

To pioneer[ˌpaɪəˈnɪə] - разрабатывать, открывать, прокладывать путь

To give (gave; given) birth to- дать начало (чему-л); приводить к возникновению; рождать

General-purpose computer - универсальная вычислительная машина

Researcher [rɪˈsɜːʧə]- исследователь, ученый

Random access memory (RAM)[ræm]- оперативная память, оперативное запоминающее устройство

Instruction [ɪnˈstrʌkʃ(ə)n] - команда

Hardware[ˈhɑːdweə] - "железо", аппаратные средства компьютера, техническое обеспечение компьютера (электронные, механические и иные "физические" компоненты компьютера)

Reliability [rɪˌlaɪəˈbɪlətɪ] - надежность

Capability[ˌkeɪpəˈbɪlətɪ] - мощность; производительность

Magnetic core memory - память [запоминающее устройство] на магнитных сердечниках

Transistor - Circuit- транзисторная схема, схема на транзисторах

Support personnel - вспомогательный персонал

Semiconductor chip - полупроводниковый чип, кристалл

To make an advance – добиться прогресса

Memory chip – микросхема памяти

Microprocessor [ˌmaɪkrə(u)ˈprəusesə] - микропроцессор

To handle [ˈhændl] - обрабатывать

Bit [bɪt] - бит

At a time - за раз, разом, единовременно

Pre-Reading Tasks

1. Before reading the text, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

- What role do you think the computers play in a modern society?
- What gave birth to computers?
- Can you name the person who developed the first computer?

2. Next, reorder the words in the mystery questions below:

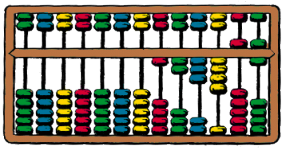
- digital Clifford Berry **What** did John Atanasoff in the first and his assistant late computer 1930's build?
- companies **What** in the very PCs introduced successful 1970's?

•early image of in the 50s**What**engineering two important changed the computer discoveries field theelectronic?

Reading Tasks

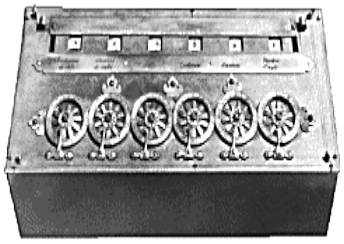
Nowread the article to check your answers and do the tasks which follow.

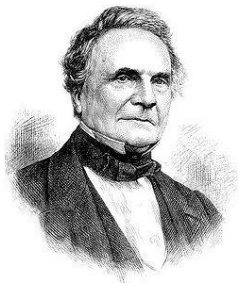
Computers and computer applications are on almost every aspect of our daily lives. As like many ordinary objects around us, we may need clearer understanding of what they are. You may ask "What is a computer?" or "What is a software", or "What is a programming language?" First, let's examine the history.



1. The history of computers starts out about 2000 years ago in Babylonia (Mesopotamia), at the birth of the abacus.

2. In 1642 Blaise Pascal (a famous French mathematician) invented an adding machine based on mechanical gears in which numbers were represented by the cogs on the wheels.

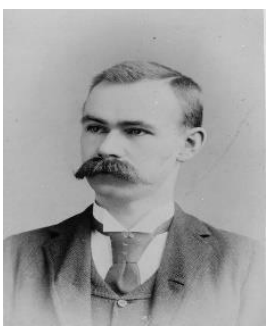




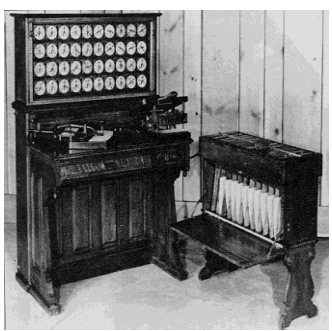
3. Englishman, Charles Babbage, invented in the 1830's a "Difference Engine" made out of brass and pewter rods and gears, and also designed a further device which he called an "Analytical Engine". His design contained the five key characteristics of modern computers:

- An input device
- Storage for numbers waiting to be processed
- A processor or number calculator
- A unit to control the task and the sequence of its calculations
- An output device

Augusta Ada Byron was an associate of Babbage who has become known as the first computer programmer.



4. An American, Herman Hollerith, developed (around 1890)



the first electrically driven device. It utilized punched cards and metal rods which passed through the holes to close an electrical circuit and thus cause a counter to advance. This machine was able to complete the calculation of the 1890 U.S. census in 6 weeks compared with 7 1/2 years for the 1880 census which was manually counted.

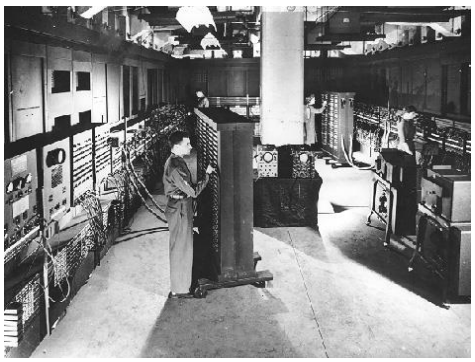
5. In 1936 Howard Aiken of Harvard University convinced Thomas Watson of IBM to invest $1 million in the development of an electromechanical version of Babbage's analytical engine. The Harvard Mark 1 was completed in 1944 and was 8 feet high and 55 feet long.

6. At about the same time (the late 1930's) John Atanasoff of Iowa State University and his assistant Clifford Berry built the first digital computer that worked electronically, the ABC (Atanasoff-Berry Computer). This machine was basically a small calculator.

7. In 1943, as part of the British war effort, a series of vacuum tube based computers (named Colossus) were developed to crack German secret codes. The Colossus Mark 2 series (pictured) consisted of 2400 vacuum tubes.

8. John Mauchly and J. Presper Eckert of the University of Pennsylvania developed these ideas further by proposing a huge machine consisting of 18,000 vacuum tubes. ENIAC (Electronic Numerical Integrator And Computer) was born in 1946. It was a huge machine with a huge power requirement and two major disadvantages. Maintenance was extremely difficult as the tubes broke down regularly and had to be replaced, and also there was a big problem with overheating. The most important limitation, however, was that every time a new task needed to be performed the machine need to be rewired.

The size of ENIACs numerical "word" was 10 decimal digits, and it could multiply two of these numbers at a rate of 300 per second, by finding the value of each product from a multiplication table stored in its memory. ENIAC was therefore about 1,000 times faster than the previous generation of relay computers. ENIAC used about 1,800 square feet of floor space, and consumed about 180,000 watts of electrical power.



In other words, programming was carried out with a soldering iron.

9. In the late 1940's John von Neumann (at the time a special consultant to the ENIAC team) developed the EDVAC (Electronic Discrete VariableAutomatic Computer) which pioneered the "stored program concept". This allowed programs to be read into the computer and so gave birth to the age of general-purpose computers.

10. University of Manchester researchers Frederic Williams, Tom Kilburn, and Geoff Toothill develop the Small-Scale Experimental Machine (SSEM), better known as the Manchester "Baby." The Baby was built to test a new memory technology developed by Williams and Kilburn - soon known as the Williams Tube – which was the first electronic random access memory for computers. The first program, consisting of seventeen instructions and written by Kilburn, ran on June 21st, 1948. This was the first program to ever run on an electronic stored-program computer.

11. Early in the 50s two important engineering discoveries changed the image of the electronic - computer field, from one of fast but unreliable hardware to an image of relatively high reliability and even more capability. These discoveries were the magnetic core memory and the Transistor – CircuitElement.

These technical discoveries quickly found their way into new models of digital computers. RAM capacities increased from 8,000 to 64,000 words in commercially available machines by the 1960s, with access times of 2 to 3 MS (Milliseconds). These machines were very expensive to purchase or even to rent and were particularly expensive to operate because of the cost of expanding programming. Such computers were mostly found in large computer centers operated by industry, government, and private laboratories - staffed with many programmers and support personnel. This situation led to modes of operation enabling the sharing of the high potential available.

12. Many companies, such as Apple Computer and Radio Shack, introduced very successful PCs in the 1970's, encouraged in part by a fad in computer (video) games. In the 1980's some friction occurred in the crowded PC field, with Apple and IBM keeping strong. In the manufacturing of semiconductor chips, the Intel and Motorola Corporations were very competitive into the 1980s, although Japanese firms were making strong economic advances, especially in the area of memory chips. By the

late 1980s, some personal computers were run by microprocessors that, handling 32 bits of data at a time, could process about 4,000,000 instructions per second.

Vocabulary Practice

1. Match the highlighted words from the text with their definitions:

1) a small machine that has a particular purpose or is part of a larger machine

2) any device used to send data from a computer to another device or user

3) to decide how something will be made, including how it will work and what it will look like, and often to make drawings of it

4) to use something, especially for a practical purpose,to make use of smth

5) the complete path of wires and equipment along which an electric current flows

6) the act or process of using numbers to find out an amount

7) a part of a computer that controls all the other parts of the system

8) a program designed to do a particular job; a piece of software\ the practical use of something, especially a theory, discovery, etc.

9) the programs used to operate a computer

10) an object or a piece of equipment that has been designed to do a particular job

11) a hardware or peripheral device used to send data to a computer

12) the part of a computer that stores information for subsequent use or retrieval.

13) to perform a series of operations on data in a computer

14) a computer that works with numbers that are represented by the digits 0 and 1

15) a glass tube that was used in the past in computers, televisions, etc., to control the flow of electricity

16) to break through (as a barrier) so as to get access to confidential information

17) a special language that programmers use to develop software programs, scripts, or other sets of instructions for computers to execute.

18) the act of keeping something in good condition by checking or repairing it regularly

19) a unit for measuring electrical power

20) the process of becoming too hot

21) to carry out

22) to begin something new or take part in the early development of something

23) to be the source of

24) a person who studies something carefully and tries to discover new facts about it

25) to stop working because of a fault, to fail

26) the act or process of multiplying

27) to keep information or facts

28) to use something, especially fuel, energy or time

29) the physical components of a computer

30) the quality or state of being fit to be trusted or relied on

31) a type of storage for computer systems that makes it possible to access data very quickly in random order.

32) an order given to a computer processor by a computer program.

33) potential for an indicated use or deployment

34) to progress or to develop in a particular activity or area of understanding

35) an integrated circuit made out of millions of capacitors and transistors that can store data or can be used to process code.

36) a piece of electronic equipment inside a computer that makes it work.

37) to deal successfully with a large amount of information

38) the smallest unit of information used by a computer

39) during one period of time without stopping

2. Translate the sentences from Russian into English

1. Интегральная схема представляет собой электронную схему, вытравленную на кремниевом кристалле.

2. Компьютер, разработанный Стивом Джобсом и Стивом Возняком, пользовался популярностью в конце 70-х – начале 80-х.

3. Персональные компьютеры обязательно должны иметь монитор и ряд других периферийных устройств.

4. Персональные компьютеры имеют следующие характеристики: малые размеры, отсутствие необходимости в обслуживании, низкая цена и универсальность.

5. Компьютер изначально был задуман для обработки информации на компьютере.

6. Взломать WhatsApp реальность или фантастика?

7. Периферийные устройства компьютеров делятся на устройства ввода, устройства вывода и внешние запоминающие устройства.

8. Программное обеспечение помогает не только решить проблемы совместимости, но и обеспечивает мощную функциональность и гибкость.

9. Существует около 8 500 языков программирования, однако, несмотря на такое разнообразие, число языков, на которых пишет большинство программистов, едва достигает десяти.

10. Персональный компьютер остается основной платформой для игр, говорится в докладе организации PC Gaming Alliance (PCGA).

11. Глава компании Microsoft Стив Баллмер представил публике компьютер-планшет, созданный совместно с Hewlett-Packard.

12. Основное отличие Midori заключается в том, что она создается как интернет-ориентированная операционная система, которая сделает программы независимыми от "железа", так как будет работать с удаленными приложениями.

13. При создании Windows Vista использовался целый ряд инноваций и технологий, направленных на ограничение возможностей злоумышленников по "взлому" компьютеров.

14. В США создано программное обеспечение, позволяющее компаниям следить за каждым нажатием клавиши на клавиатурах компьютеров своих сотрудников.

15. Команда исследователей из Республики Корея разработали трехмерный полупроводниковый чип, которая употребляет меньшее количество энергии при меньших размерах и действует, как мозг.

3. Complete these sentences using the given words. Make any changes to the phrases that are necessary.

Sophisticated/tremendous /process information / a field of research / perform a calculation /solve a problem more efficiently / capability / primitive quantum computer

Scientists and engineers from the Universities of Bristol and Western Australia have developed how to efficiently simulate a "quantum walk" on a new design for a primitive quantum computer.

Quantum computers have significant potential to open entirely new directions for _____ and to overhaul the way that we think about and use the science of computation. Modern computers already play a huge role in society - they routinely _____ and process vast amounts of data and _____at an incredible rate. However, there are some problems that they just cannot solve in a useful amount of time, no matter how fast they become. The concept of a quantum computer aims to address this, exploring uncharted computation and solving at least some of these problems that classical computers cannot.

The study published today in Nature Communications, reports strong evidence that with this method something meaningful can already be seen with _____that cannot be seen with a classical computer. The very first steps towards this have been implemented in the lab in Bristol.

Dr Ashley Montanaro, Lecturer in Applied Mathematics and EPSRC Fellow from the University of Bristol's School of Mathematics, said: "A quantum computer is a machine designed to use quantum mechanics to _____ than any possible classical computer.

Building a large-scale quantum computer is one of the biggest engineering challenges today. There's a growing worldwide effort to develop one and it needs substantial effort from a wide range of expertise -- including as part of the UK National Quantum Technologies Programme (UKNQT). The results could be_____, offering fast and cheap ways to design new materials and new pharmaceuticals.

But there is _____emerging now that can help accelerate understanding how quantum computers will work and how users can apply them. Examining the power of smaller, more primitive designs for quantum computers indicates that sooner than we thought, quantum machines could outperform the _____ of classical computing for very specific tasks -- "Boson Sampling" is a recent example that is driven by what is experimentally available very soon.

Big questions researchers face include what these primitive quantum processors can do that is useful to someone and how _____ they need to be. The results published in

today's paper help to answer this question, by looking at how to simulate particular kinds of a phenomenon called the quantum walk.

Speaking Practice

1. Work in small groupsand discuss the following questions using useful expressions on p. 5:

❖ When was the word "computer" first used?

❖ What was considered to be the first electro-mechanical binary programmable computer?

❖ What do you know about the early British computer known as the EDSAC?

❖ Where was UNIVAC 1101 or ERA 1101 to be the first computer that was capable of storing and running a program from memory designed?

❖ When did Intel introduce the first microprocessor, the Intel 4004?

2. Think of one of the latest inventions in the field of computing and IT. Prepare a short presentation providing details about them, such as key dates, inventors, areas of application, usefulness, and some other interesting information. Use the Internet if you need to check the information. Present your information to the rest of the class. Use the following useful phrases:

Welcoming

- *Good morning, ladies and gentlemen…*
- *Introducing yourself*
- *Let me introduce myself; my name is …..(and I am responsible for …)*

Introducing your presentation

- *The purpose of today's presentation is to … .*
- *In today's presentation I'd like to … show you … . / explain to you how … / give you an overview of … .*
- *In today's presentation I'd like to cover three points:*
- *firstly, … , secondly … , and finally … .*

Starting the presentation

- *To begin with … .*
- *Let's start with / start by looking at … .*

Closing a section of the presentation

- *So, that concludes* [title of the section] … .
- *So, that's an overview of … .*

- *I think that just about covers … .*

Beginning a new section of the presentation

- *Now I'd like to move on to … .*

Concluding and summarising the presentation

- *Well, that brings us to the end of the final section. Now, I'd like to summarise by … .*
- *That concludes my presentation. Now, if I can just summarise the main points.*
- *That's an overview of … . Now, just to summarise, let's quickly look at the main points again.*

Finishing and thanking

- *Thank you for your attention.*
- *That brings the presentation to an end.*

Inviting questions

- *If anyone has any questions, I'll be pleased to answer them.*

Referring to a previous point made

- *As I mentioned earlier … .*
- *As we saw earlier … .*

ReadingTEXT 2The Evolution of Computers

Laptop [ˈlæptɔp] - лэптоп, небольшой портативный компьютер

Tablet [ˈtæblət] - планшет

To bear (bore, born) resemblance to smth- иметь сходство

Unsophisticated [ˌʌnsəˈfɪstɪkeɪtɪd] -бесхитростный, простой, незамысловатый

Switch [swɪʧ] - переключатель, коммутатор

Amplifier [ˈæmplɪfaɪə] - усилитель

Cooling unit - охлаждающее устройство

In lieu of [ljuː] - вместо

In terms of–относительно, в соответствии с, с точки зрения

Integrated circuit- интегральная схема, ИС

Silicon chip - кремниевый кристалл, кремниевый чип

To herald[ˈherəld] - объявлять, уведомлять; предвещать, предрекать

Lead (led, led) to the dawn of smth –приводить к появлению чего-л.

Kit [kɪt] – комплект, набор

Assembly[əˈsemblɪ]- агрегат; комплект; блок; сборка, монтаж

Hand-held device - карманное устройство

Graphical user interface- графический интерфейс пользователя

Artificial Intelligence – искусственный интеллект

Natural Language - естественный язык

Goal [gəul] –задача, цель

Pre-Reading Tasks

1. Before reading the text on the following page, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

- What do such abbreviations as GUI, AI, and IC stand for?
- What do 1G, 2G, 3G and 4G mean?
- How many generations of computers do you know?

▪ **Next, reorder the words in the mystery questions below:**

- machine generation of language **What** used a computers first basic very?
- Improvements **What** kind through of did computer go memory and storage?
- 5G **What** the main were of the machines requirements?

Reading Tasks

Now read the article to check your answers and do the tasks which follow.

Computers in the form of personal desktop computers, laptops and tablets have become such an important part of everyday living that it can be difficult to remember a time when they did not exist. In reality, computers as they are known and used today are still relatively new. Although computers have technically been in use since the abacus approximately 5000 years ago, it is modern computers that have had the greatest and most profound effect on society. The first full-sized digital computer in history was developed in 1944. Called the Mark I, this computer was used only for calculations and weighed five tons. Despite its size and limited ability it was the first of many that would start off generations of computer development and growth.

First Generation Computers

First generation computers bore little resemblance to computers of today, either in appearance or performance. The first generation of computers took place from 1940 to 1956 and was extremely large in size. The inner workings of the computers at that time were unsophisticated. These early machines required magnetic drums for memory and vacuum

tubes that worked as switches and amplifiers. It was the vacuum tubes that were mainly responsible for the large size of the machines and the massive amounts of heat that they released. These computers produced so much heat that they regularly overheated despite large cooling units. First generation computers also used a very basic programming language that is referred to as machine language.

Second Generation Computers

The second generation (from 1956 to 1963) of computers managed to do away with vacuum tubes in lieu of transistors. This allowed them to use less electricity and generate less heat. Second generation computers were also significantly faster than their predecessors. Another significant change was in the size of the computers, which were smaller. Transistor computers also developed core memory which they used alongside magnetic storage.

Third Generation Computers

From 1964 to 1971 computers went through a significant change in terms of speed, courtesy of integrated circuits. Integrated circuits, or semiconductor chips, were large numbers of miniature transistors packed on silicon chips. This not only increased the speed of computers but also made them smaller, more powerful, and less expensive. In addition, instead of the punch cards and the printouts of previous systems, keyboards and monitors were now allowing people to interact with computing machines.

Fourth Generation Computers

The changes with the greatest impact occurred in the years from 1971 to 2010. During this time technology developed to a point where manufacturers could place millions of transistors on a single circuit chip. This was called monolithic integrated circuit technology. It also heralded the invention of the Intel 4004 chip which was the first microprocessor to become commercially available in 1971. This invention led to the dawn of the personal computer industry. By the mid-70s, personal computers such as the Altair 8800 became available to the public in the form of kits and required assembly. By the late 70s and early 80s assembled personal computers for home use, such as the Commodore Pet, Apple II and the first IBM computer, were making their way onto the market. Personal computers and their ability to create networks eventually would lead to the Internet in the early 1990s. The fourth generation of computers also saw the creation of even smaller computers including laptops and hand-held devices. Graphical user interface, or GUI, was also invented during this time. Computer

memory and storage also went through major improvements, with an increase in storage capacity and speed.

The Fifth Generation (the future)

The "fifth generation" of computers was defined by the Japanese government in 1980 when they unveiled an optimistic ten-year plan to produce the next generation of computers. This was an interesting plan for two reasons. Firstly, it is not at all really clear what the fourth generation is, or even whether the third generation had finished yet. Secondly, it was an attempt to define a generation of computers before they had come into existence. The main requirements of the 5G machines were that they incorporate the features of Artificial Intelligence, Expert Systems, and Natural Language**.** The goal was to produce machines that are capable of performing tasks in similar ways to humans, are capable of learning, and are capable of interacting with humans in natural language and preferably using both speech input (speech recognition) and speech output (speech synthesis). Such goals are obviously of interest to linguists and speech scientists as natural language and speech processing are key components of the definition. As you may have guessed, this goal has not yet been fully realized, although significant progress has been made towards various aspects of these goals.

Vocabulary Practice

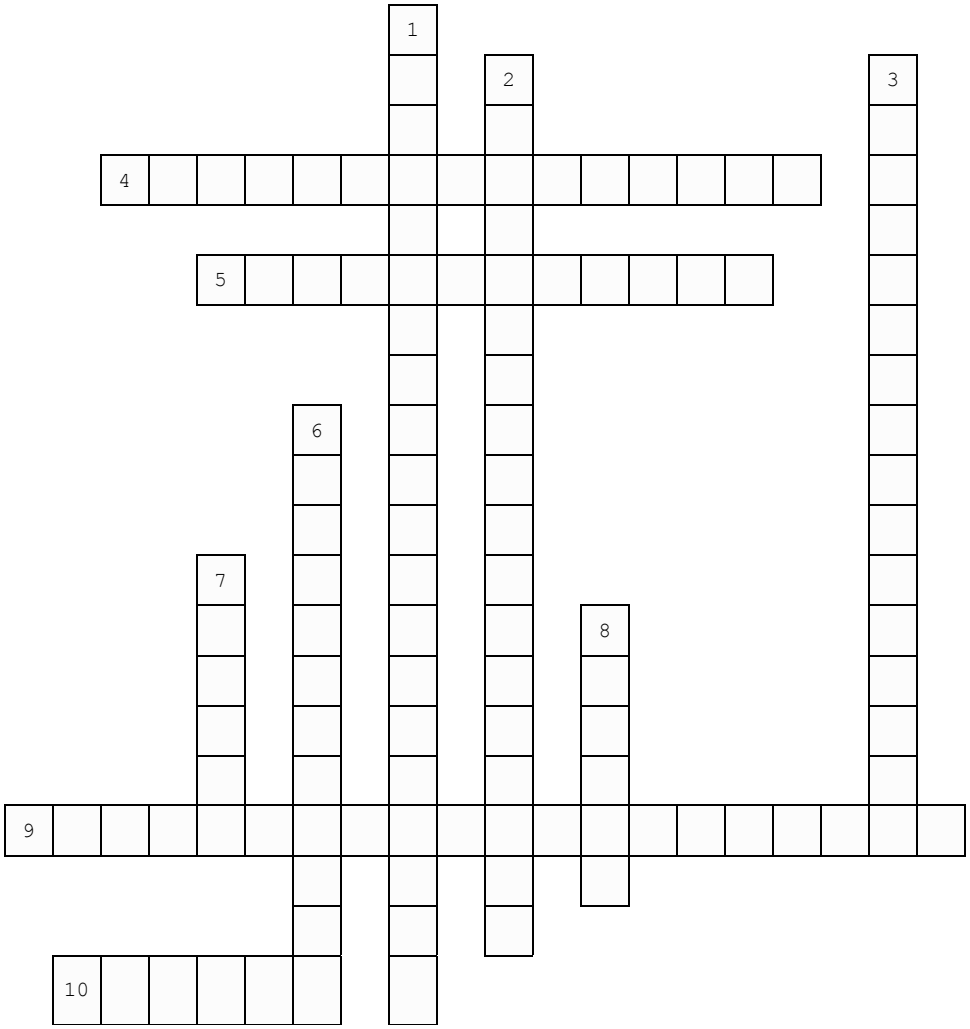
1. Match the words and expressions from the text to their meaning

A	B
1. bear resemblance to	a. aim
2. goal	b. English, Spanish, Russian
3. herald	c. instead of
4. in lieu of	d. look a lot like someone else
5. in terms of	e. of with regard to, concerning
6. Natural Language	f. signal the approach of

2. Try to explain the highlighted words and phrases in Text 2.

3. What does the writer mean by the underlined phrases? Discuss in pairs.

4. Complete the crossword.



Across

4. simple and basic, not complicated

5. a device that removes the waste heat produced by computer components

9. a way of giving instructions to a computer using things that can be seen on the screen such as symbols and menus

10. portable and compact personal computer with the same capabilities as a desktop computer

Down

1. an area of study concerned with making computers copy intelligent human behaviour

2. a small microchip that contains a large number of electrical connections and performs the same function as a larger circuit made from separate parts

3. any portable unit that can be carried and held in one's palm

6. a very small piece of silicon used to carry a complicated electronic circuit

7. an electrical component which can break an electrical circuit, interrupting the current or diverting it from one conductor to another

8. a wireless touch screen personal computer that is smaller than a notebook but larger than a smartphone

Speaking practice

1. Paraphrase the following quotations. Which one do you agree with the most? Why? Discuss in small groups.

"To be human is to be 'a' human, a specific person with a life history and idiosyncrasy and point of view; artificial intelligence suggests that the line between intelligent machines and people blurs most when a puree is made of that identity."

"Computers are magnificent tools for the realization of our dreams, but no machine can replace the human spark of spirit, compassion, love, and understanding."

2. Look at these three newspaper headlines. What do you think the story is behind each one? Discuss your ideas with a partner.

- *HP Inc lays the foundations for a digital future*
- *Intel pushes business benefits of 4th generation ultrabooks*
- *Board presentations on IT risk: Don't make these five mistake*

---

3. Can you match the years with the events? Compare your answers with the rest of the class. Try to explain your choice.

2007 2010 2014 2015

- Apple introduced Apple Watches
- University of Michigan Micro Mote is Complete
- Introduction of the iPhone 4 with retina display
- The Amazon Kindle is released

4. **Imagine that you are going to pass an examination to foreign university and you have to make a review on the article (p. 22) Get acquainted with useful set expressions to fulfill this task successfully.**

Stating your own position on a topic or subject

- The aim of this paper/essay is to…
- The argument in this paper...

- The perspective presented here is…

Stating the view of another person

- Smith claims that…
- Smith's argument is that…
- Smith's conclusion is that…
- According to Smith…
- From Smith's point of view/perspective...
- The point of Smith's article/paper/book is that…
- The substance of Smith's article/paper/book is that…
- Smith's work/data leads him to conclude…
- Some theorists, such as Smith, argue that…
- It is argued by theorists, such as Jones (2009) and Smith (2010), that…

Attributing a view to another person (when you are not quite sure)

- Smith's claim seems to be that…
- Smith seems to be claiming…
- Smith's argument seems to be that…
- The point of Smith's argument appears to be that…

Drawing a conclusion using the work of others

- Using Smith's work it is possible to show that…
- From Smith's work it can be determined that…
- One possible consequence of Smith's work is that…
- Developing Smith's work to its logical conclusion shows that...
- When Smith's argument is analysed it can be seen that…
- Analysis of Smith's data demonstrates that…

Disagreeing with the views of others

- The argument advanced here is opposed to that of Smith…
- Problems arise in Smith's work [when it is noted that]…
- Smith's argument/data is flawed because…
- Contrary to Smith's argument…
- In contrast to Smith's argument…
- Smith's data/argument/conclusion do/does not follow because…
- It does not seem to follow from Smith's argument that…

Agreeing with the views of others

- As Smith argues…
- This is also Smith's view…
- Following from Smith's argument…

- Smith's view is persuasive because…
- Smith is right in so far as...
- Not unlike Smith, I am suggesting/arguing...
- Along the lines of Smith, I argue…

Pointing out assumptions

- This assumes that…
- Smith assumes that…
- Smith's assumption is that….
- The point being assumed here is that…
- Smith's view depends on the assumption that…

UNIT 1 REVISE AND CHECK

Give the definitions and translations to the following words:

Application
Artificial Intelligence
Bit
Calculation
Capability
Cooling unit
Digital computer
General-purpose computer
Graphical user interface
Hand-held device
Hardware
Input device
Instruction
Integrated Circuit
Laptop
Memory chip
Microprocessor
Natural Language
Output device
Processor
Programming language
Random access memory
Reliability
Researcher
Semiconductor chip
Silicon chip
Software
Storage
Tablet
Vacuum tube

Can you answer these questions?

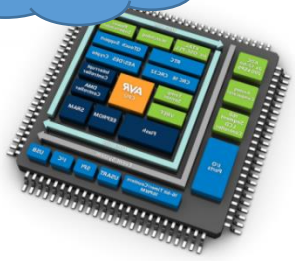
- Who first invented the computer?
- Who created a machine called The Analytical Engine?
- What device gave birth to the development of digital computer?
- What are the main features of 1Gcomputers?
- What are the main features of 2Gcomputers?
- What are the main features of 3Gcomputers?
- What are the main features of4G computers?
- What computers are related to the 5G ones?
- What are the prospects of AI nowadays?

Unit 2 Computer system

Lead-in

Discuss in pairs, using the table on p.5: Which of these terms do you know? Can you guess their role in performance of computers? Can you put the given units into three groups: hardware, software and firmware?

Hardware



Software

Firmware

Tower, a printer, monitor, keyboard, mouse, operating system, application software, word processor, webcam, scanner, graphics card, disk drive, BIOS, CD-ROM

Reading TEXT 1 Hardware and Software

Software ['sɔftweə] – программное обеспечение (ПО)

Hardware ['hɑ:dweə] - оборудование, аппаратура, аппаратное обеспечение, хардвер, "железо"

Systems software - системное программное обеспечение

Application software - прикладное программное обеспечение

Operating system - операционная система

File management utility – утилита управления файлами

Disk operating system - дисковая операционная система

To install [ɪn'stɔ:l] – устанавливать

End-user program – программа для конечного пользователя

Spreadsheet ['spredʃi:t] - электронная таблица

Database ['deɪtəbeɪs]- база данных

Word processing application - приложение текстового редактора

Software suite - комплект ПО

Web browser/ browser - веб-браузер/ браузер

To retrieve [rɪ'tri:v] - извлекать (хранимую) информацию

Hyperlink ['haɪpəˌlɪnk] - гипертекстовая связь, гиперссылка

Demand for – спрос на

Cloud computing – облачные вычисления

Computer case - системный блок

Tower - системный блок

Power supply unit - блок питания

Central processing unit (CPU) - центральный процессор

Optical disk drive - накопитель на оптических дисках

Soundcard – звуковая карта

Videcard – видео карта

Fan - вентилятор

Firmware - программно-аппаратные средства; встроенные программы; "зашитые программы" (в ПЗУ)

Read-only memory (ROM) - постоянная память, постоянное запоминающее устройство, ПЗУ

Basic Input/Output System (BIOS) - базовая система ввода/вывода

Motherboard - ['mʌðəbɔ:d] - системная плата, материнская плата

Non-volatile storage - энергонезависимое запоминающее устройство

To boot - загружаться (о компьютере, программе)

Pre-Reading Tasks

1. Before reading the text, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

- What types of software do you know? What is the difference between them?
- What do the following abbreviations such as URL, apps, DOS, OS stand for?
- What do you know about firmware? Can you give the examples?
- What units are called hardware?

2. Next, reorder the words in the mystery questions below:

- smart phones, development **What** has the of tablets, handheld and other mobile devices led to?
- is the chip located **Where** ROM?

Reading Tasks

Now read the article to check your answers and do the tasks which follow.

The term software refers to the set of electronic program instructions or data a computer processor reads in order to perform a task or operation. In contrast, the term hardware refers to the physical components that you can see and touch, such as the computer hard drive, mouse, and keyboard.

Software can be categorized according to what it is designed to accomplish. There are two main types of software: systems software and application software.

Systems Software

Systems software includes the programs that are dedicated to managing the computer itself, such as the operating system, file management utilities,

and disk operating system(or DOS). The operating system manages the computer hardware resources in addition to applications and data. Without systems software installed in our computers we would have to type the instructions for everything we wanted the computer to do!

Applications Software

Application software, or simply applications, are often called productivity programs or end-user programs because they enable the user to complete tasks such as creating documents, spreadsheets, databases, and publications, doing online research, sending email, designing graphics, running businesses, and even playing games! Application software is specific to the task it is designed for and can be as simple as a calculator application or as complex as a word processing application. When you begin creating a document, the word processing software has already set the margins, font style and size, and the line spacing for you. But you can change these settings, and you have many more formatting options available. For example, the word processor application makes it easy to add color, headings, and pictures or delete, copy, move, and change the document's appearance to suit your needs.

Microsoft Word is a popular word-processing application that is included in the software suite of applications called Microsoft Office. A software suite is a group of software applications with related functionality. For example, office software suites might include word processing, spreadsheet, database, presentation, and email applications. Graphics suites such as Adobe Creative Suite include applications for creating and editing images, while Sony Audio Master Suite is used for audio production.

A Web browser, or simply browser, is an application specifically designed to locate, retrieve, and display content found on the Internet. By clicking a hyperlink or by typing the URL of a website, the user is able to view Web sites consisting of one or more Web pages. Browsers such as Internet Explorer, Mozilla Firefox, Google Chrome, and Safari are just a few of the many available to choose from.

Demand for mobility in computing has led to the development of smart phones, tablets, and other handheld mobile devices. Mobile software

applications, or simply apps, are available to perform in much the same way as their full-blown computer software counterparts do: they are designed for specific tasks and functions (games, GPS, music, etc.). Some mobile apps are downloaded from Internet sources such as Apple's App Store, Google Play, and Amazon and are installed on the mobile device. Other apps are accessible from the Internet using cloud computing technology. Cloud-based apps are accessed by the user on a device but use information that is stored on a central computer server. Virtual office suites, Web-based email, online or mobile banking, and Facebook are just a few examples of cloud computing apps you may already use.

Computer hardware is the collection of all the parts you can physically touch. Some hardware components are easy to recognize, such as the computer case, keyboard, and monitor. However, there are many different types of hardware components. In this lesson, you will learn how to recognize the different components and what they do.

Types of Computers

Before looking at the various components, it is useful to distinguish between two different types of computers: desktop computers and laptop computers. A desktop computer consists of a computer case and a separate monitor, keyboard, and mouse. As the name suggests, this type of computer is typically placed on a desk and is not very portable.

A laptop computer has the same components but integrated into a single, portable unit.

While these two types of computers look quite different, they have the same general hardware components.

Hardware Components

Let's start with the computer case. This is the metal enclosure that contains many of the other hardware components. It comes in various shapes and sizes, but a typical tower model is between 15-25 inches high. Want to know what's inside? Okay, go get a screwdriver and let's open it up. Seriously, if you are really into computers, the best way to learn is to actually get hands-on.

The computer case contains a power supply unit to convert general-purpose electricity to direct current for the other components. The most

critical component is the motherboard, a plastic board on which several essential components are mounted. This includes the central processing unit, or CPU, the main memory, and expansions slots for other hardware components. The internal hard disk drive serves as the mass storage device for data files and software applications. An optical disk drive makes it possible to read from and write to CDs and DVDs. Other hardware components typically found inside the computer case are a sound card, a video card, and a cooling mechanism, such as a fan.

A computer system also needs input devices, such as a keyboard and a mouse. To interact with a user, a computer system also needs a display device, such as a monitor.

The hardware components described so far result in a fully functional computer system. A user can provide input using the keyboard and the mouse, and the computer can process instructions, read and write information, and display the results on the monitor. Most present-day computer systems have additional hardware components to provide more functionality. These include input devices, such as a microphone and video camera, and output devices, such as speakers. These can be integrated into the other hardware components or connected as external devices.

Additional peripheral devices can be connected to the computer systems, such as an image scanner to input paper documents as digital files, a printer to print out documents, and an external hard disk drive for extra mass storage.

The hardware components described here are all part of a personal computer. Other types of hardware are needed for a computer network and for the infrastructure that supports the Internet, but those types of hardware are not covered here.

Firmware is a combination of software and hardware. It includes the instructions to control hardware, which is just like software. It also includes hardware in the form of the actual memory chip where the instructions are stored. Technically, firmware consists of permanent software stored into read-only memory.

Computer systems use a special type of firmware known as BIOS, or Basic Input/Output System. It represents the basic code to get the computer started. You can think of BIOS as the firmware for the motherboard of your computer.

The BIOS determines what a computer can do without accessing programs from a disk. It contains all the code required to control the keyboard, display screen, disk drives, serial communications, and a number of other functions. The BIOS is typically placed in a read-only memory, or ROM, chip that comes with the computer - it is therefore often called a ROM BIOS.

ROM is a type of non-volatile storage, which means that the information is maintained even if the computer loses power. In a typical computer system, the ROM chip is located on the motherboard. This ensures that the BIOS will always be available and will not be damaged by disk failures. It also makes it possible for a computer to boot itself.

Firmware may need to be updated to fix minor bugs or add features to the device. For example, perhaps you have a DVD player connected to your TV to play movies, but it can also stream movies from online streaming services. Under most circumstances, you don't need to make any changes to your DVD player, but once in a while, you may get a message on your TV that some type of update is needed.

This will typically be a firmware update for the device. Since your device is set up to connect to the internet, the update can be downloaded and installed automatically. However, in some devices the firmware is permanent and cannot be changed.

The BIOS of a computer may need updating once in a while, but this is not very common. Modifying the BIOS is typically a task performed by a computer specialist trying to repair a computer system that has not been performing as expected.

Vocabulary Practice

1. Match the highlighted words from the text with their definitions:

2. a type of computer program that is designed to run a computer's hardware and application programs.

3. a computer program designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user.

4. system software that manages computer hardware and software resources and provides common services for computer programs.

5. an operating system with a command-line interface used on personal computers.

6. a computer program that is used, for example, when doing financial or project planning. You enter data in rows and columns and the program calculates costs, etc. from it.

7. an organized set of data that is stored in a computer and can be looked at and used in various ways

8. a collection of several applications that are bundled together and sold or distributed as a package

9. a program that lets you look at or read documents on the World Wide Web

10. a place in an electronic document that is connected to another electronic document or to another part of the same document

11. a way of using computers in which data and software are stored or managed on a network of servers (=computers that control or supply information to other computers), to which users have access over the Internet

12. the usually plastic or metal housing that contains the computer's main parts such as the motherboard, hard drive, etc.

13. the part of a computer that controls all the other parts of the system

14. a disk drive that uses laser light or electromagnetic waves within or near the visible light spectrum as part of the process of reading or writing data to or from optical discs.

15. a type of computer software that is stored in such a way that it cannot be changed or lost

16. computer memory that contains instructions or data that cannot be changed or removed

17. a type of firmware used to perform hardware initialization during the booting process (power-on startup) on IBM PC compatible computers, and to provide runtime services for operating systems and programs

18. the main printed circuit board (PCB) found in general purpose microcomputers and other expandable systems.

19. a type of computer memory that can retrieve stored information even after having been power cycled (turned off and back on)

2. Translate the sentences from Russian into English

1. Аппаратное обеспечение – это совокупность технических средств (электронных и механических устройств), обеспечивающих,

как нормальное функционирование каких-либо электронных систем – компьютеров, сетей передачи данных, так и расширяющих их основные функции.

2. Процессор способен выполнять 1,78 триллиона инструкций в секунду и содержит 621 миллион транзисторов.

3. Компания AMD устроила для журналистов презентацию своей новой видеокарты.

4. Эта компания планирует создать программное обеспечение, не требующее обновлений.

5. Ожидается, что Windows 10 будет облачной операционной системой.

6. Первый в мире центральный процессор, насчитывающий 1000 независимых программируемых ядер, был спроектирован командой исследователей из Калифорнийского университета.

7. Sapphire показала новые фотографии материнской платы Pure Platinum Z77.

8. Когда вы включаете свой компьютер, то первым делом ждете, когда в оперативную память загрузится операционная система.

9. Компания XILENCE - это всемирно известный производитель систем охлаждения и блоков питания для ПК.

10. По мнению аналитиков, пользователи социальной сети на первых порах никаких изменений не почувствуют, а вот восприятие приложений Microsoft может измениться.

11. Практически каждый покупатель компьютера, как правило, смотрит на внешний вид и интересуется производителем корпуса, не обращая внимания на характеристики установленного в нем блока питания.

12. Как удалить встроенные программы?

13. Компания Cubitek готовится пополнить свою фирменную серию компьютерных корпусов Tank еще одной моделью.

14. В приложении Microsoft Office PowerPoint 2007 гиперссылка осуществляет связь одного слайда с другим в одной и той же презентации.

15. Центральный процессор представляет собой сложную микросхему с миллионами транзисторов и множеством контактов.

3. Complete these sentences using the given words. Make any changes to the phrases that are necessary.

a Web browser\ researcher\ to utilize hardware and software\ goal/ vulnerable\ gain control\ to restoreoperating system (OS)\

The _____ is the backbone of your computer. If the OS is compromised, attackers can take over your computer -- or crash it. Now _____at North Carolina State University have developed an efficient system that _____to restore an OS if it is attacked.

At issue are security attacks in which an outside party successfully compromises one computer application (such as _____) and then uses that application to gain access to the OS. For example, the compromised application could submit a "system call" to the OS, effectively asking the OS to perform a specific function. However, instead of a routine function, the attacker would use the system call to attempt to _____of the OS.

"Our _____ is to give the OS the ability to survive such attacks," says Dr. Yan Solihin, an associate professor of electrical and computer engineering at NC State and co-author of a paper describing the new system. "Our approach has three components: attack detection; security fault isolation; and recovery."

The mechanism also allows the OS to identify the source of the attack and isolate it, so that the OS will no longer be _____ to attacks from that application.

The idea of detecting attacks and re-setting a system to a safe state is a well-known technique for _____ a system's normal functions after a failure, but this is the first time researchers have developed a system that also incorporates the security fault isolation component. This critical component prevents the OS from succumbing to the same attack repeatedly.

Speaking Practice

1. Work in small groups and discuss the following questions using useful expressions on p. 5:

- ❖ What is considered to be the brain of the computer?
- ❖ What is a storage device?
- ❖ What do you know about the cloud computing technology?
- ❖ What is the function of arithmetic and logic unit?
- ❖ Can you give the definition of cache?

2. Think of one of the latest releases of computers. Prepare a short presentation (p. 15) providing details about it such as inventors, its

hardware and software, its advantages and disadvantages. Use the Internet if you need to check the information. Present your information to the rest of the class.

Reading **TEXT 2 Peripheral Devices**

Computer peripheral – внешнее оборудование, периферийное устройство

Computer architecture - архитектура вычислительной машины

Hard drive - накопитель на жестких дисках

Random-access memory (RAM) – оперативное запоминающее устройство

Flash drive - флеш-драйв, флеш-накопитель

Data ['deɪtə] - данные, факты, сведения; информация

Expansion slot - расширительное гнездо

Hard disk drive - жесткий диск

Speaker ['spi:kə] - динамик ПК

Headphones- наушники

Wired connection – проводная связь

Wireless connection – беспроводная связь

Universal Serial Bus – универсальная последовательная шина

Pre-Reading Tasks

1. Before reading the text on the following page, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

- What do such abbreviations as Wi-fi and USB stand for?
- What do external and internal peripheral devices mean?
- How many types of connections do you know?

2. Next, ask questions to the underlined words:

- The peripheral devices fall into <u>three</u> general categories: input, output and storage devices.
- <u>Removing the monitor of a desktop computer</u> makes it pretty much useless.
- Bluetooth is good for <u>very short</u> distances.

Reading Tasks

Now read the article to check your answers and do the tasks which follow.

A computer peripheral is a device that is connected to a computer but is not part of the core computer architecture. The core elements of a computer are the central processing unit, power supply, motherboard and the computer case that contains those three components. Technically speaking, everything else is considered a peripheral device. However, this is a somewhat narrow view, since various other elements are required for a computer to actually function, such as a hard drive and random-access memory (or RAM).

Most people use the term peripheral more loosely to refer to a device external to the computer case. You connect the device to the computer to expand the functionality of the system. For example, consider a printer. Once the printer is connected to a computer, you can print out documents. Another way to look at peripheral devices is that they are dependent on the computer system. For example, most printers can't do much on their own, and they only become functional when connected to a computer system.

Types of Peripheral Devices

There are many different peripheral devices, but they fall into three general categories:

1. Input devices, such as a mouse and a keyboard
2. Output devices, such as a monitor and a printer
3. Storage devices, such as a hard drive or flash drive

Some devices fall into more than one category. Consider a CD-ROM drive; you can use it to read data or music (input), and you can use it to write data to a CD (output).

Peripheral devices can be external or internal. For example, a printer is an external device that you connect using a cable, while an optical disc drive is typically located inside the computer case. Internal peripheral devices are also referred to as integrated peripherals. When most people refer to peripherals, they typically mean external ones.

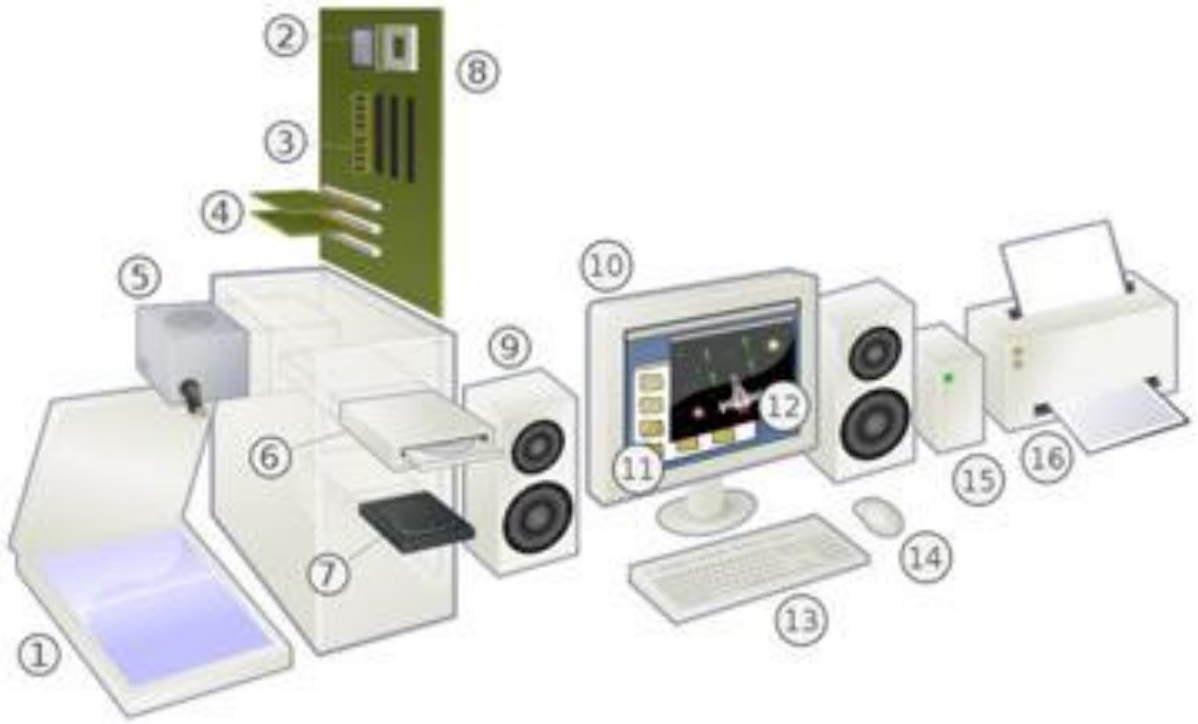
The concept of what exactly is 'peripheral' is therefore somewhat fluid. For a desktop computer, a keyboard and a monitor are considered peripherals - you can easily connect and disconnect them and replace them if needed. For a laptop computer, these components are built into the computer system and can't be easily removed.

The term 'peripheral' also does not mean it is not essential for the function of the computer. Some devices, such as a printer, can be disconnected and the computer will keep on working just fine. However,

remove the monitor of a desktop computer and it becomes pretty much useless.

Examples of Peripheral Devices

Here you can see a typical desktop computer system with a number of common peripheral devices. The central processing unit (#2), motherboard (#8) and power supply are the core computer system. Expansion slots (#4) on the motherboard make it possible to connect internal peripherals, such as a video card or sound card (not shown). Other internal peripherals shown are a hard disk drive (#7) and an optical disc drive (#6). External input peripherals are a scanner (#1), display monitor (#10), keyboard (#13) and mouse (#14). External output peripherals are a set of speakers (#9) and a printer (#16). Note that labels 11 and 12 in the figure refer to software and are not peripherals.



There are many other examples of peripherals, such as a microphone, web camera, headphones, external hard drive and flash drive. Most computer users have at least several of these peripheral devices.

How Peripherals Are Connected

Internal peripherals are directly connected to the motherboard using one of the different types of slots on the motherboard.

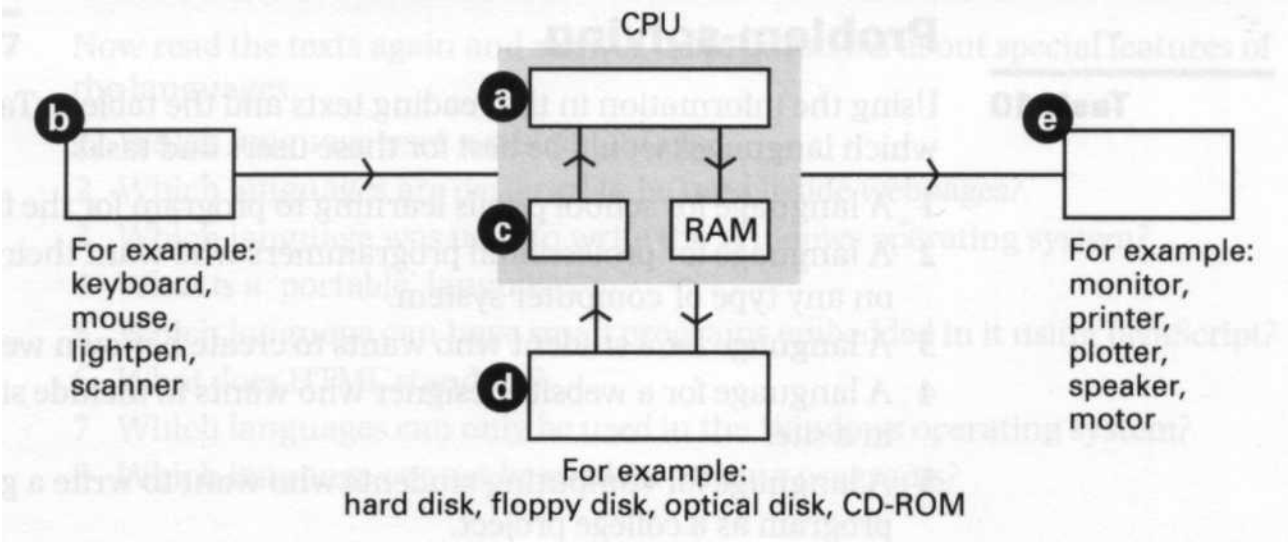
External devices can be connected using a wired connection or a wireless connection. As the name suggests, a wired connection uses a cable that needs to be plugged into the computer using a connector. The

most widely used connector is a Universal Serial Bus (or USB) connection, but several other types are used depending on the specific computer system and the type of peripheral.

A wireless connection does not require a cable. The most widely used wireless connections are Bluetooth and Wi-Fi. Bluetooth is good for very short distances, so peripherals such as a wireless mouse and keyboard typically use a Bluetooth connection. Wi-Fi is good for longer distances. If you have set up a wireless network in your home or office, you may be able to print wirelessly to a printer if it is also connected to the network.

Vocabulary Practice

1. Look at words in texts 1,2 and label the given diagram with the correct terms



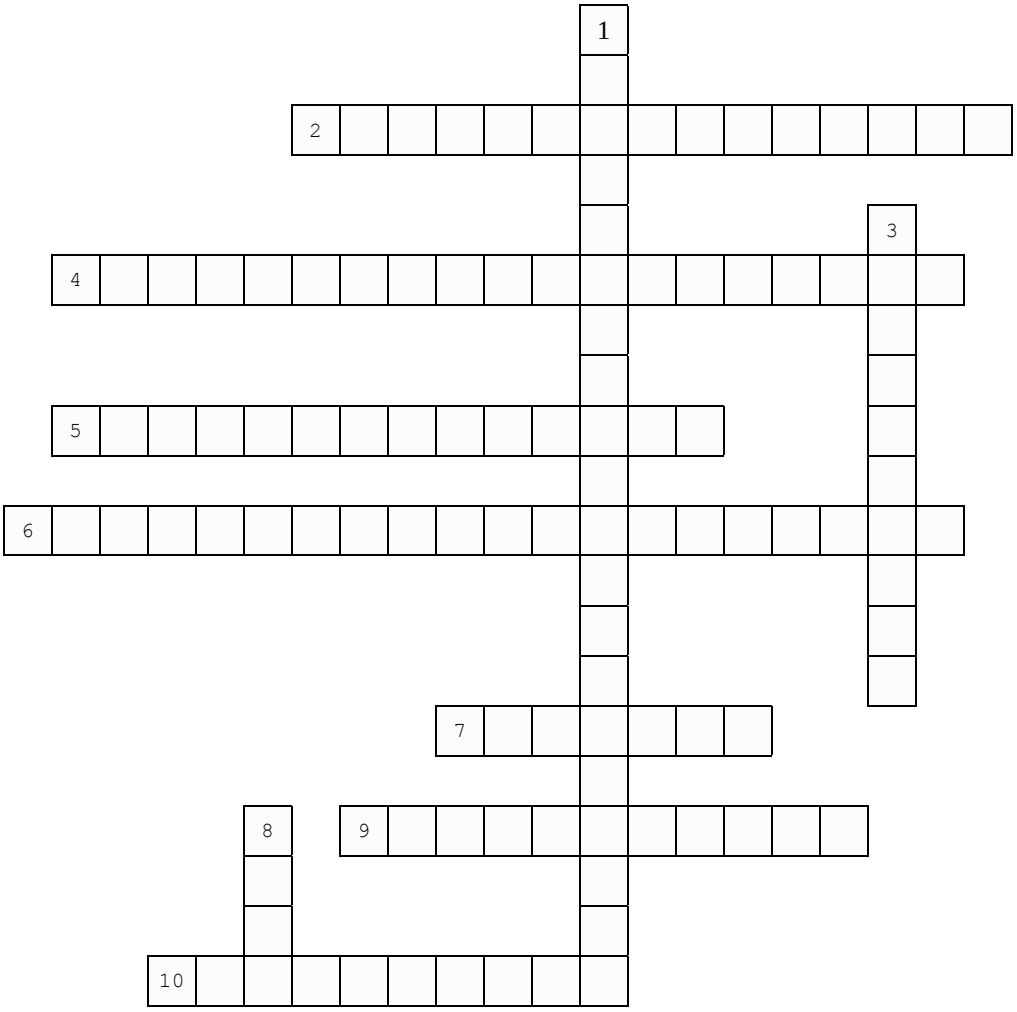
2. Match the words and expressions from the text to their meaning

A	B
1. computer case	a. tower
2. expand the functionality	b. increase in size, range, or amount
3. fall into	c. divide
4. replace	d. put in place of
5. remove	e. displace

2. Try to explain the highlighted words and phrases.

3. What does the writer mean by the underlined phrases? Discuss in pairs.

4. Complete the crossword.



Across

2. a type of low-cost, high-capacity physical storage used for random-access data in PCs and enterprise data centers

4. a device that is used to put information into or get information out of the computer

5. a socket on the motherboard that is used to insert an expansion card

6. computer memory in which data can be changed or removed and can be looked at in any order

7. a piece of equipment that sends out the sound from a CD player, radio

9. a small memory device that can be used to store data from a computer and to move it from one computer to another

10. a small speaker that is worn over or in the ear, often as one of a pair connected by a wire attached to a band

Down

1. an external serial bus interface standard for connecting peripheral devices to a computer

3. a non-volatile memory hardware device that permanently stores and retrieves information

8. information that is stored by a computer

Speaking practice

1. Paraphrase the following quotations. Which one do you agree with the most? Why? Discuss in small groups.

"It's hardware that makes a machine fast. It's software that makes a fast machine slow."

"For systems in which you already have a lot of hardware and software, change is difficult. That's why apps are so popular."

"Adapting old programs to fit new machines usually means adapting new machines to behave like old ones."

2. Look at these three newspaper headlines. What do you think the story is behind each one? Discuss your ideas with a partner.

- **Why Firmware is So Vulnerable to Hacking, and What Can be Done about it**
- **Supercomputer Powered by Mobile Chips Suggests New Threat to Intel**
- **Why Autocorrect for Passwords Is a Great Idea**

3. Read the task. Unscramble the words in groups. Compare your answers with the others. Present a brief description of each category.

Most users, whether at home or in business, are drawn to task-oriented software, sometimes called productivity software, that can make their work faster and their lives easier. The collective set of business tasks is limited, and the number of general paths towards performing these tasks is limited, too. Thus, the tasks and the software solutions fall, for the most part, into just a few categories, which can be found in most business environments. These major categories are

ONR GCSSORWPEID_____
SEARDHPESET_____
AAEASEEDMBNTGT AMAN_____
RIGHASCP_____
ACICUSNOTMINOM_____

4. Look at the advertisement for a personal computerand describe it, using useful phrases given on p.42:

DIMENSION™ 2400 DESKTOP

- Intel® Pentium® 4 Processor 2.80GHz
- Microsoft® Windows® XP Home Edition
- **512MB DDR RAM** (was 256MB)
- 80GB Hard Drive
- **17" Flat Panel Monitor**
- Integrated Intel® Extreme Graphics Card
- **DVD/CD Rewriter Combo Drive**
- Integrated Audio • Stereo Speakers
- 56k modem
- **Tiscali Broadband – only £15.99 a month PLUS FREE' Broadband Modem and NO Set-Up charge!**
- Microsoft® Works 7.0

£699 incl. VAT & Del.

or 48 monthly payments of **£19.75*** incl. VAT & Del.

Total Amount Payable £948 incl. VAT & Del.

How to read a computer ad?

- **Intel Pentium 4 processor (3GHz, 800MHz FSB)**
- **Mini-tower chassis**
- **1GB dual channel DDR2 SDRAM**
- **200GB Serial ATA hard drive (7200 r.p.m.)**
- **128MB PCI-Express video card**
- **Integrated audio**
- **48X CD-RW drive**
- **19" TFT flat panel XGA (1024 x 768) monitor**
- **Microsoft Windows XP Professional**

- The main processing chip called a 'Pentium 4' was designed and manufactured by the Intel Corporation.
  It operates at a clock speed of threegigahertz and has a front-side bus that operates at a speed of eight hundredmegahertz.
- A small, tall and narrow style of case containing the computer system.
- Synchronous dynamic random access memory with a capacity of one gigabyte. It is a high bandwidth, double data rate memory.
- A hard drive with a capacity of two hundred gigabytes that uses a type ofconnection interface known as Serial ATA i.e. it has a serial data connection rather than the original parallel connection.
- It rotates at a speed of seventhousand, two hundred revolutions per minute.
- Electronics for driving the graphics output has a memory capacity ofo ne hundred and twentyeight megabytes and uses a type of connection interface known as PCI-Express.
- Electronics for controlling the sound outputbuilt into the mainelectron ics of the computer.
- A compact disk read/write disk drive that operates at forty-eight times thespeed of the original CD drives.
- A nineteen inch, flat display screen made from thin film transistors with are solution of 1024 by 768.
- The operating system is Microsoft Windows XP Professional.

UNIT 2 REVISE AND CHECK

Give the definitions and the translation to the following words:

Software
Hardware
Systems software
Application software
Operating system
File management utility
Disk operating system
Database
Software suite
Web browser/ browser
To retrieve
To install
Hyperlink
Cloud computing technology
Computer case\Tower
Power supply unit
Central processing unit (CPU)
Optical disk drive
Sound card
Video card
Fan
Firmware
Read-only memory (ROM)
Basic Input/ Output System (BIOS)
Motherboard
Non-volatile storage
To boot
Computer peripheral
Computer architecture
Hard drive
Random-access memory (RAM)
Flash drive
Data
Universal Serial Bus (USB)
Videocard
Web browser/ browser
Wired connection
Wireless connection
Cloud computing technology
Expansion slot
Hard disk drive
Wired connection
Wireless connection
Universal Serial Bus (USB)

Can you answer the following questions?

- ✓ What is software divided into?
- ✓ Is there any difference between mobile and computer apps? What?
- ✓ What types of computers do you know? What are their similarities and differences?
- ✓ Can you enumerate the main computer hardware components?
- ✓ What does the tower consist of?
- ✓ What have you found out about firmware?
- ✓ What categories do peripheral devices fall into?
- ✓ What are internal and external peripherals?
- ✓ How are internal peripherals connected to computer?
- ✓ How are external peripherals connected to computer?

Unit 3 Programming

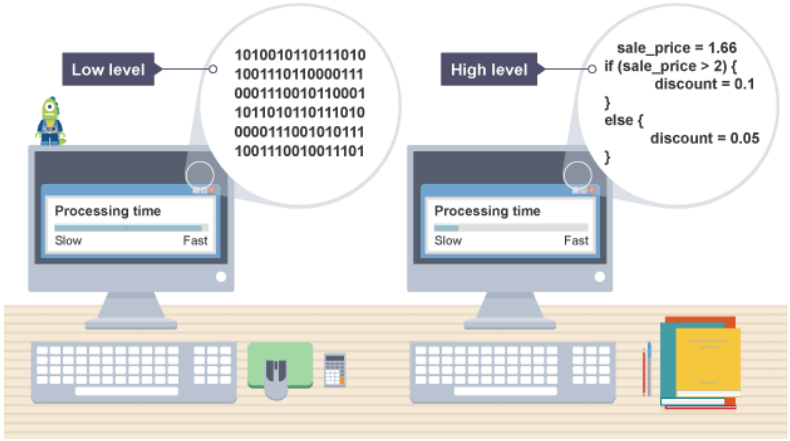
Lead-in

1. Put these basic elements of programming into the right order and match with the definitions:

1. **A_ _ _ _ _ _ _ _ _c**
2. **C _ _ _ _ _ _ _ _ _ _l**
3. **I_ _ _t**
4. **L_ _ _ _ _g**
5. **O_ _ _ _ t**

1. cycling through a set of instructions until some condition is met
2. getting your results out of the computer
3. testing to see if a condition is true or false
4. getting data and commands into the computer
5. performing mathematical calculations

2. What is this picture related to? What does it describe?



3. **Discuss in pairs, using the table on p.5: Which of these terms do you know? Can you guess their role in programming? What kind of programming languages are they?**

Java Machine code Python HTML CSS Ruby C++ Assembly

Reading TEXT 1 Basics of Programming

Programming ['prəugræmɪŋ] –программирование
Conditional – условная конструкция
Looping – введение циклов (в программу)
Event-controlled loops [luːp] -циклы, управляемые событием
Counter-controlled loops – циклы со счётчиком
To meet a predetermined condition – удовлетворять заданное условие
To echo['ekəu] - отражаться
Flow chart–блок-схема
Object Oriented Programming- объектно-ориентированное программирование
Procedural Programming [prə'siːʤ(ə)r(ə)l] - процедурное программирование
Procedure [prə'siːʤə] -процедура, подпрограмма
Routine – программа, операция
Subroutine [ˈsʌbruːtiːn] - подпрограмма
Imperative programming - императивное программирование (задающее жёсткую последовательность действий, в противоположность функциональному и логическому)
Top-down language–язык нисходящего программирования
To interact [ˌɪntər'ækt] – взаимодействовать, влиять друг на друга
Method ['meθəd] –метод
Class - класс
Blueprint ['bluːprɪnt] –шаблон, образец
Pseudocode [ˈsjuːdəʊ ‘kəʊd] - псевдокод
Code [kəud] - код
To debug [ˌdiː'bʌg] – отлаживать (программу)
To iterate ['ɪt(ə)reɪt] – повторять

Pre-Reading Tasks

1. Before reading the text, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

•What types of programming languages do you know? What is the difference between them?

•Can you give any examples?

2. Next, reorder the words in the mystery questions below:

- And describing procedures What processes involves?

•high-level **When** the first was programming language invented FORTRAN?

Reading Tasks

Now read the article to check your answers and do the tasks which follow.

Programming is somewhat like working with building blocks. The five basic elements in programming are:

- input: getting data and commands into the computer
- output: getting your results out of the computer
- arithmetic: performing mathematical calculations on your data
- conditional: testing to see if a condition is true or false
- looping: cycling through a set of instructions until some condition is met

Let's show how the five operations can be mapped.

Input

Input can come from just about anywhere: a keyboard, a touchscreen, a text file, and another program are just a few examples. Input is one of the two elements that are used by every program because every program needs some data to work with.

Arithmetic

Computers can perform all kinds of mathematical operations and functions. Not every program needs to do calculations on the data that's entered, but it may still need to do some in order to control what is happening inside the program itself.

Output

Output is the result that your program gives you. That is the whole purpose of writing a program: to ask a question and get the answer! Output can take many forms - text or graphics, either printed or on a screen, a sound - just about any form that can be interpreted and understood by a human being or another program.

Looping

Quite often, your program has to repeat an operation a number of times before the program can continue. The simplest example is adding up a column of numbers. Since a computer can only add two numbers at a time,

it has to add the first two numbers, then add the next number to the total, then add the next one and the next one until there are no more numbers to be added. There are a number of different types of loops, which are used based on how the input or calculations need to be handled. All of them are either event-controlled or counter-controlled. The control part is important - if there is no control, the loop can go on forever, or until you stop the program. Event-controlled loops can be stopped by an external event. That event could be user input, perhaps in response to a prompt like 'Any More?' or by reaching the end of an input file. Counter-controlled loops are stopped when a counter in the loop reaches a predetermined value. The loop may be counting down to zero or up to a maximum value. Think ticket sales for a ball game or concert.

Conditional

Loops need to be controlled. The loop ends when a predetermined condition is met. A programming condition looks at a program statement and figures out if it is true or false. Here is an example of an event-controlled loop:

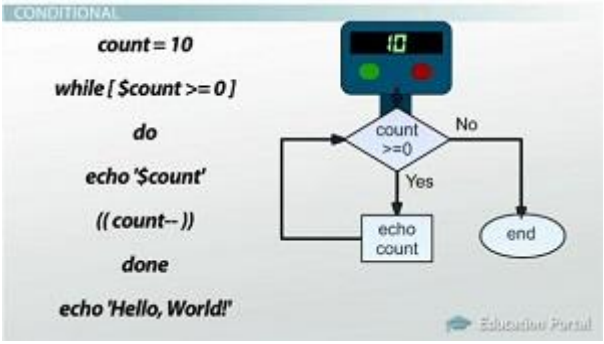
```
while read name
do
echo $name
done < namelist
```

Without going into great detail, this program is reading a file called 'namelist.' When it starts, it opens the file and gets the first name on the list. It displays each name on the screen. When it has read all the names, it stops. Here is an example of a counter-controlled loop:

```
count=10
while '$count >= 0'
do
echo '$count'
(( count - - ))
done
echo 'Hello, World!'
```

This little program counts down from ten through zero, then displays the message 'Hello, World!' The 'while' statement contains the condition, while the count is greater than or equal to zero, the loop will continue. After the count is echoed, the 'count -' statement subtracts one from the count.

When the count reaches -1, the test is no longer true, the loop ends and the program displays 'Hello, World!'.



This flow chart represents the counter-controlled loop mentioned in this les

Programming conditions are also used to determine which action the program should take, even though there is no looping.

Programming requires two important skills - the ability to analyze and understand a problem and the ability to describe that problem to a computer so it can arrive at a solution.

Object Oriented Programming vs. Procedural Programming

There are several alternative approaches to the programming process. Two of the most important approaches are procedural programming and object-oriented programming.

Procedural programming uses a list of instructions to tell the computer what to do step-by-step. Procedural programming relies on procedures, also known as routines or subroutines. A procedure contains a series of computational steps to be carried out. Procedural programming is also referred to as imperative programming. Procedural programming languages are also known as top-down languages.

Procedural programming is intuitive in the sense that it is very similar to how you would expect a program to work. If you want a computer to do something, you should provide step-by-step instructions on how to do it. It is, therefore, no surprise that most of the early programming languages are all procedural. Examples of procedural languages include Fortran, COBOL and C, which have been around since the 1960s and 70s.

Object-oriented programming, or **OOP**, is an approach to problem-solving where all computations are carried out using objects. An object is a component of a program that knows how to perform certain actions and how to interact with other elements of the program. Objects are the basic units of object-oriented programming. A simple example of an object would be a person. Logically, you would expect a person to have a name. This

would be considered a property of the person. You would also expect a person to be able to do something, such as walking. This would be considered a method of the person.

A method in object-oriented programming is like a procedure in procedural programming. The key difference here is that the method is part of an object. In object-oriented programming, you organize your code by creating objects, and then you can give those objects properties and you can make them do certain things.

A key aspect of object-oriented programming is the use of classes. A class is a blueprint of an object. You can think of a class as a concept, and the object as the embodiment of that concept. So let's say you want to use a person in your program. You want to be able to describe the person and have the person do something. A class called 'person' would provide a blueprint for what a person looks like and what a person can do. Examples of object-oriented languages include C#, Java, Perl and Python.

Key Differences

One of the most important characteristics of procedural programming is that it relies on procedures that operate on data - these are two separate concepts. In object-oriented programming, these two concepts are bundled into objects. This makes it possible to create more complicated behavior with less code. The use of objects also makes it possible to reuse code. Once you have created an object with more complex behavior, you can use it anywhere in your code.

OOP Most Popular

Object-oriented programming has become the dominant programming paradigm in today's software development. Most of the newer programming languages that have been developed over the past 10 to 20 years are object-oriented. However, this does not mean that the other paradigms have gone away, and procedural programming languages are still widely used.

Some languages combine elements of both paradigms. For example, the widely used language C++ was derived from the language C, which is a procedural language. Object-oriented components were added to C to become C++, which is therefore both a procedural and object-oriented language.

The general steps for writing a program include the following:

- Understand the problem you are trying to solve
- Design a solution

- Draw a flow chart
- Write pseudocode
- Write code
- Test and debug
- Test with real-world users
- Release program
- Iterate the steps for the next version

In order to start writing a program without having to worry too much about the details of a specific programming language, programmers use pseudocode. Pseudocode is a plain English version of the detailed steps of a computer program that can be read by non-programmers. Pseudocode is written in English so that humans can easily understand it, but it looks like programming. However, pseudocode is not an actual programming language. It contains well-defined structures that resemble programming languages, but these are not unique to one particular language.

You may also encounter pseudocode in textbooks on computer programming. Instead of focusing on the syntax of one specific language, the textbook may teach the logic of programming using pseudocode.

A Simple Example

Let's look at a very simple example. Let's say a bank customer wants to use an ATM to withdraw $100 cash from her account. You need to program for the ATM to check the customer's balance and determine if there is enough money in the account. The pseudocode would look something like this:

```
if balance is less than $100
   print 'Insufficient funds'
else
   issue $100 cash from machine
   calculate new balance
   print new balance
```

You don't need to be a programmer to understand this code. In fact, if you were to read out this code, it almost sounds like regular English. On the other hand, the logic and structure is starting to look a lot like code.

Vocabulary Practice

1. Match the highlighted words from the text with their definitions:

1. the act of an individual writing code (a set of instructions) that are to be interpreted and executed by a computer or other electronic device.
2. the process of a software program or script repeats the same instructions or processes the same information over and over until receiving the order to stop.
3. a portion of source code in an object-oriented programming language such as Java; one class per source code file.
4. one whose execution is controlled by the occurrence of an event within the loop itself
5. one that is executed a certain number of times
6. to meet the requirements given
7. a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure.
8. a programming that incorporates other segments of external code.
9. a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program. It contains a systematic order of statements, functions and commands to complete a computational task or program.
10. a set of instructions designed to perform a frequently used operation within a program
11. a sequence of instructions for performing a task that forms a program or a distinct part of one.
12. a portion of code that may be called and executed anywhere in a program.
13. an if-statement.
14. a programming paradigm that uses statements that change a program's state.
15. a language of programming where an application is constructed starting with a high-level description of what it is supposed to do, and breaking the specification down into simpler and simpler pieces, until a level has been reached that corresponds to the primitives of the programming language to be used
16. to act in such a way as to have an effect on each other

17. a programmed procedure that is defined as part of a class and included in any object of that class.
18. something which acts as a plan, model, or template for others
19. a computer programming language that resembles plain English that cannot be compiled or executed, but explains a resolution to a problem.
20. program instruction
21. to identify and remove errors
22. to make repeated use of a mathematical or computational procedure, applying it each time to the result of the previous application

2. Translate the sentences from Russian into English

1. Pascal как язык для обучения программированию уже не соответствует современным требованиям.
2. В настоящее время большинство программистов пользуются объектно-ориентированными языками C++, C# и Java.
3. Программисты создали новый язык программирования.
4. Мой друг заявил мне, что любая программа может быть написана без использования условной конструкции if/else.
5. В чем отличие объектно-ориентированного программирования от процедурного?
6. После установки программы на вашем компьютере, программа не отображается в окне.
7. Стоит заметить, что большая часть современных языков в той или иной степени поддерживает императивное программирование.
8. Цикл - это команда исполнителю многократно повторить указанную последовательность команд.
9. Под отладкой программы понимается процесс испытания работы программы и исправления обнаруженных при этом ошибок.
10. В качестве примеров даны блок-схемы очень простых алгоритмов.
11. В языке Pascal, как и в большинстве языков программирования, существует три типа циклов.
12. Цикл C++ for позволяет вашим программам повторять один или несколько операторов определенное количество раз.

13. Если программисту действие нужно выполнить несколько раз, то выполнение действия целесообразно выделить в маленькую подпрограмму — процедуру.
14. В языках программирования для оформления и использования подпрограмм существуют специальные синтаксические средства.
15. Существуют два вида подпрограмм – внутренние и внешние.
16. Нисходящее программирование - методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.
17. Если функция выполняет одни и те же действия над аргументами различных типов, то ее можно оформить в виде шаблона.
18. Язык C++ представляет собой набор команд, которые говорят компьютеру, что необходимо сделать. Этот набор команд, обычно называется исходный код или просто код.

3. Complete these sentences using the given words. Make any changes to the phrases that are necessary.

tutorial/ "generic" programming/ compiler/ low-level/"object-oriented" programming/ syntax/CGI scriptgeneral-purpose programming language/

What is C, What is C++, and What is the Difference?

C is a programming language originally developed for developing the Unix operating system. It is a _____ and powerful language, but it lacks many modern and useful constructs. C++ is a newer language, based on C, that adds many more modern programming language features that make it easier to program than C.

Basically, C++ maintains all aspects of the C language, while providing new features to programmers that make it easier to write useful and sophisticated programs.

For example, C++ makes it easier to manage memory and adds several features to allow _____ and_____. Basically, it

makes it easier for programmers to stop thinking about the nitty-gritty details of how the machine works and think about the problems they are trying to solve.

So, what is C++ used for?

C++ is a powerful _____. It can be used to create small programs or large applications. It can be used to make _____ or console-only DOS programs. C++ allows you to create programs to do almost anything you need to do. The creator of C++, Bjarne Stroustrup, has put together a partial list of applications written in C++.

How do you learn C++?

No special knowledge is needed to learn C++, and if you are an independent learner, you can probably learn C++ from online _____ or from books.

While reading a tutorial or a book, it is often helpful to type - not copy and paste (even if you can!) - the code into the _____ and run it. Typing it yourself will help you to get used to the typical typing errors that cause problems and it will force you to pay attention to the details of programming_____ . Typing your program will also familiarize you with the general structure of programs and with the use of common commands. After running an example program - and after making certain that you understand how it works - you should experiment with it: play with the program and test your own ideas. By seeing which modifications cause problems and which sections of the code are most important to the function of the program, you should learn quite a bit about programming.

Speaking Practice

1. Work in small groups and discuss the following questions using useful expressions on p. 5:

- ❖ What are the basic elements of programming?
- ❖ How will you describe loops and their types? Can you give your own examples?
- ❖ What skills do you need to program?
- ❖ Explain the difference between procedural and object-oriented programming.
- ❖ What are the stages of writing a program?

2. Work in groups. Describe the following generation of languages: the first, the second, the third, the fourth and the fifth ones. When werethey designed? Who first developed? What are key

features? How do they differ from the other generations of programming languages? What languages do they include?

3. Prepare a short presentation (p. 15) on one of the most in-demand programming languages, for example, SQL, Java, JavaScript, C#, C++, Python, PHP,Ruby on Rails and iOS/Swift. Speak about its history of development, usage, advantages and disadvantages. Use the Internet if you need to check the information. Present your information to the rest of the group.

Reading TEXT 2 HOW TO WRITE A PROGRAM?

Statement ['steɪtmənt] – оператор

Error ['erə] - ошибка

Capitalize ['kæpɪt(ə)laɪz] – писать прописными буквами

To indent[ɪn'dent] –сделать отступ, сдвинуть вправо текст программы

Boolean[ˈbuːliən]- булево [логическое] выражение

Value of true– значение «истина»

Value of false – значение «ложь»

Boolean logic – булева логика

Boolean operator – логический оператор, булев оператор

String[strɪŋ] - строка

Array [ə'reɪ] - массив

Boolean expression - булево выражение

Variable ['veərɪəbl] – переменная, изменяемая величина

Algorithm ['ælg(ə)rɪð(ə)m] -алгоритм

Searchal gorithm -алгоритм поиска, поисковый алгоритм

Data base query ['kwɪərɪ] - запрос базы данных

Sequential search [sɪ'kwen(t)ʃ(ə)l]- последовательный, логический поиск

Linear search ['lɪnɪə] - линейный поиск

Optimization [ˌɔptɪmaɪ'zeɪʃ(ə)n] -оптимизация

Binary search ['baɪnərɪ] –двоичный поиск

Integrated Development Environment- интегрированная среда разработки (программ)

Compiling - компиляция, компилирование

Text editor - текстовый редактор

Syntax checking–проверка синтаксиса

Bug [bʌg] – баг, глюк, ошибка
To run a program – запустить программу
To execute a program – выполнить программу
Machine language – машинный язык
Machine code – машинный код
Binary notation –двоичная система счисления
High-level language – язык высокого уровня
Compiler [kəm'paɪlə] - компилятор, компилирующая программа
Interpreter [ɪn'tɜːprɪtə] - интерпретатор

Pre-Reading Tasks

1. Before reading the text on the following page, work with a partner and ask and answer the questions below. Base your answers on your possible knowledge of the topic:

- Why was programming initially difficult?
- What four main paradigms are programming languages divided into?
- Are coding and programming the same?

5. Next, ask questions to the underlined words:

- Ada Lovelace was the first woman to devise an algorithm that could be processed a by a machine.
- The first actual computer "bug" was identified in 1947 as a dead moth.
- PHP was originally designed to create dynamic and more interactive web pages. It is the most widely-used, open-source and general-purpose scripting language.

Reading Tasks

Now read the article to check your answers and do the tasks which follow.

Before you write one line of code in any language, it is a good idea to write it in a simple way first to ensure you have included everything you need. The best way to set this up is by using pseudocode.

Understanding Pseudocode

To use pseudocode, all you do is write what you want your program to say in English. Pseudocode allows you to translate your statements into any language because there are no special commands and it is not standardized. Writing out programs before you code can enable you to better organize and see where you may have left out needed parts in your programs. All you have to do is write it out in your own words in short statements.

Basic Guidelines

If you would like a format for using pseudocode to create your program, here are some basic guidelines:

First, do not use language-specific commands in your statements. Pseudocode should be universal. The point of pseudocode is to design a program that can be translated into any language.

Second, remember to write only one task or statement per line. Make sure you put only one task on each line. Including too much information on one line can be confusing and increases the possibility of errors.

Third, make sure to capitalize keywords. Capitalizing keywords like 'Read,' 'Write,' or 'Display' helps to show when an action is occurring or when a specific command or process will be necessary when coding in your specific language.

Lastly, indent statements in a loop to better see the flow of your program. Indenting can become your friend. It makes things easier to read and keeps information together, especially when creating loops.

Pseudocode Text Editors

Although pseudocode is not standardized and you can use any document-creating application or software to create it, there are some text editors out there to help you organize your programs.

Boolean Logic, Operators & Expressions

Programming uses Booleans, which are used to represent values of true and false. Many operations use Boolean logic. Learn how Boolean operators and expressions are used.

Boolean Data Type

Programming uses a number of different data types. The data type of an object determines what type of values an object can have and what operations can be performed on the object. Commonly used data types include strings, numbers, lists and arrays.

The Boolean data type can only represent two values: true or false. Typically, a 1 is used to represent true, and a 0 is used to represent false. Boolean data is widely used when working with conditions. The term 'Boolean' comes from a 19th century mathematician called George Boole who came up with the original idea of what we now call Boolean logic in his book *The Laws of Thought*.

Boolean Expression

Boolean data are used in Boolean expressions, which are expressions in a programming language that produce a Boolean value. An expression in programming is any combination of values, variables and operators that produce a new value. For example, 2 + 3 is an expression, and the result is the new value 5. When you use a Boolean expression, the only logical result can be true or false.

Consider the following example where a user inputs two values and a computer program determines whether the first one is smaller than the second one or not.

x = 8
y = 7
x < y

In this example, the part '*x* < *y*' is the Boolean expression. You are asking whether *x* is less than *y*, and the answer can only be a yes or a no - which means true or false in programming. In the example, the value of *x* is in fact not smaller than the value of *y*, and the program therefore results in a Boolean value of false. In programming language, we say that the expression is evaluated and returns a value of false.

The Boolean type is the primary result of conditional statements, which are used to control workflow in program. For example, if a particular condition is true, then do this; if the condition is false, then do something else.

Boolean Operators

In addition to Boolean data, there are Boolean operators, which are used to carry out Boolean algebra. There are three main Boolean operators: AND, OR and NOT. The first two are used to combine two expressions; the third is used as a negation operator. Let's look at each of these in more detail.

The simplest Boolean operator is the NOT operator. It simply turns true into false, and vice versa. Consider the following example.

x = 8
y = 7
NOT (x < y)

This returns a value of true. We know that *x* is greater than *y*, so the expression '*x* < *y*' returns a value of false. The NOT operator turns this into a value of true.

Now, let's look at the AND and OR operators. The AND operator compares two expressions. It only returns a value of true if both expressions are true; otherwise it returns a value of false. Consider the following example:

x = 8
y = 7
z = 6
(*x* < *y*) AND (*z* < *y*)

The first expression is false, and the second expression is true. The AND operator combines both expressions, and since one of them is false, the final result is false.

Now, let's look at the OR operator. The OR operator also compares two expressions. It returns a value of true if one of the expressions is true or if both expressions are true. If both expressions are false, it returns a value of false.

x = 8
y = 7
z = 6
(*x* < *y*) OR (*z* < *y*)

The first expression is false, and the second expression is true. The OR operator combines both expressions, and since one of them is true, the final result is true.

These Boolean operators illustrate the use of Boolean logic. Boolean logic is widely used when writing programs. It is also widely used as part of search algorithms and database queries.

What Is an Algorithm?

An **algorithm** is a well-defined procedure that allows a computer to solve a problem.

How Do Algorithms Work?

A very simple example of an algorithm would be to find the largest number in an unsorted list of numbers. If you were given a list of five different numbers, you would have this figured out in no time, no computer needed. Now, how about five million different numbers? Clearly, you are going to need a computer to do this, and a computer needs an algorithm.

Here is what the algorithm could look like. Let's say the input consists of a list of numbers, and this list is called L. The number L1 would be the first number in the list, L2 the second number, etc. And we know the list is not sorted - otherwise the answer would be really easy. So, the input to the algorithm is a list of numbers, and the output should be the largest number in the list.

The algorithm would look something like this:

Step 1: Let Largest = L1

This means you start by assuming that the first number is the largest number.

Step 2: For each item in the list:

This means you will go through the list of numbers one by one.

Step 3: If the item > Largest:

If you find a new largest number, move to step four. If not, go back to step two, which means you move on to the next number in the list.

Step 4: Then Largest = the item

This replaces the old largest number with the new largest number you just found. Once this is completed, return to step two until there are no more numbers left in the list.

Step 5: Return Largest

This produces the desired result.

Notice that the algorithm is described as a series of logical steps in a language that is easily understood. For a computer to actually use these instructions, they need to be written in a language that a computer can understand, known as a programming language.

Alternative Approaches and Optimization

There are many different types of algorithms. Search algorithms are used to find an item with specific properties among a collection of items.

There are different approaches to searching, each representing a slightly different technical approach to the same problem.

In a sequential or linear search, you start by examining the first item in the list to see if it matches the properties you are looking for. If not, you continue examining each sequential item until a match is found.

Alternative algorithms may require less time to find the correct answer. This is known as optimization: the process of finding the most computationally efficient algorithms to solve a particular problem.

In the case of searching, an alternative to sequential search is the binary search. A binary search improves the algorithm by removing as much of the input data as possible without having to examine each item.

In a binary search, you would jump to the item more or less in the middle of the list. If the number you are looking for is higher, you can drop the left-hand side of the list and continue only with the right-hand side. That reduces the number of items to search through by half in just one step. You can repeat this until you have found the number you are looking for or until the remaining list is very short, and then you can run a sequential search very quickly.

Determining which algorithm is best for a given task is not as easy as it may sound. For example, in the case of sequential and binary search, the binary search is much faster but only if the list of interest is already sorted. Sorting would require another algorithm, which will take quite a bit of time. This may be worth it if the list will be searched many times. However, if you only plan to search an unsorted list once, the sequential search will be faster than first performing a sort and then a binary search.

Debugging and Compiling Code

Programmers use an Integrated Development Environment to assist in writing code in a specific programming language. Here are specific tools used by programmers, such as debugging and compiling code.

Writing Code

Computer code is essentially a list of instructions that can be run by a certain program. You can write code using a basic text editor, but it is much more effective to use a software application that is specifically designed for this purpose. Similarly, a code editor provides tools such as syntax checking. Syntax is to code what spelling and grammar are to writing English.

A code editor is also called an integrated development environment, or IDE. An IDE is a software application for formatting your code, checking syntax, as well as running and testing your code.

Testing and Debugging

A program that is free of syntax errors will execute. However, this does not mean it actually works. For example, let's say you have a file with payroll information for each employee, with each employee represented by a line. You need a computer program that reads this information line-by-line and performs some type of operation, such as calculating benefits. The results should then be written to a new file.

Once you have written your code and checked for any syntax errors, you are ready to start testing. Testing consists of determining whether the program actually does what it is supposed to do. In order to test your program, you would run the program using a test file as the input. You then examine the output to make sure it is correct.

Did the program create an output file in the desired format? Does the output file contain the correct information? Were the calculations done correctly? Were all the lines in the input file processed?

Now let's say your testing shows that the output is not as expected. Time to start debugging. A bug in a computer program is a defect - something that prevents the program from executing correctly. Debugging is the process of finding and removing bugs from a program.

One approach to debugging is to read through the original code to try to find any bugs. Now imagine that your code contains 1,000 lines, and 999 of those could actually be correct. Finding the bug by manually reading through all the lines of codes is possible but cumbersome.

To make debugging more effective and less time-consuming, programmers use a debugger. This is one of the tools in a typical IDE. A debugger helps you walk through your code in a systematic manner to find the bugs.

Debugging can tell you where the bug is located in the program but not how to fix your code. You still have to go into the code, understand its logic and then correct it. However, using a debugger can save you a lot of time; instead of having to look at 1,000 lines of code, you may only have to look at 5 lines.

Machine Code vs. High-Level Languages

For now, we have simply referred to 'running' or 'executing' a program. Let's look at what this actually involves. Computers think in terms of 1s and 0s. Machine language, or machine code, is the only language that is directly understood by the computer and does not need to be translated.

Machine code uses binary notation written as a string of 1s and 0s. A program instruction in machine language may look like something like this:

10010101100101001111101010011011100101

Binary notation is very difficult for humans to understand. This is where high-level languages come in. A high-level language is a programming language that uses English and mathematical symbols in its instructions. When programmers write code, they use a high-level language. Examples of high-level languages are C++, Fortran, Java and Python.

A high-level language is much closer to the logic of a human language, but it cannot be understood directly by a computer, and it needs to be translated into machine code. There are two ways to do this: using a compiler or an interpreter.

Compiler

A compiler is a computer program that translates a program written in a high-level language to the machine language of a computer. The high-level program is referred to as the source code.

Consider a typical computer program that processes some type of input data to produce output data. The compiler is used to translate the source code into machine code or compiled code. This does not use any of the input data. When the compiled code is executed, referred to as 'running the program,' the program finally processes the input data to produce the desired output.

When using a compiler, the entire source code needs to be compiled before the program can be executed. The resulting machine code is typically a compiled file, such as a file with a .EXE extension. Once you have a compiled file, you can run the program over and over without having to compile it again.

Interpreter

An interpreter translates source code line-by-line during execution. Consider again a computer program that processes some type of input data to produce output data. The interpreter executes the code line-by-line, which results in the desired output data. The only result is the output data - there is

no compiled code. When using an interpreter, every time you want to run the program, you need to interpret the code again line-by-line.

Compiled vs. Interpreted

Compiled code tends to be faster since the translation is completed in one step prior to the actual execution. Interpreted code, on the other hand, is more flexible and can be run interactively. For example, using interpreted code you can try out a few lines of code to see if they work without having to go through the steps of compiling and executing the program. Examples of compiled languages are C and its derivatives C++ and C#, COBOL and Fortran. Examples of interpreted languages are JavaScript, Perl, Python and Ruby.

Vocabulary Practice

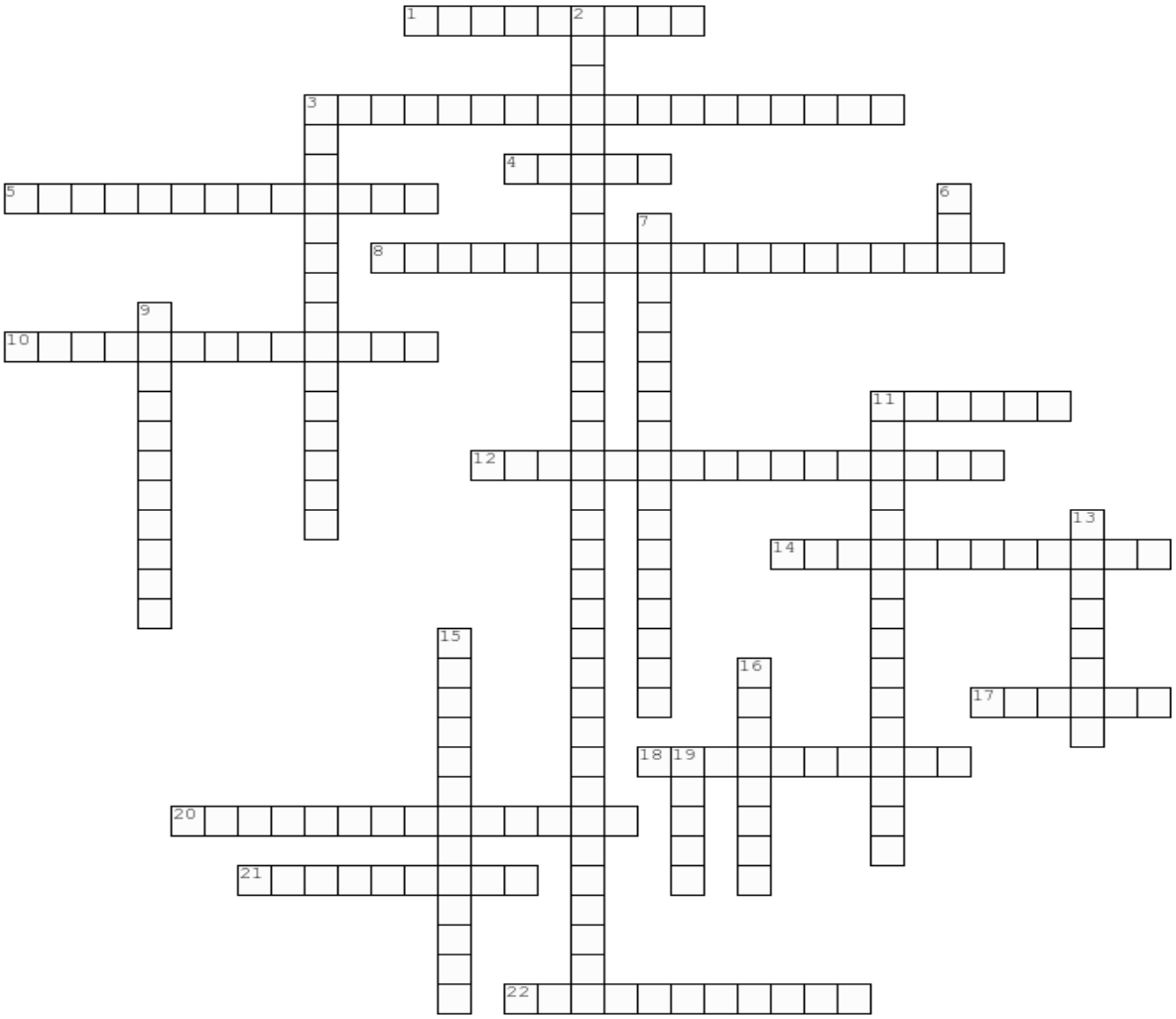
1. Match the words and expressions from the text to their meaning

A	B
1. Error	a. a fault in a machine
2. Capitalize	b. changeable
3. Toindent	c. demand for answers
4. Variable	d. make a space
5. Query	e. mistake
6. Bug	f. to begin a word with a capital letter

2. Try to explain the highlighted words and phrases.

3. What does the writer mean by the underlined phrases? Discuss in pairs.

4. Complete the crossword



Across

1. a set of steps that are followed in order to solve a mathematical problem or to complete a computer process

3. an expression in a programming language that produces a Boolean value when evaluated, i.e. one of true or false.

4. failure or fault

5. a search algorithm that finds the position of a target value within a sorted array

8. a programming language with strong abstraction from the details of the computer

10. a subset of algebra used for creating true/false statements

11. any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks)

12. simple words (AND, OR, NOT or AND NOT) used as conjunctions to combine or exclude keywords in a search, resulting in more focused and productive results.

14. a set of instructions executed directly by a computer's central processing unit (CPU)

17. to start a line of print or writing further away from the edge of the page than the other lines

18. to write or print a letter of the alphabet as a capital; to begin a word with a capital letter

20. a request for data or information from a database table or combination of tables

21. the smallest standalone element of an imperative programming language that expresses some action to be carried out.

22. a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program.

Down

2. a software application that provides comprehensive facilities to computer programmers for software development.

3. expression of a number with a base of 2 using only the digits 0 and 1 with each digital place representing a power of 2 instead of a power of 10 as in decimal notation

6. a coding error in a computer program

7. a synonym for linear search

9. computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs)

11. an algorithm that retrieves information stored within some data structure

13. a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code

15. a method for finding a target value within a list

16. a value that can change, depending on conditions or on information passed to the program

19. a container object that holds a fixed number of values of a single type.

Speaking practice

1. Paraphrase the following quotations. Which one do you agree with the most? Why? Discuss in small groups.

"A C program is like a fast dance on a newly waxed dance floor by people carrying razors."
- Waldi Ravens.

"Fine, Java MIGHT be a good example of what a programming language should be like. But Java applications are good examples of what applications SHOULDN'T be like."

"I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone."
- Bjarne Stroustrup

"PHP is a minor evil perpetrated and created by incompetent amateurs, whereas Perl is a great and insidious evil, perpetrated by skilled but perverted professionals."
- JonRibbens

2. Look at these five newspaper headlines. What do you think the story is behind each one? Discuss your ideas with a partner.

- ***Programmable Routers: More Resilient Networks?***
- ***How Computer Algorithms Shape Our Experience of the Real World***
- ***Computer Scientists Develop Video Game That Teaches How to Program in Java***
- ***Conflictive Animations Support the Development of Programming Skills***
- ***New Technique Safely Combines Programming Languages***

3. Prepare a short presentation (p. 15) on one of the approaches to the programming process such as structured programming, object oriented programming, procedural programmingand so on. Speak about the core elements, usage, advantages and disadvantages and the examples of languages. Use the Internet if you need to check the information. Present your information to the rest of the class.

4. Present to your groupmates the program written by you. Explain to them what languages and tools you used to make it work.

UNIT 3 REVISE AND CHECK

Give the definitions and the translation to the following words:

Programming
Conditional
Looping
Event-controlled loops
Counter-controlled loops
To meet a predetermined condition
To echo
Flow chart
Object Oriented Programming
Procedural Programming
Procedure
Routine
Subroutine
Imperative programming
Top-down language
Method
Class
Blueprint
Pseudocode
Code
To debug
To iterate
Statement
Error
To capitalize
To indent
Value of true
Value of false
Boolean logic
Boolean operator
String
Array
Boolean expression
Variable
Algorithm
Search algorithm
Database query
Sequential search\ Linear search
Optimization
Binary search
Integrated Development Environment
Compiling
Text editor
Syntax checking
Bug
To run a program\To execute a program
Machine language\Machine code
Binary notation
High-level language
Compiler
Interpreter

Can you answer the following questions?

- ✓ What skills should a programmer have?
- ✓ What are five basic elements in programming?
- ✓ What types are loops divided into?
- ✓ When are conditions used? Why?
- ✓ What steps does writing a program include?
- ✓ When is a pseudocode used? What are the main rules when using a pseudocode for creating a programme?
- ✓ How does algorithm work?
- ✓ Why are compiling and debugging are so crucial in programming?

APPENDIX
Articles for review

Teaching computers to understand human languages

Researchers at the University of Liverpool have developed a set of algorithms that will help teach computers to process and understand human languages.

Whilst mastering natural language is easy for humans, it is something that computers have not yet been able to achieve. Humans understand language through a variety of ways for example this might be through looking up it in a dictionary, or by associating it with words in the same sentence in a meaningful way.

The algorithms will enable a computer to act in much the same way as a human would when encountered with an unknown word. When the computer encounters a word it doesn't recognize or understand, the algorithms mean it will look up the word in a dictionary (such as the WordNet), and tries to guess what other words should appear with this unknown word in the text.

It gives the computer a semantic representation for a word that is both consistent with the dictionary as well as with the context in which it appears in the text.

In order to know whether the algorithm has provided the computer with an accurate representation of a word it compares similarity scores produced using the word representations learnt by the computer algorithm against human rated similarities.

Liverpool computer scientist, Dr Danushka Bollegala, said: "Learning accurate word representations is the first step towards teaching languages to computers."

"If we can represent the meaning for a word in a way a computer could understand, then the computer will be able to read texts on behalf of humans and perform potentially useful tasks such as translating a text written in a foreign language, summarising a lengthy article, or find similar other documents from the Internet.

"We are excitingly waiting to see the immense possibilities that will be brought about when such accurate semantic representations are used in various language processing tasks by the computers."

How insights into human learning can foster smarter artificial intelligence

Recent breakthroughs in creating artificial systems that outplay humans in a diverse array of challenging games have their roots in neural networks inspired by information processing in the brain. In a Review published June 14 in*Trends in Cognitive Sciences*, researchers from Google DeepMind and Stanford University update a theory originally developed to explain how humans and other animals learn -- and highlight its potential importance as a framework to guide the development of agents with artificial intelligence.

First published in 1995 (Psychol Rev., 102(3):419-57), the theory states that learning is the product of two complementary learning systems. The first system gradually acquires knowledge and skills from exposure to experiences, and the second stores specific experiences so that these can be replayed to allow their effective integration into the first system. The paper built on an earlier theory by influential British computational neuroscientist David Marr and on then-recent discoveries in neural network learning methods.

"The evidence seems compelling that the brain has these two kinds of learning systems, and the complementary learning systems theory explains how they complement each other to provide a powerful solution to a key learning problem that faces the brain," says Stanford Professor of Psychology James McClelland, lead author of the 1995 paper and senior author of the current Review.

The first system in the proposed theory, placed in the neocortex of the brain, was inspired by precursors of today's deep neural networks. As with today's deep networks, these systems contain several layers of neurons between input and output, and the knowledge in these networks is in their connections. Furthermore, their connections are gradually programmed by experience, giving rise to their ability to recognize objects, perceive speech, understand and produce language, and even to select optimal actions in game-playing and other settings where intelligent action depends on acquired knowledge.

Such systems face a dilemma when new information must be learned: If large enough changes are made to the connections to force the new

knowledge into the connections quickly, it will radically distort all of the other knowledge already stored in the connections.

"That's where the complementary learning system comes in," McClelland says. In humans and other mammals, this second system is located in a structure called the hippocampus. "By initially storing information about the new experience in the hippocampus, we make it available for immediate use and we also keep it around so that it can be replayed back to the cortex, interleaving it with ongoing experience and stored information from other relevant experiences." This two-system set-up therefore allows both immediate learning and also gradual integration into the structured knowledge representation in the neocortex.

"Components of the neural network architecture that succeeded in achieving human-level performance in a variety of computer games like Space Invaders and Breakout were inspired by complementary learning systems theory" says DeepMind cognitive neuroscientist Dharshan Kumaran, the first author of the Review. "As in the theory, these neural networks exploit a memory buffer akin to the hippocampus that stores recent episodes of game play and replays them in interleaved fashion. This greatly amplifies the use of actual game play experience and avoids the tendency for a particular local run of experience to dominate learning in the system."

Kumaran has collaborated both with McClelland and with DeepMind co-founder Demis Hassabis (also a co-author on the Review), in work that extended the role of the hippocampus as it was envisioned in the 1995 version of the complementary learning systems theory.

"In my view," says Hassabis, "the extended version of the complementary learning systems theory is likely to continue to provide a framework for future research, not only in neuroscience but also in the quest to develop Artificial General Intelligence, our goal at Google DeepMind."

'Hanging' computers can be life threatening

When your email program or word processor "hangs" it is annoying, you lose messages or have to reboot your computer and start that writing project again if you hadn't saved the text. But, we depending increasingly on computers in almost all walks of life, not least critical systems such as

air-traffic control, in which the computer "hanging" can be life threatening.

Now, researchers at the Università degli Studi di Napoli Federico II and at Naples company SESM SCARL have developed a software tool that works at the operating system (OS) level and can detect when a computer program "hangs" and so allow a safe exit from any given system without crashing the computer as a whole and requiring a reboot of important systems. Writing in the *International Journal of Critical Computer-Based Systems*, SESM's Gabriella Carrozza explain their detection framework. The framework allows the non-intrusive monitoring of complex systems, based on multiple sources of data gathered at the OS level and the data collected data are then combined to reveal hang failures automatically.

Faults in software represent a major threat to the smooth running of sophisticated computer systems, according to Carrozza and colleagues. Testing and static code analysis are used widely to help detect and remove "bugs" in a system during development. However, once a software system is in place and being used in a real-world application, any number of problems can still occur, perhaps revealing bugs that were missed or simply triggered by memory overloads and timing errors. Such problems can cause just one critical component of the system to "hang" without crashing the whole system and without it being immediately obvious to operators or users of the system that there is a problem until it is too late.

Current software tools simply poll the health status of system components, or analyse system log files to uncover error messages and to correlate these with problematic memory or CPU component activity. However, they cannot spot "hangs" at the time they occur because the system might otherwise respond normally, but for the hanging failure.

The new approach taken by the Italian team relies on several simple monitors which exploit the OS support to trigger alarms when the behaviour of the system differs from the nominal one. "Our experimental results show that this framework increases the overall capacity of detecting hang failures, it exhibits a 100% coverage of observed failures, while keeping low the number of false positives, less than 6% in the worst case," the team says. Response time, or latency, between a hang occurring and it being detected is about 0.1 seconds on average, while

the impact on computer performance of running the hang-detection software is, they add, negligible.

Hardware encryption developed for new computer memory technology

Security concerns are one of the key obstacles to the adoption of new non-volatile main memory (NVMM) technology in next-generation computers, which would improve computer start times and boost memory capacity. But now researchers from North Carolina State University have developed new encryption hardware for use with NVMM to protect personal information and other data.

NVMM technologies, such as phase-change memory, hold great promise to replace conventional dynamic random access memory (DRAM) in the main memory of computers. NVMM would allow computers to start instantly, and can fit more memory into the same amount of space used by existing technologies. However, NVMM poses a security risk.

Conventional DRAM main memory does not store data once the computer is turned off. That means, for example, that it doesn't store your credit card number and password after an online shopping spree. NVMM, on the other hand, retains all user data in main memory even years after the computer is turned off. This feature could give criminals access to your personal information or other data if your laptop or smart phone were stolen. And, because the data in the NVMM is stored in main memory, it cannot be encrypted using software. Software cannot manage main memory functions, because software itself operates in main memory.

NC State researchers have developed a solution using a hardware encryption system called i-NVMM.

"We could use hardware to encrypt everything," explains Dr. Yan Solihin, associate professor of electrical and computer engineering at NC State and co-author of a paper describing i-NVMM, "but then the system would run very slowly -- because it would constantly be encrypting and decrypting data.

"Instead, we developed an algorithm to detect data that is likely not needed by the processor. This allows us to keep 78 percent of main memory encrypted during typical operation, and only slows the system's performance by 3.7 percent."

The i-NVMM tool has two additional benefits as well. First, its algorithm also detects idleness. That means any data not currently in use -- such as your credit card number -- is automatically encrypted. This makes i-NVMM even more secure than DRAM. Second, while 78 percent of the main memory is encrypted when the computer is in use, the remaining 22 percent is encrypted when the computer is powered down.

"Basically, unless someone accesses your computer while you're using it, all of your data is protected," Solihin says.

i-NVMM relies on a self-contained encryption engine that is incorporated into a computer's memory module -- and does not require changes to the computer's processors. That means it can be used with different processors and different systems.

"We're now seeking industry partners who are interested in this technology," Solihin says.

The paper, "i-NVMM: A Secure Non-Volatile Main Memory System with Incremental Encryption," will be presented June 6 at the International Symposium on Computer Architecture (ISCA) in San Jose, Calif. The paper was co-authored by Dr. Siddhartha Chhabra, a former Ph.D. student at NC State. The research was supported, in part, by the National Science Foundation.

Software analyzes apps for malicious behavior

Apps on web-enabled mobile devices can be used to spy on their users. Computer scientists at the Center for Security, Privacy and Accountability (CISPA) developed software that shows whether an app has accessed private data. To accomplish this, the program examines the "bytecode" of the app in question. The researchers show their program at the upcoming computer expo Cebit in Hannover.

Last year at the end of July the Russian software company "Doctor Web" detected several malicious apps in the app store "Google Play." Downloaded on a smartphone, the malware installed -- without the permission of the user -- additional programs which sent expensive text messages to premium services. Although Doctor Web, according to its own statement, informed Google immediately, the malicious apps were still available for download for several days. Doctor Web estimates that in this way up to 25,000 smartphones were used fraudulently.

Computer scientists from the German Saarland University have now developed software which can discover such malicious apps already in the app store. The software detects pieces of code where the app accesses sensitive data and where data is sent from the mobile device. If the software detects a connection between such a "source" and such a "sink," it reports that as suspect behavior. To give an example of such a malicious source-sink combination, Erik Derr explains: "Your address book is read; hundreds of instructions later and without your permission an SMS is sent or a website is visited." Derr is a PhD candidate at the Graduate School of Computer Science and does research at the Center for IT-Security, Privacy and Accountability (CISPA), only a few yards away.

To identify a functional relation between source and sink, the computer scientists from Saarbrücken use new methods of information flow analysis. As input they provide suspicious combinations of accesses on the application programming interface. As the software needs a lot of computational power and storage, it runs on a separate server. "So far we have tested up to 3000 apps with it. The software analyzes them fast enough that the approach can also be used in practice," Derr says.

Conflictive animations support the development of programming skills

Traditional educational tools present information to students in a conventional way: what they present is true and students are expected to learn what is presented. In a PhD study completed recently at the University of Eastern Finland, Andrés Moreno, MSc, developed a tool, Jeliot ConAn, that tricks students during their learning process. Jeliot ConAn uses "conflictive animations" to teach computer programming, which is a very challenging topic for students due to its abstract nature.

The animations in Jeliot ConAn have intentional errors in them. The possibility of errors creates a new set of activities that students can engage with when learning. In Jeliot ConAn, students need to find the error in the animation and signal it in the application when they find it. If successful, the tool will let them know, if not, they will have to keep trying.

The idea of conflictive animations came out when Moreno was discussing his observations regarding Jeliot 3, a programming

visualisation tool, with UEF Professor Erkki Sutinen. In Jeliot 3, students did not pay attention to classical animations that do not have any errors in them. The idea to design conflictive animations was created from the need to engage students with animations, thus helping them understand difficult topics. During programming, students need to keep check of Jeliot ConAn in case the tool behaves in an unexpected way.

Errors and conflicts play a key role in the development of students' programming skills and learning skills in general. The idea of mastering the tool and having to keep check of it can be very empowering for students who find computers unapproachable. Students' skills of critical thinking also get improved, as they are not able to trust the tool.

The evaluation of Jeliot ConAn has led to mixed results and its benefits over classical animations are not fully confirmed yet. Further research involving a larger set of students and a longer time span is needed. However, the study offers a new set of activities that teachers can prepare for their students; these activities have a playful component in them, as students look for errors.

Moreno collected the data for the study in England, Tanzania, Mozambique and South Africa, where students were asked to use both the Jeliot 3 and Jeliot ConAn tools. Jeliot 3 was developed by Moreno and Niko Myller, PhD, at the University of Joensuu (now known as the University of Eastern Finland) under the lead of Professor Sutinen and Professor Moti Ben-Ari, and the tool has been in use all over the world for a decade already.

The results were originally published in IEEE Transactions on Learning Technologies and Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE.

Computer programming: Are two heads really better than one?

Texas Tech professor Miguel Aguirre-Urreta and his colleagues investigated the advantages and perceptions of pair programming from the programmer's standpoint.

It seems like a simple premise - two people on one project can do the job faster and easier and generate a better product.

So why, in computer programming, is there still a significant resistance to sharing the work or at least having someone on the project who can check the work being done to ensure it is of the highest quality? That

was the idea behind the research in a paper recently published by a Texas Tech University professor in the Rawls College of Business.

Miguel Aguirre-Urreta, an assistant professor in the Area of Information Systems and Quantitative Sciences (ISQS), along with colleagues from Washburn University in Kansas and Florida International University, researched the area of pair programming and why some programmers like it, some don't, what makes a good programming pair and at what point programmers come on board with the idea.

The findings of their paper, "Effectiveness of Pair Programming Perceptions of Software Professionals," has been accepted for publication in an upcoming edition of *IEEE Software magazine*.

"The thought has been that pair programming has a lot of purported advantages in terms of speed, quality and whatnot," Aguirre-Urreta said. "But we haven't seen as much of an uptake as you would expect from something that has those advantages. We wanted to see if we could understand the reasons who or which kind of project pair programming is a good idea and for which project it is wasted on, what are the perceptions people who have done this for a living have on pair programming and when it should be used."

When it comes to programming, or writing code for programs, Aguirre-Urreta's research seems to show two heads could indeed be better than one. But the success of having two programmers, called pair programming, also depends on the complexity of the project and the composition of the programming pair involved.

In instances where pair programming is used, Aguirre-Urreta's research shows programmers have a more favorable view of the technique than those who have not participated in pair programming. It also shows once a programmer is involved with pair programming, his view toward the technique is more favorable than before.

"Part of the culture and the kind of work environment is it tends to be more competitive in terms of producing quality code," Aguirre-Urreta said. "Working by yourself is part of the culture. The thing we did talk about in the research and found interesting is people who haven't tried pair programming have a very negative view of it and people who have tried it and done it for a few years have a much better perception of its benefits."

Factors of success Aguirre-Urreta said several factors are involved in determining whether or not pair programming is right for a project and what constitutes a good pairing of programmers.

Complexity of the project seems to be the first determining factor, Aguirre-Urreta said. If it is a simple project that doesn't require much time to complete, then a single programmer is likely the best solution. However, if it is a longer term project requiring a great deal or different types of code, pair programming seems to work well.

"The main advantage to pair programming would be having two people work together on the problem where you get more of a discussion between two people," Aguirre-Urreta said. "You get a better exchange of ideas. It's not a scenario where one person has a certain way of doing things or a certain approach to the problem that they can't break away from because they have someone else working with them."

In a typical pair programming situation, Aguirre-Urreta said the pair works by having one person write the code and the second person checks the quality of the code to see if it could be done better. Eventually, the roles will switch so neither programmer gets burned out doing the same thing for the length of the project.

"Presumably, the quality of the product is going to be much better," Aguirre-Urreta said. "The person doing it by himself is going to find the mistakes at some point but usually it's after someone complains to them that it's not going to work. With pair programming, you should have a better quality product, fewer bugs, a better exchange of ideas and also a knowledge sharing and trading aspect."

Pair programming, however, isn't always the best solution. For one, if the project is a small one, it would be difficult to justify having two people, and thus two salaries, working for its solution unless two people can produce quality code at a much quicker rate. Also, pairing two people means one person may have to explain his coding or work methods to the other frequently, which results in a lengthier period to produce the code.

Aguirre-Urreta said the question is whether that outweighs the factor of working alone and having no one checking the work being done, or if the lone programmer gets stuck on something and has no one there with whom to brainstorm or troubleshoot.

Then there's the factor of the actual makeup of the programming pair. It all depends on the type of project, but Aguirre-Urreta said the research

found that for projects in which pair programming is used as a training tool, having one programmer with a good amount of experience and another who is relatively new often produces the best results. Often pairing two senior programmers or two junior programmers doesn't produce the same results or quality code.

"If the goal is to produce good, nice working software but also train junior programmers and help develop programmers, pairing them with a senior programmer seems to work well," Aguirre-Urreta said. "Again, it depends on the project. There are some combinations that seem to just work better than others."

Push for pair programming Despite the purported advantages to pair programming, Aguirre-Urreta said it has been difficult to get the idea to take hold fully with the programming community.

With every new idea, technique or development, Aguirre-Urreta said there are those who are really willing to try it because they have been beaten down by the previous way of doing things. But once all the enthusiasts are immersed in the new technique, adoption of that new technique slows down, or plateaus.

Much the same can be said for pair programming. Those programmers who have embraced the idea and used it have a much more favorable view of pair programming than those who have resisted its use. It could be a matter of viewing pair programming as positive, but the way they achieve success now has worked well and there's no need to change it.

Eventually, however, Aguirre-Urreta's research discovered once programmers give pair programming a try and use it over a period of time, they eventually come around to its advantages.

"We don't know if it's six months of doing it or a whole year, or it could be three weeks," Aguirre-Urreta said. "We don't know where that click happens and the perception shifts, but we can tell and compare with people who have a fair amount of experience with pair programming to people who haven't done it, there are some marked differences in everything, from benefits to downsizing, cost, all those perceptions."

Once that happens, Aguirre-Urreta said, the change in perception comes quickly.

"We think it's actually the act of just doing it, being there and experiencing working with the other person," Aguirre-Urreta said. "They see that, indeed, their fears that it will take forever to get done are not

really realized. They see the quality of the code is indeed better and there is a huge time-saver. Fixing code is very expensive compared to producing quality code from the get-go."

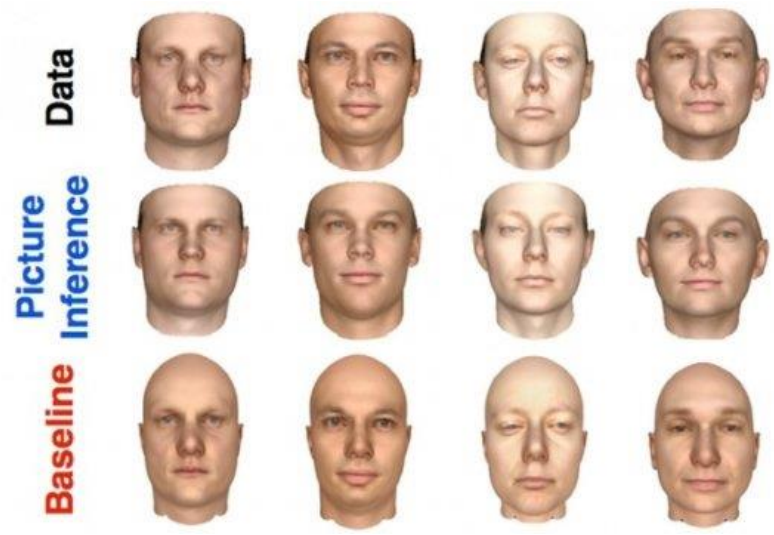
Aguirre-Urreta is hopeful this paper and its appearance in the IEEE Magazine will encourage programmers who have been reluctant to use pair programming to give it a second chance, seeing how popular it is with those who were reluctant to it at one time.

He also would like to further the research by investigating which pairs work best together. He said he and his colleagues have good models in place but haven't always had the best data until now, so he would like to plug that data into the models to see which pairings produce the best increase in productivity.

"The work we're doing now is using those same simulation models, but with the data we have now we feel is more realistic, we'll see what we get out of it," Aguirre-Urreta said. "We have data from different groups and we know how much they agree or disagree, so we can plug that into the models and get results that have some validity based on the data from professionals.

"This research should help motivate someone to say, 'if all the people who are like me and do the same work I do think it's a good idea, maybe I should try it and I don't know what I'm missing.'"

Graphics in reverse: Probabilistic programming does in 50 lines of code what used to take thousands



Two-dimensional images of human faces (top row) and front views of three-dimensional models of the same faces, produced by both a new MIT system (middle row) and one of its predecessors (bottom row).

Most recent advances in artificial intelligence - such as mobile apps that convert speech to text -are the result of machine learning, in which computers are turned loose on huge data sets to look for patterns.

To make machine-learning applications easier to build, computer scientists have begun developing so-called probabilistic programming languages, which let researchers mix and match machine-learning techniques that have worked well in other contexts. In 2013, the U.S. Defense Advanced Research Projects Agency, an incubator of cutting-edge technology, launched a four-year program to fund probabilistic-programming research.

At the Computer Vision and Pattern Recognition conference in June, MIT researchers will demonstrate that on some standard computer-vision tasks, short programs - less than 50 lines long - written in a probabilistic programming language are competitive with conventional systems with thousands of lines of code.

"This is the first time that we're introducing probabilistic programming in the vision area," says Tejas Kulkarni, an MIT graduate student in brain and cognitive sciences and first author on the new paper. "The whole hope is to write very flexible models, both generative and discriminative models, as short probabilistic code, and then not do anything else. General-purpose inference schemes solve the problems."

By the standards of conventional computer programs, those "models" can seem absurdly vague. One of the tasks that the researchers investigate, for instance, is constructing a 3-D model of a human face from 2-D images. Their program describes the principal features of the face as being two symmetrically distributed objects (eyes) with two more centrally positioned objects beneath them (the nose and mouth). It requires a little work to translate that description into the syntax of the probabilistic programming language, but at that point, the model is complete. Feed the program enough examples of 2-D images and their corresponding 3-D models, and it will figure out the rest for itself.

"When you think about probabilistic programs, you think very intuitively when you're modeling," Kulkarni says. "You don't think mathematically. It's a very different style of modeling."

Joining Kulkarni on the paper are his adviser, professor of brain and cognitive sciences Josh Tenenbaum; Vikash Mansinghka, a research

scientist in MIT's Department of Brain and Cognitive Sciences; and Pushmeet Kohli of Microsoft Research Cambridge. For their experiments, they created a probabilistic programming language they call Picture, which is an extension of Julia, another language developed at MIT.

What's old is new

The new work, Kulkarni says, revives an idea known as inverse graphics, which dates from the infancy of artificial-intelligence research. Even though their computers were painfully slow by today's standards, the artificial intelligence pioneers saw that graphics programs would soon be able to synthesize realistic images by calculating the way in which light reflected off of virtual objects. This is, essentially, how Pixar makes movies.

Some researchers, like the MIT graduate student Larry Roberts, argued that deducing objects' three-dimensional shapes from visual information was simply the same problem in reverse. But a given color patch in a visual image can, in principle, be produced by light of any color, coming from any direction, reflecting off of a surface of the right color with the right orientation. Calculating the color value of the pixels in a single frame of "Toy Story" is a huge computation, but it's deterministic: All the variables are known. Inferring shape, on the other hand, is probabilistic: It means canvassing lots of rival possibilities and selecting the one that seems most likely.

That kind of inference is exactly what probabilistic programming languages are designed to do. Kulkarni and his colleagues considered four different problems in computer vision, each of which involves inferring the three-dimensional shape of an object from 2-D information. On some tasks, their simple programs actually outperformed prior systems. The error rate of the program that estimated human poses, for example, was between 50 and 80 percent lower than that of its predecessors.

Learning to learn

In a probabilistic programming language, the heavy lifting is done by the inference algorithm -- the algorithm that continuously readjusts probabilities on the basis of new pieces of training data. In that respect, Kulkarni and his colleagues had the advantage of decades of machine-

learning research. Built into Picture are several different inference algorithms that have fared well on computer-vision tasks. Time permitting, it can try all of them out on any given problem, to see which works best.

Moreover, Kulkarni says, Picture is designed so that its inference algorithms can themselves benefit from machine learning, modifying themselves as they go to emphasize strategies that seem to lead to good results. "Using learning to improve inference will be task-specific, but probabilistic programming may alleviate re-writing code across different problems," he says. "The code can be generic if the learning machinery is powerful enough to learn different strategies for different tasks."

"Picture provides a general framework that aims to solve nearly all tasks in computer vision," says Jianxiong Xiao, an assistant professor of computer science at Princeton University, who was not involved in the work. "It goes beyond image classification -- the most popular task in computer vision -- and tries to answer one of the most fundamental questions in computer vision: What is the right representation of visual scenes? It is the beginning of modern revisit for inverse-graphics reasoning."

GLOSSARY

Algorithm - a set of steps that are followed in order to solve a mathematical problem or to complete a computer process
Application - a program designed to do a particular job; a piece of software\ the practical use of something, especially a theory, discovery, etc.
Application software - a computer program designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user
Array - a container object that holds a fixed number of values of a single type
Artificial Intelligence - an area of study concerned with making computers copy intelligent human behavior

Basic Input/Output System - a type of firmware used to perform hardware initialization during the booting process (power-on startup) on IBM PC compatible computers, and to provide runtime services for operating systems and programs
Binary notation - expression of a number with a base of 2 using only the digits 0 and 1 with each digital place representing a power of 2 instead of a power of 10 as in decimal notation
Binary search - a search algorithm that finds the position of a target value within a sorted array
Bit - the smallest unit of data in a computer which has a single binary value, either 0 or 1
Blueprint - something which acts as a plan, model, or template for others
Boolean expression - an expression in a programming language that produces a Boolean value when evaluated, i.e. one of true or false.
Boolean logic - a subset of algebra used for creating true/false statements
Boolean operator - simple words (AND, OR, NOT or AND NOT) used as conjunctions to combine or exclude keywords in a search, resulting in more focused and
Break (broke, broken) down - to stop working because of a fault, to fail
Bug - a coding error in a computer program

Calculation - the act or process of using numbers to find out an amount
Capitalize - to write or print a letter of the alphabet as a capital; to begin a word with a capital letter

Central processing unit - the part of a computer that controls all the other parts of the system
Circuit - the complete path of wires and equipment along which an electric current flows
Class - a portion of source code in an object-oriented programming language such as Java
Cloud computing - a way of using computers in which data and software are stored or managed on a network of servers (=computers that control or supply information to other computers), to which users have access over the Internet
Code - program instruction
Compiler - a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code
Computer architecture - a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform
Computer case, Tower - plastic or metal housing that contains the computer's main parts such as the motherboard, hard drive, etc.
Computer peripheral – a device that is used to put information into or get information out of the computer
Conditional - an if-statement
Consume - to use something, especially fuel, energy or time
Cooling unit- a device that removes the waste heat produced by computer components
Counter- controlled loop - one that is executed a certain number of times
Crack- to break through (as a barrier) so as to get access to confidential information

Data - information that is stored by a computer
Database - an organized set of data that is stored in a computer and can be looked at and used in various ways
Database query - a request for data or information from a database table or combination of tables
Debug - to identify and remove errors

Design - to decide how something will be made, including how it will work and what it will look like, and often to make drawings of it
Device - an object or a piece of equipment that has been designed to do a particular job
Digital computer - a computer that works with numbers that are represented by the digits 0 and 1
Disk operating system - an operating system with a command-line interface used on personal computers

Error - failure or fault
Event-controlled loop - one whose execution is controlled by the occurrence of an event within the loop itself
Expansion slot - a socket on the motherboard that is used to insert an expansion card (or circuit board), which provides additional features to a computer such as video, sound, advanced graphics, Ethernet or memory

Firmware - a type of computer software that is stored in such a way that it cannot be changed or lost
Flash drive – a small memory device that can be used to store data from a computer and to move it from one computer to another
Flow chart - a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure

Graphical user interface - a way of giving instructions to a computer using things that can be seen on the screen such as symbols and menus

Hand-held device - any portable unit that can be carried and held in one's palm
Hard disk drive - a type of low-cost, high-capacity physical storage used for random-access data in PCs and enterprise data centers
Hard drive - a non-volatile memory hardware device that permanently stores and retrieves information
Hardware - the physical components of a computer
Headphones - a small speaker that is worn over or in the ear, often as one of a pair connected by a wire attached to a band running over the head.
High-level language - a programming language with strong abstraction from the details of the computer

Hyperlink - a place in an electronic document that is connected to another electronic document or to another part of the same document

Imperative programming - a programming paradigm that uses statements that change a program's state.
Indent - to start a line of print or writing further away from the edge of the page than the other lines
Input device - a hardware or peripheral device used to send data to a computer
Integrated circuit - a small microchip that contains a large number of electrical connections and performs the same function as a larger circuit made from separate parts
Integrated Development Environment - a software application that provides comprehensive facilities to computer programmers for software development.
Interpreter - a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program.
Iterate - to make repeated use of a mathematical or computational procedure, applying it each time to the result of the previous application

Laptop - portable and compact personal computer with the same capabilities as a desktop computer
Linear search - a method for finding a target value within a list
Looping - the process of a software program or script repeats the same instructions or processes the same information over and over until receiving the order to stop

Machine code - a set of instructions executed directly by a computer's central processing unit (CPU)
Maintenance - the act of keeping something in good condition by checking or repairing it regularly
Method - a programmed procedure that is defined as part of a class and included in any object of that class
Motherboard - the main printed circuit board (PCB) found in general purpose microcomputers and other expandable systems
Multiplication - the act or process of multiplying

Non-volatile storage - a type of computer memory that can retrieve stored information even after having been power cycled (turned off and back on)
Object-oriented programming - a programming that incorporates other segments of external code.
Operating system - system software that manages computer hardware and software resources and provides common services for computer programs.
Optical disk drive - a disk drive that uses laser light or electromagnetic waves within or near the visible light spectrum as part of the process of reading or writing data to or from optical discs
Output device - any device used to send data from a computer to another device or user
Overheating - the process of becoming too hot

Perform – to carry out
Procedural programming - a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program. It contains a systematic order of statements, functions and commands to complete a computational task or program.
Procedure - a set of instructions designed to perform a frequently used operation within a program
Process - to perform a series of operations on data in a computer
Processor - a part of a computer that controls all the other parts of the system productive results.
Programming - the act of an individual writing code (a set of instructions) that are to be interpreted and executed by a computer or other electronic device
Programming language - a special language that programmers use to develop software programs, scripts, or other sets of instructions for computers to execute
Pseudocode - a computer programming language that resembles plain English that cannot be compiled or executed, but explains a resolution to a problem

Random access memory (RAM) - a type of storage for computer systems that makes it

Random-access memory (RAM) - computer memory in which data can be changed or removed and can be looked at in any order
Read-only memory - computer memory that contains instructions or data that cannot be changed or removed
Reliability - the quality or state of being fit to be trusted or relied on
Researcher - a person who studies something carefully and tries to discover new facts about it
Routine - a sequence of instructions for performing a task that forms a program or a distinct part of one.

Search algorithm - an algorithm that retrieves information stored within some data structure
Sequential search - synonyms for linear search
Silicon chip - a very small piece of silicon used to carry a complicated electronic circuit
Software - the programs used to operate a computer
Software suite - a collection of several applications that are bundled together and sold or distributed as a package
Speaker - a piece of equipment that sends out the sound from a CDplayer, radio etc.
Spreadsheet - a computer program that is used, for example, when doing financial or project planning. You enter data in rows and columns and the program calculates costs, etc. from it
Statement - the smallest standalone element of an imperative programming language that expresses some action to be carried out
Storage - the part of a computer that stores information for subsequent use or retrieval
Store - to keep information or facts
String - any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks)
Subroutine - a portion of code that may be called and executed anywhere in a program
Switch - an electrical component which can break an electrical circuit, interrupting the current or diverting it from one conductor to another
Systems software - a type of computer program that is designed to run a computer's hardware and application programs

Tablet - a wireless touch screen personal computer that is smaller than a notebook but larger than a smartphone
Text editor - computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs)
Top-down language - a language of programming where an application is constructed starting with a high-level description of what it is supposed to do, and breaking the specification down into simpler and simpler pieces, until a level has been reached that corresponds to the primitives of the programming language to be used

Unit - a small machine that has a particular purpose or is part of a larger machine
Universal Serial Bus - an external serial bus interface standard for connecting peripheral devices
Utilize - to use something, especially for a practical purpose,to make use of smth to a computer

Vacuum tube - a glass tube that was used in the past in computers, televisions, etc., to control the flow of electricity
Variable - a value that can change, depending on conditions or on information passed to the program

Watt - a unit for measuring electrical power
Web browser/ browser - a program that lets you look at or read documents on the World Wide Web

REFERENCES

1. A Brief History of the Computer [Электронныйресурс]: http://www.seattlecentral.edu/~ymoh/history_of_computer/history_of_computer.htm
2. A Short History of Computers and Computing [Электронныйресурс]: http://clas.mq.edu.au/speech/synthesis/history_computers/
3. The Evolution of Computers, 1st, 2nd, 3rd, 4th Generation, and More to Come [Электронныйресурс]: http://www.nortonsecurityonline.com/security-center/evolution-of-computers.html
4. What is Computer Hardware? - Components, Definition & Examples [Электронныйресурс]: http://study.com/academy/lesson/what-is-computer-hardware-components-definition-examples.html
5. 5 Basic Elements Of Programming [Электронныйресурс]: https://study.com/academy/lesson/5-basic-elements-of-programming.html
6. Getting Started with C or C++ [Электронныйресурс]: http://www.cprogramming.com/begin.html
7. Pseudocode: Definition & Examples [Электронныйресурс]: https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html
8. Making a presentation: language and phrases [Электронныйресурс]: http://speakspeak.com/resources/general-english-vocabulary/presentation-language-phrases
9. IEEE Standard 610-90 (Standard Glossary of Software Engineering Terminology) [Электронныйресурс]: http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf
10. Boolean Logic, Operators & Expressions [Электронныйресурс]: https://study.com/academy/lesson/boolean-logic-operators-expressions.html
11. What is a Computer Algorithm? - Design, Examples & Optimization [Электронныйресурс]: https://study.com/academy/lesson/what-is-a-computer-algorithm-design-examples-optimization.html

12. New design of primitive quantum computer finds application [Электронныйресурс]: http://www.bristol.ac.uk/news/2016/may/quantum-walk.html
13. The language of critical review [Электронныйресурс]: https://academicskills.anu.edu.au/?q=print/494
14. Hardware, Software Advances Help Protect Operating Systems From Attack [Электронныйресурс]: https://news.ncsu.edu/2011/01/wmssolihinos/
15. What Are Peripheral Devices of a Computer? - Definition, Examples & Types [Электронныйресурс]: http://study.com/academy/lesson/what-are-peripheral-devices-of-a-computer-definition-examples-types.html
16. How to read a computer ad [Электронныйресурс]: http://www.samos.aegean.gr/english/EnglishforIT/Exercises/IcseReadingComprehensionOnline/math_unit_2_reading.htm
17. Teaching computers to understand human languages [Электронныйресурс]: https://www.sciencedaily.com/releases/2016/05/160506105650.htm
18. How insights into human learning can foster smarter artificial intelligence [Электронныйресурс]: https://www.sciencedaily.com/releases/2016/06/160614133609.htm
19. 'Hanging' computers can be life threatening [Электронныйресурс]: https://www.sciencedaily.com/releases/2011/08/110826085153.htm
20. Hardware encryption developed for new computer memory technology [Электронныйресурс]: https://www.sciencedaily.com/releases/2011/05/110517110308.htm
21. Software analyzes apps for malicious behavior [Электронныйресурс]: https://www.sciencedaily.com/releases/2014/03/140307084013.htm
22. Conflictive animations support the development of programming skills [Электронныйресурс]: http://www.eurekalert.org/pub_releases/2014-09/uoef-cas092614.php
23. Computer programming: Are two heads really better than one? [Электронный ресурс]:https://www.sciencedaily.com/releases/2015/11/151124082459.htm

24. Graphics in reverse: Probabilistic programming does in 50 lines of code what used to take thousands [Электронныйресурс]: https://www.sciencedaily.com/releases/2015/04/150413110630.htm

25. What's old is new [Электронныйресурс]: http://news.mit.edu/2015/better-probabilistic-programming-0413