Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

А.В. Терещенко

ENGLISH FOR PROGRAMMERS. PART I

Учебное пособие
для магистрантов ФСУ, ФВС

Томск
2025

УДК 811.111:004(075.8)
ББК 81.432.1я73
Т 350

Рецензент:

Балонкина Ольга Викторовна,

кандидат филологических наук, доцент кафедры иностранных языков ФГАОУ ВО «Томский государственный университет систем управления и радиоэлектроники»

А.В. Терещенко

English for Programmers. Part I: учебное пособие для магистрантов ФСУ, ФВС / А.В. Терещенко. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2025. – 80 с.

Учебное пособие предназначено для магистрантов факультета систем управления (ФСУ) и факультета вычислительных систем (ФВС). Пособие содержит четыре раздела: “Career in IT” (Карьера в области информационных технологий), “Computers” (Компьютеры), “Programming languages and paradigms” (Языки программирования и парадигмы программирования), “Database management systems” (Системы Управления Базами Данных, СУБД). Каждый раздел пособия включает комплекс лексико-грамматических упражнений (Language Development), два текста для чтения (базового и повышенного уровня сложности), глоссарий, задания для проектной работы. Раздел “Focus on Writing” дает представление об особенностях написания резюме при устройстве на работу, знакомит с правилами описания графиков, таблиц, диаграмм и схем. Раздел “Improve your translation skills” содержит упражнения по переводу профессионально-ориентированных фрагментов текста с русского языка на английский язык.

Одобрено на заседании кафедры иностранных языков (ИЯ), протокол № 6 от 12.11.2024 года

УДК 811.111:004(075.8)
ББК 81.432.1я73

CONTENTS

Введение ……...…..4

Unit 1 “Career in IT” ……6

Unit 2 “Computers” ……23

Unit 3 “Programming languages and paradigms” ……38

Unit 4 “Database management systems” ……54

Заключение ……..……....65

Список условных сокращений …… 66

Список использованных источников ……67

Приложение ……68

ВВЕДЕНИЕ

Настоящее пособие предназначено для магистрантов ФСУ и ФВС, обучающихся по следующим направлениям подготовки: *«Программная инженерия», «Информатика и вычислительная техника», «Прикладная информатика».*

Уверенное владение английским языком необходимо современному IT-специалисту, поскольку для значительной части языков программирования английский язык используется в интерфейсах и командных строках, большинство сред разработки (в т.ч. визуальных) представлено исключительно на английском языке. Знание профессиональной терминологии позволит IT-специалистам лучше разбираться в технической документации.

Пособие состоит из четырех разделов:

- раздел 1 “Carrer in IT” («Карьера в области информационных технологий»);
- раздел 2 “Computers” («Компьютеры»);
- раздел 3 “Programming languages and paradigms” («Языки программирования и парадигмы программирования»);
- раздел 4 “Database management systems” («Системы Управления Базами Данных, СУБД»).

Структура каждого радела включает:

- обсуждение в парах/небольших группах вопросов по теме *(Lead-in);*
- работу с двумя текстами профессиональной направленности *(Reading I and Reading II);*
- изучение лексики *(List of vocabulary);*
- комплекс лексико-грамматических упражнений *(Language development).*

Следует обратить внимание, что тексты для чтения отличаются по уровню сложности. Второй текст для чтения *(Reading II)* повышенного уровня сложности, содержит вокабуляр профессиональной направленности и сложные грамматические конструкции, поэтому автор рекомендует использовать этот текст для самостоятельной работы.

В пособии уделяется внимание совершенствованию навыков перевода с русского языка на английский предложений, содержащих профессиональную терминологию *(подраздел “Improve your translation skills”).*

Автор пособия считает немаловажным развитие навыков письменной речи у магистрантов, поскольку грамотная письменная речь – фундамент правильной устной речи. Каждый раздел пособия включает темы, направленные на совершенствование навыков письма. Так, в первом и втором разделах рассматриваются отдельные письменные жанры (сопроводительное письмо, резюме при устройстве на работу в международную IT-компанию, описание графиков, схем, алгоритмов). В третьем и четвертом разделах рассказывается о некоторых языковых особенностях письма (передача оттенков модальности в письменной речи, глаголы формального стиля, лингвистические средства нейтрализации речевого высказывания).

При изучении материала пособия предполагается выполнение магистрантами проектной работы *(Project work),* поиск информации в Интернете *(задание: Surf the Internet).*

В пособии предусмотрена проектная работа *(Project work),* которая способствует совершенствованию умений работать в команде, понимать собеседника и формировать собственную точку зрения на основе полученного опыта. Преподаватель может использовать раздел *“Project work”* как для самостоятельной работы учащихся, так и в качестве итогового задания по пройденной теме.

В «Приложении» к настоящему учебному пособию представлены дополнительные тексты и комплекс упражнений к текстам. Тексты можно использовать для организации самостоятельной работы магистрантов, как источник дополнительной информации к таким разделам пособия, как Unit 2 “Computers” («Компьютеры»), Unit 3 “Programming languages and paradigms” («Языки программирования и парадигмы программирования»).

Автор надеется, что материал, представленный в пособии, будет способствовать активному вовлечению студентов в изучение иностранного языка профессиональной направленности. Работа с текстами (базового и повышенного уровня сложности) позволит расширить словарный запас, выполнение комплекса лексико-грамматических упражнений позволит лучше освоить лексику и уверенно использовать иностранный язык в ситуациях общения.

Успехов в получении новых знаний!

UNIT 1

CAREER IN IT

LEAD-IN

1. Why have you chosen to get education for a job in IT sphere?
2. What IT specialists do you know?
3. What are the responsibilities of different computer specialists?
4. What skills do you already have for your job? What skills do you need to develop?

Different types of IT jobs and their responsibilities

a) Here you can find the list of responsibilities.

b) Look through the list of responsibilities. Find unknown words, use the dictionary to consult their meaning.

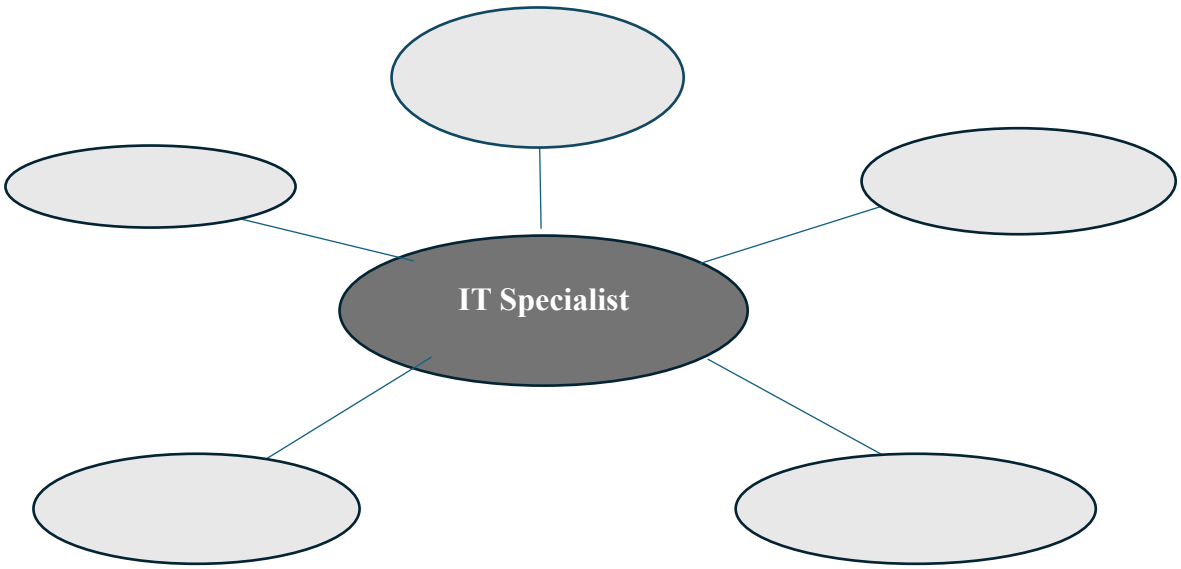
c) If you work in IT, what are your responsibilities? What are your colleagues responsible for? Discuss it with your partner.

to work on the full range of development activities – analysis, design, coding, testing and implementation
to perform formal analysis of operational needs
to run data-processing equipment (data control and editing)
to facilitate systems integration
to provide messaging, data storage, networking
to develop and maintain web-based applications
to possess analytical problem solving skills
to handle customer support calls
to set up equipment
to maintain security of documents and customers
to install, configure and maintain software and hardware systems

to assess potential risks
to conduct trainings to new hires, users and technical teams as needed
to recommend process improvements
to ensure system reliability, security, integrity and performance
to conduct computer diagnostics
to extract and analyze information
to conduct computer diagnostics
to have good communication, persuasion and sensitivity
to deal with project management and problem solving
to be able to work in a team
to have patience and diplomacy, to stay calm under pressure
to have mobility and business awareness

MAKING A CLUSTER

a) Choose any IT job and make a cluster presenting responsibilities. Use ideas from table above to make your cluster.
b) Be ready to ask and answer questions while you work in pairs.



READING I

Learn new words and expressions before you read the text.

fast growing	быстро развивающийся, востребованный, быстро растущий
appear, *v.*	появляться, возникать
arise, *v.*	возникать, формироваться *(context)*
pursue, *v.*	стремиться, преследовать цель, проявлять интерес, рассматривать, придерживаться
pursue a career, *expr.*	выбрать профессию, делать карьеру *(context)*
pros and cons, *expr.*	преимущества и недостатки
entry-level job	низшая должность специалиста (с дипломом, но без опыта работы), стартовая должность, должность при приеме на работу (context)
be an exception, *expr.*	быть исключением
get ahead, go ahead, *ph.v.*	преуспевать, добиться успеха, достичь успеха, опережать, достичь цели (context)
salary	зарплата, оклад, должностной оклад
varied, *adj.*	различный, многообразный, разнообразный
versatile, *adj.*	многогранный, многосторонний, разнообразный, разноплановый (context)
software engineer	разработчик программного обеспечения, инженер-программист
data analyst	аналитик данных
cybersecurity expert	специалист по кибербезопасности, эксперт по информационной безопасности в компьютерных сетях
strengths	сильные стороны, положительные качества (context)
non-strenuous, *adj.*	не требующий усилий
assignment, *n.*	задание, задача, обязанность *(context)*
release, *v.*	выпускать программный продукт
handle, *v.*	управлять, решать, обращаться с чем-либо

meet deadline, *fixed expr.*	исполнять работу в срок, своевременно выполнять работу
stay ahead, *ph.v.*	быть впереди, работать на опережение *(context)*

Career in IT: advantages and disadvantages

Computer and information systems are one of the most fast growing industries in the world. With so many new information technologies appearing, it is no surprise that new job opportunities arise constantly and more and more people are interested in pursuing a career in this market. However, before choosing an IT career, it's a good idea to learn its pros and cons.

IT Career Advantages

✓ *High income*

Entry-level jobs may be the exception, but once you get ahead in IT, you are looking at a very high salary. IT professionals are well-paid and live a financially stable life, which is enough for many to consider getting into the industry.

✓ *Varied job opportunities*

IT is an amazingly versatile industry. You can choose from a wide range of job opportunities, such as becoming a software engineer, video game designer, web designer, 3D animator, data analyst, systems analyst, cybersecurity expert, and so much more. Wherever your strengths lie, IT will have a place for you.

✓ *Non-strenuous work*

Sitting at a desk for at least eight hours a day has its disadvantages, such as developing poor posture or gaining weight. However, sitting in a comfy office is hardly strenuous work. In IT you won't find a problem of working outdoors in bad weather conditions. Moreover, you might get the perk of working from home.

✓ *Dynamic work assignments*

In IT, no two projects are the same. You can be making a mobile app one day and a web platform the next. Dynamic work of this type requires a dynamic personality. If you enjoy challenges, then IT will be a good fit for you.

IT Career Disadvantages

✓ *Stressful environment*

Deadlines are very important in IT. Projects need to be planned out, developed, tested, and sent back to the client or released on a strict schedule. The development team usually has no control

over this schedule, since managers handle that side of the business. So, you can't make many errors or work slowly. If you decide to work freelance, then you will face clients who have no IT knowledge and cannot explain what they want from you or the software you're making for them. So, be prepared to deal with a lot of miscommunication and constant stress to meet deadlines in this kind of work.

✓ *Little free time*

Most jobs require sacrificing free time. IT is not an exception. Even if you're doing everything right, it may happen that you need to cancel a family picnic or a romantic dinner, or work weekends when faced with a problem or a project that is running late.

✓ *Continual training and education*

This last con might be considered a pro for some. IT is constantly developing, new software, technologies, and programming languages are constantly being released, and it's not easy to keep up with all of that. To stay ahead of your competitors, you will need to constantly educate yourself. This isn't always an easy thing to do, especially if you are working long hours.

LANGUAGE DEVELOPMENT

Exercise 1.1. Find the right answer. Sometimes more than one answer can be correct.

1. What is true about IT industry?
    a. New job opportunities appear continually.
    b. IT industry grows at a fast pace.
    c. There are no cons to an IT career.
    d. More and more people want to sell things at markets.

2. What is true about IT career advantages?
    a. It is a lie that your strengths can be useful in IT.
    b. IT work involves a lot of walking on foot.
    c. In IT you may get the perk of working outdoors.
    d. There is a wide range of job opportunities to choose from.

3. What is NOT true about IT career disadvantages?
    a. There are strict deadlines for software releases.
    b. The schedule is usually controlled by the managers.
    c. Clients always know exactly what software they want you to make for them.
    d. If you are doing everything right, you finish things on time and don't have to work late.

4. What is NOT true about sitting at a desk for 8 hours a day?
    a. It can be a disadvantage.
    b. It causes a person to lose weight.
    c. It is strenuous work.
    d. Some people prefer it to working outdoors in poor weather conditions.

5. What is true about stress on an IT job?
    a. It is often caused by miscommunication.
    b. Trying to keep up with constantly evolving technologies can be stressful.
    c. You are only stressed on weekdays during working hours.
    d. You live in a constant chase to meet deadlines.

Exercise 1.2. a) Arrange the words in appropriate order to make up questions.

1. opportunities/do/why/constantly/job/arise/new/in IT sphere/?
……

2. /enough/getting/for many people/what/IT/to consider/industry/is/into
……

3. job/what/in IT/are/opportunities/there/?
……

4. working/are/at/the disadvantages/an office/what/of/?
……

5. working/an advantage/at/is/an office/what/of/?
……

6. your/need/you/competitors/do/of/to stay ahead/what/?
……

7. if you/will/to work/you/problems/decide/what/face/freelance/?
……

8. a problem at work /when/you/cancel/need to/may/what/faced with?
……

9. does/work/require/what/a dynamic/?
……

b. Work in pairs and ask each other questions from part A of this exercise.

Exercise 1.3. Use the appropriate preposition from the box to complete the sentences.

in with (x 2) from (x 2) on out at (x 2) of for

1. If you enjoy challenges, then IT will be a good fit ____ you.
2. Sitting ____ a desk for at least eight hours a day has its disadvantages.
3. Projects need to be planned _____, developed, tested, and sent back to the client.
4. Be prepared to deal _____ a lot of miscommunication.
5. To stay ahead____your competitors, you need to constantly educate yourself.
6. Sometimes clients can not explain what they want _____ you.
7. In IT sector you can choose _____ a wide range of job opportunities.
8. More and more people are interested _____ pursuing a career in IT.
9. It's not easy to keep ____ with new technologies constantly being released.
10. You may need to work weekends when faced _____ a problem.

11. Once you get ahead in IT, you are looking _____ a very high salary.

Exercise 1.4. Revise the vocabulary from the text "Career in IT: advantages and disadvantages". Fill in the blanks with an appropriate vocabulary item from the box.

salary range training environment pursue face a good fit competitors require

1. A career in IT makes sure you are not stuck in a particular position for a long period, giving you the option of a better work ____with better benefits.
2. If you constantly learn new things, you will be able to stay ahead of your____.
3. To adapt to the constant change in IT industry, you will need regular ____.
4. If you can not deal with stress well, you may not be ____ for IT industry.
5. Before you ____ a career in IT, you need to know its pros and cons.
6. Systems analyst has an average annual ____ of $76,300.
7. Working in IT allows you to dive into a wide ____ of industries and sectors.
8. If you decide to work freelance, then you will ____ clients who have no IT knowledge.
9. Managers ____ meeting deadlines no matter what it takes.

Exercise 1.5. Match the paragraphs (1-7) with the corresponding advantage (a-g).

a. High income
b. Varied job opportunities
c. Non-strenuous work
d. Dynamic work assignments
e. Stressful environment
f. Little free time
g. Continual training and education

1. Working from home, you don't have to worry about questionable lunch break food choices, since you can eat out of your own kitchen!
2. You will need to constantly undergo training for what's new and potentially better than what you've been using up until that point.
3. You don't necessarily have to be good at math, logical operations, and algorithms if you want a job in computing technology – almost all kinds of skill sets are welcome here.
4. "Statista" website declares that IT professionals working in aerospace and defense, communications, public relations, and advertising, as well as the pharmaceutical, medical, and biotech industry, are enjoying the highest salary.
5. Sometimes you will need to put aside your free time and get the project done, no matter what it takes.
6. If you like to take on new problems to solve, you will enjoy working in IT. Even if you work in one specific branch – for example, virtual reality – you will meet different clients who will have different assignments for you.
7. IT projects often deal with large sums of money, so if they encounter technical glitches, managers will put pressure on you to make sure that all issues are resolved as quickly as possible.

READING II

Learn new words and expressions before you read the text.

system-level software	системное программное обеспечение, системное ПО
embedded system	встроенная система, встраиваемая система
asses, *v.*	оценивать, давать оценку, определять, анализировать *(context)*
feasible solution, *expr.*	возможное решение, допустимое решение, приемлемое решение, осуществимое решение *(context)*
implement a project	осуществить проект, реализовать проект
troubleshooter, *n.*	специалист, выявляющий неисправности *(context)*
end-user, *adj.*	пользовательский (context)
maintain, *v.*	обслуживать, поддерживать, обеспечивать функционирование
administer, *v.*	администрировать, осуществлять административные функции
good understanding of code, *expr.*	хорошее понимание программного кода
test script	тестовая процедура, сценарий тестирования, тестовый скрипт *(проф.)*
macros	макрокоманда, макроопределение, макроэлементы, макросы *(проф.)*
pitfall, *n.*	ошибка, проблема, недостаток
pitch for, *phr. v.*	представить, подать идею *(context)*
refine specifications	совершенствовать спецификации, стандарты, детализировать технические характеристики
database management system	система управления базами данных, СУБД *(проф.)*
database integrity	целостность базы данных (БД)

Paragraphs 1-10 represent descriptions of different jobs in computing. Read paragraphs (1-10) and match the description with the job in computing (a-j).

Jobs in computing

A. Business analyst
B. Software engineer
C. Systems analyst
D. Software tester
E. Technological consultant
F. Network engineer
G. Technical support manager
H. Web developer
I. Computer programmer
J. Database administrator

1. The work of a _____ typically includes designing and programming system-level software: operating systems, database systems, embedded systems and so on. They understand how both software and hardware function. The work can involve talking to clients and colleagues *to assess* and define what solution or system is needed, which means there's a lot of interaction as well as full-on technical work. _____ are often found in electronics and telecommunications companies.

2. ____ investigate and analyze business problems and then design information systems that provide a feasible solution, typically in response to requests from their business or a customer. They gather requirements and identify the costs and the time needed *to implement* the project. The job needs a mix of business and technical knowledge, and a good understanding of people. It's a role for _____ to move into and typically requires a few years' experience from graduation.

3. These are the professional troubleshooters of the IT world. Many _____ work for hardware manufacturers and suppliers solving the problems of business customers or consumers, but many work for end-user companies supporting, monitoring and maintaining workplace technology and responding to users' requests for help. Some lines of support require professionals with specific experience and knowledge, but _____ can also be a good way into the industry for graduates.

4. _____ is one of the more technically demanding IT jobs. Broadly speaking the role involves setting up, *administering,* maintaining and upgrading communication systems, local area networks and wide area networks for an organization. _____ are also responsible for security, data storage and disaster recovery strategies. It is a highly technical role and you'll gather a wide range of specialist technical certifications as you progress. A telecoms or computer science-related degree is needed.

5. _____ covers everything to do with building websites and all the infrastructure that sits behind them. The job is still viewed as the trendy side of IT years after it first emerged. These days _____is a pretty technical job and involves some hardcore programming as well as the more creative side of designing the user interfaces of new websites. The role can be found in organizations large and small.

6. Bugs can have a massive impact on the productivity and reputation of an IT firm. _____ try *to anticipate* all the ways an application or system might be used and how it could fail. They don't necessarily program but they do need a good understanding of code. _____ prepare test scripts

and macros, and analyze results, which are fed back to the project leader so that fixes can be made. _____ can also be involved at the early stages of projects in order to anticipate pitfalls before work begins. You can potentially get to a high level as a_____.

7. Typically the _____ provide technical expertise to, and develop and implement IT systems for external clients. They can be involved at any or all stages of the project lifecycle: pitching for a contract; refining a specification with the client team; designing the system; managing part or all of the project; after sales support or even developing the code. A technical degree is preferred, but not always necessary.

8. ____ are true midfielders, equally happy talking with technology people, business managers and end users. They identify opportunities for improvement to processes and business operations using information technology. The role is project based and begins with analyzing a customer's needs, gathering and documenting requirements and creating a project plan to design the resulting technology solution. _____ need technology understanding, but don't necessarily need a technical degree.

9. _____ are responsible for the storage, organization, and management of electronic data. Their job is connected with developing and maintaining the computer database of various business organizations. In performing their duties, _____ install and test new database management systems. They carry out assessments to identify user needs and develop database solutions effective in meeting those needs. They also implement security plans to maintain database integrity and protect against cyber-attacks. _____ ensure company database systems *function* efficiently and meet the requirements of an organization.

10. _____ write and test code that allows computer applications and software programs to function properly. They turn the program designs created by software developers and engineers into instructions that a computer can follow. In addition, they test newly created applications and programs to ensure that they produce the expected results. If they do not work correctly, computer _____ check the code for mistakes and fix them. They work closely with software developers, and in some businesses their duties overlap. When such overlap occurs, they can do work that is typical of developers, such as designing programs.

LANGUAGE DEVELOPMENT

Exercise 1.6. a) Find English equivalents for the following Russian expressions.

1. тестировать созданные приложения и программы
2. обеспечивать эффективную работу баз данных компании
3. защищать от кибератак
4. разрабатывать пользовательские интерфейсы вебсайтов
5. операционная система
6. анализировать потребности клиентов
7. быть вовлеченным во все этапы жизненного цикла программного продукта
8. проверять код на наличие ошибок

b) Make up sentences of your own using expressions from part a of this exercise.

Exercise 1.7. Find *italicized words* **in the text "Jobs in Computing". Match them with their synonyms.**

a. to look forward to

b. to serve, to operate

c. to put into effect

d. to value, to estimate

e. to manage

IMPROVE YOUR TRANSLATION SKILLS

Learn vocabulary from Unit 1 "CAREER IN IT" and translate the following sentences from Russian into English.

1. Многие мечтают построить карьеру в сфере IT, поскольку это динамично развивающаяся современная отрасль.
2. Работая в сфере IT, необходимо постоянно обучаться и совершенствовать свои навыки, поскольку новые проекты требуют нестандартных решений.
3. Примеры профессий в сфере IT включают: инженер-программист, тестировщик, веб-разработчик, администратор баз данных, специалист технической поддержки, аналитик, руководитель проектов.
4. Высокая заработная плата считается несомненным плюсом работы в IT сфере.
5. Иногда заказчик не может объяснить требования к финальной версии программного продукта, поэтому разработчику приходится сталкиваться с непониманием и стрессом.

FOCUS ON WRITING

Cover letter

Read the instructions and answer the following questions:

1. What is the cover letter for?
2. What parts does it contain?
3. What recommendations are given to you?

A cover letter is a document that you submit as part of your job application, alongside your resume or CV. Covering letters are not just sent as a courtesy but are an introduction to your potential employer. They are designed to complement your CV and provide extra information about you. The covering letter is the first

impression a potential employer will have of you and without a good impact, they may not progress far with your CV.

The purpose of a cover letter is to introduce you and briefly summarize your professional background. On average, it should be around *250 to 400 words* long. A cover letter should be no longer than one page with a font size between 10-14 points. Be sure to include your contact information and address it directly to the hiring manager, using their name. If you are not sure who to address the letter to, write "Dear Hiring Manager." If the role you are applying for has a reference number or code, be sure to include it in your letter so that human resources are able to accurately track your application. The reference code is usually included.

A good cover letter is supposed to impress the hiring manager and convince them you're worth interviewing as a candidate. So, how can your cover letter achieve this?

First of all, it should *complement* your resume, not copy it. Your cover letter is your chance to elaborate on important achievements, skills, or anything else that your resume doesn't give you the space to cover.

For example, if you have an employment gap on your resume, the cover letter is a great place to explain why it happened and how it helped you grow as a person. If this is your first time writing a cover letter, writing about yourself might seem complicated. But don't worry – you don't need to be super creative or even a good writer.

You have to follow this tried and tested cover letter structure:

- ✓ **Introduction: 1st paragraph**

State clearly in your opening sentence the purpose for your letter and a brief professional introduction.

Specify why you are interested in that specific position and organization.

Provide an overview of the main strengths and skills you will bring to the role.

- ✓ **Body: 2-3 paragraphs**

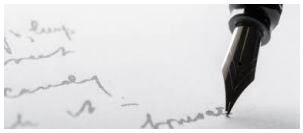
Cite a couple of examples from your experience that support your ability to be successful in the position or organization.

Try not to simply repeat your resume in paragraph form, complement your resume by offering a little more detail about key experiences.

Discuss what skills you have developed and connect these back to the target role.

- ✓ **Closing: last paragraph**

Restate succinctly your interest in the role and why you are a good candidate. Thank the reader for their time and consideration.



Writing Exercise 1.

1. Look at the example of a cover letter. Point out several useful phrases that you can use in your cover letter.
2. Does cover letter from exercise 1 correspond to the instructions given above?

3. What should you add to the letter to make it better?

Thomas Beaker
717-555-0156
thomasbeaker@email.com
January 2, 2024

Dear Hiring Manager,
I am excited to apply for the position of IT Professional with Staples Plus, Inc. With over 10 years of experience in various departments and industries, I'm excited to explore the chance of working for such a well-known company. I'm always looking to further my knowledge in this industry and keep up with the latest trends and research, and I feel your organization is a place where I can do just that.

I believe this position would be a great opportunity for me to expand my skills while providing top-notch services to your organization.

Since graduating with a bachelor's degree in information technology, I have held three positions, all of which have given me the opportunity to hone my skills and grow in my profession. In my last position, I developed specialized software that assisted employees in productivity management, increasing overall productivity by 15% throughout the organization. I also received the award of "Best IT Professional of the Year" from my previous employer thanks to my knowledge, skills and commitment to providing only the highest quality work.

Thank you for taking the time to consider my application. I look forward to hearing from you and learning more about the open IT Professional position at Staples Plus, Inc. I believe my skills, combined with the opportunities your organization can offer me, will help me grow as a professional and allow me to make a positive impact on your company.

Sincerely,
Thomas Beaker

CV[1]

If you want to apply for a job you should present the information about yourself correctly. You can do it with the help of CV. You will be given two examples of CV. What parts does a typical CV consist of?

Template 1: Dennis Scherrer is applying for the position of IT engineer

Template 2: Andrew Green is applying for the position of software support engineer, Linux and Java expert, SQL specialist

1. CV (short for Lat. *curriculum vitae*)

DENNIS SCHERRER

IT Engineer

d.scherrer38@email.com
(123) 456-7890
Houston, TX
LinkedIn

EDUCATION

Bachelor of Science
Computer Science
Texas A&M University
2005 - 2009
College Station, TX

SKILLS

- Python
- Microsoft 365
- Agile Project Management
- Network Infrastructure
- Troubleshooting Windows/Apple OS
- VPN Maintenance
- Verbal Communication
- Customer Service

CERTIFICATIONS

- MCSE
- CCNA

WORK EXPERIENCE

IT Engineer

Loomis Armored US, LLC
2020 - current Houston, TX

- Hired 11 technicians and instructed them in Agile project management, increasing efficiency by 39%
- Drafted troubleshooting guides for common technical strategies, decreasing average ticket resolution time by 48%
- Collaborated with 13 techs to upgrade VPN security, including updating encryption methods and adding antivirus protection, reducing chances of a breach by 67%
- Developed and enhanced product security systems, meeting 100% of client requirements

Network Engineer

ADP
2017 - 2020 Houston, TX

- Created and reorganized SQL queries and scripts for internal troubleshooting, decreasing work tickets by 28%
- Analyzed escalated tickets and coached junior techs to resolve 84% of excessive escalations
- Analyzed diagnostic data to understand causes/correlations of network issues and presented results to internal staff
- Collaborated with staff to resolve network issues and implement fixes, resulting in 31% fewer malfunctions

Systems Support Engineer

Two Sigma
2012 - 2017 Houston, TX

- Managed 7 daily work tickets, prioritizing urgent needs and scheduling projects to resolve tickets within 2 hours
- Trained 8 junior techs to manage tickets, diagnose common problems, and maintain workflows
- Developed solutions for software/hardware compatibility
- Installed and upgraded internal applications and documentation, reducing installation errors by 12%

IT Support Engineer

Capital One
2009 - 2012 Houston, TX

- Resolved 12 network/software Level I tickets per shift
- Provided technical support over the phone, email, and desktop chat, responding to all messages within 4 hours
- Developed online FAQ articles to address common issues, reducing the average number of tickets by 39%
- Diagnosed and repaired network malfunctions including file deletion, failed account entry, slow computer speed, and 3rd party software compatibility issues

ANDREW GREEN

Software Support Engineer | Linux & Java Expert | SQL Specialist

+1-(234)-555-1234
help@enhancv.com
linkedin.com
Austin, Texas

SUMMARY

Dedicated Software Support Engineer with over 5 years of experience specializing in Linux environments, Java application development, and SQL database management. Exceptional analytical skills and a proven track record of implementing solutions that enhance system performance.

EXPERIENCE

Software Support Engineer

TechGiant Solutions

06/2019 - Present | Austin, Texas

- Resolved over 300 complex customer software issues, achieving a 95% customer satisfaction rating and contributing to the team's target of reducing client complaints by 30%.
- Implemented a streamlined process for monitoring and resolving tickets in Jira, reducing average ticket resolution time by 20%.
- Assisted in developing and deploying updates for corporate systems, which decreased system downtime by 15% during peak usage hours.
- Provided data gathering and reporting solutions for 5 key projects, ensuring accurate data was available for decision-making.
- Played a key role in a team that delivered a software enhancement ahead of schedule, resulting in a 10% increase in system efficiency.
- Coordinated successfully with development teams on 10 major projects to implement testing and deployment plans.

Java Developer

InnoSoft Technologies

01/2017 - 05/2019 | San Antonio, Texas

- Developed and maintained Java applications, successfully reducing the codebase by 20% while maintaining functionality.
- Optimized SQL queries for the product, decreasing loading times by 25% and enhancing the user experience.
- Lead a small team in implementing consumer REST APIs, which increased system interoperability by 40%.
- Formulated continuous integration practices that improved deployment frequency by 50%, resulting in faster time-to-market for new features.
- Managed version control with Git for a team of 10 developers, ensuring smooth workflow and minimization of code conflicts.

Backend Developer

CreativeTech Solutions

08/2014 - 12/2016 | Dallas, Texas

- Contributed to a key project which increased the efficiency of backend processes by 35% through performance tuning and code optimization.
- Collaborated on 3 enterprise-level software products, enhancing functionality and user satisfaction.
- Automated repetitive tasks using scripting, which saved approximately 10 hours of manual work per week for the development team.
- Mentored 2 junior developers, improving their productivity by 20% and boosting team output.

EDUCATION

Master of Science in Computer Science

Texas A&M University

01/2011 - 01/2013 | College Station, Texas

Bachelor of Science in Computer Science

The University of Texas at Austin

01/2007 - 01/2011 | Austin, Texas

PROJECTS

Customer Support Ticketing System

Developed an open source ticket management system, improving user experience and ticket tracking for customer support teams. github.com/andrewhgreen/cust-support-ticket-system

Real-time Data Aggregation Tool

Created a lightweight tool for real-time data aggregation from multiple sources, enhancing data accessibility for analysis. github.com/andrewhgreen/data-aggreg-tool

ACHIEVEMENTS

Led Software Optimization Project

Directed a team of 5 to optimize company software, leading to a 35% enhancement in performance metrics and subsequent user engagement.

Award for Customer Satisfaction Excellence

Received the quarterly 'Customer Hero' award for maintaining a 95% customer satisfaction rate over a 6-month period at TechGiant Solutions.

Successful REST API Integration

Spearheaded the integration of REST API that resulted in a 40% increase in system interoperability and streamlined service consumption.

Mentorship and Team Productivity

Mentored junior developers, resulting in a team productivity increase of 20% and enhanced project deliverables.

SKILLS

Linux | Java | SQL | Python | Java Spring Framework | Maven

COURSES

Linux Server Management Certification

Intensive course covering advanced Linux server setup, management, and security administered by Linux Academy.

Certified Java Developer

Comprehensive Java development program focusing on Java SE and Java Spring patterns, accredited by Oracle University.

 **Writing Exercise 2.**

You are looking for a job. Find on the Internet or in the local newspaper an advertisement of a job you would like to have.

a) Write CV and cover letter to apply for a job. Remember that CV and cover letter should be written in an appropriate style.
b) Check your grammar and punctuation.

VOCABULARY LIST OF UNIT 1 "CAREER IN IT"

administer, *v.*
appear, *v*
arise, *v*
asses, *v.*
assignment, *n.*
be an exception, *expr.*
cover letter
CV
cybersecurity expert
data analyst
database integrity
database management system
embedded system
end-user, *adj.*
entry-level job
fast growing
feasible solution, *expr.*
get ahead, go ahead, *ph.v.*

good understanding of code, *expr.*
handle, *v.*
implement a project
macros, *n.*
maintain, *v.*
meet deadline, *fixed expr.*
non-strenuous, *adj.*
pitch for, *phr. v.*
pitfall, *n.*
pros and cons, *expr.*
pursue a career, *expr.*
pursue, *v.*
refine specifications
release, *v.*
resume, *n*.
salary, *n.*
software engineer
stay ahead, *ph.v.*
strength, *n.*
system-level software
test script
troubleshooter, *n.*
varied, *adj.*
versatile, *adj.*

UNIT 2

COMPUTERS

LEAD-IN

1. What types of computers do you know on the basis of size, functionality, data handling?

2. What type of computer do you have at your workplace? What type of computer do you have at home?

3. What do you use your personal computer for?

4. What do you prefer using for listening to music, watching films, surfing the net: a PC or a smartphone? Explain your choice.

READING I

Learn new words and expressions before you read the text.

ENIAC (Electronic Numerical Integrator and Computer)	электронный цифровой интегратор и калькулятор,
gain popularity, *fixed expr.*	завоевать популярность, набирать популярность, стать популярным
PDA (personal digital assistant)	персональный цифровой помощник, персональный электронный помощник, карманный компьютер
workstation	рабочая станция
file server	файловый сервер *(context)*
print server	сервер печати, станция печати *(context)*
web server	веб-сервер, сервер, подключенный к сети Интернет *(context)*
refer to, *v.*	относиться, касаться, соотносить с, ссылаться на *(context)*
legacy system	устаревшая система *(проф.),* неподдерживаемая система *(проф.),* система предыдущего поколения
mainframe, *n.*	базовое вычислительное устройство, центральный компьютер *(проф.)*
IBM	компания-поставщик аппаратного и программного обеспечения, IT-сервисов и консалтинговых услуг

replace, *v.*	замещать, сменить, приходить на смену *(context)*
emerge, *v.*	появляться, возникать, оказываться *(context)*
tablet, *n.*	планшет
wearable computer	носимый компьютер
information appliance	информационный бытовой прибор, информационная бытовая электроника
embedded system	встроенная система, встраиваемая система
multi-user operating system	многопользовательская операционная система
set up, *phr.v.*	устанавливать, собирать, регулировать *(context)*
security server	сервер безопасности, сервер защиты
database server	сервер баз данных, сервер БД
portable device	портативное устройство
perform, *v.*	выполнять, исполнять, совершать, осуществлять *(context)*
flexibility, *n.*	гибкость
computing device	вычислительное устройство, вычислительный прибор
set of requirements	набор требований
non-volatile memory	энергонезависимая память *(проф.)*
reboot, *v.*	перезапустить систему, перезагрузить
reset, *v.*	вернуть в исходное состояние, «сбросить», перезапустить *(проф.)*

Classification of computers

Computers were not always things you could carry around with you, or even have in your bedroom. Sixty years ago, computers such as ENIAC, were as big as entire apartments. They were difficult to use and cost a lot of money to build and operate. Thus, computers were only used by large organizations such as governments, international corporations, and universities. Throughout the 1950s and 1960s more and more companies wanted computers, even if they didn't always have a good reason to own one. As a result, computers gradually became smaller, cheaper, and more practical to own.

In the 1970s and 1980s a new type of computer started to gain in popularity. It was called the PC or personal computer used for general purposes. For the first time in history, computers were now for everyone. The PC started a revolution which affects nearly everything we do today. PCs are everywhere you look today. At home, at the office, and everywhere in between. Many people still mistakenly believe the term PC is synonymous with a desktop computer. This is not really true. Really, any computer you use by yourself for general purposes could be called a PC. You probably already own at least one of these types of PCs: laptop, desktop computer, PDA or personal digital assistant, workstation. Besides PCs, there are other types of computers you probably see at work. They are used for specific purposes. These include: file servers, print servers, web servers.

Types of computers go in and out of fashion as times changes. Older kinds of computers which were very popular in the 20th century (1900's) are now referred to as legacy systems. These include: mainframes, minicomputers, IBM clones. New types of computers are always coming out and replacing existing computer types. Examples of new types of computers emerging would be netbooks, tablet, and even wearable computers.

On the basis of functionality computers can be divided into servers, workstations, information appliances and embedded systems. Workstations are the computers designed to primarily to be used by single user at a time. They run multi-user operating systems. They are the ones which we use for our day-to-day personal work. Servers are dedicated computers which are set-up to offer some services to the clients. They are named depending on the type of service they offered: security server, database server. Information appliances are the portable devices which are designed to perform a limited set of tasks like basic calculations, playing multimedia, browsing internet etc. They are generally referred as the mobile devices.

They have very limited memory and flexibility. Embedded computers are the computing devices which are used in other machines to serve limited set of requirements. They follow instructions from the non-volatile memory and they are not required to execute reboot or reset. Embedded systems can be found in traffic lights, TV sets, refrigerators, coffee machines and many more devices. Embedded systems are typically controlled by inexpensive, specialized processors which can only handle very specific tasks.

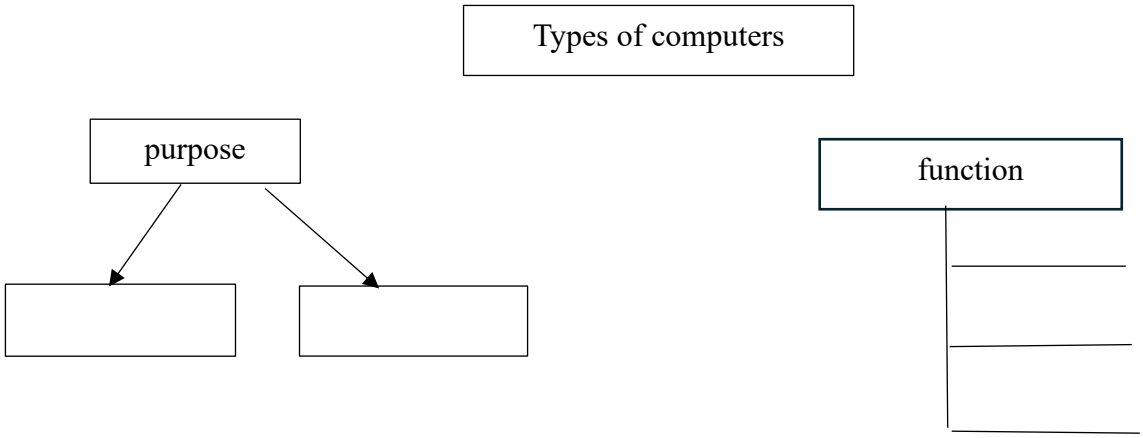
LANGUAGE DEVELOPMENT

Exercise 2.1. Read the text and define the sentences as True or False.

1. First computers were as big as the size of the room.
2. First computers were widely spread.
3. PC is another name for a desktop computer.

4. Old computers refer to as legacy systems.
5. In terms of use the computers are divided into digital, hybrid and analog ones.
6. Embedded computers are as the same as wearable computers.
7. Workstation can be used for high-speed calculations by a group of people.
8. Servers are computers which are used to process huge amounts of information.

Exercise 2.2. Complete the mind map for the text.



Exercise 2.3. Work in pairs and in turns define the terms in your own words.

1. A desktop computer is _____.
2. A laptop is _____.
3. An embedded system is _____.
4. PC or personal computer is _____.
5. A router is _____.
6. A server is _____.
7. A supercomputer is _____.
8. A tablet is _____.
9. A wearable computer is_____.
10. A worskation is _____.

Exercise 2.4. Match the types of computers classified on the basis of data handling with their definitions.

1. Analog	A. It is a computer that accepts analog signals, converts them to digital and processes them in digital form.
2. Digital	B. It is a form of computer that uses electrical, mechanical quantities to model the problem being solved.
3. Hybrid	C. It is a computer that performs calculations and logical operations with quantities represented as digits, usually in the binary number system of “0” and “1”. It can perform

	mathematical calculations, organize and analyze data, control industrial and other processes.

Exercise 2.5. Use expressions from the box to complete the sentences.

desktop computer embedded system legacy system router server supercomputer tablet wearables workstation PDA (personal digital assistant)

1. _____ are seen in nearly every office and home today.
2. Nearly all electronics you can buy today are controlled by _____.
3. Mainframes and minicomputers used to be cutting edge; now they are known as _____ .
4. In the 1990's the modern traveling businessman depended on his _____to organize all his contacts and appointments.
5. The Wi-Fi _____allowed the entire office to share a single Internet connection.
6. Because _____ hold large amounts of centralized data, it is critical to have daily backup routines in place.
7. A typical _____ may have up to 100 processors and speeds are measured in tens of gigaflops.
8. Bill Gates predicted the rise of the _____PC in 2001, but not much happened until Apple released the first iPad in 2010.
9. Smart watches and VR goggles are two popular forms of _____.
10. The engineer's latest 3d modeling project was so GPU intensive, that he wouldn't even begin it until his new _____ arrived.

Exercise 2.6. Search the web and find information about the new types of computers developed. Present the information to your groupmates in 10-15 sentences. You should say:

- what sort of computer was invented;
- where the computer was invented;
- who invented the computer;
- what new technology the computer uses;
- what new features it has.

READING II

Before you read the text "Computers of the future", work in pairs and discuss the following questions:

- How could developments in computing impact industry, governments and society?

- What technology or gadget would you most like to see by 2030?

- What sphere of computer development can be dangerous for the humanity, if any?

Learn new words and expressions before you read the text.

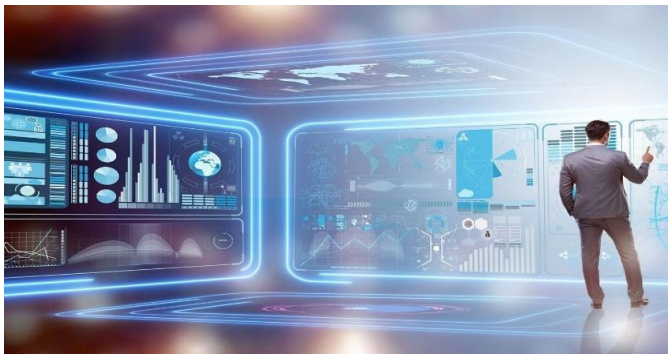
artificial intelligence	искусственный интеллект
neural network	нейронная сеть, нейросеть
recognition	Распознавание
evident, *adj.*	очевидный, явный, ясный
cutting-edge application	современное приложение
target, *v.*	выявлять, планировать, затрагивать, установить цель *(context)*
incarnation, *n.*	воплощение
full-blown, *adj.*	полнофункциональный, полномасштабный *(context)*
binary notation, *expr.*	представление чисел в двоичной системе счисления
digit	число, знак, цифра от 0 до 9, символ
rely on, *phr.v.*	полагаться на, исходить из, рассчитывать на, основываться на *(context)*
DNA strand	цепь ДНК
subject, *v.*	подвергать, склонять к чему-либо, предрасполагать *(context)*
simultaneously, *adv.*	одновременно
parallel processing	параллельная обработка данных, параллельное исполнение
fingerprinting	создание цифрового отпечатка *(context)*
occur, *v.*	случаться, происходить
plague, *v.*	приносить вред, становиться проблемой
sequentially, *adv.*	последовательно
complex computations	сложные вычисления
to process billions of bits	обрабатывать миллиарды битов

bandwidth	производительность, пропускная способность

Computers of the future

Match the texts (A-E) with their subtitles (1-6). You have one extra subtitle you do not need to use.

1. DNA computers
2. New computers, new possibilities
3. Quantum computers
4. Wearable computers
5. Optical computers
6. Virtual reality



A. Future computers promise to be even faster than today's computers. Perhaps they will become the size of coins and offer artificial intelligence features like expert intelligence, neural network pattern recognition features, or natural language capabilities. Already evident are some evolving cutting-edge applications for computer technology.

B. The latest trend in computing is wearable computers. Common computer applications (e-mail, database, multimedia, calendar) are integrated into watches, cell phones, and even clothing. Many other wearables target outdoors enthusiasts allowing them to track their location, altitude, calories burned, steps, speed. The Apple iWatch, now in its nineth incarnation, is one of the best reviewed wearables to date. This small watch has many of the functionalities of a full-blown smartphone. It lets you perform normal texting and email duties. It has a built-in cell phone. It even has a built-in electrical heart sensor that you can use to take an electrocardiogram and share it instantly with your doctor. A company called MC10 is even advertising skin patches that will track various biological processes happening in your body. Wearables are indeed a new horizon in personal computing.

C. A DNA-based computer would be radically different from a conventional computer. Instead of storing data on silicon chips, converting data to binary notation (0s and 1s), and performing computations on the binary digits, DNA computing would rely on data found in patterns of molecules in a synthetic DNA strand. Each strand represents one possible answer to the problem. A set of strands is manufactured so that all possible answers are included. To find a solution, the DNA computer subjects all the strands simultaneously to a series of chemical reactions that imitate mathematical computations. The advantage of DNA computing is that it works in parallel, processing all possible answers simultaneously. DNA-based computers could be used to perform parallel processing applications, DNA fingerprinting, and the decoding of strategic information such as banking, military, and communications data.

D. The first application of quantum theory and computers occurred in 1981 at Argonne National Laboratory. In 1985, a quantum parallel computer was proposed. Today, physicists and computer scientists still hope that the imprecision of subatomic particles can be used to solve problems that thus far remain unsolved. The quantum computer would overcome some of the problems that have plagued conventional computers: namely, sequentially following rules and representing data as a series of switches corresponding to 0 or 1. By using subatomic particles, quantum computers will have the ability to represent a number of different states simultaneously. Manipulating these small subatomic particles will allow researchers to solve more complex problems, such as performing complex computations, predicting weather conditions, helping chip designers create circuits that are presently impossibly complex.

E. As microprocessor chip designers reach physical limitations that prevent them from making chips faster, they are searching for other materials to conduct data through the electrical circuits of computer systems. Optical computing could allow computers to perform parallel processing tasks more efficiently and increase the speed and complexity of computers by allowing them to process billions of bits simultaneously. Optical computers might use fiber-optic cable, optical chips, or wireless optical networks to process and transmit data. A more recent development – optical chips – could cut the cost of optical communication by using Dense Wave Division Multiplexing technology to carry more information over a fiber. This would give users increased bandwidth for connecting to the Internet. Optical networks could be used to improve free-space optics, video delivery, and voice communications.

LANGUAGE DEVELOPMENT

Exercise 2.7. Complete the table using information from the text.

#	Advancement	Advantage over current systems	Application
1.	wearable computers		
2.		parallel processing	
3.			can perform complex computations, predict weather conditions, help chip designers create complex circuits
4.	quantum computer		

Exercise 2.8. Match the word with its synonym.

1. to evolve	A. to use together
2. to track	B. to cope with
3. to share	C. to trouble
4. to overcome	D. to forecast
5. to plague	E. possibilities
6. to predict	F. at the same time
7. functionalities	G. at the same time
8. sequentially	H. to develop
9. simultaneously	I. to mimic
10. to imitate	J. subsequently

Exercise 2.9. a) Make collocations.

b) Use some collocations in the sentences of your own.

1. expert	processing
2. pattern	networks
3. cutting-edge	strand
4. silicon	cable
5. binary	notation
6. parallel	intelligence
7. DNA	applications
8. subatomic	circuits
9. electrical	particles
10. fiber-optic	chips
11. wireless	Recognition

Exercise 2.10. Discuss in pairs.

1. What type of computer do you consider most convenient to use for personal needs and why?
2. What computer would you prefer using: a desktop computer, a laptop or a tablet. Why?
3. Do you want a more powerful computer? If so, what computer do you want?
4. If you could buy a new computer, what would you like to buy?

Exercise 2.11. Paraphrase the following quotations. Which one do you agree with the most? Why? Discuss in small groups.

Quotation 1
"To be human is to be 'a' human, a specific person with a life history and idiosyncrasy[1] and point of view; artificial intelligence suggests that the line between intelligent machines and people blurs most when a mash is made of that identity."

Quotation 2
"Computers are magnificent tools for the realization of our dreams, but no machine can replace the human spark of spirit, compassion, love, and understanding."

Exercise 2.12. Think of your own predictions for 2040s. Choose one of the spheres listed below and describe what innovations will have been implemented by 2040s through advances in computer technology. Prepare a presentation (6-8 slides) and share your ideas with the group.

- Computation
- Artificial intelligence
- Lifestyle
- Communication

IMPROVE YOUR TRANSLATION SKILLS

Learn vocabulary from Unit 2 "Computers" and translate the following sentences from Russian into English.

1. Персональные компьютеры имеют следующие характеристики: малые размеры, отсутствие необходимости в обслуживании, низкая цена и универсальность.
2. Программное обеспечение помогает не только решить проблемы совместимости, но и обеспечивает мощную функциональность и гибкость.
3. При создании Windows Vista использовался целый ряд инноваций и технологий, направленных на ограничение возможностей злоумышленников по "взлому" компьютеров.
4. Персональный компьютер остается основной платформой для игр, говорится в докладе организации PC Gaming Alliance (PCGA).
5. Microsoft представила новую версию планшета Surface Pro, оснащенную процессором Qualcomm Snapdragon X, который позволяет увеличить производительность устройства в отдельных задачах на величину до 90 % по сравнению с моделью предыдущего поколения.

1. (transl.) характерная особенность, индивидуальная отличительная черта

FOCUS ON WRITING

Addressing to visuals, tables and figures in academic texts

If you are engaged into academic research at your department, it may be permissible and appropriate to insert tables, figures and other graphics in the text of your scientific article. These graphics may have been copied, adapted from sources of information or may be from your own research. They need to be relevant, correctly labelled and referenced – unless they are entirely your own work. It is important that tables and figures are used purposefully (i.e. with good reason) and referenced correctly.

DO NOT:

- restructure data from an information source into another format (e.g. a graph, a flowchart) without referencing the author of your information. You may structure the graph, but the author still 'owns' the research;
- just 'plonk' a table or figure into your writing. You need to refer to its existence and relevance to your argument in the preceding text;
- give extensive descriptions in your writing of the contents of a table or diagram. The information in a table or diagram tells its own story – it's your job to point out its significance to your argument.

Although visuals do largely speak for themselves, it is usual to help the reader interpret them by briefly commenting on their main features using the language of change.

Table 1. Verbs that are used to describe visuals

You describe increasing tendencies	You describe decreasing tendencies
grow	fall (to)/ decline (to)
raise/rise to	drop (to)
increase	decrease (to)/reduce (to)
climb to	collapse
step up	cut
expand	go down (to)/fall (to)
improve	break
shoot up/soar	push down
peak at	bottom out
boom	dip (to)

Table 2. Adjectives and nouns that are used to describe visuals

a slight drop
a gradual fall
a sharp decrease
a dramatic growth
a huge boom
a steep increase
a substantial climb
a considerable change
a significant reduction
a marked change

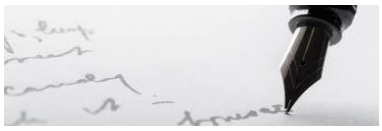
Table 3. Adverbs that are used to describe visuals

substantially
considerably
significantly
markedly
moderately
minimally
swiftly
quickly
suddenly
slightly
gradually
steadily
sharply
dramatically
hugely
rapidly
slowly

To introduce visuals and name the parts of a diagram you can use phrases from Table 4.

Table 4. Phrases to introduce visuals, parts of a graph, diagram

Introducing visuals	Naming the parts of a graph, diagram
I'd like you to look at this graph... Let me show you this pie chart... Let's have a look at this model... Let's turn to this map... To illustrate my point let's look at some diagrams... As you can see from these figures... If you look at these photographs you'll see... If you look at this bar chart you'll notice... If you look at this histogram you'll appreciate... If you look at this flow chart you'll understand … If you look at this matrix...	The vertical axis represents total annual … The horizontal axis shows our … The curve The solid line The dotted line The broken line The shaded area The unshaded section The dotted column The coloured segment The red bar



Writing Exercise 1.

Place the correct letter in the right box. The box where you need to write a letter is coloured.

slump		rise		recover		plunge		pick up		bounce back	
drop		soar		climb		fall		rally		take off	
crash		slide		stabilize		escalate		decline		flatten out	
rocket		dip		fluctuate		plummet		hold steady		expand	

Letters:

A  **B**  **C**  **D**  **E**  **F**  **G**

Writing Exercise 2.

a) Find a graph or diagram on the Internet and describe it using the main ideas and vocabulary from *Focus on Writing* section.
b) If you have published academic research article, you can describe visuals from it.
c) You should write 100-120 words. Check your grammar and punctuation.

VOCABULARY LIST OF UNIT 2 "COMPUTERS"

artificial intelligence
bandwidth
binary notation, *expr.*
complex computations
computing device
cutting-edge application
database server
diagram
digit
DNA strand
embedded system
emerge, *v.*
ENIAC (Electronic Numerical Integrator and Computer)
evident, *adj.*
file server
fingerprinting
flexibility, *n.*
full-blown, *adj.*
gain popularity, *fixed expr.*
graphic
IBM

incarnation, *n.*
information appliance
legacy system
mainframe, *n.*
multi-user operating system
neural network
non-volatile memory
occur, *v.*
parallel processing
PDA (personal digital assistant)
perform, *v.*
plague, *v.*
portable device
print server
reboot, *v.*
recognition
refer to, *v.*
rely on, *phr.v.*
replace, *v.*
reset, v.
security server
sequentially, *adv.*
set of requirements
set up, *phr.v.*
simultaneously, *adv.*
subject, *v.*
tablet, *n.*
target, *v.*

to process billions of bits
visuals
wearable computer
web server
workstation

UNIT 3

PROGRAMMING LANGUAGES AND PARADIGMS

LEAD-IN

1. What programming languages do you know?
2. What are some of the best programming languages to learn?
3. What are their special features?
4. What is, in your opinion, the easiest/most difficult part in writing a code?

In the second column of the table below you see programming language specific features. Match the specific feature to the language it characterizes.

	A. It is one of the newest programming languages to be used on a wide-scale. *Programming language:*
C	B. It is ancestor to many of the advanced programming languages. *Programming language:*
Java	C. It requires less time, less lines of code, and less concepts to be taught to reach a given goal. *Programming language:*
Python	
Ruby	D. Coding in this language is stricter and has a steeper learning curve than other languages. *Programming language:*
	E. It is one of the most used programming languages in the world. *Programming language:*
	F. It is often considered to be the easiest language to learn, owing to its simplicity, readability and straight forward syntax.

C **Java** **Python** **Ruby**	*Programming language:*
	G. With this language you'll be able to access and manipulate the most important computer parts like the filesystem, graphics, and sound for any fairly sophisticated and modern program. *Programming language:*
	H. It is used for developing low level applications. *Programming language:*
	I. One of the truly greatest perks of working with this language is that it's completely open-sourced and free. *Programming language:*
	J. There is a sense of beauty in its coding that makes this one of the best programs for beginners. *Programming language:*
	K. This language is used to develop enterprise level application and video games. *Programming language:*
	L. It is regarded as combination of some of the most famous features of Lisp, Pearl and Eiffel. *Programming language:*

READING I

Learn new words and expressions before you read the text.

artificial language	искусственный язык
instruct, *v.*	давать команду, сообщать, давать указания *(context)*
perform computation	выполнять вычисления
binary code	двоичный код
symbolic programming language	символьный язык программирования
assembly language	язык ассемблера
hardware, *n.*	аппаратное обеспечение *(в отличие от программного)*

restrict, *v.*	ограничивать *(context)*
high-level language	высокоуровневый язык программирования
low-level language	низкоуровневый язык программирования
assembler	ассемблер, машинный язык
interpreter	интерпретатор
compiler	компилятор, компилирующая программа
source code	исходная программа, исходный код программного обеспечения (ПО)
object code	выходная программа
convert, *v.*	преобразовывать *(context)*
markup language	язык разметки
markup tag	тег разметки *(в гипертекстовом документе)*
pre-defined tag	предопределенный тег *(проф.)*
enable, *v.*	позволять, делать возможным, поддерживать *(context)*
ambiguous, *adj.*	неясный, неоднозначный
debugging	отладка *(проф.)*
framework	интегрированная среда, базовое средство разработки, комплекс программ, фреймворк *(проф.)*
ancestor, *n.*	предок
script	скрипт, сценарий *(проф.)*
syntax	синтаксис
software	программное обеспечение (ПО)
functional programming	функциональное программирование
object-oriented programming	объектно-ориентированное программирование (ООП)

Fill in the gaps with words or word combinations from the box.

compiler high-level language programs assembler machine code assembly language low-level languages markup languages

A programming language is an artificial language that can be used to control the behavior of a computer. It is used to write computer (1) _____ which instruct a computer to perform computation and/or control mechanical devices, for example, in a robot. Since the nature of scientific research is to explore the unknown world and test new theories, programming languages are usually the most important interface between scientists and computers. Programming languages allow people to communicate with computers.

Unfortunately for us, computers can't understand spoken English or any other natural language. The only language they can understand directly is (2) ____ which consists of 1s and 0s (binary code).

Machine code is too difficult to write. For this reason, we use symbolic languages to communicate instructions to the computer. For example, (3) _____ use abbreviations such as ADD, SUB, MPY to represent instructions. The program is then translated into machine code by a piece of software called (4) _____. Machine code and assembly languages are called (5) _____ because they are closer to the hardware. They are quite complex and restricted to particular machines. To make the programs easier to write, and to overcome the problem of intercommunication between different types of computer, software developers designed (6) ___, which are closer to the English language.

Programs written in high—level languages must be translated into machine code by (7) ____ or an interpreter. A compiler translates the source code into object code — that is, it converts the entire program into machine code in one go. On the other hand, an interpreter translates the source code line by line as the program is running.

It is important not to confuse programming languages with (8) _____, used to create web documents. Markup languages use instructions, known as markup tags, to format and link text files. Some examples include:

- ✓ HTML, which allows us to describe how information will be displayed on web pages.
- ✓ XML, which stands for EXtensible Markup Language. While HTML uses pre-defined tags, XML enables us to define our own tags; it is not limited by a fixed set of tags.
- ✓ VoiceXML, which makes Web content accessible via voice and phone. VoiceXML is used to create voice applications that run on the phone, whereas HTML is used to create visual applications (for example, web pages).

A major purpose of programming languages is to provide instructions to a computer. As such, programming languages differ from most other forms of human expression in that they require a greater degree of precision and completeness. When using a natural language to communicate with other people, human authors and speakers can be ambiguous and make small errors, and still expect their intent to be understood. However, computers do exactly what they are told to do, and cannot understand the code if it is unclear.

LANGUAGE DEVELOPMENT

Exercise 3.1. Answer the questions.

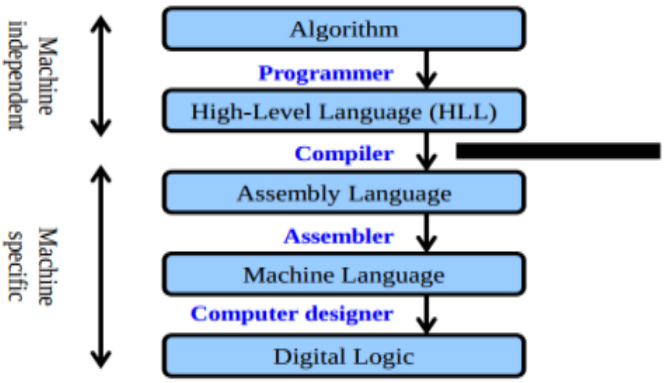
1. Can any computers understand human language?
2. What is the function of assembler?
3. Why did software developers design high-level languages?
4. What is the difference between compiler and interpreter?
5. Why are HTML and VoiceXML called markup languages?

Exercise 3.2. Complete the sentence using one word from the box.

applications	**debugging**	**framework**	**ancestor**	**object**	**script**	**syntax**	**coding**

1. Many famous websites were developed on this web development ____.
2. One of the reasons to pick up this language is that its ____ is easy to understand.
3. It is more convenient to develop web based ____ in Python.
4. It requires less time to write a simple ____ in this language.
5. ____ in Java is knowing how and why the project works.
6. This is the first purely ____ oriented programming language.
7. C language is an ____ to many of the advanced programming languages.
8. With the help of this language you will learn things like ____ programs, memory management, and how computers work.

Exercise 3.3. Match the terms with their definitions. The diagram below will help you.



1. program	a. the language used to write a computer program before it is turned into a machine code
2. machine code	b. a language for writing computer programs that is similar to machine code
3. assembly language	c. a set of instructions which computer uses to do a specific task
4. assembler	d. a software that translates assembly language into machine code
5. low-level language	e. a set of 1s and 0s that gives instructions to a computer
6. high-level language	f. a program that changes computer instructions into machine code
7. compiler	g. a language for writing computer programs that looks more like human language
8. functional programming	h. a programming paradigm based on the concept of "objects" which contain data and code
9. object-oriented programming	i. a technique of programming that stress on an evaluation of functions
10. markup language	j. a computer language that uses tags to define elements within a document

Exercise 3.4. Solve the anagrams in the second column and match them with the words in the first column to complete the phrases. The first one is done as an example for you.

1. high-level language (1-f)

1. high-level	a. esorscp
2. computer	b. orgmigpramn
3. programming	c. asweg
4. problem -	d. nliogvs
5. step up to the	e. ltepa
6. perfect	f. egguanal
7. provide a	g. rargmop
8. source	h. tfi

9. compile a	i. tunsiloo
10. calculate	j. ecdo

Exercise 3.5. Discuss the following questions with your groupmate:

1. What does a "computer-literate" person mean?
2. What is the role of programmers in our life?
3. What is the difference between program and data?
4. How do you cope with computer programming?
5. Describe your latest experience in writing a program. What programming language did you use to write it?

READING II

Before you read the text "Programming languages and paradigms", work in pairs and discuss the following questions:

- What do you know about programming paradigms?
- Is there any difference between programming language and programming paradigm? Which one if any?
- What are the reasons for using programming languages and paradigms?
- What are the four main types of programming paradigms?

Learn new words and expressions before you read the text.

central processing unit (CPU)	центральный процессор, центральный вычислительный блок
substitute, *v.*	заменить
execute instruction, *expr.*	выполнять инструкции
regardless of, *prep.*	независимо от, вне зависимости от
executable program	исполняемая программа
concurrent program	многопоточная программа *(проф.)*
constraint-based programming	программирование в ограничениях *(проф.)*
discrete synchronous programming	дискретное синхронное программирование *(проф.)*

embody, *v.*	реализовать, воплощать (context)
data abstraction	абстракция данных
sequence of events	последовательность событий
grasp, *v.*	воспринимать, улавливать смысл *(context)*
tedious, *adj.*	длительный, трудоемкий *(context)*
error prone method	подверженный ошибкам метод
implementation, *n.*	реализация

The difference between programming paradigms and programming languages is that programming language is an artificial language that has vocabulary and sets of grammatical rules to instruct a computer to perform specific tasks. Programing paradigm is a particular way (i.e., a 'school of thought') of looking at a programming problem.

The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal. Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions. High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages. Each different type of CPU has its own unique machine language. Assembly languages are lying between machine languages and high-level languages. Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers.

Machine languages consist of numbers only. Lying above high-level languages are languages called fourth-generation languages (usually abbreviated 4GL). 4GLs are far removed from machine languages and represent the class of computer languages closest to human languages. Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it.

There are two ways to do this: compile the program and interpret the program. A program that executes instructions is written in a high-level language. There are two ways to run programs written in a high-level language. The most common is to compile the program. To transform a program written in a high-level programming language from source code into object code. Programmers write programs in a form called source code. Source code must go through several steps before it becomes an executable program.

There are currently 27 paradigms exist in the world. Most of them are of similar concepts extending from the 4 main programming paradigms (see Table 5). Programming languages should support many paradigms. Let us name 4 main programming paradigms: the imperative paradigm, the functional paradigm, the logical paradigm, the object-oriented paradigm. Other possible programming paradigms are: the visual paradigm, one of the parallel/concurrent paradigms and the constraint based paradigm. The paradigms are not exclusive but reflect the different emphasis of language designers. Most practical languages embody features of more than one paradigm.

Each paradigm supports a set of concepts that makes it the best for a certain kind of problem. For example, object-oriented programming is best for problems with a large number of related data abstractions organized in a hierarchy. Logic programming is best for transforming or navigating complex symbolic structures according to logical rules.

Discrete synchronous programming is best for reactive problems, i.e., problems that consist of reactions to sequences of external events. Programming paradigms are unique to each language within the computer programming domain, and many programming languages utilize multiple paradigms. The term paradigm is best described as a "pattern or model." Therefore, a programming paradigm can be defined as a pattern or model used within a software programming language to create software applications.

Popular mainstream languages such as Java or C++ support just one or two separate paradigms. This is unfortunate, since different programming problems need different programming concepts to solve them cleanly, and those one or two paradigms often do not contain the right concepts.

It is helpful to understand the history of the programming language and software in general to better grasp the concept of the programming paradigm. In the early days of software development, software engineering was completed by creating binary code or machine code, represented by 1s and 0s. These binary manipulations caused programs to react in a specified manner.

This early computer programming is commonly referred to as the "low-level" programming paradigm. This was a tedious and error prone method for creating programs. Programming languages quickly evolved into the "procedural" paradigm or third generation languages including COBOL, Fortran, and BASIC. These procedural programming languages define programs in a step-by-step approach.

The next evolution of programming languages was to create a more logical approach to software development, the "object oriented" programming paradigm. This approach is used by the programming languages of Java, Smalltalk, and Eiffel. This paradigm attempts to abstract modules of a program into reusable objects.

In addition to these programming paradigms, there is also the "declarative" paradigm and the "functional" paradigm. While some programming languages strictly enforce the use of a single paradigm, many support multiple paradigms. Some examples of these types include C++, C#, and Visual Basic. Each paradigm has unique requirements on the usage and abstractions of processes within the programming language.

Nevertheless, Peter Van Roy says that understanding the right concepts can help improve programming style even in languages that do not directly support them, just as object-oriented programming is possible in C with the right programmer attitude. By allowing developers flexibility within programming languages, a programming paradigm can be utilized that best meets the business problem to be solved. As the art of computer programming has evolved, so too has the creation of the programming paradigm. By creating a framework of a pattern or model for system development, programmers can create computer programs to be the most efficiency within the selected paradigm.

Table 5. Programming paradigms

Imperative/ Algorithmic	Declarative		Object-Oriented
	Functional Programming	Logic Programming	
Algol Cobol PL/1 Ada C Modula-3 Esterel	Lisp Haskell ML Miranda APL	Prolog	Smalltalk Simula C++ Java

LANGUAGE DEVELOPMENT

Exercise 3.6. Give Russian equivalents for the following word combinations.

1. constraint programming
2. discrete synchronous programming
3. software applications
4. concurrent processes
5. step-by-step approach
6. external event
7. assembly language
8. source code

Exercise 3.7. Find English equivalents for the following word combinations.

1. низкоуровневое программирование
2. уязвимый для ошибок
3. объектный код, объектная программа
4. преобразовывать исходные тексты программы в объектные модули
5. реагирующие проблемы *(проблемы, которые изменяют свое поведение в ответ на конкретные ситуации)*
6. стадия компиляции
7. программа, работающая с абстрактными типами данных
8. оценочная функция

Exercise 3.8. Translate the following sentences into Russian.

1. Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.
2. In computer science, a low-level programming language is a programming language that provides little or no abstraction from a computer's instruction set architecture.
3. Early systems were frequently error-prone and difficult to modify because they made widespread use of global data.
4. Functional programming a program can be thought of as a combination of stateless function evaluation.

PROJECT WORK

Project 1.

When one first encounters a new programming language, the first question is usually: What can this language 'do'?

Choose two different programming languages, which are based on different paradigms (such as data-oriented, imperative, object-oriented, or scripting) and:

a) compare and contrast what each language can do;

b) discuss why you should use each one, highlighting what type of applications they are most suitable for.

Project 2.

There are numerous articles that address why software projects fail. Common software issues include: poor software quality; inability to cope with changing requirements; hard to maintain software; modules that do not fit together; unacceptable performance. Choose one programming paradigm (such as data-oriented, imperative, object-oriented or scripting), and discuss what characteristics it has that would either help, or hinder, software project development.

IMPROVE YOUR TRANSLATION SKILLS

Learn vocabulary from Unit 3 "Programming languages and paradigms" and translate the following sentences from Russian into English.

1. Программисты не используют все существующие языки; чтобы развиваться в профессии, достаточно знать 4-5 самых популярных языков.
2. Стоит заметить, что большая часть современных языков в той или иной степени поддерживает императивное программирование.
3. Под отладкой программы понимается процесс испытания работы программы и исправления обнаруженных при этом ошибок.
4. Если программисту действие нужно выполнить несколько раз, то выполнение действия целесообразно выделить в маленькую подпрограмму – процедуру.
5. Язык С++ представляет собой набор команд, которые говорят компьютеру, что необходимо сделать. Этот набор команд, обычно называется исходный код.

FOCUS ON WRITING

Caution in formal writing

It is wise to use a cautious tone in your writing, because very often you are discussing issues in which there is no absolutely right answer, or absolutely correct definition, or absolutely perfect solution. If you present something as being the best way, it might easily be shown not to be the best way.

Therefore, it is usually better to "suggest", rather than "state". Areas where caution is particularly important include:

a) outlining a hypothesis that needs to be tested, (e.g. in an introduction of research paper);
b) discussing the results of a study, which may not be conclusive;
c) commenting on the work of other writers.

Table 6 contains some phrases that convey a cautious tone.

Table 6. Phrases that make the tone of your writing less categorical

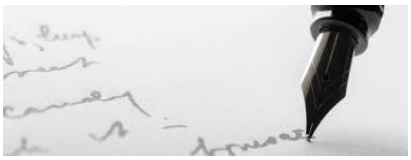
Introductory verbs	seem, tend, look like, appear to be, think, believe, doubt, be sure, indicate, suggest, estimate
Certain lexical verbs	believe, assume, suggest
Certain modal verbs	will, must, would, may, might, could, can
Adverbs of frequency	often, sometimes, usually, generally, occasionally
Modal adverbs	probably, possibly, perhaps, conceivably *(compare with less tentative adverbs like certainly, definitely, clearly)*
Modal adjectives	probable, possible *(compare with less tentative adjectives like certain, definite, clear),* likely, unlikely
Modal nouns	assumption, possibility, probability
That clauses	It could be the case that … It might be suggested that … It appears that … It may be that … It is likely that … This suggests that …

To-clause + adjective	It may be possible to obtain It is important to develop It is useful to study

Another way to express caution is to use *quite, rather* or *fairly* before an adjective (e.g. a fairly accurate summary, quite a significant correlation, a rather inconvenient location). *Quite* is often used before the article. It is often used positively, whereas *rather* tends to be used negatively. When referring to sources, the verb used indicates the degree of caution appropriate.

Compare:
1. Before the launch of unlocked version, rumors states that unlocked version of iPhone … *(positive)*
2. Microsoft CEO suggests that more hardware would open… *(cautious)*

Other verbs that imply tentative or cautious findings are: think, consider, hypothesize, believe, claim, presume.



Writing Exercise 1.

Rewrite the following sentences in a more cautious way.

1. Older students cope better with computer apps than younger ones.
2. Computer manuals are difficult to understand.
3. Application software, also known as an application or an app, is computer software designed to help the user to perform specific tasks.
4. There are cases where this would have been the only possible method of transmission.
5. These apps also make it easy to figure out what to remove if you're running out of storage space.

Formality in verbs

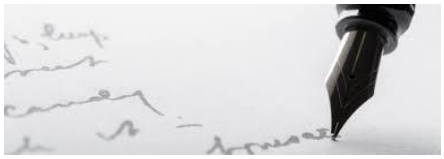
A distinctive feature of academic writing style is choosing the more formal alternative when selecting a verb, noun, or other part of speech. English often has two, or more, choices to express an action or occurrence. The choice is often between a phrasal or prepositional verb (verb + preposition) and a single verb, the latter with Latinate origins. Often in lectures and other instances of everyday spoken English, the verb + preposition is used; however, for written academic style, the preferred choice is a single verb wherever possible. This is one of the most dramatic stylistic shifts from informal to formal style.

Less formal: Researchers *looked at* the way strain *builds up* around a fault.
Academic: Researchers *observed* the way strain *accumulates* around a fault.

A feature of most academic writing is a tendency to use rather formal verbs to express the writer's meaning accurately. Study the list of verbs in Table 7 below and find the meaning in each case. Some of these verbs, e.g. hold, are used in academic writing with a special meaning.

Table 7. Formal verbs and their use

to adapt	The simulation *has* also *been adapted* extensively for use in Asian …
to arise	Inverse problems *arise* in many branches of science and mathematics, …
to carry out	The largest study *was carried out* as a sub-project of the Academy of ...
to characterize	Today's era *is characterized* by dynamic, yet focused IT partnership.
to clarify	AT helps *to clarify* any contradictions in an AS and…
to concentrate on	The final part of that study *concentrated on* the use of 'video-recall' ...
to demonstrate	… and new experimental evidence *has demonstrated* that
to determine	The IDA evidence code *was* experimentally *determined*...
to discriminate	… a failure *to discriminate* between the GFP devices and SCPDs …
to focus on	… this term *focused on* integrated resource management
to generate	Clients may *generate* byte-range requests without having received this header.
to hold	Once you've added items to it, Melon *will hold* the bits and pieces of …
to identify	More than 30 reasons *have been identified* ...
to indicate	The survey *indicates* that opportunities for new developments have ...
to overcome	Let us see what exactly computer rage is, and how you can *overcome* it.
to prove	This paper *proves* that interactive systems are …
to yield	Such surveys *yield* single-digit percentages at best …



Writing Exercise 2.

Choose a verb from the box that reduces the informality of each sentence. You may need to change the tense to fit the sentence.

assist determine establish investigate reduce fluctuate increase raise create eliminate

1. Expert Systems can *help out* the user in the diagnosis of problems.
2. This program was *set up* to improve access to medical care.
3. Research expenditures have *gone up* to nearly $350 million.
4. Researchers have *found out* that this drug has serious side effects.
5. The use of optical character readers (OCRs) should *cut down* the number of problems with mail service.
6. Building a nuclear power plant will not *get rid of* the energy problem completely.
7. Researchers have been *looking into* this problem for 15 years now.
8. This issue was *brought up* during the investigation.
9. Engineers can *come up with* better designs using CAD.
10. The emission levels have been *going up and down.*

VOCABULARY LIST OF UNIT 3 "PROGRAMMING LANGUAGES AND PARADIGMS"

artificial language
ambiguous, *adj.*
ancestor, *n.*
assembler
assembly language
binary code
central processing unit (CPU)
compiler
concurrent program
constraint-based programming
convert, *v.*

data abstraction
debugging
discrete synchronous programming
embody, *v.*
enable, *v.*
error prone method
executable program
execute instruction, *expr.*
framework
functional programming
grasp, *v.*
hardware, *n.*
high-level language
implementation, *n.*
instruct, *v.*
interpreter
low-level language
markup language
markup tag
object code
object-oriented programming
perform computation
pre-defined tag
regardless of, *prep.*
restrict, *v.*
script
sequence of events
software

source code
substitute, *v.*
symbolic programming language
syntax
tedious, *adj.*

UNIT 4

DATABASE MANAGEMENT SYSTEMS

LEAD-IN

1. What is a database?
2. What kind of information can you find in databases?
3. What is the purpose of database management system?
4. What examples of database management systems do you know? What are their differences?

Oracle, MySQL, Microsoft SQL Server, PostgreSQL are the most popular database systems. Match the specific feature to the databases system it characterizes.

	1. It is one of the most popular databases for web-based applications. It's free, and it is frequently updated with features and security improvements.
	2. It is ideal for large organizations that handle enormous databases and need a variety of features.
Oracle	
MySQL	3. It is ideal for organizations that need a robust database management tool but are on a budget.
Microsoft SQL Server	4. It is available on Linux as well as Windows-based platforms.
PostgreSQL	5. PostgreSQL is one of several free popular databases, and it is frequently used for web databases.
	6. You are able to access visualizations on mobile devices.
	7. It is ideal for organizations with a limited budget that want the ability to select their interface and use JSON.
	8. It works very well with other Microsoft products.

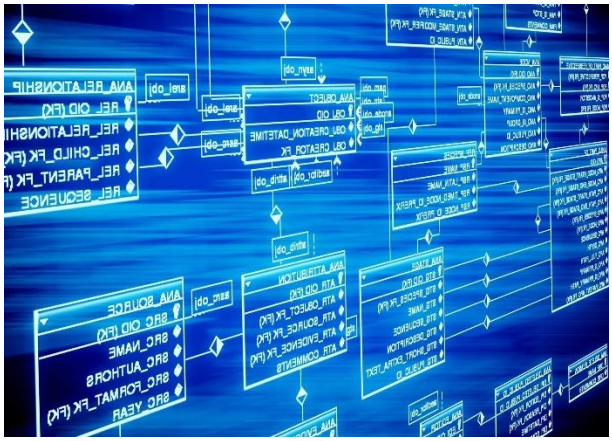
	9. The first version of this database management tool was created in the late 70s.

Learn new words and expressions before you read the text.

database	база данных
database management system (DBMS)	система управления базами данных, СУБД
related data	связанная с чем-либо информация
implicit property	неявное свойство *(проф.)*
coherent, *adj.*	связный, когерентный *(context)*
inherent meaning	исходное значение
preconceived application	заранее заданное применение
reliable, *adj.*	надежный
complexity, *n.*	сложность, коэффициент сложности
entry, *n.*	ввод, введенные данные *(context)*
general-purpose software	программное обеспечение (ПО) общего назначения
constraint of data	ограничение данных *(context)*
query, *n.*	запрос
malfunction, *n.*	неисправность
unauthorized access	неавторизованный доступ, несанкционированный доступ
malicious access	злонамеренный доступ
deploy, *v.*	развертывать

READING I

Databases



Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science.

A database is a collection of related data. By data, we mean facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

In other words, a database has some source from which data is derived, some degree of interaction with events in the real world, and an audience that is actively interested in its contents. The end users of a database may perform business transactions (for example, a customer buys a camera) or events may happen (for example, an employee has a baby) that cause the information in the database to change. In order for a database to be accurate and reliable at all times, it must be a true reflection of the miniworld that it represents; therefore, changes must be reflected in the database as soon as possible.

A database can be of any size and complexity. For example, the list of names and addresses referred to earlier may consist of only a few hundred records, each with a simple structure. On the other hand, the computerized catalog of a large library may contain half a million entries organized under different categories – by primary author's last name, by subject, by book title – with each category organized alphabetically.

A database may be generated and ma"ntai'ed manually or it may be computerized. For example, a library card catalog is a database that may be created and maintained manually. A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a database management system.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the

processes of defining, constructing, manipulating, and sharing databases among various users and applications. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database.

An application program accesses the database by sending queries or requests for data to the DBMS. A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database.

Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time. Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access. A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time.

It is not absolutely necessary to use general-purpose DBMS software to implement a computerized database. We could write our own set of programs to create and maintain the database. In either case – whether we use a general-purpose DBMS or not – we usually have to deploy a considerable amount of complex software. In fact, most DBMSs are very complex software systems.

LANGUAGE DEVELOPMENT

Exercise 4.1. Make a summary of database management systems specific features after reading the text "Databases". Your summary should contain 6-8 sentences. Be ready to present your summary to your groupmates.

Exercise 4.2. a) Match the words from two columns to make up an expression.
b) Use some expressions in sentences of your own.

1. random	a) data
2. malicious	b) applications
3. preconceived	c) malfunction
4. software	d) access
5. facilitate	e) coherent
6. retrieve	f) process
7. inherent	g) meaning
8. logically	h) assortment

Exercise 4.3. Find English equivalents for the following Russian word combinations.

1. выполнять выборку данных
2. поддерживать базы данных
3. надежная база данных
4. редактировать информацию
5. хранить данные
6. заранее заданный набор
7. связанные данные
8. запрос данных
9. присущее значение
10. получать доступ к информации
11. обновлять информацию
12. совместное использование данных

Exercise 4.4. Find on the Internet some additional facts about Oracle, MySQL, Microsoft SQL Server, PostgreSQL. Work in pairs, be ready to ask and answer questions.

READING II

Learn new words and expressions before you read the text.

interrelated data	взаимосвязанные данные
disparate information	несопоставимая информация
numerical analysis	числовой анализ
bulk, *n.*	массив данных
spreadsheet, *n.*	электронная таблица
tabular form	табличная форма
attribute, *n.*	определяющий элемент, признак, критерий *(context)*
flat-file database	одноуровневая база данных *(проф.)*
cross-reference record	перекрестная запись
aggregate calculation	обобщенное вычисление, агрегированное вычисление *(проф.)*
referential constraint	справочное ограничение *(проф.)*
referential integrity	ссылочная целостность, целостность на уровне ссылок

How databases work

In the early days of computing, a database generally consisted of a single file that was divided into data blocks that in turn consisted of records and fields within records. The COBOL language was (and is) particularly suited to reading, processing, and writing data in such files. This flat file database model is still used for many simple applications including “home data managers.” However, for more complex applications where there are many files containing interrelated data, the flat file model proves inadequate.

All computer databases are made up of tables. These tables contain records and each record is made up of some number of fields, columns and rows which are designed to provide an organized or arranged mechanism for managing, storing and retrieving information. A field is a single piece of information; a record is one complete set of fields; and a file is a collection of records. For example, a telephone book is analogous to a file. It contains a list of records, each of which consists of three

fields: name, address, and telephone number. An alternative concept in database design is known as Hypertext. In a Hypertext database, any object, whether it is a piece of text, a picture, or a film, can be linked to any other object. Hypertext databases are particularly useful for organizing large amounts of disparate information, but they are not designed for numerical analysis.

To access information from a database, you need a database management system (DBMS). This is a collection of programs that enables you to enter, organize, and select data in a database.

Databases are useful as one can manipulate data, update records in bulk, perform complex calculations and retrieve records that match particular criteria. The collected information could be in any number of formats (electronic, printed, graphic, audio, statistical, combinations). There are physical (paper/print) and electronic databases. A database could be as simple as an alphabetical arrangement of names in an address book or as complex as a database that provides information in a combination of formats.

Some databases provide numeric information, such as statistics or demographic information. Examples of these are Census Bureau databases and databases containing stock market information. You can also find databases that collect only image information (EBSCOhost image collection), audio information (MP3 or wav files), or a combination of any of the above types. CNN's site has a search option that provides access to news articles and the original video and audio files that accompanied them. Meta-databases are databases that allow one to search for content that is indexed by other databases. GOLD is an example of this kind of database.

There are thousands of different types and manufacturers of computer databases. A few common examples are Microsoft Access, Oracle and MySQL. Some computer databases are completely free, while others cost tens of thousands of dollars. Databases are used around the world by nearly every business in the twenty-first century, including vast government databases that contain huge amounts of information about citizens.

If you're familiar with spreadsheets like Microsoft Excel, you're probably already accustomed to storing data in tabular form. It's not much of a stretch to make the leap from spreadsheets to databases. Just like Excel tables, database tables consist of columns and rows. Each column contains a different type of attribute and each row corresponds to a single record.

For example, imagine that we were building a database table that contained names and telephone numbers. We'd probably set up columns named "FirstName", "LastName" and "TelephoneNumber." Then we'd simply start adding rows underneath those columns that contained the data we're planning to store. If we were building a table of contact information for our business that has 50 employees, we'd wind up with a table that contains 50 rows.

Databases are actually much more powerful than spreadsheets in the way you're able to manipulate data. Here are just a few of the actions that you can perform on a database that would be difficult if not impossible to perform on a spreadsheet:

- Retrieve all records that match certain criteria
- Update records in bulk
- Cross-reference records in different tables
- Perform complex aggregate calculations

You can correlate information from multiple tables in a database by creating foreign key relationships between the tables.

LANGUAGE DEVELOPMENT

Exercise 4.5. a) Translate the following sentences into Russian.

1. Some can manipulate only one collection of data – a table – at a time; these database programs are called *flat-file database managers.*
2. When a firewall detects suspicious activity, it sends an alert in the form of a *pop-up window* or email to notify the computer's user or the network manager that someone might have tried to break in.
3. When a calculation uses *an aggregate function,* it's called an aggregate calculation which you create by defining a new calculated field.
4. In the context of relational databases, a foreign key is a *referential constraint* between two tables.
5. *Referential integrity* is a relational database concept in which multiple tables share a relationship based on the data stored in the tables, and that relationship must remain consistent.

b) Work in pairs and take turns defining *italicized* **word combinations from part a.**

Exercise 4.6. Find Russian equivalents to the following word combinations.

1. flat-file database
2. interrelated data
3. relational model
4. manipulating data table
5. access information
6. data bulk
7. physical database
8. numerical information
9. pop-up window
10. candidate key

Exercise 4.7. Find English equivalents for the following words and word combinations.

1. целостность ссылочных данных, ссылочная целостность
2. отразить изменения
3. обращаться с запросом к базе данных
4. система управления базами данных (СУБД)
5. таблица
6. библиографическая база данных
7. полнотекстовая база данных
8. графическая информация
9. суммарные вычисления
10. внешний ключ

IMPROVE YOUR TRANSLATION SKILLS

Learn vocabulary from Unit 4 "Database management systems" and translate the following sentences from Russian into English.

1. База данных – это структурированная информация, которая хранится в связанных электронных таблицах.
2. Передача больших массивов данных — это программный механизм, разработанный для перемещения большого объема данных путем методов блокирования, компрессии и буферизации для того, чтобы оптимизировать время переноса.
3. Внешний ключ – это поле в реляционной таблице, которое совпадает с потенциальным ключом другой таблицы.
4. Ссылочная целостность означает, что внешний ключ в любой таблице должен всегда соотноситься с соответствующей строкой в справочной таблице.
5. При создании базы данных Oracle использовался язык Java с расширением PL/SQL, благодаря чему удалось добиться важных преимуществ, в частности, удобное масштабирование и обширный функционал.

FOCUS ON WRITING

Modals

Modal verbs used in academic writing tend to have three main meanings:

- Ability

May and *can* are similar but can is more common.
Study the following: The assessment … *may* be made in a variety of ways.

- Degrees of certainty

Will and *should* are used for predictions of near certainty (will is stronger). Study the following: Improved soft *should* lead to lower cost.

- *May* and *might* both suggest possibility.
  Study the following: Landfill carbon sequestration *might* supplement fossil fuel combustion.
  Would and *could* are used in conditional situations (not always with if).
  Study the following: … estimates of the model's parameters *could* conceivably be computed.

  Modal verbs that express degrees of certainty are presented in Table 8.

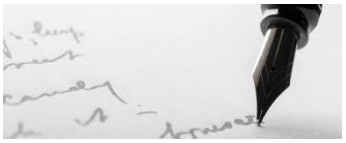
Table 8. Degrees of certainty

could/might	weak inference/low possibility
may	stronger/perhaps, quite possibly
should	strong/moderate possibility, probable

must	very strong/certainly
will	strongest/very certainly
might	low possibility
may	low possibility
won't/wouldn't	
can't/couldn't	impossibility

Degrees of obligation

Must suggests strong obligation; *should* is for recommendations.
Study the following: … to obtain a total estimate … several approximations *must* be used. … a primary research emphasis … *should* then be on identifying …



Writing Exercise 1.

Complete the following sentences with a suitable modal verb of ability.

1. The question is whether democracy _____ survive in such difficult conditions.
2. Fifty years ago a new house _____ be bought for $1500.
3. Students _____be expected to write more than one long essay a week.
4. The mistakes of past historians _____now be clearly seen.
5. Jenkins (1976) argued that aluminum _____be used in place of steel.

Writing Exercise 2.

Complete the following sentences with a suitable modal verb of certainty.

1. It _____ not be surprising if the company were bought by Microsoft.
2. Various situations _____ lead to a user's loss of confidence.
3. Other studies confirm that a permanent shift in system use _____ occur.
4. By 2020 most children _____ have internet access by the age of five.
5. If the pressure is lowered, the reaction _____ take place more quickly.

Writing Exercise 3.

Use a suitable modal verb of obligation to complete the following sentences.

1. Students studying abroad _____ take some of their favourite gadgets with them.
2. All reports _____ be returned to the main administrator by June 6th.
3. First-year undergraduates _____ take at least three modules from the list below.

Writing Exercise 4.

Use a modal verb in place of *italicized* words and phrases. You might need to reconstruct the sentence rather than substitute words.

1. The scientist *found it impossible* to present her analysis because she had not collected all her data.
2. There is a *low possibility* that the shipment arrived yesterday afternoon.
3. It is *quite likely* that such software is available at the market.
4. Partners *are likely* to see benefits from his simulation program.

VOCABULARY LIST OF UNIT 4 "DATABASE MANAGEMENT SYSTEMS"

aggregate calculation
attribute, *n.*
bulk, *n.*
coherent, *adj.*
complexity, *n.*
constraint of data
cross-reference record
database
database management system (DBMS)
deploy, *v.*
disparate information
entry, *n.*
flat-file database
general-purpose software
implicit property
inherent meaning
interrelated data

malfunction, *n.*
malicious access
numerical analysis
preconceived application
query, *n.*
referential constraint
referential integrity
related data
reliable, *adj.*
spreadsheet, *n.*
tabular form
unauthorized access

ЗАКЛЮЧЕНИЕ

Целью пособия “English for Programmers. Part I” рассматривается совершенствование умений в области профессиональной коммуникации по следующим темам: “Career in IT” («Карьера в области информационных технологий»), “Computers” («Компьютеры»), “Programming languages and paradigms” («Языки программирования и парадигмы программирования»), “Database management systems” «Системы Управления Базами Данных, СУБД».

Пособие можно использовать как основное, так и дополнительное, наряду с имеющимися пособиями для магистрантов (“English for Master’s Students”, “English for Graduate Students”).

Разнообразные лексико-грамматические упражнения, задания на совершенствования навыков перевода, проектные задания, отдельный блок материала по развитию навыков письма способствуют активному вовлечению магистрантов в учебный процесс, повышая мотивацию к изучению иностранного языка профессиональной направленности.

Автор уверен, что знание английского языка служит надежным плацдармом для карьерного роста в условиях значительной конкуренции на рынке труда, особенно в IT сфере. Уверенное владение иностранным языком открывает значительные горизонты для реализации личностного потенциала и амбициозных карьерных планов.

СПИСОК УСЛОВНЫХ СОКРАЩЕНИЙ

adj.	*adjective,* имя прилагательное
adv.	*adverb,* наречие
context	при переводе слова учитывался контекст предложения/фразы
expr.	*expression,* выражение
fixed expr.	*fixed expression,* фиксированное выражение
Lat.	*Latin,* слово латинского происхождения
n.	*noun,* имя существительное
ph.v.	*phrasal verb,* фразовый глагол
prep.	*preposition,* предлог
transl.	*translation,* перевод
v.	*verb,* глагол
проф.	профессионально-ориентированная лексика

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алешугина, Е. А., Лошкарева, Д. А. Профессионально-ориентированный английский язык для специалистов в области информационных технологий / Е. А. Алеуштина, Д. А. Лошкарева. – Нижний Новгород : Издательство ННГАСУ, 2020. – 104 с.

2. Беседина, В.Г. English for IT students / В. Г. Беседина. – Барнаул : Издательство АлтГТУ, 2024. – 85 с.

3. Волченкова, К. Н., Колегова, И. А., Истомина, Е. М., Серяпина, Ю. С. English for IT specialists / К. Н. Волченкова, И. А. Колегова, Е. М. Истомина, Ю. С. Серяпина. – Челябинск : Издательский центр ЮУрГУ, 2021. – 145 с.

4. Малашенко, Е.А. English for IT students / Е. А. Малашенко. – Минск : МГЭУ им. А. Д. Сахарова, 2014. – 140 с.

5. Шамсутдинова, Э. Х. English for programmers / Э. Х. Шамсутдинова. – Казань : Издательство Казанского университета, 2016. – 93 с.

6. Иллюстрации, представленные в настоящем пособии, заимствованы из ресурса: URL: https://yandex.ru/images/ (дата обращения: 12.06.2025)

ПРИЛОЖЕНИЕ

ADDITIONAL TEXTS AND EXERCISES

ADDITIONAL TEXT 1

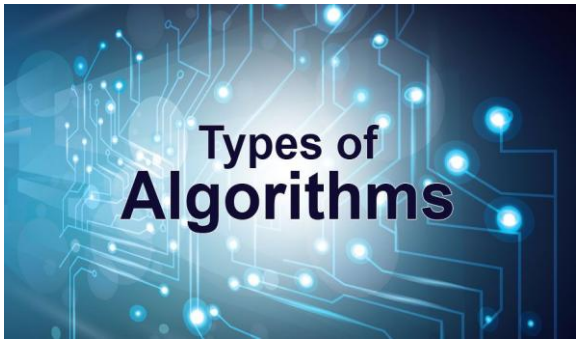
Algorithms. Types of Algorithms

Learn new words and expressions before you read the text.

implement, *v.*	разрабатывать (например, программный продукт), реализовывать совместимость (различных платформ), устанавливать программный продукт
devise, *v.*	создавать план, задумывать, придумывать, изобретать *(context)*
appropriate, adj.	приемлемый, необходимый, обоснованный, пригодный, возможный *(context)*
problem-solving method	метод решения проблемы, процедура решения проблемы
computation, *n.*	машинное вычисление
data structure, *expr.*	структура данных
go hand in hand	сопровождать, быть тесно связанным, быть неразрывно связанным с чем-либо
byproduct, *n.*	побочный продукт
endproduct, *n.*	конечный продукт
conversely, *adv.*	противоположно, в противном случае, напротив
manage complexity, *expr.*	бороться со сложностями *(context)*
decompose, *v.*	подвергать декомпозиции, разбирать на составные части, разбивать
subtask, *n.*	подзадача

fraction, *n.*	доля, дробь, дробная часть числа
tailor, *v.*	адаптировать, корректировать, разрабатывать для специальных задач *(context)*
sophisticated mathematical analysis	сложный математический анализ
analysis of algorithms	анализ алгоритмов, изучение алгоритмов
additition, *n.*	сложение
multiplication, *n.*	умножение
division, *n.*	деление
integer, *n.*	число, целочисленная переменная *(context)*
polynomial, *n.*	полином
data fitting, *expr.*	аппроксимация данных, выравнивание данных
integration, *n.*	сборка, интеграция информационных систем *(context)*
priority queue, *expr.*	приоритезированная очередь, очередь с приоритетом *(проф.)*
heap data structure, *expr.*	пирамидальная структура данных
heap sort, *expr.*	древовидная сортировка
indirect heap, *expr.*	нелинейная сортировка, динамическая сортировка
merging	слияние
“divide-and-conquer” paradigm	парадигма «разделяй и властвуй» *(проф.)*
reasonable, *adj.*	приемлемый, адекватный, обоснованный, оправданный *(context)*
restricted range, *expr.*	ограниченная область
digital property, *expr.*	цифровое свойство
radix, *n.*	основание алгоритмов, основание, основной объем выборки, основание системы счисления *(context)*
insert, *v.*	включать, подставлять, вносить *(context)*
stack, *n.*	комплекс, стек *(проф.)*

priority assignment, *expr.*	приоритезированная задача *(проф.)*
intrinsic, *adj.*	встроенный, предопределенный
sequential searching, *expr.*	последовательный поиск
hashing	хеширование, перемещение, рандомизация, размещение с использованием функции расстановки *(проф.)*
biased data, *expr.*	неравномерно-распределенные данные
quicksort	быстрая сортировка



When one writes a *computer program,* one is generally implementing a method of solving a problem, which has been previously devised. This method is often independent of the particular computer to be used: it's likely to be equally appropriate for many computers. In any case, it is the method, not the computer program itself. The term *algorithm* is universally used in *computer science* to describe problem-solving methods suitable for implementation as computer programs.

Algorithms are the "stuff" of computer science: they are central objects of study in many, if not most, areas of the field. Most algorithms of interest involve complicated methods of organizing the data involved in the *computation.* Objects created in this way are called data structures, and they are central objects of study in computer science. Thus, algorithms and data structures go hand in hand: data structures exist as the byproducts or endproducts of algorithms, and thus need to be studied in order to understand the algorithms. Simple algorithms can give rise to complicated data structures and, conversely, complicated algorithms can use simple data structures.

When a very large computer program is to be developed, a great deal of effort must go into understanding and defining the problem to be solved, managing its complexity, and decomposing it into smaller *subtasks* which can be easily implemented. It is often true that many of the algorithms required after the decomposition are trivial to implement. However, in most cases there are a few algorithms the choice of which is critical since most of the system resources will be spent running those algorithms.

The sharing of programs in computer systems is becoming more widespread, so that while it is true that a serious computer user will use a large fraction of the algorithms, he may need to implement only a somewhat smaller fraction of them. However, implementing simple versions of basic algorithms helps us to understand them better and thus use advanced versions more effectively in the future.

Also mechanisms for sharing software on many computer systems often make it difficult to tailor standard programs perform effectively on specific tasks, so that the opportunity to reimplement basic algorithms frequently arises. Computer programs are often overoptimized. It may be worthwhile

to take pains to ensure that an implementation is the most efficient possible only if an algorithm is to be used for a very large task or is to be used many times. In most 7 situations, a careful, relatively simple implementation will branch: the programmer can have some confidence that it will work, and it is likely to run only five or ten times slower than the best possible version, which means that it may run for perhaps an extra fraction of a second. By contrast, the proper choice of algorithm in the first place can make a difference of a factor of a hundred or a thousand or more, which translates to minutes, hours, days or more in running time. Often several different algorithms (or implementations) are available to solve the same problem.

The choice of the very best algorithm for a particular task can be a very complicated process, often involving sophisticated *mathematical analysis.* The branch of computer science where such questions are studied is called analysis of algorithms. Many of the algorithms that we will study have been shown to have very good performance through analysis, while others are simply known to work well through experience. We will not dwell on comparative performance issues: our goal is to learn some reasonable algorithms for important tasks. But we will try to be aware of roughly how well these algorithms might be expected to perform. Below are brief descriptions of the basic algorithms.

Algorithms for doing elementary arithmetic operations such as addition, *multiplication,* and *division* have a very long history, dating back to the origins of algorithm studies in the work of the Arabic mathematician al-Khowdrizmi, with roots going even further back to the Greeks and the Babylonians. Computers have built-in capabilities to perform arithmetic on integers and floating-point representations of real numbers; for example, Pascal allows numbers to be of type integer or real, with all of the normal arithmetic operations defined on both types. Algorithms come into play when the operations must be performed on more complicated mathematical objects, such as polynomials or matrices.

So, mathematical algorithms include fundamental methods from arithmetic and numerical analysis. We study methods for addition and multiplication of integers, polynomials, and matrices as well as algorithms for solving a variety of mathematical problems which arise in many contexts: random number generation, solution of simultaneous equations, data fitting, and integration. There are several sorting applications in which a relatively simple algorithm may be the method of choice. Sorting programs are often used only once (or only a few times). If the number of items to be sorted is not too large (say, less than five hundred elements), it may well be more efficient just to run a simple method than to implement and debug a complicated method.

Elementary methods are always suitable for small files; it is unlikely that a sophisticated algorithm would be justified for a small file, unless a very large number of such files are to be sorted. Other types of files that are relatively easy to sort are ones that are already almost sorted or ones that contain large numbers of equal keys. Simple methods can do much better on such well-structured files than general-purpose methods. So, sorting methods for rearranging files into order are of fundamental importance. A variety of methods are priority queues (e.g. elementary implementations, heap data structure, algorithms on heaps, heap sort, indirect heaps, advanced implementations, selection (finding the smallest element (or finding the smallest elements) in a file), and merging.

The “keys” used to define the order of the records in files for many sorting applications can be very complicated. For example, consider the ordering function used in the telephone book or a library catalogue. Because of this, it is reasonable to define sorting methods in terms of the basic operations of “comparing” two keys and “exchanging” two records. Most of the methods we have studied can be described in terms of these two fundamental operations. For many applications, however, it is possible to take advantage of the fact that the keys can be thought of as numbers from some restricted range. Sorting methods, which take advantage of the digital properties of these

numbers are called radix sorts. These methods do not just compare keys: they process and compare pieces of keys.

Radix sorting algorithms treat the keys as numbers represented in a base-M number system, for different values of M (the radix) and work with individual digits of the numbers. In many applications, records with keys must be processed in order, but not necessarily in full sorted order and not necessarily all at once. Often a set of records must be collected, then the largest processed, then perhaps more records collected, then the next largest processed, and so forth. An appropriate data structure in such an environment is one, which supports the operations of inserting a new element and deleting the largest element. This can be contrasted with queues (delete the oldest) and stacks (delete the newest). Such a data structure is called a priority queue. In fact, the priority queue might be thought of as a generalization of the stack and the queue, since these data structures can be implemented with priority queues, using appropriate priority assignments.

Applications of priority queues include simulation systems (where the keys might correspond to "event times" which must be processed in order), job scheduling in computer systems (where the keys might correspond to "priorities" which indicate which users should be processed first), and numerical computations (where the keys might be computational errors, so the largest can be worked on first). A fundamental operation intrinsic to a great many computational tasks is searching. Searching methods for finding things in files are also of fundamental importance. Normally we think of the information as divided up into records, each record haling a key for use in searching. The goal of the search is to find all records with keys matching a given search key.

The purpose of the search is usually to access information within the record (not merely the key) for processing. Sequential Searching is the simplest method for searching is simply to store the records in an array, then, look through the array sequentially each time a record is sought. Sequential List Searching is the seqsearch program, which uses purely sequential access to the records, and thus can be naturally adapted to use a linked list representation for the records.

Binary Search: if the set of records is large, then the total search time can be significantly reduced by using a search procedure based on applying the "divide-and-conquer" paradigm: divide the set of records into two parts, determine which of the two parts the key being sought be longs to, then, concentrate on that part. Binary tree search is a simple, efficient dynamic searching method, which qualifies as one of the most fundamental algorithms in computer science.

A completely different approach to searching from the comparison based tree structures of the last section is provided by hashing (Separate Chaining, Open Addressing, Analytic Results): directly referencing records in a table by doing arithmetic transformations on keys into table addresses. If we were to know that the keys are distinct integers from 1 to N, then we could store the record with key *i* in table position *i,* ready for immediate access with the key value. Hashing is a generalization of this trivial method for typical searching application. Several searching methods proceed by examining the search keys one bit at a time (rather than using full comparisons between keys at each step). These methods, called radix searching methods (Digital Search Trees, Radix Search Tries, Multiway Radix Searching, Patricia), work with the bits of the keys themselves, as opposed to the transformed version of the keys used in hashing.

As with radix sorting methods, these methods can be useful when the bits of the search keys are easily accessible and the values of the search keys are well distributed. The principal advantages of radix searching methods are that they provide reasonable worst-case performance without the complication of balanced trees; they provide an easy way to handle variable-length keys; some allow some savings in space by storing part of the key within the search structure; and they can provide

very fast access to data, competitive with both binary search trees and hashing. The disadvantages are that biased data can lead to degenerate trees with bad performance (and data comprised of characters is biased) and that some of the methods can make very inefficient use of space. These methods are related to each other and similar to sorting methods.

Some of these algorithms are used as the basis for other algorithms. Selection and merging are complementary operations in the sense that selection splits a file into two independent files and merging joins two independent files to make one file. The relationship between these operations also becomes evident if one tries to apply the "divide-and-conquer" paradigm to create a sorting method. The file can either be rearranged so that when two parts are sorted the whole file is sorted or broken into two parts to be sorted and then combined to make the whole file sorted. An elementary sorting method that is often taught in introductory classes is bubble sort: keep passing through the file, exchanging adjacent elements, if necessary; when no exchanges are required on some pass, the file is sorted. Quicksort is the sorting algorithm, which is probably more widely used than any other.

The study of algorithms is interesting because it is a new field (almost all of the algorithms are less than twenty-five years old) with a rich tradition (a few algorithms have been known for thousands of years). New discoveries are constantly being made, and few algorithms are completely understood.

EXERCISES TO THE ADDITIONAL TEXT 1

Exercise 1.1. Give the translation to the following words and words combinations presented in the table below. If necessary, consult with the dictionary.

English word or word combination	*Russian equivalent*
1. heap data structure	a)
2. insertion sort	b)
3. general-purpose method	c)
4. adjacent elements	d)
5. shell sort	e)
6. divide-and-conquer algorithm	f)
7. bubble sort	g)
8. variable-length coding	h)
9. quicksort	i)
10. sorting algorithm	j)

Exercise 1.2. Translate the following sentences from English into Russian.

1. In 1830, Charles Babbage invented the Analytical Engine, which was different from its predecessors because, based on the results of its own computations, it could make decisions such as sequential control, branching, and looping.
2. The utilities use different algorithms that emphasize storage efficiency at the expense of speed.
3. The software simultaneously repositions the transported pixels and the polygons they form to warp the emerging image so the pixels move steadily toward the positions they occupied in the picture from which they came.

4. The compression program uses some variation of a scheme generally called LZ (after its creators, Lempel and Ziv) adaptive dictionary-based algorithm.
5. Although the algorithm was initially designed for small datasets, its efficiency significantly diminishes when applied to large-scale data due to its exponential time complexity.
6. While many algorithms rely on recursive strategies to simplify problem-solving, some, like dynamic programming, optimize performance by storing intermediate results to avoid redundant calculations.
7. Since the sorting algorithm's stability is crucial for maintaining the relative order of equal elements, developers often prefer algorithms like merge sort over quicksort in such scenarios.
8. Although machine learning models depend heavily on the quality of training data, the underlying algorithms must also be carefully tuned to prevent overfitting and ensure generalization.
9. Because graph traversal algorithms such as breadth-first search and depth-first search have different use cases, selecting the appropriate one depends on whether the goal is shortest path discovery or connectivity analysis.
10. While cryptographic algorithms are designed to secure data through complex mathematical operations, advances in computational power continually challenge their robustness, necessitating ongoing research into more secure methods.

Exercise 1.3. Translate the sentences from Russian into English. You may need to consult with the dictionary while doing the exercise.

1. Центральный процессор все ставит в режим ожидания, что позволяет записать адрес возврата в стек.
2. Цифровая подпись обычно создается путем вычисления свертки сообщения или значения хеш-функции.
3. Операционная система ставит эту операцию в очередь для последующего выполнения в указанное время в соответствии с запросами, которые передаются на запоминающее устройство.
4. Чип хранит характеристики разных музыкальных инструментов в виде набора математических описаний, называемых алгоритмами.
5. Несмотря на то, что алгоритм сортировки обладает высокой стабильностью и эффективностью при работе с большими объемами данных, его реализация требует значительных затрат памяти, что может стать критичным в условиях ограниченных ресурсов.
6. Хотя алгоритмы поиска кратчайшего пути широко применяются в навигационных системах, их эффективность зависит от структуры графа и наличия эвристических функций, что зачастую усложняет их настройку.
7. Поскольку алгоритмы машинного обучения основываются на обработке больших массивов данных и сложных математических моделей, их обучение требует значительных вычислительных ресурсов и времени, особенно при использовании глубоких нейронных сетей.
8. Несмотря на то, что алгоритмы шифрования обеспечивают высокий уровень безопасности данных, развитие квантовых вычислений ставит под угрозу их стойкость, что вызывает необходимость разработки новых криптографических методов.
9. Хотя алгоритмы часто применяются для решения задач оптимизации благодаря своей простоте и скорости, они не всегда дают оптимальный результат.
10. Поскольку алгоритмы динамического программирования позволяют эффективно решать задачи с оптимизацией и повторяющимися подзадачами, их применение

значительно упрощает разработку решений для сложных задач, таких как вычисление последовательностей или маршрутов.

Exercise 1.4. Work with *the italicized* words from the text. Give definitions to them using the dictionary. The first definition is presented as an example for you.

The italicized word from the text	*Definition*
1. Algorithm	**a)** a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
2. Computer program	**b)**
3. Computer science	**c)**
4. Computation	**d)**
5. Subtask	**e)**
6. Mathematical analysis	**f)**
7. Multiplication	**g)**
8. Division	**h)**

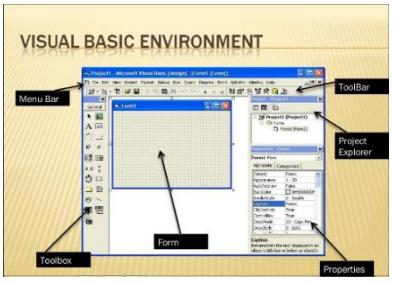
ADDITIONAL TEXT 2

Learn new words and expressions before you read the text.

punch, *v.*	наносить *(context)*
teletype, *n.*	буквопишущий телеграфный аппарат, дистанционное печатающее устройство
CRT terminal	терминал с устройством отображения на ЭЛТ, экранный терминал
compiler, *n.*	компилятор *(проф.)*
assembler, *n.*	ассемблер *(проф.)*
timesharing system, *expr.*	система с распределением времени
editing facility, *expr.*	средства редактирования
cumbersome, *adj.*	утомительный, громоздкий, проблематичный *(context)*
Integrated Development Interface (IDE)	интегрированная среда разработки
vector graphics, *expr.*	векторная графика
event handling, *expr.*	управление событиями, обработка событий *(context)*
large-scale schematic	крупномасштабная схема
context-free programming, *expr.*	контекстно-независимое программирование *(проф.)*

toolkit, *n.*	инструментарий, пакет инструментальных средств разработки
low-latency execution, *expr.*	режим выполнения с малой задержкой
uptime, *n.*	период работоспособного состояния, аптайм *(проф.)*
protocol stack, *expr.*	пакет протоколов, стек протоколов
operating system settings, *expr.*	установочные параметры операционной системы
deployment, *n.*	внедрение
Remote Procedure Call (RPC)	сервер вызова удаленных данных
wrapper, *n.*	оболочка (например, функциональная) *(context)*
widget, *n.*	виджет
top-level window, *expr.*	окно высшего уровня

Visual Programming Environment



The first programmers used pencil and paper to sketch out a series of commands or punched them directly on cards for input into the machine. However, as more computer resources became available, it was a natural thought that programs could be used to help programmers create other programs. The availability of Teletype or early CRT terminals on timesharing systems by the 1960s encouraged programmers to write simple text editing programs that could be used to create the computer language source code file, which in turn would be fed to the compiler to be turned into an executable program.

The *assemblers* and BASIC language implementations on the first personal computers also included simple editing facilities. More powerful *programming editors* soon evolved, particularly in academic settings. One of the best known is EMACS, an editor that contains its own LISP like language that can be used to write macros to automatically generate program elements. With the many other utilities available in the UNIX *operating system,* programmers could now be said to have a programming environment— a set of tools that can be used to write, compile, run, debug, and analyze programs. More tightly integrated programming environments also appeared.

The UCSD “p-system” brought together a program editor, compiler, and other tools for developing Pascal programs. While this system was somewhat cumbersome, in the mid-1980s Borland International released (and steadily improved) Turbo Pascal. This product offered what became known as an “integrated development interface” or IDE. Using a single system of menus and windows, the programmer could edit, compile, run, and debug programs without leaving the main window.

The release of Visual Basic by Microsoft a few years later brought a full graphical user interface (GUI). Visual Basic not only ran in Windows, it also gave programmers the ability to design programs by arranging user interface elements (such as menus and dialog boxes) on the screen and then attaching code and setting properties to control the behavior of each interface object. This approach was soon extended by Microsoft to *development environments* for C and C++ (and later, Java) while Borland released Delphi, a visual Pascal development system. So what is visual programming environment?

VPE is software, which allows the use of visual expressions (such as graphics, drawings, animation or icons) in the process of programming. These visual expressions may be used as graphical interfaces for textual programming languages. They may be used to form the syntax of new visual programming languages leading to new paradigms such as programming by demonstration or they may be used in graphical presentations of the behaviour or structure of a program.

Today visual programming environments are available for most languages. Indeed, many programming environments can host many different languages and target environments. Examples include Microsoft's Visual Studio.NET and the open-source Eclipse, which can be extended to new languages via plug-ins. In short, VPE is a visual programming environment, which supports general purpose visual programming. Theoretically, it can be used to create any application or system.

It is based on both structural (*vector graphics,* graphs, capable of event handling) and nonstructural graphics (raster graphics without event handling), therefore graphs in schematics can be used to define inputs, outputs, bodies, groups, wires, and connectors of elements as well as event handling related to them. VPE is very suitable for designing complex large-scale schematics. VPE is visual development environment, or integrated development environment, for rich visual programming, with the following features:

1. *Context-free programming* environment: no strictly specialized;
2. Multi-platform: Linux, *BSD, Mac OS X, Solaris, Windows;
3. Multi-language: Python, C, C++, C#, Java, JavaScript, Tcl, Lua;
4. Multi GUI toolkit: Gtk, Qt, Tk (Tkinter), Fltk, Swing, Forms;
5. Decentralized and distributed systems;
6. Real-time systems;
7. Low-latency execution with very fast feedback;
8. Continuous execution and document modification;
9. With 100% uptime of application;
10. Running existing protocol stacks/suits in different threads, or even processes, and communicating with them;
11. No need to restart application after document (e.g. SCADA) is modified to accept local or global changes.

In the VPE, visual elements can be graphically designed, programmed in one of supported programming languages (such as Python or C), saved and arranged in *libraries.* After basic elements are built and stored into the libraries, they can be used for the construction of schematics, which are basically documents. It is possible to save a document for later use. Full object serialization is supported by default, so after closing and reopening of the application, and loading the previously saved document in the application, it is theoretically possible to repeat the same conditions under which the document was previously operating.

The initial code is located in the document, but event handlers are located in the elements. Elements can be programmed in various languages, such as: Python, Tcl, C, etc. Primarily in the

application itself, but it is advisable to use the Python programming language because it can directly affect the state and behavior of the program/document and all its parts. For instance, it is possible to change the visual appearance of main application, or even operating system settings.

Whole environment is dynamic, meaning that it can be changed at runtime, and adapted to the specific needs. For instance, when designing, it is necessary to see all the options in the VPE, but after deployment, it is needed to prohibit the possibility of drawing and programming to avoid errors and unexpected situations, and reduce the chance for error. VPE is a multi-platform application that runs on all modern operating systems. It is written for use on the Linux operating system, but can also be used in other operating systems. VPE is primarily designed to create SCADA applications. SCADA applications require high precision in numerical operations, which is one of the reasons why Python programming language is used. In addition to accuracy, SCADA systems also require highly *reliability.*

After designing the general solution using the Python programming language as proof of concept, it is suggested to optimize the solution for the greater execution speed and lower memory usage in the C programming language. VPE comes with all the necessary standard libraries for most of basic operations, and these are mainly operating system libraries, such as: threading, processing, socket, http, ftp, RPC client/server, etc. Main reference implementation of VPE is written in Python, Tcl, and C programming languages.

GUI toolkits used are Tk (Tkinter) and Gtk (PyGtk), but in parallel the application can use other GUI toolkits, such as: Qt, Wx, Forms, Swing, SWT, etc, and their wrappers for different programming languages. Different widgets are available only in an independent top-level window that corresponds to that GUI toolkit. The reason for this is the inability of a GUI toolkit to recognize widgets from other GUI toolkits. It should have in mind that VPE is not a GUI editor, but abstract visual editor with a programmable document and elements. VPE is the next generation of visual programming languages, similar to Auto CAD & LabVIEW, allowing complex programmable graphics. VPE can create context-free applications allowing for example telecommunications and other electronics equipment working together in same document, in other words, different unrelated devices can friendly work together in a complex system.

EXERCISES TO THE ADDITIONAL TEXT 2

Exercise 2.1. Give the translation to the following words and words combinations presented in the table below. If necessary, consult with the dictionary.

English word or word combination	*Russian equivalent*
1. handling	a)
2. distributed system	b)
3. low-latency execution	c)
4. raster graphics	d)
5. context-free	e)
6. real-time system	f)
7. visual editor	g)

Exercise 2.2. Answer the questions using information from the text.

1. What was VPE primarily designed for?
2. What is GUI?
3. Describe the importance of Visual interface design.
4. Name the feature of VPE.
5. Why it is important to study detailed code or functions of windows/visual programming, rather there are many professional tools like Visual Studio etc. available that do the same in mush less time as compared to write a detailed code?
6. What are the reports used in graphics application?

Exercise 2.3. Translate the sentences from English into Russian. You may need to consult with the dictionary while doing the exercise.

1. The first application was developed in the 1950s for handling radioactive materials.
2. The AWT (Abstract Windowing Toolkit) is a set of classes that provide a graphical user interface.
3. Knuth also did important work in areas such as LR (left-to-right, rightmost) parsing, a context-free parsing approach used in many program language interpreters and compilers or detail.
4. Raster data is easy to work with, but the “coarseness” of the grid means that it does not capture much local variation.
5. A wrapper is used to compress and encrypt software that is being sold over the Internet.
6. While navigating through the integrated development environment intuitive interface, the developer meticulously customized the layout to optimize workflow efficiency and enhance visual clarity for complex project management.
7. Although the visual debugging tools provided real-time feedback on code execution, the programmer had to carefully interpret layered graphical representations to identify subtle inconsistencies in the application's user interface.
8. Since the visual environment supports drag-and-drop component assembly, designers can rapidly prototype user interfaces; however, they must still write underlying code to implement advanced functionalities that are not accessible through visual means alone.
9. Because the development environment integrates version control with visual tools, team members can seamlessly compare interface changes over time, but understanding these differences requires familiarity with both graphical and textual representations.
10. Although the software offers extensive customization options for visual themes and layout configurations, configuring these settings often involves navigating complex menus and understanding underlying configuration files to achieve a cohesive development workspace.

Exercise 2.4. Translate the sentences from Russian into English. You may need to consult with the dictionary while doing the exercise.

1. Операционная система реального времени (ОСРВ) это операционная система, нацеленная на обработку запросов приложений системы реального времени.
2. В компьютерном программировании это графический элемент пользовательского интерфейса.

3. Распределенные вычисления – это область вычислительных машин и систем вычисления, которые изучают распределенные системы.
4. Начиная с 1970-х операционная система UNIX предусматривала в себе строчный редактор (ed или ex) и экранный редактор (vi).
5. Последовательное упорядочение объекта означает сохранение значения объекта в любой момент времени.
6. В современных визуальных средах разработки, таких как Visual Studio Code и JetBrains Rider, интеграция систем контроля версий осуществляется через встроенные плагины, что значительно упрощает управление проектами и обеспечивает более эффективное командное сотрудничество.
7. Использование drag-and-drop интерфейсов в визуальных средах разработки позволяет разработчикам быстро создавать пользовательские интерфейсы без необходимости писать код вручную, однако при этом возникает риск снижения гибкости и точности настройки элементов.
8. Визуальные среды разработки, поддерживающие автоматическую генерацию кода на основе графических моделей или диаграмм, значительно ускоряют процесс прототипирования и позволяют снизить вероятность ошибок при реализации бизнес-логики.
9. Несмотря на преимущества визуальных сред разработки в плане интуитивности и быстроты создания приложений, их ограниченная гибкость по сравнению с текстовыми редакторами иногда приводит к сложности реализации нестандартных решений или оптимизации производительности.
10. Современные визуальные среды разработки активно внедряют функции анализа кода и автоматического исправления ошибок, что способствует повышению качества программного обеспечения и снижению времени на этапы тестирования и отладки.

Exercise 2.5. Work with *the italicized* words from the text. Give definitions to them using the dictionary. The first definition is presented as an example for you.

The italicized word from the text	*Definition*
1. Vector graphics	**a)** Vector graphics are digital images created using a series of commands or mathematical formulas that define lines, shapes, and curves in a two-dimensional or three-dimensional space.
2. Assembler	**b)**
3. Programming editor	**c)**
4. Operating system	**d)**
5. Development environment	**e)**
6. Library	**f)**
7. Context-free programming	**g)**
8. Reliability	**h)**