Computers

Computers are found everywhere in daily life: in phones, smart watches, cars… the list goes on. In this topic, you'll investigate computer **hardware** and **software**. You'll find out about the stored program concept and the role of hardware components, including the **CPU**, **memory**, **buses** and the **clock**, in the fetch-decode-execute cycle.

You'll learn why computers need both primary and secondary storage and how data is physically stored on different types of devices (**magnetic**, **optical**, **solid state**). You will explore the concept of an embedded system and find out about applications of **embedded systems**, including the **Internet of Things**. You'll also learn about the role played by the **operating system** in a computer, different types of **utility software** and characteristics of **high- and low-level programming** languages, including how they are translated into machine code.

What is a computer system?

Learning outcomes

By the end of this section you should be able to:

- identify hardware and software components of a computer system.
- describe the input-process-output model.

How many computers do you have in your house? There are probably a lot more than you think. Let's find out why.

Figure 3.1 How many of these things would you think of as computers?

You might be surprised to know that every device shown in the photograph is, or contains, a computer. A computer is a machine that takes some kind of **input** from its surroundings and then **processes** it to produce an **output**.

Computer systems consist of **hardware** and **software**. Hardware refers to the physical components, such as a keyboard, sensors, a hard drive, the central processing unit (CPU), memory, a graphics card or a screen. Software is the set of programs running on the hardware which direct the computer to perform specific operations.

The clever bit in computing is the activity that goes on between the inputs and outputs – the processing. Processing involves following a set of instructions – a **program** – to do something to the input in order to generate an output. You might have met this idea in maths already if you have studied 'function machines'. You might also have processed ingredients according to a recipe to make a cake; or put together steps according to a choreography to make a dance; or musical notes in harmonic combinations to make a melody.

Key terms

Input: data sent to a computer for processing.

Process: a program that is loaded into main memory and being executed. An input is acted upon to generate an output.

Output: data that was processed by and sent out from a computer.

Hardware: the physical components of a computer system.

Software: the set of programs running on the computer which direct it to perform specific operations.

Program: an algorithm that has been converted into program code so that it can be executed by a computer.

Summary

Computers come in many shapes and sizes.

- A computer system consists of hardware and software.
- All computers operate according to the input-process-output model.

3.2 Software

Learning outcomes

By the end of this section you should be able to:

- explain the purpose of an operating system and describe how it manages files, processes, peripheral devices and users.
- describe how utility software tools are used for file repair, backup, data compression, disk defragmentation and protecting against malware.
- explain why it is important to develop robust software and describe how audit trails and code reviews help to identify code vulnerabilities.

Software is the set of programs that run on a computer system. There are three main types of software.

- **Operating system** – a program that controls and manages the hardware and all other software on a computer and provides an interface for **liveware** (users).
- **Application software** – programs or apps designed for end users, such as a web browser, a spreadsheet, a console game, an email client or a payroll application.
- **Utility software** – programs that add functionality to a computer or improve its performance in some way, such as disk repair, compression or anti-malware.

Key terms

Operating system: a program that controls and manages the hardware and other software on a computer.

Liveware: users of a computer system.

Application software: programs or apps designed for end users such as a web browser, an email client or a spreadsheet.

Utility software: programs that add functionality to a computer or improve its performance in some way. These include backup and disk repair.

Directory: a location for storing files – sometimes called a folder.

Sub-directory: a directory inside another directory.

File: a collection of related data. The operating system keeps a record of a file's name, type and attributes.

Sector: a subdivision of a track on a disk. Each sector holds a fixed amount of data.

The operating system

The operating system (OS) is the most important software on a computer as it enables the hardware to communicate effectively with the software and provides an interface for users to interact with the computer.

You will probably already know several operating systems, for example Microsoft® Windows®, Linux, Android, iOS etc.

When the computer starts up, the OS is loaded and performs a number of key functions.

File management

The OS is responsible for organising and keeping track of files stored on a computer's secondary storage. It uses a hierarchical tree structure.

The top node of the tree is the root directory. Users have to have special administrator privileges to access the root directory. Nodes lower down are either **directories**, **sub-directories** or **files**. Every file in the system has a unique path so that it can be easily located. This also allows the same filename to be used more than once.

When there's a new file to be stored, the OS splits it up into blocks and stores each block in an empty **sector** on the hard drive.

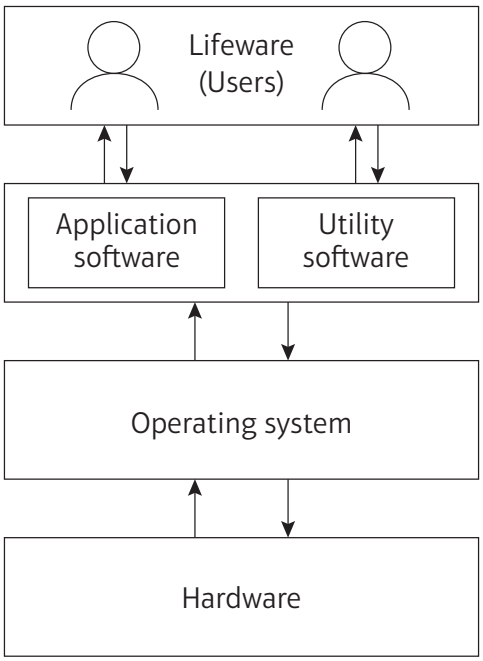


Figure 3.2.1 The OS manages hardware, software and liveware

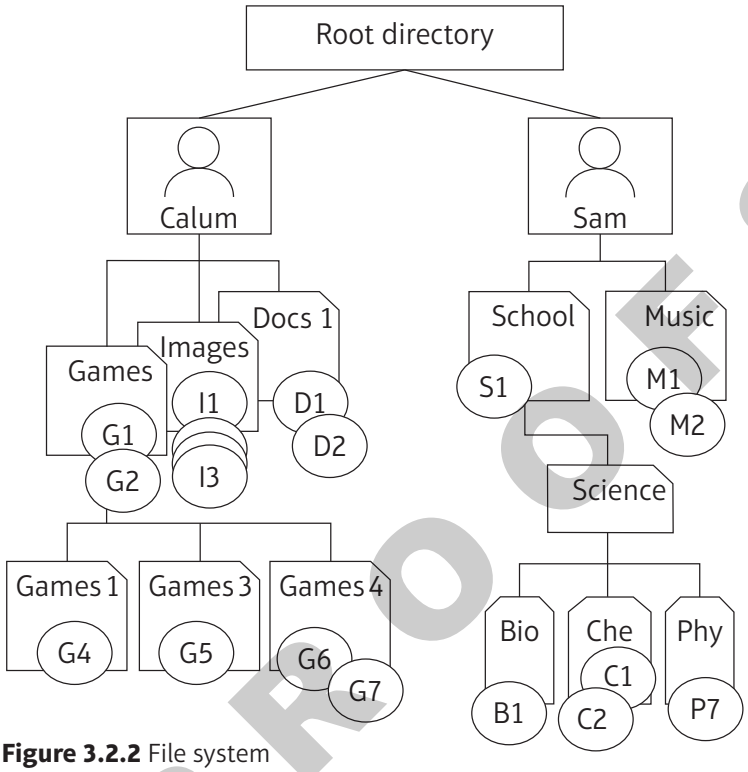


Figure 3.2.2 File system

It keeps a record of the start location and sequence number of each block so that it can retrieve and reassemble the file when it is required.

The OS provides a number of functions for users to manage their files and directories, such as save, open, copy, duplicate, rename and delete.

File permissions are used to control who can access a file and what they are allowed to do with it.

- Read: users can view the content of a file but not alter it.
- Write: users can read and amend the content of a file.
- Execute: users can execute the file.
- Delete: users can delete a file.

Key terms

File permissions: control the ability of users to view, change and execute files.

Activity

Sally and Ahmed work for a company that makes printers. Ahmed writes the user manuals for the printers. Sally programs the robots that assemble the printers.

Decide what levels of access they should be given to files containing:

a) designs for new printers
b) user manuals for printers already on the market
c) programs to control the robots
d) first drafts of a user manual for a new printer to be launched later win the year.

Practical activity

A text file is stored on a hard disk. The file holds information from one side of a sheet of paper.

The sheet of paper is represented as a grid, 80 columns wide and 66 rows long.

Each cell in the grid contains a single 7-bit ASCII character.

The hard disc allocates space in blocks of 1024 bytes.

Construct an expression to show the number of blocks required to store the file.

You do not need to do the calculation.

Process management

Programs are stored on a hard drive or some other form of secondary storage. They are loaded into the computer's main memory in order to be executed.

A **process** is a program that is loaded into main memory and being executed. Some processes are visible to the end user as application software, but others run unnoticed in the background.

There are usually several processes loaded into a computer's memory at any one time – all of them vying for use of its resources. The OS is responsible for allocating each of them a share of the CPU, main memory and peripherals and for keeping track of their progress. This allows several applications to be available at the same time and is known as **multitasking**.

Multitasking enables a computer to appear to execute several processes **concurrently** (at the same time). Like spinning plates, the OS must keep all the processes moving forward as quickly as possible towards completion. Ideally, while one process is waiting for a printer, another is using the CPU and a third is waiting for data to be loaded into memory from secondary storage.

The OS uses a **scheduling algorithm** to ensure that each process gets a share of CPU time. Strategies for doing this include:

- **first in, first out**: processes queue in the order in which they arrive. The process at the head of queue gets to use the CPU until it is finished or is held up waiting for another resource like a printer
- **shortest job first**: the process which is closest to finishing goes next
- **round robin**: each process is assigned a **time slice**. Processes wait their turn in a queue. The process at the head of the queue goes next. At the end of its time slice, if it isn't finished, it goes to the end of the queue and the next process is taken from the front of the queue. Some processes have a higher priority than others and get more time slices.

Key terms

Process: a program that is loaded into main memory and being executed.

Scheduling algorithm: used to allocate each process running on a computer a share of the CPU.

Time slice: the maximum time a process is allowed to execute in the CPU on each turn.

Multitasking: in an operating system, this enables several applications to be available at the same time.

Concurrent processing: several processes are executed simultaneously. In a computer with a single CPU this is achieved by giving each process a time slice.

Key terms

Paging algorithm: used to move programs from RAM to an area of the hard drive allocated as virtual memory and back again when needed once main memory is full.

Peripheral device: a hardware device such as a printer or a keyboard attached to a computer.

Device driver: a program that relays instructions and data between the operating system and a peripheral device. Each device has its own driver.

The CPU is not the only resource that is in demand. Sometimes there's not enough memory to store all the processes being executed. When RAM gets full, the OS has to create an area of virtual memory on the hard drive. It uses a **paging algorithm** to swap processes out of RAM into virtual memory and back again when it is their turn to use the CPU.

Slow peripheral devices, such as printers, can sometime cause hold-ups. Often there are several processes competing for use of a peripheral device. When this happens, the OS decides which one of them gets to use the device and for how long.

Peripheral management

Most computer systems include a number of **peripheral devices**, such as printers, keyboards and headphones. Each peripheral attached to a computer has its own **device driver** – a piece of software that relays instructions and data between the OS and the peripheral device.

An operating system comes with some built-in drivers, but when a new device is added its device driver has to be installed in order for it to work.

When a process needs to access a peripheral, it sends a request to the OS. The OS passes on the necessary instructions and data to the appropriate device driver.

Computer Science in action: Drivers

Windows® Vista® was a highly controversial operating system in the early days after its launch in 2007. When installed, a number of peripherals, such as printers and keyboards, no longer worked. Despite a long Beta period, this compatibility issue was not picked up prior to launch. The problem was mostly due to increased security built into Vista, but it meant that users were no longer able to carry out actions they were used to doing in its predecessor, Windows XP®.

Activity

Draw up a table to compare the benefits and drawbacks of the three scheduling algorithms mentioned above.

Key terms

User interface: the way the user interacts with the computer system.

GUI: graphical user interface.

WIMP: windows, icons, menus and pointers – the term used for operating systems like Microsoft® Windows® and macOS.

CLI: command-line interface.

User management

The OS also controls the way in which users interact with the computer. It provides and runs the **user interface**. Sometimes this is a graphical user interface (**GUI**) with windows, icons, menus and pointers (**WIMP**) like Microsoft® Windows® or macOS, or sometimes it is just a text, command-line interface (**CLI**).

The choice of user interface depends on the applications a computer is expected to run. A computer acting as a web server might only need a

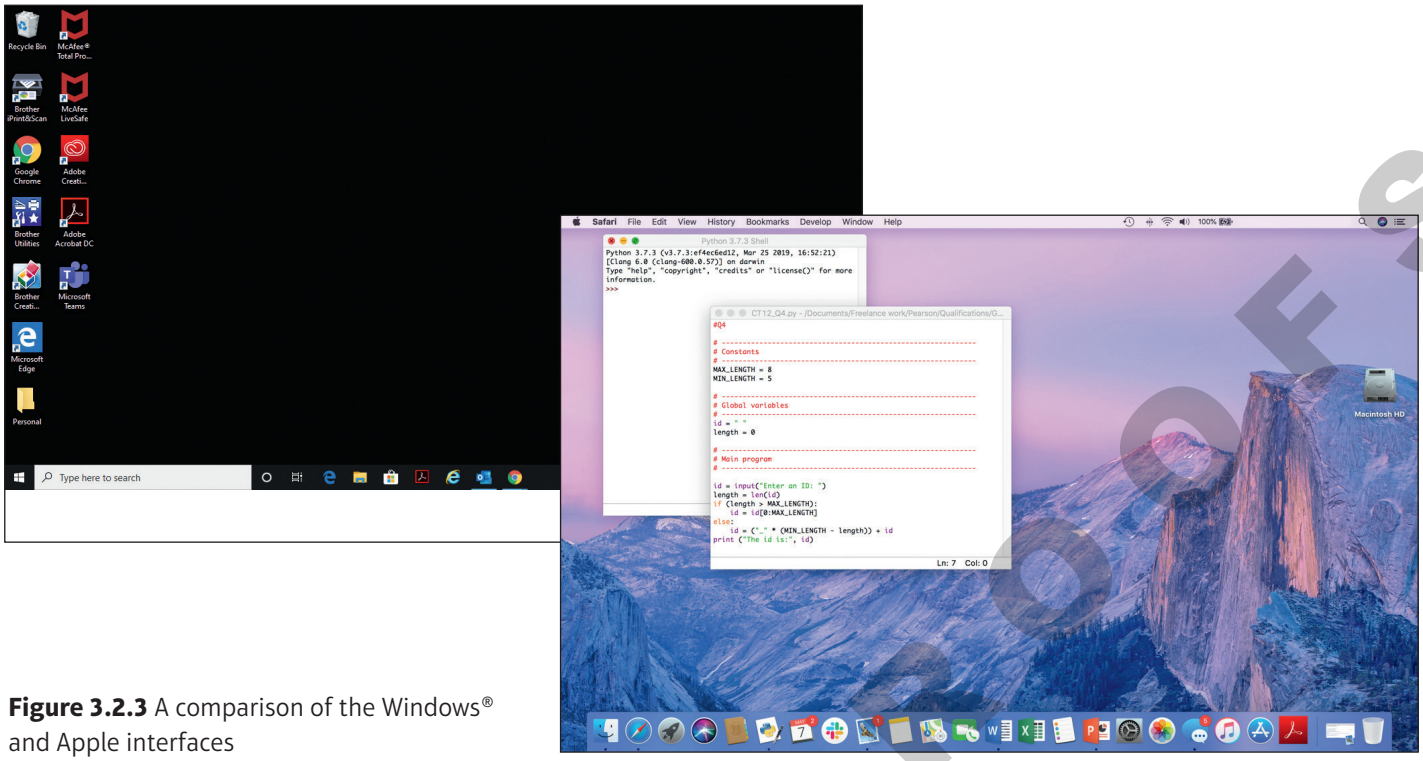


Figure 3.2.3 A comparison of the Windows® and Apple interfaces

command-line interface because it is intended only ever to be managed by technicians. A computer used for video-editing will definitely need a graphical user interface.

The OS is responsible for **access control** – determining who is allowed to use the computer system. It uses passwords or some form of biometric scan, such as fingerprint or face recognition to **authenticate** the identity of a person seeking access.

Authority to use the facilities of the computer system is based on who you are, for example learner users might not be able to print, whereas tutor users can. In such systems, there is usually also a super-user (sometimes called administrator or root) who can access all resources and also control the access rights of other users.

You can find out more about access control of networks in Topic 4.

Utility software

There are a number of free and proprietary (paid-for) tools available that help minimise the impact of data loss or damage caused by events such as a system failure, a cyberattack, a fire or a flood.

File repair

A file can become **corrupt**, unrecognisable or damaged as a result of a **cyberattack** or a device malfunction. When this happens, it will need to be repaired before it can be re-opened or read. This is the job of a **file repair tool**.

Key terms

Access control: using some form of authentication to control who has access to a computer system or network.

Authentication: the process of checking the identity of an individual attempting to gain access to a network. Methods of authentication include passwords, swipe cards and biometric scans.

Corrupt file: a file that has been damaged in some way and can no longer be processed properly.

Cyberattack: any kind of malicious attack on a computer system or network with intent to cause damage or harm.

Key terms

File repair: the process of extracting as much reusable data from a corrupted file as possible.

Backup: the process of creating backup copies of data and storing them in a separate location to the live working environment – possibly in the cloud – to protect against data loss and damage.

Disaster recovery plan: a detailed plan for responding to a disaster. It specifies roles and responsibilities, procedures for backing up and recovering data files and arrangements for relocating staff to an alternative site or enabling them to work from home.

It scans the damaged file, extracts as much data from it as possible and stores it in a new usable file.

Backup

It's always a good idea to make a **backup** of important data and store it on a different storage device.

This is the purpose of a backup tool. It can be configured to automatically backup data stored on a computer's hard drive to another location, such as a network drive or the cloud.

Should a file get lost or damaged, the most recent backup can be restored.

You can find out more about different backup strategies in Topic 5.

Computer Science in action: Backing up

In industry and business, backing up data usually forms an important part of a company's **disaster recovery planning**. This is a set of protocols and procedures for backing up data on a regular basis and for being able to recover such data quickly in the event of its loss. Data can be lost through malicious cyberattack, because of human error, or due to unforeseen events such as a power failure or flooding. Storing, protecting and retrieving data are an important part of managing most businesses, and there are thousands of companies that specialise in backing up data and recovering systems when they fail. Companies are spoiled for choice!.

Key term

Compression: changing the format of a data file so that the size of the file becomes smaller.

Data compression

No matter how much storage you have, somehow you always manage to fill up the space with files.

A data **compression** tool is used to compress files to reduce their file size, freeing up storage space and speeding up their transfer across networks.

Data compression is commonly used for audio and video files which can be very large.

You can find out more about lossless and lossy methods of compression in Topic 2.

Disk defragmentation

When a file is saved to a disk, it is split up into blocks. Ideally, these blocks are stored in sequential segments on the disk, so that the file can be retrieved quickly. This gets more and more difficult to achieve as files are added and deleted.

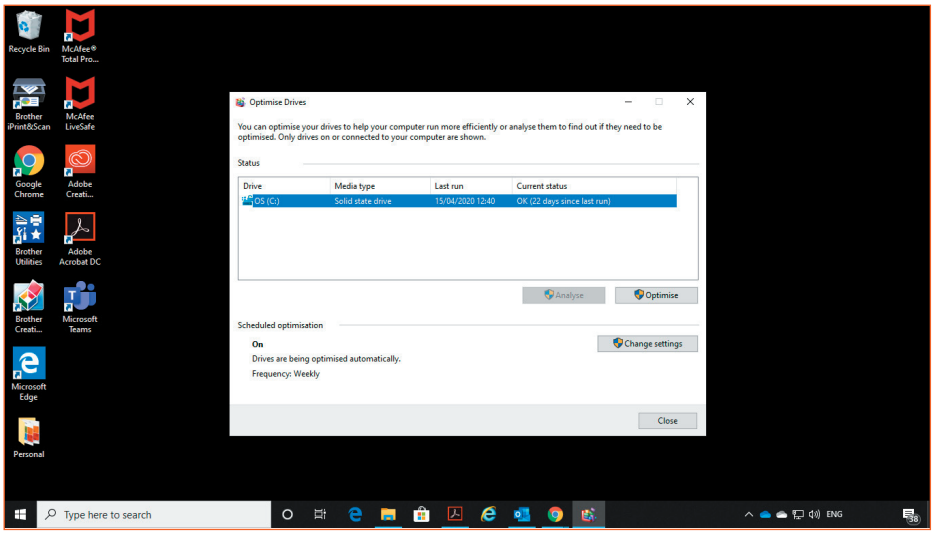


Figure 3.2.4 A screenshot showing the Microsoft® Windows® defragmenter

Over time, space on a disk becomes **fragmented** and the operating system has to store file blocks wherever free space is available. When a new file is written to the disk, it ends up being scattered all over the disk surface, slowing down a computer's performance.

A **disk defragmenter** is designed to address this problem. It rearranges all the files so that every file is stored in sequential sectors and free space is consolidated (grouped together).

Key terms

Fragmented: parts of a file are scattered across a disk rather than being stored sequentially.

Disk defragmenter: a tool that moves file blocks on a disk, so they are closer to each other, in order to speed up disk access and improve performance.

Activity

Here is a diagram of a hard disk on which five files (A, B, C, D and E) are stored. Each file is split up into several blocks. Redraw the diagram to show one possible arrangement of the blocks after a disk defragmenter has been run.

B3	D3	D1	C1	A4	E3	B4	D2	A2	E2	B2	C2	A1	D4	C3	E1	A3	B1

Anti-malware

Anti-malware tools are designed to eliminate malicious software (malware) from a computer.

You can find out more about how they work in Topic 5.

Key term

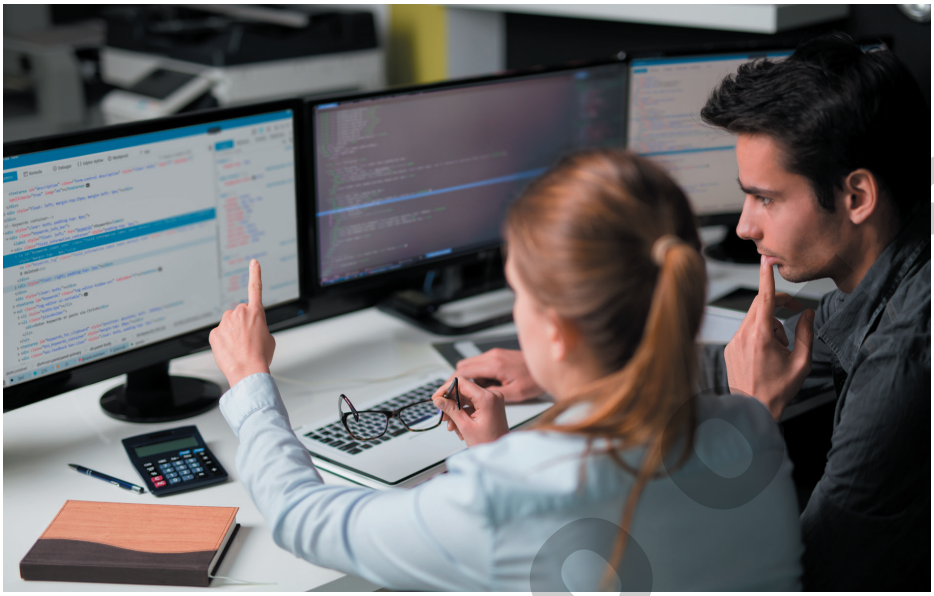
Anti-malware: utility software that detects and provides protection against attacks from malicious software.

Practical Activity

Create more flashcards to add to your collection for this topic.

Robust software

When working on a new software application, programmers always have to be mindful of what could go wrong when the program is executed. Some eventualities are easier to predict than others.



Programmers discussing the code on their screen.

- A user who doesn't know what they are doing may select an incorrect option or supply an inappropriate input.
- Data required by the program may somehow become corrupted.
- A hardware fault may occur.
- A peripheral may not be available when required.

While it's impossible to predict and allow for every eventuality, the goal should be to produce **robust software** that is capable of handling the unexpected without crashing, generating incorrect output or revealing sensitive data.

A program may be robust and do the job for which it was developed, but that doesn't mean that it doesn't have any weak spots.

A **code vulnerability** is a flaw in a program that compromises security. It can be exploited by an attacker to gain access to a computer system, enabling them to steal or corrupt data, introduce malware or take over control of the computer for some purpose of their own.

Just like any other software, programming languages can have **language-specific vulnerabilities**. For example, programs written in C and C++ languages can access memory directly which – though it can be useful – also makes them a target for **hackers** who can use this 'back door' to introduce malicious code into a computer system.

Programmers need to be aware of these language-specific code vulnerabilities and avoid using them.

Other languages may not have this particular vulnerability, but they will have others. When selecting which language to use for a particular purpose, a programmer should take known vulnerabilities into account.

Key terms

Robust software: code that can handle unexpected actions without crashing or producing incorrect output.

Code vulnerability: a flaw in a program that compromises security, enabling an attacker to gain access to a computer system and the data within it.

Language-specific vulnerabilities: a known flaw in a programming language that can pose a threat to security.

Hacker: someone who seeks out and exploits security vulnerabilities in code in order to gain unauthorised access to a computer system or network.

Using ready-made library modules rather than writing all the code from scratch may speed up development but can sometimes have undesirable consequences. They may have security flaws and should never be used without first checking that their code does not contain any vulnerabilities.

Bad programming practices

There may be many different programmers working on a big software project. Some may have developed bad habits. Pressure to meet deadlines may force even a good programmer to take shortcuts.

Bad programming habits that can cause a program to fail or be insecure include:

- not having a properly thought-through design – it's easier to write well-structured, robust code when it has been properly planned
- not sticking to agreed coding practices and standards – standards help ensure that every programmer writes code in a similar style, which makes it easier to integrate and more readable
- making do with a temporary fix rather than spending time finding a proper permanent solution
- writing unstructured code instead of separating it into subprograms – properly structured code is easier to read and debug
- insufficient testing – without adequate testing, code is likely to be unreliable and error-prone.

Issues like these can be minimised by having regular code reviews. Code reviews are designed to spot instances of poor programming practice, find any vulnerabilities in the code and check its efficiency.

Code reviews

There are two main types of code review.

- **Review by another programmer** – this is probably someone who is more senior and has more experience. The reviewer will look at the code produced, checking to see if any bad programming practices have been used or if any code vulnerabilities are present. This is fairly labour intensive and time consuming.
- **An automated review** – here, a specialist piece of software is used to examine the code. The software will highlight potential issues such as common vulnerabilities in the programming language or obvious bad programming practice. It probably won't identify every issue but it will flag up common and well-known problems.

Activity

1 How does modular testing help keep code secure?

2 'Cyclomatic complexity' is a software metric used to measure the complexity of code. Find out how software metrics help improve the quality of code.

Sometimes a combination of both types will be used. If issues are found, then the original programmer will be asked to improve the code and another code review will take place.

Audit trail

When a software application is being developed, it is important to be able to track who made what changes and when. This is the purpose of an audit trail. An audit trail reduces errors and improves accountability.

Should a security issue arise, it can be tracked back to the version of the code when it first appeared.

Computer Science in action: Audit trails – everyone's using them

Github (https://github.com/) enables teams of developers to work on large and complex software projects. As of January 2020, the version control system had over 40 million users. The system allows different programmers to check out code, work on the code, then merge the changes back together. It can track who changed what and when they did it. It's very useful for finding out who changed the name of a variable to 'pickles'!

Exam-style question

1 The operating system (OS) manages the computer's resources.

a) Explain how the OS uses scheduling to share use of the CPU between processes. (2 marks)

b) Describe how the OS organises files on a hard drive. (3 marks)

2 A company recycles digital devices. It stores recycling statistics on a server. Users do not all have the same access to files stored on the server. Explain the type of access a student on work experience at the company should be given to the statistics file. (2 marks)

3 Utility tools helps to maintain and optimise computer performance.

a) Explain how a defragmenter tool can help improve the performance of a computer. (2 marks)

b) Give two reasons for using a data compression tool. (2 marks)

c) Explain why it is important to keep anti-malware up to date. (2 marks)

4 Explain the purpose of a code review. (2 marks)

5 Making do with a temporary fix rather than taking time to find a robust solution is bad programming practice. Explain why this is the case. (2 marks)

Summary

- An operating system performs a number of important functions that enable the hardware of a computer system to communicate with the software. It also provides a user interface and manages user access.
- Utility software comprises a set of tools that enable a user to repair, backup and compress files, defragment a disk, and protect their devices against malware.
- Tools such as audit trails and code reviews help programmers to produce robust code, free from security vulnerabilities.