Cambridge Assessment International Education

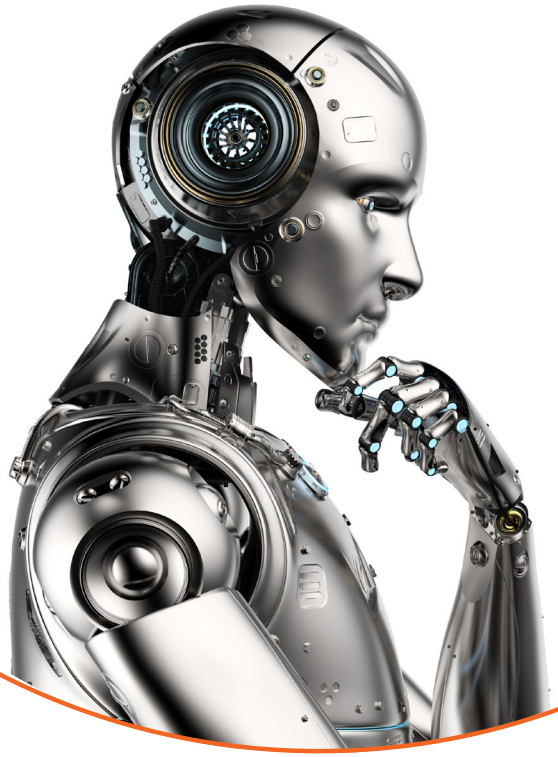
Syllabus

Cambridge IGCSE™ (9–1)

Computer Science 0984

Use this syllabus for exams in 2026, 2027 and 2028.
Exams are available in the June and November series.

This syllabus is **not** available in all administrative zones.
Please check the syllabus page at **www.cambridgeinternational.org/0984** to see if this syllabus is available in your administrative zone.



Version 6

For the purposes of screen readers, any mention in this document of Cambridge IGCSE refers to Cambridge International General Certificate of Secondary Education.

Cambridge Pathway

Why choose Cambridge International?

Cambridge International prepares school students for life, helping them develop an informed curiosity and a lasting passion for learning. We are part of the University of Cambridge.

Our Cambridge Pathway gives students a clear path for educational success from age 5 to 19. Schools can shape the curriculum around how they want students to learn – with a wide range of subjects and flexible ways to offer them. It helps students discover new abilities and a wider world, and gives them the skills they need for life, so they can achieve at school, university and work.

Our programmes and qualifications set the global standard for international education. They are created by subject experts, are rooted in academic rigour and reflect the latest educational research. They provide a strong platform for learners to progress from one stage to the next, and are well supported by teaching and learning resources. Learn more about our research at **www.cambridgeassessment.org.uk/our-research/**

We believe education works best when curriculum, teaching, learning and assessment are closely aligned. Our programmes develop deep knowledge, conceptual understanding and higher-order thinking skills, to prepare students for their future. Together with schools, we develop Cambridge learners who are confident, responsible, reflective, innovative and engaged – equipped for success in the modern world.

Every year, nearly a million Cambridge students from 10 000 schools in 160 countries prepare for their future with the Cambridge Pathway.

School feedback: 'We think the Cambridge curriculum is superb preparation for university.'

Feedback from: Christoph Guttentag, Dean of Undergraduate Admissions, Duke University, USA

Quality management

Cambridge International is committed to providing exceptional quality. In line with this commitment, our quality management system for the provision of international education programmes and qualifications for students aged 5 to 19 is independently certified as meeting the internationally recognised standard, ISO 9001:2015. Learn more at **www.cambridgeinternational.org/about-us/our-standards/**

© Cambridge University Press & Assessment September 2023

Cambridge Assessment International Education is part of Cambridge University Press & Assessment. Cambridge University Press & Assessment is a department of the University of Cambridge.

Cambridge University Press & Assessment retains the copyright on all its publications. Registered centres are permitted to copy material from this booklet for their own internal use. However, we cannot give permission to centres to photocopy any material that is acknowledged to a third party even for internal use within a centre.

Contents

Why choose Cambridge International? **2**

1 Why choose this syllabus? **4**

2 Syllabus overview **7**

Aims 7

Content overview 8

Assessment overview 9

Assessment objectives 10

3 Subject content **11**

Computer systems 11

Algorithms, programming and logic 25

4 Details of the assessment **32**

Paper 1 – Computer Systems 32

Paper 2 – Algorithms, Programming and Logic 32

Mathematical requirements 33

Flowchart symbols 33

Logic gate symbols 34

Pseudocode 35

Command words 49

5 What else you need to know **50**

Before you start 50

Making entries 51

Accessibility and equality 51

After the exam 52

How students and teachers can use the grades 53

Changes to this syllabus for 2026, 2027 and 2028 54

Important: Changes to this syllabus

For information about changes to this syllabus for 2026, 2027 and 2028, go to page 54.

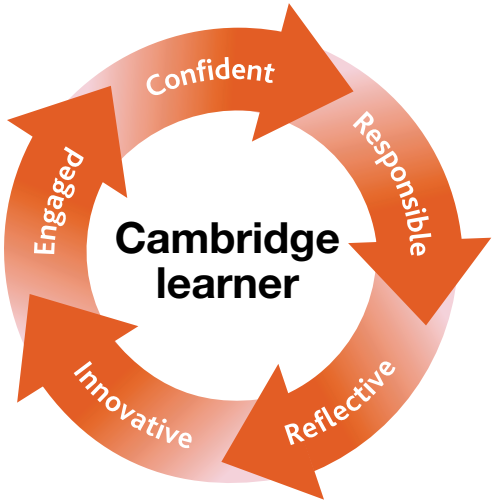
1 Why choose this syllabus?

Key benefits

Cambridge IGCSE is the world's most popular international qualification for 14 to 16 year olds, although it can be taken by students of other ages. It is tried, tested and trusted.

Students can choose from 70 subjects in any combination – it is taught by over 5000 schools in 150 countries.

Our programmes promote a thorough knowledge and understanding of a subject and help to develop the skills learners need for their next steps in education or employment.



Cambridge IGCSE (9–1) Computer Science provides an ideal foundation in computer science. Learners gain confidence in computational thinking and programming, an appreciation of automated and emerging technologies and the benefits of their use. They develop an understanding of the main principles of problem-solving by creating computer-based solutions using algorithms and a high-level programming language. Learners also develop a range of technical skills including the ability to test effectively and to evaluate solutions.

Our approach in Cambridge IGCSE (9–1) Computer Science encourages learners to be:

confident, interested in learning about computer science and using technical language to communicate their knowledge and understanding

responsible, working systematically, safely and securely when using technology

reflective, learning from their experiences when creating programs and using technology; understanding how technology impacts society

innovative, solving unfamiliar problems and designing computer programs creatively and independently

engaged, keen to develop computer science skills and further their understanding of developments in the use of technology.

School feedback: 'The strength of Cambridge IGCSE qualifications is internationally recognised and has provided an international pathway for our students to continue their studies around the world.'

Feedback from: Gary Tan, Head of Schools and CEO, Raffles Group of Schools, Indonesia

International recognition and acceptance

Our expertise in curriculum, teaching and learning, and assessment is the basis for the recognition of our programmes and qualifications around the world. The combination of knowledge and skills in Cambridge IGCSE (9–1) Computer Science gives learners a solid foundation for further study. Candidates who achieve grades 9 to 4 are well prepared to follow a wide range of courses including Cambridge International AS & A Level Computer Science.

Cambridge IGCSEs are accepted and valued by leading universities and employers around the world as evidence of academic achievement. Cambridge students can be confident that their qualifications will be understood and valued throughout their education and career, in their home country and internationally. Many universities require a combination of Cambridge International AS & A Levels and Cambridge IGCSEs or equivalent to meet their entry requirements.

UK ENIC, the national agency in the UK for the recognition and comparison of international qualifications and skills, has carried out an independent benchmarking study of Cambridge IGCSE and found it to be comparable to the standard of the GCSE in the UK. This means students can be confident that their Cambridge IGCSE qualifications are accepted as equivalent to UK GCSEs by leading universities worldwide.

Learn more at **www.cambridgeinternational.org/recognition**

School feedback: ‘Cambridge IGCSE is one of the most sought-after and recognised qualifications in the world. It is very popular in Egypt because it provides the perfect preparation for success at advanced level programmes.’

Feedback from: Managing Director of British School of Egypt BSE

Supporting teachers

We believe education is most effective when curriculum, teaching and learning, and assessment are closely aligned. We provide a wide range of resources, detailed guidance, innovative training and targeted professional development so that you can give your students the best possible preparation for Cambridge IGCSE. To find out which resources are available for each syllabus go to our School Support Hub.

The School Support Hub is our secure online site for Cambridge teachers where you can find the resources you need to deliver our programmes. You can also keep up to date with your subject and the global Cambridge community through our online discussion forums.

Find out more at **www.cambridgeinternational.org/support**

Support for Cambridge IGCSE			
Planning and preparation • Syllabuses • Schemes of work • Specimen Question Papers and Mark Schemes • Teacher guides	**Teaching and assessment** • Endorsed resources • Online forums	**Learning and revision** • Example candidate responses • Past papers and mark schemes • Specimen paper answers	**Results** • Candidate Results Service • Principal examiner reports for teachers • Results Analysis

Sign up for email notifications about changes to syllabuses, including new and revised products and services, at **www.cambridgeinternational.org/syllabusupdates**

Syllabuses and specimen materials represent the final authority on the content and structure of all of our assessments.

Professional development

Find the next step on your professional development journey.

- Introductory Professional Development – An introduction to Cambridge programmes and qualifications.
- Extension Professional Development – Develop your understanding of Cambridge programmes and qualifications to build confidence in your delivery.
- Enrichment Professional Development – Transform your approach to teaching with our Enrichment workshops.
- Cambridge Professional Development Qualifications (PDQs) – Practice-based programmes that transform professional learning for practising teachers. Available at Certificate and Diploma level.

Find out more at:
www.cambridgeinternational.org/support-and-training-for-schools/professional-development/

Supporting exams officers

We provide comprehensive support and guidance for all Cambridge exams officers.
Find out more at: **www.cambridgeinternational.org/eoguide**

2 Syllabus overview

Aims

The aims describe the purposes of a course based on this syllabus.

The aims are to enable students to develop:

- computational thinking skills
- an understanding of the main principles of solving problems using computers
- the skills necessary to solve computer-based problems using a high-level programming language
- an understanding of the component parts of computer systems and how they interrelate
- an understanding of the internet as a means of communication and its associated risks
- an understanding of the development and use of automated and emerging technologies.

Cambridge Assessment International Education is an education organisation and politically neutral. The contents of this syllabus, examination papers and associated materials do not endorse any political view. We endeavour to treat all aspects of the exam process neutrally.

Content overview

Candidates study the following topics:

Computer systems

1 Data representation
2 Data transmission
3 Hardware
4 Software
5 The internet and its uses
6 Automated and emerging technologies

Algorithms, programming and logic

7 Algorithm design and problem-solving
8 Programming
9 Databases
10 Boolean logic

Assessment overview

All candidates take two components. Candidates will be eligible for grades 9 to 1.

All candidates take:	**and:**
Paper 1 1 hour 45 minutes Computer Systems 50% 75 marks Short-answer and structured questions Questions will be based on Topics 1–6 of the subject content All questions are compulsory No calculators are permitted Externally assessed	**Paper 2** 1 hour 45 minutes Algorithms, Programming and Logic 50% 75 marks Short-answer and structured questions and a scenario-based question Questions will be based on Topics 7–10 of the subject content All questions are compulsory No calculators are permitted Externally assessed

Information on availability is in the **Before you start** section.

Assessment objectives

The assessment objectives (AOs) are:

AO1

Demonstrate knowledge and understanding of the principles and concepts of computer science.

AO2

Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems.

AO3

Provide solutions to problems by:

- evaluating computer systems
- making reasoned judgements
- presenting conclusions.

Weighting for assessment objectives

The approximate weightings allocated to each of the assessment objectives (AOs) are summarised below.

Assessment objectives as a percentage of the qualification

Assessment objective	Weighting in IGCSE %
AO1	40
AO2	40
AO3	20
Total	100

Assessment objectives as a percentage of each component

Assessment objective	Weighting in components %	
	Paper 1	Paper 2
AO1	60	20
AO2	20	60
AO3	20	20
Total	100	100

3 Subject content

This syllabus gives you the flexibility to design a course that will interest, challenge and engage your learners. Where appropriate you are responsible for selecting resources and examples to support your learners' study. These should be appropriate for the learners' age, cultural background and learning context as well as complying with your school policies and local legal requirements.

Computer Science is a practical subject and a range of practical exercises must be integral to the teaching of this qualification. It is important that learners develop their computational thinking skills by doing practical problem-solving and programming using appropriate resources. It is also expected that learners have the opportunity in class to write their own programs, as well as executing (running), testing and debugging them.

Any equipment and facilities should be adequate for learners to be able to satisfy the requirements of the syllabus. The hardware facilities needed will depend on the number of learners but must be sufficient for all learners to have enough time to practise their programming skills. Learners also need to have access to a system with direct-access file capability on backing store and hardcopy facilities.

Computer systems

1 Data representation

1.1 Number systems

Candidates should be able to:	Notes and guidance
1 Understand how and why computers use binary to represent all forms of data	• Any form of data needs to be converted to binary to be processed by a computer • Data is processed using logic gates and stored in registers
2 (a) Understand the denary, binary and hexadecimal number systems	• Denary is a base 10 system • Binary is a base 2 system • Hexadecimal is a base 16 system
(b) Convert between (i) positive denary and positive binary (ii) positive denary and positive hexadecimal (iii) positive hexadecimal and positive binary	• Values used will be integers only • Conversions in both directions, e.g. denary to binary or binary to denary • Maximum binary number length of 16-bit
3 Understand how and why hexadecimal is used as a beneficial method of data representation	• Areas within computer science that hexadecimal is used should be identified • Hexadecimal is easier for humans to understand than binary, as it is a shorter representation of binary
4 (a) Add two positive 8-bit binary integers	
(b) Understand the concept of overflow and why it occurs in binary addition	• An overflow error will occur if the value is greater than 255 in an 8-bit register • A computer or a device has a predefined limit that it can represent or store, for example 16-bit • An overflow error occurs when a value outside this limit should be returned

1.1 Number systems continued

Candidates should be able to:	Notes and guidance
5 Perform a logical binary shift on a positive 8-bit binary integer and understand the effect this has on the positive binary integer	• Perform logical left shifts • Perform logical right shifts • Perform multiple shifts • Bits shifted from the end of the register are lost and zeros are shifted in at the opposite end of the register • The positive binary integer is multiplied or divided according to the shift performed • The most significant bit(s) or least significant bit(s) are lost
6 Use the two's complement number system to represent positive and negative 8-bit binary integers	• Convert a positive binary or denary integer to a two's complement 8-bit integer and vice versa • Convert a negative binary or denary integer to a two's complement 8-bit integer and vice versa

1.2 Text, sound and images

Candidates should be able to:	Notes and guidance
1 Understand how and why a computer represents text and the use of character sets, including American standard code for information interchange (ASCII) and Unicode	• Text is converted to binary to be processed by a computer • Unicode allows for a greater range of characters and symbols than ASCII, including different languages and emojis • Unicode requires more bits per character than ASCII
2 Understand how and why a computer represents sound, including the effects of the sample rate and sample resolution	• A sound wave is sampled for sound to be converted to binary, which is processed by a computer • The sample rate is the number of samples taken in a second • The sample resolution is the number of bits per sample • The accuracy of the recording and the file size increases as the sample rate and resolution increase

1.2 Text, sound and images continued

Candidates should be able to:	Notes and guidance
3 Understand how and why a computer represents an image, including the effects of the resolution and colour depth	• An image is a series of pixels that are converted to binary, which is processed by a computer • The resolution is the number of pixels in the image • The colour depth is the number of bits used to represent each colour • The file size and quality of the image increase as the resolution and colour depth increase

1.3 Data storage and compression

Candidates should be able to:	Notes and guidance
1 Understand how data storage is measured	• Including: – bit – nibble – byte – kibibyte (KiB) – mebibyte (MiB) – gibibyte (GiB) – tebibyte (TiB) – pebibyte (PiB) – exbibyte (EiB) • The amount of the previous denomination present in the data storage size, e.g.: – 8 bits in a byte – 1024 mebibytes in a gibibyte
2 Calculate the file size of an image file and a sound file, using information given	• Answers must be given in the units specified in the question. Calculations must use the measurement of 1024 and not 1000 • Information given may include: – image resolution and colour depth – sound sample rate, resolution and length of track
3 Understand the purpose of and need for data compression	• Compression exists to reduce the size of the file • What the impact of this is, e.g.: – less bandwidth required – less storage space required – shorter transmission time

1.3 Data storage and compression continued

Candidates should be able to:	Notes and guidance
4 Understand how files are compressed using lossy and lossless compression methods	• Lossy compression reduces the file size by permanently removing data, e.g. reducing resolution or colour depth, reducing sample rate or resolution • Lossless compression reduces the file size without permanent loss of data, e.g. run length encoding (RLE)

2 Data transmission

2.1 Types and methods of data transmission

Candidates should be able to:	Notes and guidance
1 (a) Understand that data is broken down into packets to be transmitted	
(b) Describe the structure of a packet	• A packet of data contains a: – packet header – payload – trailer • The packet header includes the: – destination address – packet number – originator's address
(c) Describe the process of packet switching	• Data is broken down into packets • Each packet could take a different route • A router controls the route a packet takes • Packets may arrive out of order • Once the last packet has arrived, packets are reordered
2 (a) Describe how data is transmitted from one device to another using different methods of data transmission	• Including: – serial – parallel – simplex – half-duplex – full-duplex
(b) Explain the suitability of each method of data transmission, for a given scenario	• Including the advantages and disadvantages of each method
3 Understand the universal serial bus (USB) interface and explain how it is used to transmit data	• Including the benefits and drawbacks of the interface

2.2 Methods of error detection

Candidates should be able to:	Notes and guidance
1 Understand the need to check for errors after data transmission and how these errors can occur	• Errors can occur during data transmission due to interference, e.g. data loss, data gain and data change
2 Describe the processes involved in each of the following error detection methods for detecting errors in data after transmission: parity check (odd and even), checksum and echo check	• Including parity byte and parity block check
3 Describe how a check digit is used to detect errors in data entry and identify examples of when a check digit is used, including international standard book numbers (ISBN) and bar codes	
4 Describe how an automatic repeat query (ARQ) can be used to establish that data is received without error	• Including the use of: – positive/negative acknowledgements – timeout

2.3 Encryption

Candidates should be able to:	Notes and guidance
1 Understand the need for and purpose of encryption when transmitting data	
2 Understand how data is encrypted using symmetric and asymmetric encryption	• Asymmetric encryption includes the use of public and private keys

3 Hardware

3.1 Computer architecture

Candidates should be able to:	Notes and guidance
1 (a) Understand the role of the central processing unit (CPU) in a computer	• The CPU processes instructions and data that are input into the computer so that the result can be output
(b) Understand what is meant by a microprocessor	• A microprocessor is a type of integrated circuit on a single chip
2 (a) Understand the purpose of the components in a CPU, in a computer that has a Von Neumann architecture	• Including: – units: arithmetic logic unit (ALU) and control unit (CU) – registers: program counter (PC), memory address register (MAR), memory data register (MDR), current instruction register (CIR) and accumulator (ACC) – buses: address bus, data bus and control bus
(b) Describe the process of the fetch–decode–execute (FDE) cycle, including the role of each component in the process	• How instructions and data are fetched from random access memory (RAM) into the CPU, how they are processed using each component and how they are then executed • Storing data and addresses into specific registers • Using buses to transmit data, addresses and signals • Using units to fetch, decode and execute data and instructions
3 Understand what is meant by a core, cache and clock in a CPU and explain how they can affect the performance of a CPU	• The number of cores, size of the cache and speed of the clock can affect the performance of a CPU
4 Understand the purpose and use of an instruction set for a CPU	• An instruction set is a list of all the commands that can be processed by a CPU, and the commands are machine code
5 Describe the purpose and characteristics of an embedded system and identify devices in which they are commonly used	• An embedded system is used to perform a dedicated functions. For example in, domestic appliances, cars, security systems, lighting systems or vending machines. This is different to a general purpose computer that is used to perform many different functions. For example in, a personal computer (PC) or a laptop

3.2 Input and output devices

Candidates should be able to:	Notes and guidance
1 Understand what is meant by an input device and why it is required	• Including: – barcode scanner – digital camera – keyboard – microphone – optical mouse – QR code scanner – touch screen (resistive, capacitive and infra-red) – two-dimensional (2D) and three-dimensional (3D) scanners
2 Understand what is meant by an output device and why it is required	• Including: – actuator – digital light processing (DLP) projector – inkjet printer – laser printer – light emitting diode (LED) screen – liquid crystal display (LCD) projector – liquid crystal display (LCD) screen – speaker – 3D printer
3 (a) Understand what is meant by a sensor and the purposes of sensors	• Limited to: – acoustic – accelerometer – flow – gas – humidity – infra-red – level – light – magnetic field – moisture – pH – pressure – proximity – temperature
(b) Identify the type of data captured by each sensor and understand when each sensor would be used, including selecting the most suitable sensor for a given context	

3.3 Data storage

Candidates should be able to:	Notes and guidance
1 Understand what is meant by primary storage	• Primary storage is directly accessed by the CPU • Including the role of: – random access memory (RAM) – read only memory (ROM) • Including why a computer needs both RAM and ROM, and the difference between them
2 Understand what is meant by secondary storage	• Secondary storage is not directly accessed by the CPU and is necessary for more permanent storage of data
3 Describe the operation of magnetic, optical and solid-state (flash memory) storage and give examples of each	• Magnetic storage uses platters which are divided into tracks and sectors. Data is read and written using electromagnets. Including hard disk drive (HDD) • Optical storage uses lasers to create and read pits and lands. Including: CD, DVD and Blu-ray • Solid-state (flash memory) uses NAND or NOR technology. Transistors are used as control gates and floating gates. Including: solid-state drive (SSD), SD card and USB drive
4 Describe what is meant by virtual memory, how it is created and used and why it is necessary	• Pages of data are transferred between RAM and virtual memory when needed
5 Understand what is meant by cloud storage	• Cloud storage can be accessed remotely in comparison to storing data locally
6 Explain the advantages and disadvantages of storing data on the cloud in comparison to storing it locally	• Physical servers and storage are needed to store data in cloud storage

3.4 Network hardware

Candidates should be able to:	Notes and guidance
1 Understand that a computer needs a network interface card (NIC) to access a network	
2 Understand what is meant by, and the purpose of, a media access control (MAC) address, including its structure	• A network interface card is given a MAC address at the point of manufacture • MAC addresses are usually written as hexadecimal • MAC addresses are created using the manufacturer code and the serial code
3 (a) Understand what is meant by, and the purpose of, an internet protocol (IP) address (b) Understand that there are different types of IP address	• An IP address is allocated by the network and it can be static or dynamic • Including the characteristics of, and differences between, IPv4 and IPv6
4 Describe the role of a router in a network	• A router sends data to a specific destination on a network • A router can assign IP addresses • A router can connect a local network to the internet

4 Software

4.1 Types of software and interrupts

Candidates should be able to:	Notes and guidance
1 Describe the difference between system software and application software and provide examples of each	• System software provides the services that the computer requires, including operating system and utility software • Application software provides the services that the user requires
2 Describe the role and basic functions of an operating system	• Including: – managing files – handling interrupts – providing an interface – managing peripherals and drivers – managing memory – managing multitasking – providing a platform for running applications – providing system security – managing user accounts

4.1 Types of software and interrupts continued

Candidates should be able to:	Notes and guidance
3 Understand how hardware, firmware and an operating system are required to run applications software	• Applications are run on the operating system • The operating system is run on the firmware • The bootloader (firmware) is run on the hardware
4 Describe the role and operation of interrupts	• Including: – how an interrupt is generated – how it is handled using an interrupt service routine – what happens as a result of the interrupts • Software interrupts include division by zero and two processes trying to access the same memory location • Hardware interrupts include pressing a key on the keyboard and moving the mouse

4.2 Types of programming language, translators and integrated development environments (IDEs)

Candidates should be able to:	Notes and guidance
1 Explain what is meant by a high-level language and a low-level language, including the advantages and disadvantages of each	• Advantages and disadvantages include: – ease of reading and writing code, e.g. low-level is hard to read – ease of debugging code – machine independence – direct manipulation of hardware
2 Understand that assembly language is a form of low-level language that uses mnemonics, and that an assembler is needed to translate an assembly language program into machine code	
3 Describe the operation of a compiler and an interpreter, including how high-level language is translated by each and how errors are reported	• A compiler translates the whole code at once before executing it, producing an executable file • An interpreter translates and executes the code line-by-line • A compiler provides an error report for the whole code if errors are detected • An interpreter stops execution when an error is found

4.2 Types of programming language, translators and integrated development environments (IDEs) continued

Candidates should be able to:	Notes and guidance
4 Explain the advantages and disadvantages of a compiler and an interpreter	• Including an understanding that an interpreter is mostly used when developing a program and a compiler is used to translate the final program
5 Explain the role of an IDE in writing program code and the common functions IDEs provide	• Including: – code editors – run-time environment – translators – error diagnostics – auto-completion – auto-correction – prettyprint

5 The internet and its uses

5.1 The internet and the world wide web

Candidates should be able to:	Notes and guidance
1 Understand the difference between the internet and the world wide web	• The internet is the infrastructure • The world wide web is the collection of websites and web pages accessed using the internet
2 Understand what is meant by a uniform resource locator (URL)	• A URL is a text-based address for a web page; it can contain the protocol, the domain name and the web page/file name
3 Describe the purpose and operation of hypertext transfer protocol (HTTP) and hypertext transfer protocol secure (HTTPS)	
4 Explain the purpose and functions of a web browser	• The main purpose of a web browser is to render hypertext markup language (HTML) and display web pages • Functions include: – storing bookmarks and favourites – recording user history – allowing use of multiple tabs – storing cookies – providing navigation tools – providing an address bar

5.1 The internet and the world wide web continued

Candidates should be able to:	Notes and guidance
5 Describe how web pages are located, retrieved and displayed on a device when a user enters a URL	• Including the role of: – the web browser – IP addresses – the domain name server (DNS) – the web server – HTML
6 Explain what is meant by cookies and how they are used, including session cookies and persistent cookies	• Cookies are used for functions, including: – saving personal details – tracking user preferences – holding items in an online shopping cart – storing login details

5.2 Digital currency

Candidates should be able to:	Notes and guidance
1 Understand the concept of a digital currency and how digital currencies are used	• A digital currency is one that only exists electronically
2 Understand the process of blockchain and how it is used to track digital currency transactions	• Blockchain, in its basic form, is a digital ledger, that is a time-stamped series of records that cannot be altered

5.3 Cyber security

Candidates should be able to:	Notes and guidance
1 Describe the processes involved in, and the aim of carrying out, a range of cyber security threats	• Including: – brute-force attack – data interception – distributed denial of service (DDoS) attack – hacking – malware (virus, worm, Trojan horse, spyware, adware, ransomware) – pharming – phishing – social engineering

5.3 Cyber security continued

Candidates should be able to:	Notes and guidance
2 Explain how a range of solutions are used to help keep data safe from security threats	• Including: – access levels – anti-malware, including anti-virus and anti-spyware – authentication (username and password, biometrics, two-step verification) – automating software updates – checking the spelling and tone of communications – checking the URL attached to a link – firewalls – privacy settings – proxy-servers – secure socket layer (SSL) security protocol

6 Automated and emerging technologies

6.1 Automated systems

Candidates should be able to:	Notes and guidance
1 Describe how sensors, microprocessors and actuators can be used in collaboration to create automated systems	
2 Describe the advantages and disadvantages of an automated system used for a given scenario	• Including scenarios from: – industry – transport – agriculture – weather – gaming – lighting – science

6.2 Robotics

Candidates should be able to:	Notes and guidance
1 Understand what is meant by robotics	• Robotics is a branch of computer science that incorporates the design, construction and operation of robots • Examples include factory equipment, domestic robots and drones
2 Describe the characteristics of a robot	• Including: – a mechanical structure or framework – electrical components, such as sensors, microprocessors and actuators – programmable
3 Understand the roles that robots can perform and describe the advantages and disadvantages of their use	• Robots can be used in areas including: – industry – transport – agriculture – medicine – domestic settings – entertainment

6.3 Artificial intelligence

Candidates should be able to:	Notes and guidance
1 Understand what is meant by artificial intelligence (AI)	• AI is a branch of computer science dealing with the simulation of intelligent behaviours by computers
2 Describe the main characteristics of AI as the collection of data and the rules for using that data, the ability to reason, and it can include the ability to learn and adapt	
3 Explain the basic operation and components of AI systems to simulate intelligent behaviour	• Limited to: – expert systems – machine learning • Expert systems have a knowledge base, a rule base, an inference engine and an interface • Machine learning is when a program has the ability to automatically adapt its own processes and/or data

Algorithms, programming and logic

See section 4 for the:

- standard flowchart symbols that must be used by students when drawing flowcharts
- logic gate symbols that must be used by students when drawing logic circuits
- format in which pseudocode will appear in examinations.

In Paper 2, candidates are expected to write solutions to given problems using the methods and concepts listed in topics 7 to 10.

Where the solution to a problem involves coding, candidates are required to write solutions in pseudocode. Solutions written in programming code will **not** be awarded marks. The exception is the 15-mark scenario question.

In the 15-mark scenario question, candidates are required to write solutions to the given problem using pseudocode or their choice of one of the following programming languages: Python, Visual Basic or Java. Solutions written in other programming languages will **not** be awarded marks.

7 Algorithm design and problem-solving

Candidates should be able to:	Notes and guidance
1 Understand the program development life cycle, limited to: analysis, design, coding and testing	• Including identifying each stage and performing these tasks for each stage: – analysis: abstraction, decomposition of the problem, identification of the problem and requirements – design: decomposition, structure diagrams, flowcharts, pseudocode – coding: writing program code and iterative testing – testing: testing program code with the use of test data
2 (a) Understand that every computer system is made up of sub-systems, which are made up of further sub-systems	
(b) Understand how a problem can be decomposed into its component parts	• Including: – inputs – processes – outputs – storage
(c) Use different methods to design and construct a solution to a problem	• Including: – structure diagrams – flowcharts – pseudocode
3 Explain the purpose of a given algorithm	• Including: – stating the purpose of an algorithm – describing the processes involved in an algorithm

7 Algorithm design and problem-solving continued

Candidates should be able to:	Notes and guidance
4 Understand standard methods of solution	• Limited to: – linear search – bubble sort – totalling – counting – finding maximum, minimum and average values
5 (a) Understand the need for validation checks to be made on input data and the different types of validation check	• Including: – range check – length check – type check – presence check – format check – check digit
(b) Understand the need for verification checks to be made on input data and the different types of verification check	• Including: – visual check – double entry check
6 Suggest and apply suitable test data	• Limited to: – normal – abnormal – extreme – boundary • Extreme data is the largest/smallest acceptable value • Boundary data is the largest/smallest acceptable value and the corresponding smallest/largest rejected value
7 Complete a trace table to document a dry-run of an algorithm	• Including, at each step in an algorithm: – variables – outputs – user prompts
8 Identify errors in given algorithms and suggest ways of correcting these errors	
9 Write and amend algorithms for given problems or scenarios, using: pseudocode, program code and flowcharts	• Precision is required when writing algorithms, e.g. `x > y` is acceptable but `x is greater than y` is not acceptable • See section 4 for flowchart symbols • See section 4 for pseudocode

8 Programming

8.1 Programming concepts

Candidates should be able to:	Notes and guidance
1 Declare and use variables and constants	
2 Understand and use basic data types	• Including: – integer – real – char – string – Boolean
3 Understand and use input and output	
4 (a) Understand and use the concept of sequence	
(b) Understand and use the concept of selection	• Including: – `IF` statements – `CASE` statements
(c) Understand and use the concept of iteration	• Including: – count-controlled loops – pre-condition loops – post-condition loops
(d) Understand and use the concepts of totalling and counting	
(e) Understand and use the concept of string handling	• Including: – length – substring – upper – lower • The first character of the string can be position zero or one

8.1 Programming concepts continued

Candidates should be able to:	Notes and guidance
(f) Understand and use arithmetic, relational and logical operators	• Arithmetic, limited to: – `+` – `-` – `/` – `*` – `^` (raised to power of) – `MOD` – `DIV` • Relational, limited to: – `=` – `<` – `<=` – `>` – `>=` – `<>` (not equal to) • Logical, limited to: – `AND` – `OR` – `NOT`
5 Understand and use nested statements	• Including nested selection and iteration • Candidates will **not** be required to write more than three levels of nested statements
6 (a) Understand what is meant by procedures, functions and parameters (b) Define and use procedures and functions, with or without parameters (c) Understand and use local and global variables	• Procedures and functions may have up to three parameters
7 Understand and use library routines	• Including: – MOD – DIV – ROUND – RANDOM

8.1 Programming concepts continued

Candidates should be able to:	Notes and guidance
8 Understand how to create a maintainable program	• Including appropriate use of: – meaningful identifiers – the commenting feature provided by the programming language – procedures and functions – relevant and appropriate commenting of syntax • Use meaningful identifiers for: – variables – constants – arrays – procedures and functions

8.2 Arrays

Candidates should be able to:	Notes and guidance
1 Declare and use one-dimensional (1D) and two-dimensional (2D) arrays	
2 Understand the use of arrays	• Including the use of variables as indexes in arrays
3 Write values into, and read values from, an array using iteration	• The first index can be zero or one • Including nested iteration

8.3 File handling

Candidates should be able to:	Notes and guidance
1 Understand the purpose of storing data in a file to be used by a program	
2 Open, close and use a file for reading and writing	• Including: – read and write single items of data – read and write a line of text

9 Databases

Candidates should be able to:	Notes and guidance
1 Define a single-table database from given data storage requirements	• Including: – fields – records – validation
2 Suggest suitable basic data types	• Including: – text/alphanumeric – character – Boolean – integer – real – date/time
3 Understand the purpose of a primary key and identify a suitable primary key for a given database table	
4 Read, understand and complete structured query language (SQL) scripts to query data stored in a single database table	• Limited to: – `SELECT` – `FROM` – `WHERE` – `ORDER BY DESCENDING` – `ORDER BY ASCENDING` – `SUM` – `COUNT` – `AND` – `OR` • Identifying the output given by an SQL statement that will query the given contents of a database table

10 Boolean logic

Candidates should be able to:	Notes and guidance
1 Identify and use the standard symbols for logic gates	• See section 4 for logic gate symbols
2 Define and understand the functions of logic gates	• Including: – NOT – AND – OR – NAND – NOR – XOR (EOR) – the binary output produced from all the possible binary inputs • NOT is a single input gate • All other gates are limited to two inputs
3 (a) Use logic gates to create given logic circuits from a: (i) problem statement (ii) logic expression (iii) truth table	• Circuits must be drawn for the statement given, without simplification • Logic circuits will be limited to a maximum of three inputs and one output
(b) Complete a truth table from a: (i) problem statement (ii) logic expression (iii) logic circuit	• An example truth table with three inputs, for completion:

A	B	C	Output
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(c) Write a logic expression from a: (i) problem statement (ii) logic circuit (iii) truth table	

4 Details of the assessment

Paper 1 – Computer Systems

Written paper, 1 hour 45 minutes, 75 marks

This question paper consists of short-answer and structured questions set on Topics 1–6 of the subject content.

All questions are compulsory, and candidates answer on the question paper.

This paper assesses all assessment objectives, AO1, AO2 and AO3, and assesses the full grade range, A* to G.

This paper is externally assessed.

Calculators are **not** allowed in this examination.

Paper 2 – Algorithms, Programming and Logic

Written paper, 1 hour 45 minutes, 75 marks

This question paper consists of short-answer and structured questions set on Topics 7–10 of the subject content.

All questions are compulsory, and candidates answer on the question paper.

The questions require candidates to have practical programming experience.

Knowledge of programming language syntax is not examined; in all cases the logic is more important than the syntax.

This paper assesses all assessment objectives, AO1, AO2 and AO3, and assesses the full grade range, A* to G.

This paper is externally assessed.

Calculators are **not** allowed in this examination.

Scenario question

The final question in Paper 2 is a 15-mark unseen scenario question using the methods and concepts listed in Topics 7-10 of the subject content.

Candidates will be required to write an algorithm using pseudocode or program code for the context provided.

It is expected that candidates should spend 30 minutes answering this question.

Teachers are advised to familiarise themselves with the updated Paper 2 specimen paper and mark scheme for first assessment 2023, which provide an example of the scenario question, how it will be marked, and include an indicative 15-mark response.

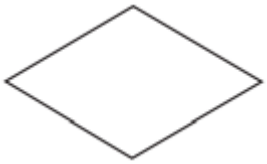
Mathematical requirements

Calculators are **not** permitted in IGCSE Computer Science examinations.

Candidates should be able to:

- add, subtract, multiply and divide
- use averages, random numbers, decimals, fractions, percentages and ratios
- use both positive and negative integers, and real numbers
- use arithmetic and logical operators
- use different number systems, including binary, denary and hexadecimal
- use methods of counting, totalling and rounding.

Flowchart symbols

	Flow line	An arrow represents control passing between the connected shapes.
	Process	This shape represents something being performed or done.
	Subroutine	This shape represents a subroutine call that will relate to a separate, non-linked flowchart.
	Input/Output	This shape represents the input or output of something into or out of the flowchart.
	**Decision**	This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes.
	Terminator	This shape represents the ‘Start’ and ‘Stop’ of the process.

Logic gate symbols

	NOT
	AND
	OR
	NAND
	NOR
	XOR (EOR)

Pseudocode

The following information sets out how pseudocode will appear within the examinations of this syllabus. The numbers and letters that appear at the end of a sub-heading provide a cross reference to the relevant section of the subject content.

General style

Font style and size

Pseudocode is presented in `Courier New`. The size of the font will be consistent throughout.

Indentation

Lines are indented by four spaces to indicate that they are contained within a statement in a previous line. Where it is not possible to fit a statement on one line, any continuation lines are indented by two spaces from the margin. In cases where line numbering is used, this indentation may be omitted. Every effort will be made to make sure that code statements are not longer than a line of code, unless this is necessary.

Note that the `THEN` and `ELSE` clauses of an `IF` statement are indented by only two spaces. Cases in `CASE` statements are also indented by only two spaces.

Case

Keywords are in upper case, e.g. `IF`, `REPEAT`, `PROCEDURE`.

Identifiers are in mixed case with upper case letters indicating the beginning of new words, e.g. `NumberOfPlayers`.

Meta-variables – (symbols in the pseudocode that should be substituted by other symbols) are enclosed in angled brackets `< >`.

Example – meta-variables

```
REPEAT
     <Statements>
UNTIL <Condition>
```

Lines and line numbering

Each line representing a statement is numbered. However, when a statement runs over one line of text, the continuation lines are not numbered.

Comments

Comments are preceded by two forward slashes: `//`. The comment continues until the end of the line. For multi-line comments, each line is preceded by `//`.

Normally the comment is on a separate line before, and at the same level of indentation as, the code it refers to. Occasionally, however, a short comment that refers to a single line may be at the end of the line to which it refers.

Example – comments

```
// This procedure swaps
// values of X and Y
PROCEDURE SWAP(X : INTEGER, Y : INTEGER)
    Temp ← X    // temporarily store X
    X ← Y
    Y ← Temp
ENDPROCEDURE
```

Variables, constants and data types

Basic data types (8.1.2)

The following keywords are used to designate basic data types:

- `INTEGER` a whole number
- `REAL` a number capable of containing a fractional part
- `CHAR` a single character
- `STRING` a sequence of zero or more characters
- `BOOLEAN` the logical values `TRUE` and `FALSE`

Literals

Literals of the above data types are written as follows:

- Integer written as normal in the denary system, e.g. `5`, `-3`
- Real always written with at least one digit on either side of the decimal point, zeros being added if necessary, e.g. `4.7`, `0.3`, `-4.0`, `0.0`
- Char a single character delimited by single quotes, e.g. `'x'`, `'c'`, `'@'`
- String delimited by double quotes. A string may contain no characters (i.e. the empty string), e.g. `"This is a string"`, `""`
- Boolean `TRUE`, `FALSE`

Identifiers

Identifiers (the names given to variables, constants, procedures and functions) are in mixed case using Pascal case, e.g. `FirstName`. They can only contain letters (`A–Z`, `a–z`) and digits (`0–9`). They must start with a capital letter and not a digit. Accented letters and other characters, including the underscore, should not be used.

As in programming, it is good practice to use identifier names that describe the variable, procedure or function to which they refer. Single letters may be used where these are conventional (such as `i` and `j` when dealing with array indices, or `X` and `Y` when dealing with coordinates) as these are made clear by the convention.

Keywords should never be used as identifier names.

Identifiers should be considered case insensitive, for example, `Countdown` and `CountDown` should not be used as separate variables.

Variable declarations (8.1.1)

Declarations are made as follows:

```
DECLARE <identifier> : <data type>
```

Example – variable declarations

```
DECLARE Counter : INTEGER
DECLARE TotalToPay : REAL
DECLARE GameOver : BOOLEAN
```

Constants (8.1.1)

It is good practice to use constants if this makes the pseudocode more readable, and easier to update if the value of the constant changes.

Constants are declared by stating the identifier and the literal value in the following format:

```
CONSTANT <identifier> ← <value>
```

Example – `CONSTANT` declarations

```
CONSTANT HourlyRate ← 6.50
CONSTANT DefaultText ← "N/A"
```

Only literals can be used as the value of a constant. A variable, another constant or an expression must never be used.

Assignments

The assignment operator is ←

Assignments should be made in the following format:

```
<identifier> ← <value>
```

The identifier must refer to a variable (this can be an individual element in a data structure, such as an array or an abstract data type). The value may be any expression that evaluates to a value of the same data type as the variable.

Example – assignments

```
Counter ← 0
Counter ← Counter + 1
TotalToPay ← NumberOfHours * HourlyRate
```

Arrays

Declaring arrays (8.2.1)

Arrays are fixed-length structures of elements of identical data type, accessible by consecutive index numbers. It is good practice to explicitly state what the lower bound of the array (i.e. the index of the first element) is because this defaults to either 0 or 1 in different systems. Generally, a lower bound of 1 will be used.

Square brackets are used to indicate the array indices.

1D and 2D arrays are declared as follows (where `l`, `l1`, `l2` are lower bounds and `u`, `u1`, `u2` are upper bounds):

```
DECLARE <identifier> : ARRAY[<l>:<u>] OF <data type>
DECLARE <identifier> : ARRAY[<l1>:<u1>, <l2>:<u2>] OF <data type>
```

Example – array declaration

```
DECLARE StudentNames : ARRAY[1:30] OF STRING
DECLARE NoughtsAndCrosses : ARRAY[1:3, 1:3] OF CHAR
```

Using arrays (8.2.1)

In the main pseudocode statements, only one index value is used for each dimension in the square brackets.

Example – using arrays

```
StudentNames[1] ← "Ali"
NoughtsAndCrosses[2,3] ← 'X'
StudentNames[n+1] ← StudentNames[n]
```

An appropriate loop structure is used to assign the elements individually.

Example – assigning a group of array elements

```
FOR Index ← 1 TO 30
    StudentNames[Index] ← ""
NEXT Index
```

Common operations

Input and output (8.1.3)

Values are input using the `INPUT` command as follows:

```
INPUT <identifier>
```

The identifier should be a variable (that may be an individual element of a data structure, such as an array).

Values are output using the `OUTPUT` command as follows:

```
OUTPUT <value(s)>
```

Several values, separated by commas, can be output using the same command.

Examples – `INPUT` and `OUTPUT` statements

```
INPUT Answer
OUTPUT Score
OUTPUT "You have ", Lives, " lives left"
```

Arithmetic operations (8.1.4 (f))

Standard arithmetic operator symbols are used:

+	addition
–	subtraction
*	multiplication
/	division
^	raised to the power of

Examples – arithmetic operations

```
Answer ← Score * 100 / MaxMark
Answer ← Pi * Radius ^ 2
```

The integer division operators `MOD` and `DIV` can also be used.

`DIV(<identifier1>, <identifier2>)`
Returns the quotient of `identifier1` divided by `identifier2` with the fractional part discarded.

`MOD(<identifier1>, <identifier2>)`
Returns the remainder of `identifier1` divided by `identifier2`

The identifiers are of data type integer.

Examples – MOD and DIV

```
DIV(10, 3) returns 3
MOD(10, 3) returns 1
```

Multiplication and division have higher precedence over addition and subtraction (this is the normal mathematical convention). However, it is good practice to make the order of operations in complex expressions explicit by using parentheses.

Relational operations (8.1.4 (f))

The following symbols are used for relational operations:

=	equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
<>	not equal to

The result of these operations is always of data type `BOOLEAN`.

In complex expressions, it is advisable to use parentheses to make the order of operations explicit.

Logical operations (8.1.4 (f))

The only logical operators used are `AND`, `OR` and `NOT`. The operands and results of these operations are always of data type `BOOLEAN`.

In complex expressions, it is advisable to use parentheses to make the order of operations explicit.

Examples – logical operations

```
IF Answer < 0 OR Answer > 100
  THEN
    Correct ← FALSE
  ELSE
    Correct ← TRUE
ENDIF
```

String operations (8.1.4 (e))

`LENGTH(<identifier>)`

Returns the integer value representing the length of string. The identifier should be of data type string.

`LCASE(<identifier>)`

Returns the string/character with all characters in lower case. The identifier should be of data type string or char.

`UCASE(<identifier>)`

Returns the string/character with all characters in upper case. The identifier should be of data type string or char.

`SUBSTRING(<identifier>, <start>, <length>)`

Returns a string of length `length` starting at position `start`. The identifier should be of data type string, length and start should be positive, and data type integer.

Generally, a start position of 1 is the first character in the string.

Example – string operations

`LENGTH("Happy Days")` will return `10`

`LCASE('W')` will return `'w'`

`UCASE("Happy")` will return `"HAPPY"`

`SUBSTRING("Happy Days", 1, 5)` will return `"Happy"`

Other library routines (8.1.7)

`ROUND(<identifier> or <expression>, <places>)`

Returns the value of the identifier or the result of an expression rounded to `places` number of decimal places.

The identifier should be of data type real; `places` should be data type integer.

`RANDOM()`

Returns a random number between 0 and 1 inclusive.

Example – `ROUND` and `RANDOM`

```
Value ← ROUND(RANDOM() * 6, 0) // returns a whole number between 0 and 6
```

Selection

`IF` statements (8.1.4 (b) and 8.1.5)

`IF` statements may or may not have an `ELSE` clause.

`IF` statements without an `ELSE` clause are written as follows:

```
IF <condition>
  THEN
    <statements>
ENDIF
```

`IF` statements with an `ELSE` clause are written as follows:

```
IF <condition>
  THEN
    <statements>
  ELSE
    <statements>
ENDIF
```

Note that the `THEN` and `ELSE` clauses are only indented by two spaces. (They are, in a sense, a continuation of the `IF` statement rather than separate statements.)

When `IF` statements are nested, the nesting should continue the indentation of two spaces.

Example – nested `IF` statements

```
IF ChallengerScore > ChampionScore
  THEN
    IF ChallengerScore > HighestScore
      THEN
        OUTPUT ChallengerName, " is champion and highest scorer"
      ELSE
        OUTPUT Player1Name, " is the new champion"
    ENDIF
  ELSE
    OUTPUT ChampionName, " is still the champion"
    IF ChampionScore > HighestScore
      THEN
        OUTPUT ChampionName, " is also the highest scorer"
    ENDIF
ENDIF
```

CASE statements (8.1.4 (b))

CASE statements allow one out of several branches of code to be executed, depending on the value of a variable.

CASE statements are written as follows:

```
CASE OF <identifier>
  <value 1> : <statement>
  <value 2> : <statement>
  ...
ENDCASE
```

An OTHERWISE clause can be the last case:

```
CASE OF <identifier>
  <value 1> : <statement>
  <value 2> : <statement>
  ...
  OTHERWISE <statement>
ENDCASE
```

It is best practice to keep the branches to single statements as this makes the pseudocode more readable. Similarly, single values should be used for each case. If the cases are more complex, the use of an IF statement, rather than a CASE statement, should be considered.

Each case clause is indented by two spaces. They can be considered as continuations of the CASE statement rather than new statements.

Note that the case clauses are tested in sequence. When a case that applies is found, its statement is executed, and the CASE statement is complete. Control is passed to the statement after the ENDCASE. Any remaining cases are not tested.

If present, an OTHERWISE clause must be the last case. Its statement will be executed if none of the preceding cases apply.

Example – formatted CASE statement

```
INPUT Move
CASE OF Move
  'W' : Position ← Position - 10
  'E' : Position ← Position + 10
  'A' : Position ← Position - 1
  'D' : Position ← Position + 1
  OTHERWISE OUTPUT "Beep"
ENDCASE
```

Iteration

Count-controlled (FOR) loops (8.1.4 (c))

Count-controlled loops are written as follows:

```
FOR <identifier> ← <value1> TO <value2>
    <statements>
NEXT <identifier>
```

The identifier must be a variable of data type `INTEGER`, and the values should be expressions that evaluate to integers.

The variable is assigned each of the integer values from `value1` to `value2` inclusive, running the statements inside the `FOR` loop after each assignment. If `value1 = value2`, the statements will be executed once, and if `value1 > value2`, the statements will not be executed.

An increment can be specified as follows:

```
FOR <identifier> ← <value1> TO <value2> STEP <increment>
    <statements>
NEXT <identifier>
```

The increment must be an expression that evaluates to an integer. In this case the `identifier` will be assigned the values from `value1` in successive increments of `increment` until it reaches `value2`. If it goes past `value2`, the loop terminates. The `increment` can be negative.

Example – nested `FOR` loops

```
Total ← 0
FOR Row ← 1 TO MaxRow
    RowTotal ← 0
    FOR Column ← 1 TO 10
        RowTotal ← RowTotal + Amount[Row, Column]
    NEXT Column
    OUTPUT "Total for Row ", Row, " is ", RowTotal
    Total ← Total + RowTotal
NEXT Row
OUTPUT "The grand total is ", Total
```

Post-condition (`REPEAT`) loops (8.1.4 (c))

Post-condition loops are written as follows:

```
REPEAT
    <statements>
UNTIL <condition>
```

The condition must be an expression that evaluates to a Boolean. The statements in the loop will be executed at least once. The condition is tested after the statements are executed, and if it evaluates to `TRUE`, the loop terminates, otherwise the statements are executed again.

Example – `REPEAT UNTIL` statement

```
REPEAT
    OUTPUT "Please enter the password"
    INPUT Password
UNTIL Password = "Secret"
```

Pre-condition (`WHILE`) loops (8.1.4 (c))

Pre-condition loops are written as follows:

```
WHILE <condition> DO
    <statements>
ENDWHILE
```

The condition must be an expression that evaluates to a Boolean. The condition is tested before the statements, and the statements will only be executed if the condition evaluates to `TRUE`. After the statements have been executed, the condition is tested again. The loop terminates when the condition evaluates to `FALSE`.

The statements will not be executed if, on the first test, the condition evaluates to `FALSE`.

Example – `WHILE` loop

```
WHILE Number > 9 DO
    Number ← Number - 9
ENDWHILE
```

Procedures and functions

Procedures and functions are defined at the start of the code.

Defining and calling procedures (8.1.6 (b))

A procedure with no parameters is defined as follows:

```
PROCEDURE <identifier>
    <statements>
ENDPROCEDURE
```

A procedure with parameters is defined as follows:

```
PROCEDURE <identifier>(<param1>:<datatype>, <param2>:<datatype>...)
    <statements>
ENDPROCEDURE
```

The `<identifier>` is the identifier used to call the procedure. Where used, `param1`, `param2`, etc. are identifiers for the parameters of the procedure. These will be used as variables in the statements of the procedure.

Procedures should be called as follows:

```
CALL <identifier>
```

```
CALL <identifier>(Value1,Value2...)
```

These calls are complete program statements.

When parameters are used, `Value1`, `Value2...` must be of the correct data type as in the definition of the procedure.

When the procedure is called, control is passed to the procedure. If there are any parameters, these are substituted by their values, and the statements in the procedure are executed. Control is then returned to the line that follows the procedure call.

Example – use of procedures with and without parameters

```
PROCEDURE DefaultLine
    CALL LINE(60)
ENDPROCEDURE

PROCEDURE Line(Size : INTEGER)
    DECLARE Length : INTEGER
    FOR Length ← 1 TO Size
        OUTPUT '-'
    NEXT Length
ENDPROCEDURE

IF MySize = Default
  THEN
    CALL DefaultLine
  ELSE
    CALL Line(MySize)
ENDIF
```

Defining and calling functions (8.1.6 (b))

Functions operate in a similar way to procedures, except that in addition they return a single value to the point at which they are called. Their definition includes the data type of the value returned.

A function with no parameters is defined as follows:

```
FUNCTION <identifier> RETURNS <data type>
   <statements>
ENDFUNCTION
```

A function with parameters is defined as follows:

```
FUNCTION <identifier>(<param1>:<datatype>, <param2>:<datatype>...) RETURNS <data type>
   <statements>
ENDFUNCTION
```

The keyword `RETURN` is used as one of the statements within the body of the function to specify the value to be returned. Normally, this will be the last statement in the function definition.

Because a function returns a value that is used when the function is called, function calls are not complete program statements. The keyword `CALL` should not be used when calling a function. Functions should only be called as part of an expression. When the `RETURN` statement is executed, the value returned replaces the function call in the expression and the expression is then evaluated.

Example – definition and use of a function

```
FUNCTION SumSquare(Number1:INTEGER, Number2:INTEGER) RETURNS INTEGER
    RETURN Number1 * Number1 + Number2 * Number2
ENDFUNCTION

OUTPUT "Sum of squares = ", SumSquare(10, 20)
```

File handling

Handling files (8.3.2)

It is good practice to explicitly open a file, stating the mode of operation, before reading from or writing to it. This is written as follows:

```
OPENFILE <File identifier> FOR <File mode>
```

The file identifier will be the name of the file with data type string. The following file modes are used:

- `READ` for data to be read from the file
- `WRITE` for data to be written to the file. A new file will be created and any existing data in the file will be lost.

A file should be opened in only one mode at a time.

Data is read from the file (after the file has been opened in `READ` mode) using the `READFILE` command as follows:

```
READFILE <File Identifier>, <Variable>
```

When the command is executed, the data item is read and assigned to the variable.

Data is written into the file after the file has been opened using the `WRITEFILE` command as follows:

```
WRITEFILE <File identifier>, <Variable>
```

When the command is executed, the data is written into the file. Files should be closed when they are no longer needed using the `CLOSEFILE` command as follows:

```
CLOSEFILE <File identifier>
```

Example – file handling operations

This example uses the operations together, to copy a line of text from `FileA.txt` to `FileB.txt`

```
DECLARE    LineOfText   : STRING
OPENFILE   "FileA.txt"   FOR READ
OPENFILE   "FileB.txt"   FOR WRITE
READFILE   "FileA.txt",  LineOfText
WRITEFILE  "FileB.txt",  LineOfText
CLOSEFILE  "FileA.txt"
CLOSEFILE  "FileB.txt"
```

Command words

Command words and their meanings help candidates know what is expected from them in the exams. The table below includes command words used in the assessment for this syllabus. The use of the command word will relate to the subject context.

Command word	What it means
Calculate	work out from given facts, figures or information
Compare	identify/comment on similarities and/or differences
Define	give precise meaning
Demonstrate	show how or give an example
Describe	state the points of a topic / give characteristics and main features
Evaluate	judge or calculate the quality, importance, amount, or value of something
Explain	set out purposes or reasons / make the relationships between things clear / say why and/or how and support with relevant evidence
Give	produce an answer from a given source or recall/memory
Identify	name/select/recognise
Outline	set out the main points
Show (that)	provide structured evidence that leads to a given result
State	express in clear terms
Suggest	apply knowledge and understanding to situations where there are a range of valid responses in order to make proposals / put forward considerations

5 What else you need to know

This section is an overview of other information you need to know about this syllabus. It will help to share the administrative information with your exams officer so they know when you will need their support. Find more information about our administrative processes at **www.cambridgeinternational.org/eoguide**

Before you start

Previous study

We recommend that learners starting this course should have studied a broad curriculum such as the Cambridge Lower Secondary programme or equivalent national educational framework.

Guided learning hours

We design Cambridge IGCSE syllabuses to require about 130 guided learning hours for each subject. This is for guidance only. The number of hours a learner needs to achieve the qualification may vary according to each school and the learners' previous experience of the subject.

Availability and timetables

All Cambridge schools are allocated to one of six administrative zones. Each zone has a specific timetable. Find your administrative zone at **www.cambridgeinternational.org/adminzone**. This syllabus is **not** available in all administrative zones. To find out about availability check the syllabus page at **www.cambridgeinternational.org/0984**

You can view the timetable for your administrative zone at **www.cambridgeinternational.org/timetables**

You can enter candidates in the June and November exam series.

Check you are using the syllabus for the year the candidate is taking the exam.

Private candidates can enter for this syllabus. For more information, please refer to the *Cambridge Guide to Making Entries*.

Combining with other syllabuses

Candidates can take this syllabus alongside other Cambridge International syllabuses in a single exam series. The only exceptions are:

- Cambridge IGCSE Computer Science (0478)
- Cambridge O Level Computer Science (2210)
- syllabuses with the same title at the same level.

Cambridge IGCSE, Cambridge IGCSE (9–1) and Cambridge O Level syllabuses are at the same level.

Making entries

Exams officers are responsible for submitting entries to Cambridge International. We encourage them to work closely with you to make sure they enter the right number of candidates for the right combination of syllabus components. Entry option codes and instructions for submitting entries are in the *Cambridge Guide to Making Entries*. Your exams officer has access to this guide.

Exam administration

To keep our exams secure, we produce question papers for different areas of the world, known as administrative zones. We allocate all Cambridge schools to an administrative zone determined by their location. Each zone has a specific timetable.

Some of our syllabuses offer candidates different assessment options. An entry option code is used to identify the components the candidate will take relevant to the administrative zone and the available assessment options.

Support for exams officers

We know how important exams officers are to the successful running of exams. We provide them with the support they need to make entries on time. Your exams officer will find this support, and guidance for all other phases of the Cambridge Exams Cycle, at **www.cambridgeinternational.org/eoguide**

Retakes

Candidates can retake the whole qualification as many times as they want to. Information on retake entries is at **www.cambridgeinternational.org/retakes**

Language

This syllabus and the related assessment materials are available in English only.

Accessibility and equality

Syllabus and assessment design

At Cambridge International, we work to avoid direct or indirect discrimination in our syllabuses and assessment materials. We aim to maximise inclusivity for candidates of all national, cultural or social backgrounds and candidates with protected characteristics, which include special educational needs and disability, religion and belief, and characteristics related to gender and identity. We also aim to make our materials as accessible as possible by using accessible language and applying accessible design principles. This gives all candidates the fairest possible opportunity to demonstrate their knowledge, skills and understanding and helps to minimise the requirement to make reasonable adjustments during the assessment process.

Access arrangements

Access arrangements (including modified papers) are the principal way in which Cambridge International complies with our duty, as guided by the UK Equality Act (2010), to make 'reasonable adjustments' for candidates with special educational needs (SEN), disability, illness or injury. Where a candidate would otherwise be at a substantial disadvantage in comparison to a candidate with no SEN, disability, illness or injury, we may be able to agree pre-examination access arrangements. These arrangements help a candidate by minimising accessibility barriers and maximising their opportunity to demonstrate their knowledge, skills and understanding in an assessment.

Important:

Requested access arrangements should be based on evidence of the candidate's barrier to assessment and should also reflect their normal way of working at school. This is explained in the *Cambridge Handbook* **www.cambridgeinternational.org/eoguide**

- For Cambridge International to approve an access arrangement, we will need to agree that it constitutes a reasonable adjustment, involves reasonable cost and timeframe and does not affect the security and integrity of the assessment.
- Availability of access arrangements should be checked by centres at the start of the course. Details of our standard access arrangements and modified question papers are available in the *Cambridge Handbook* **www.cambridgeinternational.org/eoguide**
- Please contact us at the start of the course to find out if we are able to approve an arrangement that is not included in the list of standard access arrangements.
- Candidates who cannot access parts of the assessment may be able to receive an award based on the parts they have completed.

After the exam

Grading and reporting

Grades 9, 8, 7, 6, 5, 4, 3, 2 or 1 indicate the standard a candidate achieved at Cambridge IGCSE (9–1).

9 is the highest and 1 is the lowest. 'Ungraded' means that the candidate's performance did not meet the standard required for grade 1. 'Ungraded' is reported on the statement of results but not on the certificate.

In specific circumstances your candidates may see one of the following letters on their statement of results:

- Q (PENDING)
- X (NO RESULT).

These letters do not appear on the certificate.

On the statement of results and certificates, Cambridge IGCSE is shown as INTERNATIONAL GENERAL CERTIFICATE OF SECONDARY EDUCATION (IGCSE).

How students and teachers can use the grades

Assessment at Cambridge IGCSE has two purposes:

1 to measure learning and achievement

The assessment confirms achievement and performance in relation to the knowledge, understanding and skills specified in the syllabus.

2 to show likely future success

The outcomes help predict which students are well prepared for a particular course or career and/or which students are more likely to be successful.

The outcomes help students choose the most suitable course or career.

Changes to this syllabus for 2026, 2027 and 2028

The syllabus has been updated. This is version 6, published September 2026.

You must read the whole syllabus before planning your teaching programme. We review our syllabuses regularly to make sure they continue to meet the needs of our schools. In updating this syllabus, we have made it easier for teachers and students to understand, keeping the familiar features that teachers and schools value.

Changes to version 6 of the syllabus, published September 2026

Changes to syllabus content	• Information on page 41 relating to the `ROUND` function has been updated.

Changes to version 5 of the syllabus, published December 2025

Other changes	• The weblink on page 50 has been updated.

Changes to version 4 of the syllabus, published June 2025

Changes to syllabus content	• The names of operators on pages 33 and 40 have been updated. • The number of parameters for a function and procedure has been updated.

Changes to version 3 of the syllabus, published January 2025

Changes to syllabus content	• The names of operators on page 28 have been updated.

Changes to version 2 of the syllabus, published May 2024

Changes to syllabus content	• We have updated the guidance at the start of the Algorithms, programming and logic section on page 25 to clarify the following: – In Paper 2, candidates are expected to write solutions to given problems using the methods and concepts listed in topics 7 to 10. – Where the solution to a problem involves coding, candidates are required to write solutions in pseudocode. Solutions written in programming code will **not** be awarded marks. The exception is the 15-mark scenario question. – In the 15-mark scenario question, candidates are required to write solutions to the given problem using pseudocode or their choice of one of the following programming languages: Python, Visual Basic or Java. Solutions written in other programming languages will **not** be awarded marks. • We have updated the guidance at the start of the Paper 02 guidance on page 32 to clarify that the Scenario question is based on the methods and concepts listed in topics 7 to 10.

Changes to version 1 of the syllabus, published September 2023

Changes to syllabus content	• The wording for learning outcome 1.1.6 on page 12 has been updated. Candidates should be able to use the two's complement number system to represent positive and negative 8-bit binary integers • Learning outcome 1.3.2 on page 13 has been updated. Calculations must use the measurement of 1024 and not 1000 • A minor change has been made to learning outcome 2.1.1 (b) on page 14 for clarity • The abbreviation of the fetch–decode–execute (FDE) cycle has been added to learning outcome 3.1.2 (b) on page 16 • Learning outcome 3.3.3 on page 18 has been updated to include further examples of magnetic, optical and solid-state (flash memory) storage • Learning outcome 7.5 (b) on page 26 has been updated to include the purpose of each verification check and writing algorithms to implement each verification check • Learning outcome 9.4 on page 30 has been updated to include the following functions: – `ORDER BY DESCENDING` – `ORDER BY ASCENDING` – `AND` – `OR` • Some minor formatting changes have been made to the Post-condition (`REPEAT`) loops (8.1.4 (c)) section on page 45 • The Procedures and functions section on page 46 has been updated to clarify that procedures and functions are defined at the start of the code • The File Handling section on page 48 has been updated to add formatting information to the coding in the example • Some minor formatting changes have been made throughout the learning outcomes to improve clarity.

Significant changes to the syllabus are indicated by black vertical lines either side of the text.

Any textbooks endorsed to support the syllabus for examination from 2023 are still suitable for use with this syllabus.

You should take account of the changes described above when using these textbooks.

School feedback: ‘While studying Cambridge IGCSE and Cambridge International A Levels, students broaden their horizons through a global perspective and develop a lasting passion for learning.’

Feedback from: Zhai Xiaoning, Deputy Principal, The High School Affiliated to Renmin University of China

We are committed to making our documents accessible in accordance with the WCAG 2.1 Standard. We are always looking to improve the accessibility of our documents. If you find any problems or you think we are not meeting accessibility requirements, contact us at **info@cambridgeinternational.org** with the subject heading: Digital accessibility. If you need this document in a different format, contact us and supply your name, email address and requirements and we will respond within 15 working days.

Cambridge Assessment International Education, The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA, United Kingdom
t: +44 (0)1223 553554 email: info@cambridgeinternational.org www.cambridgeinternational.org

© Cambridge University Press & Assessment September 2023