ICT Level 2 – Programming Concepts

1 of 19 – Welcome

Welcome to this session on programming concepts.

By the end of this session you will:

* Know what hierarchical structures are
* Understand what high-level programming language is
* Understand what low-level programming language is
* Understand how these individual components help to build programs

2 of 19 – Introduction to programming concepts

**What is computer programming?**

**Computer programming** is a very creative process that involves a number of different tasks.

The **programming process** as a whole can be thought of as the way a technology-user teaches their computer something. This is because when computer programs are built, it is with a **specific aim or function in mind**.

Programs might be designed to help your machine perform certain tasks better, or they may even be designed for entertainment purposes – such as a better gaming experience, for example.

To build these programs correctly though, you need a **good knowledge of how a computer works**.

3 of 19 – Hierarchical structures

A **hierarchy** is a method of displaying various things or people – or, when talking about computers, this is likely to be software – in a way that explains their importance.

Typically, there is the top level of something displayed at the top of the hierarchy, and from this there are **various branches** that explain the different layers that exist beneath.

Hierarchical structures are also used inside computers to display the software that is present, and the different jobs this software is capable of doing.

How different programs might affect or control the computer’s hardware can be thought of in terms of a hierarchy too.

4 of 19 – Hierarchy of software in a computer

From the structure below we can see how different software works together to transmit different information to different parts of a computer system, resulting in the right display for the user:

1. High-level and low-level programming languages
2. Machine code – programs are mixed with machine code
3. Computer hardware
4. User interface – machine code is performed by the CPU, which controls the hardware, and displays the proper user interface

We will look at these terms in more detail in the following sections.

5 of 19 – Hardware: a quick reminder!

It is important to remember what is meant by the term computer **hardware** – the hardware is anything that is used either inside or alongside a computer (unlike software, which is something that is added or installed).

Software can be thought of as the thing that controls the hardware, given that software introduces different instructions or commands that the central processing unit must translate and distribute across the necessary computer parts. Consider the following process:

1. Software introduces instructions in machine code
2. Central processing unit identifies the instructions and then distributes them
3. The different components of the computer then perform the instructions of the software

6 of 19 – Giving a computer instructions

When a computer programmer is introducing a new set of instructions, these instructions need to be given to the machine in a way that it will understand – or at least, using a language that the computer can properly translate. The language of computers is called **machine code**.

This is made up of **binary units** that the computer – or rather, the central processing unit within the computer – is able to understand, and then break down further into tasks for each individual component to complete.

Before this stage though, computer programmers first have to write their programs using one of two languages: **high-level programming language** or **low-level programming language**.

7 of 19 – What is a high-level programming language?

High-level programming language is **quite close to natural human speech**, and quite far away from the language of computers.

It was originally **designed to help simplify the programming process**, as programmers could write their work in this language without the added complication of machine code. Examples of high-level programming languages are:

* Visual Basic
* Java
* C# (pronounced C Sharp)
* C++

There are a number of other types of high-level programming languages available beyond these mentioned, too!

8 of 19 – High-level programming language – continued

Computer programming is now so popular amongst technology users that some software – Microsoft Office, for example – has introduced **Visual Basic for Applications** into their software programmes.

The effect of this new high-level addition is that computer users can now tweak their applications. For example, an Excel user would be able to change their application in order to hide or highlight rows of their document that are or are not needed.

It is important to remember that these high-level programming languages – wherever they are being used – must later be **translated into machine code**, for the computer to run through the instructions properly.

9 of 19 – What is a low-level programming language?

Low-level programming languages are a little closer to machine code – closer than high-level programming languages, that is – so low-level languages need fewer things to be changed before the computer can understand the instructions being delivered.

When programmers are using low-level programming languages, they must also use an assembler. This is a space designed to help write programs using low-level languages – so it makes the programming processing slightly easier to complete.

An assembler can be thought of as a translator in many ways. This is because as a programmer inputs their low-level programming language as a series of instructions, the assembler then re-writes the language into machine code, meaning the computer will be able to understand it.

10 of 19 – Using high and low-level programming languages

Each language type comes with its own advantages and disadvantages that programmers have to consider.

High-level programming languages are **easier to use** – they are **less time-consuming**, and it is often **easier to find and fix bugs** with these language types, too.

Also, because it is easier to use, it often means that programmers have a **higher level of productivity**.

However, while the languages are easier to use, they do then **have to be translated** – so the time saved is quickly spent on this translation process instead. Also, programs that have been written with high-level programming languages have also been known to **run slower** than low-level alternatives.

11 of 19 – Using high and low-level programming languages – continued

Low-level programming languages are **fast-acting**, meaning programs written in low-level tend to operate more easily and quickly. Also, because low-level programming languages are a little **closer to machine code**, these language types can **communicate more easily with the computer’s hardware**.

The use of an **assembler** (remember: a low-level programming language translator) can cut down the time it takes to introduce low-level languages to a computer system, too!

That being said, low-level programming languages do also have some disadvantages.

Not only are they **more difficult** for programmers to use, they tend to be a little **more time-consuming** and **harder to fix** if, for example, there is a bug or a glitch. Low-level programming languages can be **difficult to maintain** as well.

12 of 19 – Types of programming

When a programmer is plotting how to write a new computer program, something else that might be considered is whether the programming will be **functional** or **object-oriented**.

These two different programming types basically group programs together in slightly different ways, for example:

* **Object-oriented** – which is used by many programmers now – will group programs together by treating them as objects. Similar programs are grouped together like a family of objects – so they may be structured in a similar way – but they will have their own obvious characteristics
* **Functional programming** – rather than looking at the programs as a whole, will look at their functions instead, with a particular emphasis on input and output functions. Functional programming relies heavily on data analysis and, while useful, it has historically been used less than alternative methods

13 of 19 – Application software

Application software is the official term for **programs or applications that have been written and developed** – so they are the final products at the end of the programming process.

Common examples of application software – remember: these programs can be used to help with a certain task, or simply to entertain – might be things such as:

* Word processors
* Internet browsers
* Database systems

**Remember:** everything that runs on your computer has one of these programming languages behind it. It has taken a lot of work to write and build them to the standards they are at now.

14 of 19 – Question 1

How is a hierarchical structure used in computer programming?

1. It explains how hardware influences software
2. It shows which piece of hardware is the most important
3. It explains how software influences hardware
4. It helps you to isolate problems with the computer’s hardware

The correct answer is C, it explains how software influences hardware.

15 of 19 – Question 2

What is the language of computers?

1. Binary code
2. Machine code
3. Computer code
4. Hardware code

The correct answer is B, machine code.

16 of 19 – Question 3

Indicate whether the following statements are true or false.

C#, C++, and Java are all examples of high-level programming languages.

True

False

The correct answer is: True

Visual Basic is an example of a low-level programming language.

True

False

The correct answer is: False

Low-level programming languages should be used with an assembler.

True

False

The correct answer is: True

An assembler is something that reads low-level programming languages and re-writes them as high-level.

True

False

The correct answer is: False

17 of 19 – Question 4

Using the following choice of words; **low-level**, **system**, **bugs**, **Application**, **processors**, **machine code**, **browsers**, **completed**, **difficult** and **translations**, fill in the blanks for the paragraph below:

**Blank** software is a term that covers all **blank** programs or applications. Once the program has been written – using a high or **blank** programming language – then it can be introduced to a computer **blank**. However, programmers have to make sure that all of the necessary **blank** have taken place first, so the instructions are in **blank**. Completed application software can take many forms; some common examples might be word **blank** or internet **blank**. Programmers have to be careful during the writing process though, as both types of programming language can be **blank** to use, and both can develop **blank**.

The correct paragraph should read:

**Application** software is a term that covers all **completed** programs or applications. Once the program has been written – using a high or **low-level** programming language – then it can be introduced to a computer **system**. However, programmers have to make sure that all of the necessary **translations** have taken place first, so the instructions are in **machine code**. Completed application software can take many forms; some common examples might be word **processors** or internet **browsers**. Programmers have to be careful during the writing process though, as both types of programming language can be **difficult** to use, and both can develop **bugs**.

18 of 19 – End

Well done. You have completed this session on programming concepts.

In this session we have covered:

* What hierarchical structures are
* What high-level programming language is
* What low-level programming language is
* How these individual components help build programs

If you have any questions about any of these topics, make a note and speak to your tutor for more help.