ICT Level 2 – High and Low-level Programming Languages

1 of 19 – Welcome

Welcome to this session on high and low-level programming languages.

By the end of this session you will:

* Know and understand the uses of high-level programming languages
* Know and understand the uses of low-level programming languages
* Know what type of programming jobs each language is best suited to

2 of 19 – Introduction to programming languages

**What are programming languages?**

Programming languages is the phrase used to describe the high- and low-level programming speak that computer programmers use in order to write new programs.

Languages can either be high-level or low-level and, after writing a program in either of these language types, the programmer then has to convert the language used into something that the computer will properly understand.

**Remember:** this computer language is called machine code.

There are a number of different types of high and low-level programming languages available.

3 of 19 – An introduction to high and low-level languages

High-level programming languages are preferred by many programmers, largely because they are slightly easier to use (this is because they are closer to our human language than they are to the computer language). Also, if you have to repeat code or build instructions, high-level programming languages are much easier for this type of task.

Low-level programming languages work slightly differently. Instead of a little bit of code being written to cover many actions, low-level languages are more intricate – meaning there is a lot of code used for each individual action (even if the action is only a small one).

Overall, low-level programming languages involve a lot more detail, which can make them quite time-consuming to write – although they do have one or two advantages, which we will think about later.

4 of 19 – High-level programming languages – a closer look

Not only are high-level programming languages easier to use, but they are also very powerful – meaning that they can be used to write a number of different types of program. Also, because of how close these languages are to our own language, the programs written with high-level language tend to flow a little more easily too (meaning the programs are thought to be smooth-running).

In terms of the uses of high-level programming languages, there are a number of different programs to consider:

* Imperative code
* Event-driven programs
* Procedural programs
* Object-oriented programs (OOP)

5 of 19 – Imperative code: what is it?

**Imperative code: what is it?**

Imperative code is something used by many computer processors as it allows them to break down the individual processes within the program.

While declarative code might explain what a programmer wants, it does not always explain fully how to get from the start point to the end point.

This is the most notable difference with imperative code, as this type of coding breaks everything down into a step-by-step pattern – so it can make for a very detailed program at the end of the writing process, and it also provides the machine with clear instructions on how to run the program.

It allows the program to handle the inputs, processes, and outputs – which are the key functions in any software – but it also tells the program precisely which steps to follow to get there.

6 of 19 – Procedural programs: what are these?

The key feature of procedural programs is that they have a specific start and end point to them. Before GUI operating systems were introduced, procedural programs were normal – common, even – in computers, as their one start point made them easier for computer users to interact with them.

**Remember:** GUI means Graphical User Interface.

Procedural programs are used a little differently now though. Instead of responding to one start point, modern technology can now properly process programs that have more than one start point and so the code behind these programs no longer has to write a fixed beginning – it can write multiple beginnings instead!

7 of 19 – Event-driven programs

Event-driven programs are designed to respond to an event – or, to think of it in a different way, they are designed to respond to a certain action. So when using an event-driven program, the click of a computer mouse can be considered an ‘event’, which will trigger something else inside the program.

Because GUI operating systems now offer many different events – or actions – the event-driven programs have also had to be changed slightly.

Event-driven programs now have to be built to respond to numerous events taking place in numerous parts of a computer system, as individual systems offer more options for users than ever before.

8 of 19 – Object-orientated programs: what are they?

Object-orientated programs – or OOP, for short – might sound familiar to you (these were introduced in an earlier session).

This approach does not see programs as a list of steps that need to be followed, but as a group of objects instead.

Each object exists on its own and works independently – having its own characteristics, for example – but it has a job that it must perform in order to work within or alongside other objects (or rather, programs).

**Note:** This area in particular can be a little tricky to understand, so do not worry too much about understanding each part of the process – knowing that it is used is enough for now.

9 of 19 – Question 1

What are some of the advantages to using high-level programming languages?

Choose all that apply:

1. They are useful for intricate codes
2. They are easier to use
3. They are powerful
4. They are cheaper to write
5. They make smooth-running programs
6. They are time-consuming

The correct answers are B, C and E, they are easier to use, they are powerful, and they make smooth-running programs.

10 of 19 – Question 2

What does imperative code do?

1. It breaks down inputs, processes, and outputs into a step-by-step process
2. It states what a program will do without explaining the steps it will take to get there

The correct answer is A, it breaks down inputs, processes, and outputs into a step-by-step process.

11 of 19 – Question 3

Indicate whether the following statements are true or false.

GUI stands for Graphic User Intel.

True

False

The correct answer is: False

Procedural programs have a specific start and end point.

True

False

The correct answer is: True

Event-driven programs are designed to respond to an ‘event’ on the computer.

True

False

The correct answer is: True

Procedural programs can only feature one start point, which is a disadvantage to using them.

True

False

The correct answer is: False

12 of 19 – Low-level programming languages – a closer look

**Remember:** Low-level programming languages are closer to computer language than high-level languages are.

When programmers are using low-level programming languages, they will also being using an assembler. An assembler is something that low-level programming language can be written into, and the assembler will then re-structure, or translate, the language into something the computer system will understand i.e. machine code.

Another useful feature of the assembler is that while it is translating low-level programming languages into machine code, it can also scan the code for any errors in the mnemonics before the code is finalised.

Mnemonics are abbreviations that stand for instructions, for example: MOV would be a move instruction.

13 of 19 – Low-level programming languages – continued

**Remember:** machine code is made up of different binary units. Binary is the type of ‘language’ that can be understood by different elements within a computer system.

A programmer might decide to completely ignore the option of using an assembler though, and instead input binary figures straight into the computer itself.

However, not only is this difficult and time-consuming for a programmer, there is also something of a language barrier as the programmer can only see the program as numbers (binary numbers, that is) rather than in the format that they originally wrote the program in.

Understanding a program – especially when it is still being developed and changed – is just easier to do without the added complication of binary, which is why assemblers are used so often.

14 of 19 – Low-level programming languages – the advantages

While low-level programming languages are thought to be harder to control than their high-level language alternatives, low-level options are used by many experienced programmers – although it is worth noting that experience is the key to using low-level languages effectively.

They are thought to be more time-efficient – as they do not have to be translated by the programmer – and they are more friendly to your computer, as low-level options are closer to the language that computers understand anyway.

There are even some types of low-level programming languages that computer hardware can understand immediately, without a translation taking place. Remember though that even when a translation is needed, low-level languages can always rely on the assembler.

15 of 19 – Question 4

Using the following choice of words; **errors**, **assembler**, **low-level**, **computer**, **input** and **machine**, fill in the blanks for the paragraph below:

**Blank** languages are considered to be quite close to **blank** languages, which is why it is easy to make them computer friendly. By using an **blank**, low-level programming languages can be re-arranged as **blank** code. Assemblers are also useful for detecting **blank** and will highlight these before the translated code is **blank** into the computer.

The correct paragraph should read:

**Low-level** languages are considered to be quite close to **computer** languages, which is why it is easy to make them computer friendly. By using an **assembler**, low-level programming languages can be re-arranged as **machine** code. Assemblers are also useful for detecting **errors** and will highlight these before the translated code is **input** into the computer.

16 of 19 – Question 5

What are some of the advantages to using low-level programming languages?

Choose all that apply:

1. They are time-efficient
2. They are easier to use
3. It is easier to translate them
4. They are more intricate than other options
5. It can take more time to use these languages
6. Some can immediately interact with your computer’s hardware

The correct answers are A, C and F, they are time-efficient, it is easier to translate them, and some can immediately interact with your computer’s hardware.

17 of 19 – Question 6

Some programmers might decide to use a low-level programming language without using an assembler. Are there any disadvantages to this?

1. No, it can sometimes be a quicker way of inputting code into the computer system
2. Yes, as this can create a language barrier and might prevent the programmer from seeing mistakes

The correct answer is B, yes, as this can create a language barrier and might prevent the programmer from seeing mistakes.

18 of 19 – End

Well done. You have completed this session on high and low-level programming languages.

In this session we have covered:

* The uses of high-level programming languages
* The uses of low-level programming languages
* What type of programming jobs each language is best suited to

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