ICT Level 2 – Planning a Program

1 of 19 – Welcome

Welcome to this session on planning a program. In this session we will look at putting programming terms into practice and consider some of the questions a programmer might ask before they begin their work.

By the end of this session, you will:

* Know the main questions to ask before beginning a program design
* Understand how to answer these questions effectively, and in enough depth
* Know how to use these questions, and answers, as part of the planning process

2 of 19 – Introduction to planning a program

**Why is a program plan important?**

While there are many technical tools and terms that can be used in programming itself, there is also a lot of planning that needs to happen too, before a programmer can arrive at the construction stage.

If a programmer can develop a detailed plan for their work ahead of actually starting it, they will reduce the risk of wasting time, or producing a program that does not perform the right task or fix the right problem.

There are a number of key questions that can be considered as part of this planning process, and these questions are designed to address a number of key elements and concerns within programming.

3 of 19 – What’s the problem?

Programs are almost always designed with a specific problem (that needs to be fixed) or a specific task (that needs to be completed) in mind.

For example: a word processor is designed to make writing easier, as well as improving a user’s ability to edit their work.

The aim of any given program is to simplify an existing task – the word processor example shows this in practice as well. This is why, ahead of beginning program design or construction, it is important to ask: what’s the problem?

When a programmer has isolated their problem, they can then start thinking about ways in which this can be fixed, which gives them early directions about what they need from their program.

4 of 19 – Who is using this?

Whenever a program is developed it is designed with an ideal user in mind – this is the person a programmer imagines will use their software.

From this imagined person a programmer can then develop their ideas based around who they suspect will be using the product, and also what these imagined users are likely to want from the product.

For this stage a programmer has to set aside their personal preferences and consider the user’s needs before anything else. User interface is an important element of program design in technology systems, and how a user can interact with any given software has to be a primary concern.

**Remember:** user interface refers to the parts of a computer system that a user can in some way interact with.

5 of 19 – Where are they using this?

Another question that has to be asked in the early planning stages is where a programmer imagines a user will use this software; not geographically, but rather what computer system a user might be using, and what computer system the programmer has in mind for their work.

Although programs are designed for computer systems, not all computer systems are alike, which means that certain program designs will not always be appropriate for certain technology systems. Some further things to consider are:

* What programming language will be used
* What hardware might be needed
* How large, or small, the program must be for the system

6 of 19 – Where are they using this? continued

The programming language that a programmer chooses to use is particularly important here, as this will influence which computer system, or systems, the program can be run on.

A programming language can decide the portability of any given program.

Portability is the term used to describe the programs that can be run on different computer systems with little or no changes being made, i.e. the software is portable.

However, it is important to remember that not all programming languages can be transferred like this. There are some computer systems that will only run software written in a certain programming language, meaning that the chosen language could limit the accessibility – and perhaps even the overall popularity – of a program.

7 of 19 – Can you write this?

Beginner programmers might find it particularly hard to write the necessary code for an entire program – this is after all, an advanced activity to attempt.

This is why, when planning a program, it is important to isolate which parts can be built by the programmer and which parts should be outsourced to specialists.

Big ideas are fun to have, but it is important – for user experience, and for the overall effectiveness of the program – that the program is built as well as can be, which may mean allowing more experienced programmers to take over certain parts of the development.

Programmers do not have to outsource their work entirely, though. Sometimes there is the opportunity to collaborate with other programmers, and there are several ways of doing this.

8 of 19 – Can you write this? continued

A common example of how programmers can collaborate on programming projects, or source more experienced programmers for help, is through the use of Visual Studio.

**Remember:** Visual Studio is a virtual environment that can be used by programmers to design and plan their software.

Programmers will use Visual Studio, or a similar system, to design their program up to the point that they are able to. When their program is developed enough, they will then source another programmer to assist with their developments.

These may be programmers who are hoping to collaborate on a program, or they may be more experienced programmers who are hired to finish the development of a program. Either way, the original programmer can ensure that their work is being continued by people who are qualified to do so.

9 of 19 – Annotations

You might remember talk of annotations from an earlier session. Annotations are little notes or comments that can be added to stretches of code, and they are useful in a number of ways.

Their main two functions are to highlight which piece of code is performing a certain action. They can also be used as an editing tool.

A programmer may use annotations to mark up a problematic part of their program – if, for example, certain code is not working properly, or not performing the task it was intended for.

Collaborative work and outsourcing programmers for work provide a real life look at how these annotations might be used in practice. If a programmer is giving their work to someone else – either for help, or simply for a second opinion – annotations will be a useful tool for instructing the other programmer.

10 of 19 – Beyond these questions

Within these main questions there are sets of sub-questions, meaning these big ideas can be broken down into smaller ones if the programmer wants them to be.

Two smaller questions to consider might be:

* What features will this program include?
* What features must this program include?

These two questions return to the concerns of what problem the program is fixing and who will be using the program, but they allow programmers to approach these areas in different ways.

While the features a program will include might be elements that appeal to the programmer, asking what must be included ensures that the programmer also has a list of essentials – these will be things that the program definitely needs, and the ideal user is probably looking for.

11 of 19 – Programming timeline

Answering these questions in as much depth as possible will give a programmer a clear idea of:

* What their program should do
* Who it should be aimed at
* How it will be run
* How it will be built

This leaves one final question to consider: how long will it take?

Programming is not an easy job to do, especially when a programmer is just beginning their work, and a timeline will be particularly useful in providing a guiding framework for how much time should be spent on certain tasks. It will also give an idea of when the program will be ready for testing and, eventually, public release.

12 of 19 – How programming terms relate to this

Once a programmer has the basic skeleton of their program, from answering the earlier questions, they will be ready to plan things in more detail – using some of the programming terms and techniques that were covered in earlier sessions.

At this stage, a programmer might introduce the use of a flow chart. This will allow them to put their ideas into a practical shape, considering things such as where the program will start and end, and the various steps that will be taken to arrive at certain instructions or commands.

Plotting out the design of a program on a flow chart will also give an early idea of what code needs to be written into which parts of the program.

13 of 19 – Flow charts and programming terms

Once a program design is plotted onto a flow chart, a programmer will be able to observe where certain actions need to take place, and where certain programming tools can be introduced. For example:

If Point A of a flow chart has a connector (an arrow) to Point B, the programmer knows that there needs to be some kind of data input at Point A in order for the user to arrive at B.

Connectors are used to show the movement of the program, so a programmer will need data input from the user in order to trigger this movement of events.

By taking the design of a program and introducing it to a flow chart structure, a programmer will build a real life illustration of how their program should move and function – which makes it a little easier for them to start the real construction.

14 of 19 – Question 1

There are a number of questions a programmer should ask before constructing their software. Why are these questions important?

Choose **two** answers that apply:

1. It ignores the user and focuses on the program
2. It prevents time wasting
3. It isolates a number of potential problems to fix
4. It isolates a specific problem or task
5. It helps a programmer to delay construction

The correct answers are B and D, it prevents time wasting and it isolates a specific problem or task.

15 of 19 – Question 2

Read the statements below and decide which ones are true and which ones are false.

When a programmer is developing an idea, it is important that they do not get distracted by their ideal user.

True

False

The correct answer is: False

A programming language will influence the portability of any given program.

True

False

The correct answer is: True

Portability refers to the ease with which a user can use a program on the move.

True

False

The correct answer is: False

A programming language can hugely impact the accessibility, and popularity, of a program.

True

False

The correct answer is: True

16 of 19 – Question 3

Why might it be important for a programmer to outsource some of their developments?

1. It stops the programmer from getting bored with the project
2. It means that the programmer has less work to do
3. It means that parts of the program can be built by a specialist programmer
4. It means that the programmer can take a holiday

The correct answer is C, it means that parts of the program can be built by a specialist programmer.

17 of 19 – Question 4

Using the following choice of words; **timeline**, **to develop**, **the public**, **ideal**, **time**, **computer systems**, **testing** and **problem**, fill in the blanks for the paragraph below:

When a programmer is designing a project, not only do they need to remember the **blank** or task, the **blank** user, the intended **blank** and the skill needed – but they also need to remember how much **blank** they have, too! That is why developing a **blank** is an important part of planning; it allows a programmer to decide how long each part of the program should take **blank**, which can be a useful guide to have. This will also be useful for giving a programmer a finish line too, as a timeline will show them when the program should be ready for **blank**, and when the program could be ready for **blank**.

The correct paragraph should read:

When a programmer is designing a project, not only do they need to remember the **problem** or task, the **ideal** user, the intended **computer systems** and the skill needed – but they also need to remember how much **time** they have, too! That is why developing a **timeline** is an important part of planning; it allows a programmer to decide how long each part of the program should take **to develop**, which can be a useful guide to have. This will also be useful for giving a programmer a finish line too, as a timeline will show them when the program should be ready for **testing**, and when the program could be ready for **the public**.

18 of 19 – End

Well done! You have now completed this session on planning a program.

In this session we have covered:

* The main questions to ask before beginning your program design
* How to answer these questions effectively, and in enough depth
* How to use these questions, and answers, as part of the planning process

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