ICT Level 2 – Writing Computer Programs – Programming Terms Part 2

1 of 20 – Welcome

Welcome to this session on writing computer programs: programming terms (part 2).

By the end of this session, you will:

* Have been reminded what is meant by data input and output
* Understand, in more depth, what is meant by subroutines
* Understand, in more depth, what is meant by annotations
* Know and understand the five different data types that are available

2 of 20 – Introduction to writing computer programs: programming terms

**What are programming terms?**

Broadly speaking, programming terms can cover any word or technique that somehow relates to the process of programming software.

The terms that we have looked at in earlier sessions, such as flow charts, terminators, and variables, have all been examples of these, as they are specific tools or practices that can be used by programmers.

The terms that we are going to look at in this session are also examples of programming terms, as they are designed to help an individual not only construct a program but construct a program in the most straightforward and sensible way for them and their work.

3 of 20 – Data input and output

You might remember the terms input and output from an earlier session on programming terms (they are often used when designing programming flow charts). As a quick reminder though:

* **Data input** refers to data that has been entered into, or communicated to, the software (this involves the user giving information or instructions to the program)
* **Data output** refers to the data that follows data input, meaning it is the end result of whatever the user has instructed the program to do – this might be displayed on the monitor of the computer, or it might result in a certain action taking place

**Remember:** data input can be any number of things – not just an instruction! It may be a number typed into a box, or even a button click, for example.

4 of 20 – Subroutines: a closer look

A **subroutine** can be called a number of different things, depending on the programming language that is being used. In other programming languages subroutines might also be referred to as: a procedure, a function, a routine, a method, or a subprogram.

Whatever the preferred term, subroutines can also be broadly referred to as callable units, as this term covers all of the other names for subroutines too.

Subroutines are a powerful programming tool that are used in most programming languages. They work by structuring small amounts of code into groups, or sections, meaning there will eventually be a subroutine developed for each individual part of the program.

In modern programming, subroutines are something that will develop naturally – meaning code will automatically group itself together to create subroutines as the program is being written.

5 of 20 – Subroutines – continued

The **subroutines** that develop automatically tend to occur whenever a new event handler is introduced to the program.

An **event handler** is the code that runs whenever an event occurs in a program – these events could be a mouse movement, or a button click, and it is the movement or click that then triggers the release of this code.

These subroutines can also be used repeatedly throughout any one program, whenever the same code sequence is repeated. This means that they are particularly useful for cutting down on development time – and development costs, too – and they can also make writing the program a little easier.

Subroutines are also believed to **increase the reliability** of a program as well.

6 of 20 – Annotations: a closer look

**Annotations** are often used by professional developers – rather than those who are developing a program for enjoyment or practice.

Annotations are **similar to comment boxes** that can be written into, or attached to, the different areas of code that exist within the program.

You might be familiar with comment boxes on word documents, for example, that allow a reader to add a comment attached to a specific word or sentence – programming annotations rely on this exact practice.

They are often used to **indicate what a certain part of the code is doing**, or should be doing, and once that stretch of code has been completed and tested then the annotations are simply ignored.

There are **one or two key uses** for using annotations in programming, beyond marking up the purpose of certain pieces of code.

If, for example, a different programmer is asked to check or edit the code, then annotations will be a really useful tool, enabling the editing programmer to move around the program properly, and focus on the right problems or areas.

Annotations can also be an **important learning tool**. They are sometimes added to example code for the purpose of showing trainee programmers how annotations work and can be used, and also for showing them specific code sequences that have been used or edited in the program.

Overall, annotations can be thought of as a particularly useful tool for keeping track of how everything in the program is working.

7 of 20 – Data types: a closer look

You might remember data types from an earlier session where they were mentioned alongside variables. Data types are included alongside variables (in declarations) to tell a user what type of data can be input, and output.

Data types are an important part of programming because programs rely on data so much! In programming, data is used in calculations, comparisons, and for data output, too.

This is why it is important for a program to be using the most appropriate data type, as it means that the data being entered (input) and the data being delivered back (output) are of the best type for that particular program, making for a better user experience overall.

Using the right data type for a program will have a huge impact on the system’s use of memory, too!

8 of 20 – Data types: characters and strings

A **character** data type can allow for any single character, meaning a letter or number, to be input on its own.

A **string** data type works a little differently though, as this type allows for a string of characters to be input which can be made up of any combination of letters, numbers, and even spaces.

Examples of these data types in action can be seen in the table below:

|  |  |
| --- | --- |
| Data type | Example of data |
| Character | J |
| String | J S Thompson |

9 of 20 – Data types: integer and real

An **integer** data type – which you might remember from an earlier session – is a data type that allows for whole numbers, meaning you cannot input decimal points or fractions.

A **real** data type also allows for numbers to be input, but they do not need to be whole numbers exclusively, so while real data can be a whole number it can also be a fractional number, or rather a decimal number, as well.

Examples of these data types in action can be seen in the table below:

|  |  |
| --- | --- |
| Data type | Example of data |
| Integer | 218 |
| Real | 18.75 |

10 of 20 – Data types: Boolean

The **Boolean** data type works a little differently to the other types mentioned. Named after George Boole, this data type is rooted in algebraic systems of logic and relies on a two value system (meaning, you can only input one of two types of data).

These value systems can be thought of as true or false.

So an example of a Boolean data type would simply be (see table below):

|  |  |
| --- | --- |
| Data type | Example of data |
| Boolean | FALSE |

11 of 20 – Data structures: what are they?

The last important term that we will introduce in this session is **data structures**. These are tools that can be used by programmers as a way to organise their data; different data will be suited to different structures, and so there are one or two different options available.

The right data structure will improve the programmer’s access to data and may also improve the value or quality of it as well. These structures can also be particularly useful for helping a programmer approach and execute certain tasks involving their data, because they have easier access to individual parts.

The specific structures we will consider now are:

* A record data structure
* An array data structure

12 of 20 – A record data structure

A record data structure works in a similar way to a database structure, as both of these structures are made up of labelled fields where specific information can be stored.

A real life example of this style of structure would be something like (see table below):

|  |  |  |
| --- | --- | --- |
| Name | Age | Profession |
| Freddie | 46 | Accountant |

In this example the units across the top are the **fields** being used within this structure. Data is then input into the appropriate field areas so it can be accessed later by the programmer.

13 of 20 – An array data structure

An array structure works a little differently. In programming, when the term array is used, the programmer will be referring to a collection of objects, or items, that are all the same size as each other, and are all the same type of object as well.

Each individual object within the structure will have its own subscript – this might be a symbol, or a number – which identifies that object as unique.

So an array structure organises same-size-and-type variables in a way that allows them to be easily searched through, meaning a programmer can easily isolate one specific variable contained inside the structure by searching for its subscript.

Using an array structure might take a little longer but it is also a very useful way to maximise a program’s memory.

14 of 20 – Question 1

Indicate whether the following statements are true or false.

Data output refers to data that is put into a program by a user, like a mouse or button click.

True

False

The correct answer is: False

Subroutines have many names; a more general term for them is callable units.

True

False

The correct answer is: True

Subroutines always have to be organised by the programmer as they will not develop on their own.

True

False

The correct answer is: False

Annotations are like comment boxes for code, and they can be useful learning tools.

True

False

The correct answer is: True

15 of 20 – Question 2

What are the advantages of annotations?

Choose all that apply:

1. They can help during editing and maintenance
2. They often make the code look neater
3. They explain which code is performing which action
4. They can be useful to programmers who are still learning
5. They have no key uses and are often ignored

The correct answers are A, C and D, they can help during editing and maintenance, they explain which code is performing which action and they can be useful to programmers who are still learning.

16 of 20 – Question 3

Which data type is used for whole numbers?

1. Real
2. Integer
3. Boolean
4. String

The correct answer is B, integer.

17 of 20 – Question 4

Which of the following is an example of a string data type?

1. 3.17
2. TRUE
3. D A Webster
4. D

The correct answer is C, D A Webster.

18 of 20 – Question 5

Using the following choice of words; **structures**, **organise**, **array**, **database**, **subscripts**, **record**, **fields** and **memory space**, fill in the blanks for the paragraph below:

Data **blank** are an important tool in computer programming as they **blank** data in a way that helps the programmer to store and retrieve it – and they can be useful for maximising **blank**, too. Two types of data structures to consider are **blank** data structures and array data structures. A record data structure is similar to a **blank** structure, as they both rely on **blank** that separate different items of data. An **blank** structure works a little differently, as this provides items with individual **blank** instead.

The correct paragraph should read:

Data **structures** are an important tool in computer programming as they **organise** data in a way that helps the programmer to store and retrieve it – and they can be useful for maximising **memory space**, too. Two types of data structures to consider are **record** data structures and array data structures. A record data structure is similar to a **database** structure, as they both rely on **fields** that separate different items of data. An **array** structure works a little differently, as this provides items with individual **subscripts** instead.

19 of 20 – End

Well done. You have completed this session on writing computer programs: programming terms (part 2).

In this session we have covered:

* What is meant by data input and output
* What is meant by subroutines
* What is meant by annotations
* The five different data types

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