8. In-Out, In-Out, Shake it all About (Input-Output)

Communicating with the outside world

Humans think. The human mind is locked inside the body that is a human. Humans do more than just think though – they take note of things happening around them and react to it, taking actions accordingly that affect the outside world. The human mind is like a computer processor. The processor can process information – the computer equivalent of thinking – but to be of any use it needs to be able to sense and interact with the outside world. This is where **input-output** comes in. **Input** is the way information from the outside world enters a computer. **Output** is they way the computer presents information back out to the outside world and so alters it. Humans have a variety of input mechanisms: the five senses. Humans can see (take in visual input), hear (aural input), smell, taste and touch. Each is a different way the human brain obtains information from the outside world to process, think about, react to. Those are not the only way of sensing the outside world. Some fish can sense electric fields. Some birds can sense the Earth's magnetic field and use it to navigate. Counsellor Troy in *Star Trek: The Next Generation* can sense emotions. Computers also have a range of input mechanisms. The most common ones are the ability to detect key presses on a keyboard and movement of a mouse. Some ticket machines and information booths have touch screens allowing the computer to sense touch. Computers connected to the Internet can receive messages from other computers. Many other mechanisms are possible that allow a computer program access to information about the outside world. These are ways the computer senses the outside world.

Humans can also affect the outside world. They can do this by using speech or movement of various parts of the body. Throwing a ball, smiling, kissing, winking, and pushing a button are all ways that people can change things outside the confines of their heads using movement. They are all forms of output. Computers can output to a screen, or by sending messages to other computers along telephone lines.

Consider what it would be like to be able to see, hear, smell and taste but to be unable to move or speak. Such a person would have a perfectly normal intelligence and would be able to sense all that was happening around them. However they would have no way of communicating with the outside world: no way to communicate with their family and friends, no way to express their desires. All they could do is watch and think. It would be a form of hell, and unfortunately it can happen to people who have a major stroke. They regain consciousness in a hospital bed with no way of communicating. It is called locked-in syndrome. Someone with locked-in syndrome has input but no means of output. In fact, people with locked-in syndrome are often not totally paralysed. They may be able to blink one eye. That is enough with ingenuity for them to communicate if slowly as we will see in a later chapter.

In the song "Pinball Wizard" by *The Who,* Tommy is deaf, dumb and blind. He has severely limited input abilities being unable to see or hear. Despite this he is still brilliant at Pinball. How? Because he "plays by touch alone". Blind people can still read using Braille – reading by touch. They are using a different sense to achieve the same input.

A person who has locked in syndrome and who has lost all their senses so they can neither see, nor hear, nor taste, smell or feel, has no way of sensing the outside world and no way of communicating with it. They can do nothing to affect their surroundings. Their mind is cut off from the world: hell on earth. A computer with no input nor output is similarly unable to do anything useful in the outside world. That is why computers are supplied with a range of devices that allow communication: keyboards, mice, screens, printers and Internet connections. These are the things that ultimately make the processing power useful. However, to be used, the computer programs must include instructions that allow the information from the outside world to be used and allow information to be sent out. When I am working I often have music on in the background. Once I am engrossed in my task I do not actually listen to the music. It could be something I hate but I would not notice. My mind is not executing any instructions to process the lyrics being sung. Similarly, I do not speak much as a general rule – only when I have something to say – only when I decide to say something. My daughter, two at the time of writing, often does not seem to have any restrictions she chatters constantly, almost as though she is giving a commentary on everything she is thinking. She is running a program at those times in her head that outputs all she thinks. At other times, she sits silently reading books. Now she is executing input instructions but no output instructions.

If we were to design an instruction to do output, what would it need to consist of? It would need to indicate at least what was to be output and where it was to be output to. What do we need from an input instruction in a program? It must indicate somehow where the input is coming from (that is what device), and it must indicate where the information received is to be put (usually what variable to store it in until it can be further processed).

Interrupts

Persistent Storage

One special form of input-output that computers use is to and from **persistent storage**. The aim of this kind of input-output is a form of long term memory outside the computer.

When I am on holiday on the beach I write my daughter's name in the sand for fun. As long as such messages are only for the afternoon then that is a perfectly good place to leave them. When we return the next day, though the messages are gone – the tide comes in and washes them away. There would be no point me leaving a message when she was not there for her to see the next day as it would be gone. Sand messages on a beach are **non-persistent**: they do not stay around permanently. I can write names and smooth them away as much as I like during the afternoon, but that is as long as they last. Persistence thus does not mean it is impossible to delete a piece of information, just that it will stay around until you actively delete it (by smoothing the sand yourself say). Computers use **non-persistent storage** as their everyday memory. That is why it is so important when using a computer to remember to save things and why it is so easy to "lose" hours of work if your daughter decides to press the off button while you are typing. When you save something you transfer it to **persistent storage**: a place where it will stay until actively replaced.

Vandals writing graffiti do want persistence. They want their graffiti to still be there the next time they come by. They therefore do not write their graffiti using something that will wash away the first time it rains, but instead use permanent markers and aerosol sprays. This gives **persistent storage** for their handiwork.

Before writing was invented and in common use, knowledge and stories were passed orally. The only way the knowledge was preserved was by it being constantly passed from one person to the next. The Koran is still, I am told by a Muslim friend, passed on in this way being learnt word perfect by each new generation to ensure the words are exactly those of the original. The memories of Muslims throughout the world are acting as a storage method for the words in the original Koran. Languages and culture are passed on in the same way from generation to generation. The problem is that if one generation decide not to do the storage, the knowledge may be lost forever, if that is the only storage method used. Many languages have disappeared because a generation has lost interest. it takes great dedication, as with the Koran, to use this method of ensuring knowledge is not lost. Writing was invented as a form of persistent storage. The written word stays around much longer than any individual person. That is why we can learn much about ancient civilisations that have long been extinct.

Libraries are a form of persistent storage. The things they store are books. I just tried to order a book from my local bookshop about Computers and the Mind. I was told it was impossible for me to get as it was out of print. However, that just means I cant buy it. I can still get a copy to read as the British Library keeps a copy of every single book printed. The British Library's mission is to act as a persistent storage for books to ensure that no English book ever printed disappears.

The main mechanism for persistent storage in computers is the **file**. Most variables used in a computer program that hold strings, integers, etc are non-persistent and so are lost as soon as a program ends (if not sooner). Passing on knowledge orally is like keeping a computer switched on permanently to ensure information is not lost. The problem is it needs constant attention (or more specifically for the computer, constant electricity). Similarly when the life force leaves a person and they die, their memories cease to exist too (though unfortunately switching a person back on is not currently possible after they die). On the other hand, something placed in a **file** will still be there even if the computer is switched off and it is days before you return to it. Storing something in a file is like storing something by writing it down on paper rather than trying to remember it in your head. Persistent storage is something that by its nature exists outside the program, in the same way as paper exists outside the head of the person who writes on it. It must still be there even if the program itself disappears.

Putting something in a file is a form of **output**. Getting something out of a file is **input**. This is because information is leaving from or arriving in the program. Programs can also get input from things like a keyboard or a mouse, and most commonly they output to a screen. Notice that writing something to a screen is not like writing it on paper. The screen is not persistent, paper is. The aim of outputting to a screen is not to do with **storage** but just presentation so a human can see the information. When outputting to a file, the aim is long term storage.

A **file** can be thought of as a special kind of value or object that has an existence outside the program as well as in it. In a program a file can be stored in a variable, so that it can be passed around. However, if we think of a variable as a box to hold something, a file variable is like a box with a hole in it. Things that are stored in the box end up somewhere else: in the file. The file itself is not stored in the box, the box just provides a link to it. The file is elsewhere, outside the confines of the program. The file itself has a name that it is known as in the outside world that is different to the name of the variable that is linked to it in the program. The variable's identifier (the name of the box) is just the temporary name used in the program. The file name is the name used in the real world that is the permanent name of that file.

This is all a little like the wardrobe in "*The Lion the Witch and the Wardrobe*". A normal wardrobe contains clothes, just as a normal variable is used to store things like numbers, but that special wardrobe contained a link to a different world. Step into the wardrobe and you would step out the back into the magical world of "*Narnia*". Similarly when you put something into a file variable it escapes the confines of the program and into the wider world outside. As we said the variable has a name associated with it, but that may be different to the name of the associated file in the outside world. Similarly everyone called the piece of furniture in the side room "The wardrobe", but the name of the thing it was connected to had a different name "Narnia". The name of a variable may be different to the name of the file that it is connected to.

Buffered Input-Output
Input-output (whether to files, the screen, keyboard or otherwise) is often **buffered.** What this means essentially is input-output requests are stored and then a whole bunch of them done in one go. Libraries often do this. Small village libraries have all the books on the open shelves, so if you want to take a book out of the library (output), then you just go and find the book, take it off of the shelf and check it out at the desk. This is not buffered output – the single request for a book is done on its own. However, if you want to take several books out at once, you do not go and get one, get it stamped, then go back and find the next one, get it stamped and so on. Instead you go into the library, find all the books that you wish to take out, and take them all to the library desk at once, and get them all stamped in one go. You are buffering all the requests. As you collect the books you tuck them under your arm until eventually you have a stack of them. Your arm is acting as an **output buffer**: a place where the books are collected until you can process them all in one go. Why do you do it this way rather than one at a time? It is basically because it is quicker and more efficient. Finding a book is slow and involves wandering round the shelves, but by getting several at once, lots of time and legwork can be saved.

When you bring a book back, you hand it over to the librarian at the desk. He or she does not then immediately go and put it back on the shelf where it lives. Instead they will probably have a temporary "returned book" shelf which holds all the books brought back that day. At the end of the day, or if it becomes full, the librarian puts all those books back in their correct places at once. The "returned book shelf" acts as an **input buffer**. It is used to gather together all the books so they can be returned efficiently in one go. Of course there is a trade-off between the size of the shelf and how often the librarian has to do a tour round the library taking books back. The bigger the shelf, the less frequently the librarian has to do the rounds but the more books are unavailable for other people to take out (or at least they are harder to find).

Big libraries that are used as reference libraries have books that are in areas deep in the library where only the librarians are allowed to go. The British Library in London is like this. Rather than going to get a book off the shelf yourself, you have to put in a request and a librarian goes and finds it. However, you have to wait before your request is processed. In the British Library items ordered take about 70 minutes to arrive (presumably meaning they get them every hour and take about 10 minutes fetching them). The librarian collects a whole series of requests from different people and gets them at the same time. The library is using an **output buffer** to process requests.

Most airports have long corridors linking the terminal to the departure gates, often with moving walkways to get people there quickly. This is like a non-buffered input-output system. Passengers leave the terminal as and when they like. Stansted airport uses a buffered system. You do not go to the gate when you decide, but instead wait for a tram. It fills up with a group of people who have all been waiting to go. They are all then transported together.

Computers use input and output buffers to buffer data that is to be written to output devices such as the screen or files. The data is collected in a buffer and then actually input or output all at once when the buffer is full. If we think of an output file variable as a box with a hole in it. Buffered file output is like having another big box under the hole. Anything dropped into the file variable box, falls through the hole and into the bigger box. There it waits as other things fall through the hole and join it. Eventually the box is full, at which point a trap door opens due to the weight and everything dropped into it, falls down a tube that is connected to the file – travelling down the tube in the same order it entered the box. Thus everything ends up in the file just as it would have done if it the pipe had been connected directly from the first box, the only difference is that it arrives there in groups.

Programming languages may also provide an instruction to empty the input or output buffer now – the equivalent of the head librarian telling a junior librarian to clear the returned book shelf now rather than wait until it is full. For example, when typing at a keyboard, the characters typed in are not processed until a return key is hit. They are stored in a buffer and shown on the screen in the meantime. The return key is used as an instruction to mean “clear the buffer and process the characters typed so far”.

In the British Library most items are stored on site and so take 70 minutes to arrive. Others are stored off-site and so take longer – up to 2 days. For off site books, the buffers are emptied, and so the requests processed, less frequently. This also demonstrates the point that things stored in some places can be accessed more quickly.

In some programming languages, the buffering is invisible. In others, such as Java, it is explicit, instructions exist for separately creating a file and creating a buffer that is then connected to it.