



So what is computer programming all about?

- A program is a sequence of instructions that the computer understands, and contains Iterations, Conditions, I/O and Data Manipulation. It is written in a High Level Language, not binary (0100101101...). The language converts the 'English like' instructions into machine code binary for the computer. There are essentially two 'flavours' compiled and interpreted.
 - Compiled takes the hand written instructions and converts them into a binary executable image (an .exe file). FAST!
 - Interpreted needs a program to read each line, convert it into binary, ruin it and then get the next line of code. SLOW ☹

Example overleaf has been written in a language called Perl (The Swiss Army Chainsaw of Computer languages, because it runs on almost every computer) it's an Interpreted language

The Problem to solve...



Export email VCARDS from iCloud to MS Outlook

The problem is that iCloud dumps all of them, and Outlook wants one at a time. So we have to break up a huge file into individual VCARDS and import them separately.

The VCARD

FROM iCloud

TO MS Outlook ...

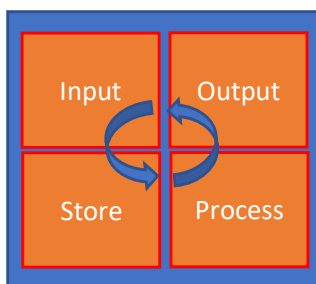
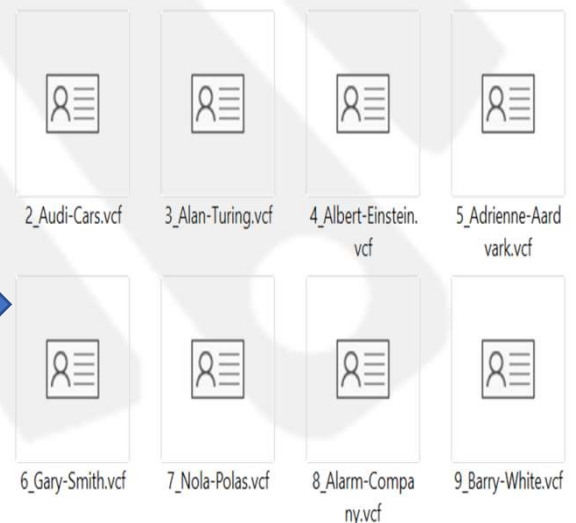
```

BEGIN:VCARD
VERSION:3.0
PRODID:--APPLE INC.//IOS 13.6//EN
FN:ALAN KEVIN;;;
ORG:GOLF,
EMAIL,TYPE=INTERNET,TYPE=HOME,TYPE=PREF:KEVIN@HOTMAIL.CO
TEL,TYPE=PHONE,TYPE=CELL,TYPE=VOICE:07930123456
REV:2020-07-26T13:09:43Z
END:VCARD

BEGIN:VCARD
VERSION:3.0
PRODID:--APPLE INC.//IOS 14.8//EN
FN:ALAN ACCIDENT,AUDI...
TEL,TYPE=CELL,TYPE=VOICE,TYPE=PREF:08458000000
REV:2020-07-21T10:22:29Z
END:VCARD

BEGIN:VCARD
VERSION:3.0
PRODID:--APPLE INC.//IOS 14.8//EN
FN:ALAN
EMAIL,TYPE=INTERNET,TYPE=HOME,TYPE=PREF:ALAN@HOTMAIL.CO
TEL,TYPE=PHONE,TYPE=CELL,TYPE=VOICE:0788712345
TEL,TYPE=HOME,TYPE=VOICE:0160454321
REV:2021-10-29T15:04:47Z
END:VCARD

BEGIN:VCARD
VERSION:3.0
PRODID:--APPLE INC.//IOS 13.6//EN
FN:A.CAROL...
ORG:COMPUTER PALS,
EMAIL,TYPE=INTERNET,TYPE=HOME,TYPE=PREF:CAROL@VODAMAIL.COM
TEL,TYPE=CELL,TYPE=VOICE,TYPE=PREF:+27726012345
TEL,TYPE=HOME,TYPE=VOICE:+27283122257
ITEM1.ADR,TYPE=HOME,TYPE=PREF:::331N/MAIN
RD,LONDON::W1A1AA;
ITEM1.X-ABADR.ZA
REV:2020-07-21T10:22:28Z
END:VCARD
    
```



The Program



ComputerPals
Wot Iz
KNOWLEDGE

Input file from iCloud



```
BEGIN:VCARD
VERSION:3.0
PRODID:-//APPLE INC//iCloud
N:CAROL;
FN:CAROL AT
ORG:COMPUTERPAIS;
EMAIL;TYPE=INTERNET;
TEL;TYPE=CELL;TYPE=VOICE;
REV:2020-07-21T10:22:2Z
END:VCARD
```

PERL Program (in Blue) to chop up the iCloud data for Outlook

Programming is just made up from **LOOPS**, **IF...THEN** decisions, **Reading** and **Writing** ☺

\$filename = 'iCloud.infile';

The output from icloud is called iCloud.infile, it contains 176 entries, which we need to break it into 176 VCARDS for Microsoft Outlook. The file is a TEXT file, so we need to manipulate the data to work with Outlook

if (open(my \$infile,<',\$filename)) { ← NOTE: Curley {}s indicate top and bottom of a LOOP.

Open the file or if you can't (go to the bottom [the other }]) and send a warning#) into a variable (the '\$' symbol identifies a container called a 'variable' to put stuff) in this case called 'infile' The end of each row contains hidden stuff; line feed, [LF] carriage return [CR] so we **chomp** it off. The variable 'row' contains the contents without the [CR][LF]

While something keep going

* while (my \$row = <\$infile) {

chomp \$row;

\$lump = \$lump . \$row . "\n";

Create a new variable called 'lump' and add the contents of the row into it, and add another [CR] (/n). This will slowly build into the 'new' outlook files contents.

We are looking to see if the row contains "FN:" (it will have the contacts Full Name in it. ie FN: Carol At)

Take another copy of 'row' call it 'modrow' we need that later Do clever stuff- replace all spaces with a '-' (this is called regex = REGular EXpression manipulation. Another dark art ☺

Calculate and put the length of the row into variable called 'clen' Extract users name from position 3 onwards (0,1,2,3) to the end of the row, using the substring command

The name row originally looked like this FN: Carol At = now it is just Carol-At ☺. Computing uses different 'types' of data A 'string' is a collection of letters, rather than a Number.

IF Something THEN

if (\$row =~ /FN\:/){

\$modrow = \$row;

\$modrow =~ s/\s+/-/g;

\$clen = length(\$modrow);

\$name=substr(\$modrow,3,\$clen);

IF Something THEN

if (\$row =~ /END:VCARD/){

\$Icounter++;

\$filename2 = \$Icounter . "_" . \$name . ".vcf";

Using format 'Number_First-Last.vcf'

So the new unique vcf's file name is like this: 2_Carol-At.vcf

Now we have a new file name **open** it for **writing** into

(the > symbol) and call it 'outfile' (See open for read above '<')

open(my \$outfile, '>',\$filename2) or die "Could not open file '\$filename' \$!";

Lets now fill this new file with the **rows** we collected in 'lump'

Print into 'outfile' the 'lump' (all the previous rows, with [CR][LF])

Now close the file, which saves it to disk

Empty lump to ""... to be filled again next loop ...

Read the program using these '{' inside which loops occur, this closing } bracket tells the program to go back to its the corresponding opening one (marked *) up at the top, and the sequence repeats until the very end of the file

print \$outfile \$lump . "\n";
close \$outfile;
\$lump = "";
}

2_Audi-Cars.vcf

} else #{

warn "Could not open file '\$filename' \$!";

* }

close \$infile;

Error handling if the file could not be opened

