A recipe for programming

How is a computer program like a recipe? **Paul Curzon** explains, and as a bonus, tells you how to cook a quick pasta dish.

Programmers are the master chefs of the computing world – except the recipes they invent don't just give us a nice meal, they change the way we live.

Programs are very similar to recipes. They both give instructions that, if followed, achieve something. There is a difference between them, though, and it has to do with language. When chefs invent recipes they write them out in human languages like English. Programmers write programs in special languages. Why's that? It's all about being precise enough to be sure exactly the same thing happens every time. Recipes are often ambiguous which is why when I follow one it sometimes goes wrong. Programs tie down every last detail.

Let's apply some ideas from programming languages to making meals. One of my favourite recipes is a hummus-based pasta dish (see box) so we'll use that.

Hummus and Tomato Pasta

Serves 2

This is a very quick 20-minute after work dish.

Olive oil
1 teaspoon of whole cumin seeds
1 large chopped onion
400g chopped plum tomatoes
200g hummus
150g pasta

1. Add the pasta to a large pan of boiling water. Simmer for 10 minutes.
2. Fry the cumin in the olive oil for a few minutes. Add the onions and fry gently.
3. Stir in the tomatoes and the hummus and leave to simmer for 5 minutes.
4. Drain the pasta and serve.

Structure it!

The first thing to notice about a recipe book is there is a clear structure. Each recipe is obviously separate from the others. Each has a title and a brief description of how it might be used. Each has an ingredients list and then a series of steps to follow. Programs follow a similar structure.

Cookery books use page layout to show their structure. Programmers use language: grammar, symbols and keywords. A keyword is a word that means something special. Once you have decided a word is a keyword you only ever use it for that purpose.

Let's invent a keyword RECIPE to mean we're starting a new recipe. The only time that word will appear in our recipes is to start a new recipe. What follows it will always be the name of the recipe. We will also need to know when the name ends. To make that clear we will use a special symbol made up of open and close brackets ().

We also want to be absolutely sure what is part of this recipe and what isn't. We will use curly brackets: everything between the brackets is part of the named recipe.

```
RECIPE Hummus and Tomato Pasta ()
{
  ...
}
```

No comment?

Recipes usually include a brief description that isn't part of the actual instructions. It is just there to help someone understand when you might use the recipe. Programs have descriptions like this too. Programmers call them 'comments'. Remove the comments and the recipe will still work. We need a clear way to show when a comment starts and ends. We will start them with a special symbol '/*' and end them with '*/'.

```
RECIPE Hummus and Tomato Pasta ()
{
  /*  Serves 2
      This is a very quick 20-minute after
      work dish.
  */
  ...
}
```

Variable storage

What comes next in a recipe is usually a list of ingredients. The idea is to list everything you need so you can have it all ready before you start. I often have a problem following recipes, though, as they don't list absolutely everything. Mid-recipe I might suddenly find I need a frying pan…when mine is crusted with burnt cheese sauce from last night! To avoid that, let's list all the pans we need too. For our recipe we need a frying pan and a saucepan.

Something used to store things (like pans do) in a program is called a 'variable'. Program variables hold things like numbers. The equivalent of the ingredients list 'declares' the variables. Declarations give each variable a unique name used to refer to it and also give each a 'type' – is it a saucepan or a frying pan we need? To be clear about when a declaration ends we add in some punctuation. Programming languages tend to use a semicolon for that – it's a bit like a full stop in English.

```
Saucepan    pan1;
Fryingpan   pan2;
```

This says that in the rest of the recipe when we say pan1 (the variable name) we mean a particular pan: a saucepan (its type). When we say pan2 we mean a particular frying pan.

New assignment

We will make a distinction between things to hold stuff, like pans (variables) and the actual ingredients that go in them: 'values'. We will also follow the TV chefs and start by setting out all the ingredients in little dishes at the start so they are at hand – and make that part of the instructions.

We will need to declare a dish to hold each ingredient, giving its type and giving the dish a name. At the same time we will say what should be put in it before the recipe proper is started. We will use an '=' symbol to mean put something in a variable (i.e., dish or pan). In programs, this action of putting something in a variable is called 'assignment'. So, for example, we will declare that we need a dish to hold the hummus (called hummusDish). We assign 200g of hummus to it.

```
Dish hummusDish = 200g hummus;
```

We are now ready for the recipe proper. We can use assignment as a precise way of moving things from one place to another too. So if we say, for example:

```
pan2  = oilDish;
```

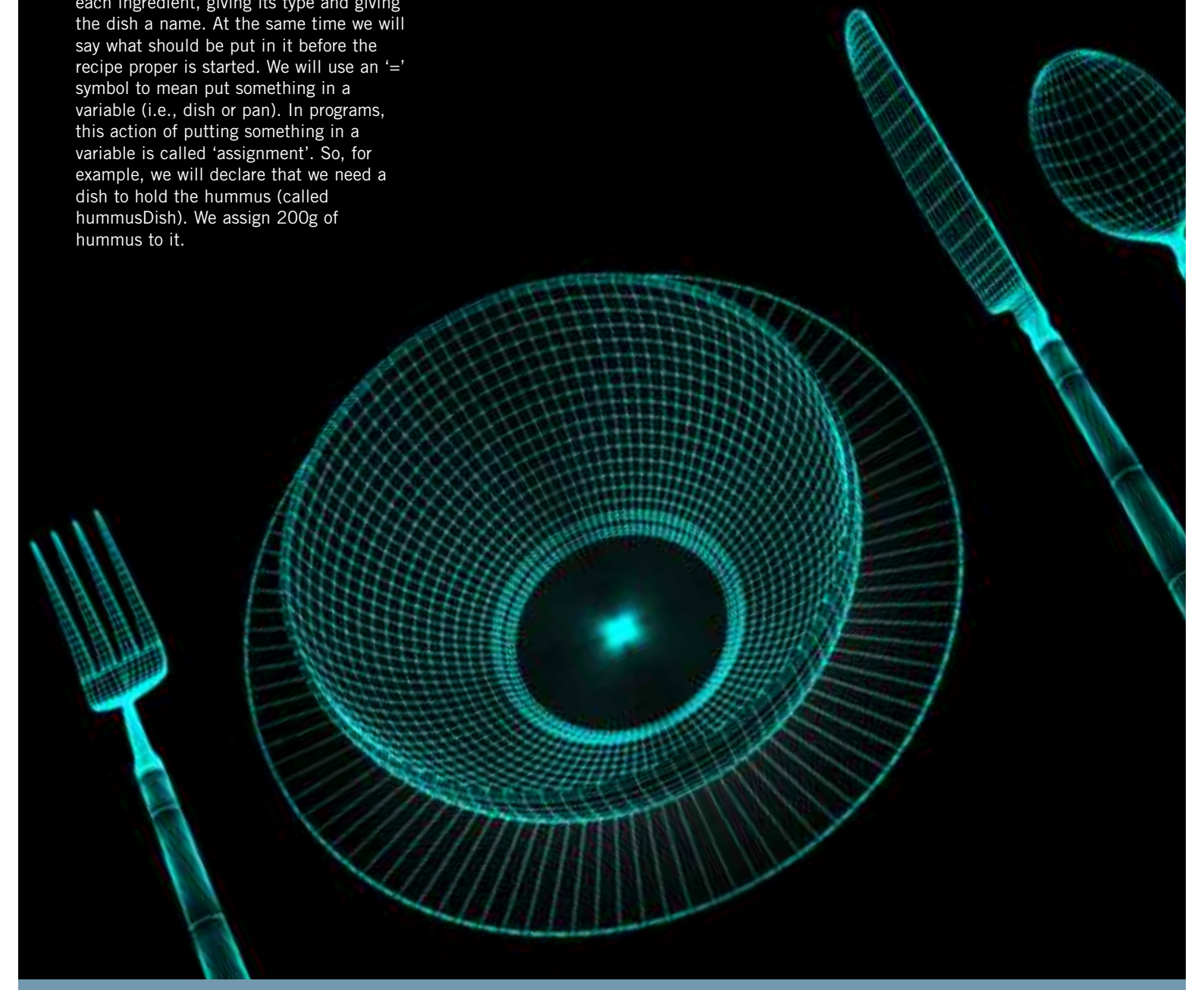
We mean empty the contents of the dish of oil into the frying pan. Programs are slightly different here, as when they do an assignment they don't move things from one place to the other, they copy it. That would be like having a dish that automatically refilled itself whenever it was emptied.

Assignment does NOT move things around, it makes new copies

Often we want to add to whatever is already in a pan. Programmers leave nothing to doubt and say explicitly that is what they mean:

```
pan2  = pan2 + onionDish;
```

This tells us to mix what is in the onion dish with what is in the frying pan, and then leave the result in the frying pan. We will use the + symbol to mean add together and stir. *(continued on the next page)*



A recipe for programming

(continued from page 15)

Methods in my madness

So far all we've done is put ingredients in things and copied them around. To make a meal we need to do various basic cooking things like heat a pan or drain a pan. Rather than spell out every step of how you do that in every recipe, we will use a short hand. We create mini-recipes that say how to do it and just refer to them by name. They are often called 'methods' by programmers. Each is written out just like our recipe. In fact to a programmer our recipe is a method too. When we want to use it we just give its name followed by any extra information needed. For example to heat a pan, we need to know which pan, how high a heat and for how long. We write, for example:

```
Heat (pan1, medium, 12 minutes);
```

This format helps make sure we don't miss something (like the time for example). We need similar methods for draining a pan and serving the meal. We won't give the actual instructions here. In a full program they would be written down step-by-step too and not left to chance.

Time to do it right

We have come up with a language for recipes similar to the ones used for programming. We've used symbols, keywords and very precise punctuation – the language's 'syntax' – to help us be precise. On its own that's not enough – each part of the language has to have a very clear meaning too – the language's 'semantics'. Together they make sure in following a recipe we know exactly what each step involves. There is then less scope for a cook (or computer) to get it wrong. Computers, of course, have no intelligence of their own. All they can do is exactly follow the instructions someone wrote for them (a bit like me cooking).

Here's what our complete recipe looks like as a program.

```
RECIPE Hummus and Tomato Pasta ()
{
   /* Serves 2. This is a very quick after work dish.
      It only takes about 20 minutes from start to finish. */
                        Saucepan  pan1;
                        Fryingpan pan2;

   /* Ingredients */     Dish oilDish = 1 tablespoon of olive oil;
                        Dish cuminDish = 1 teaspoon of whole cumin seeds;

                        Dish onionDish =  1 large onion, chopped;
                        Dish tomatoDish = 400g chopped plum tomatoes;
                        Dish hummusDish = 200g hummus;

                        Dish pastaDish =  150g pasta;
                        Kettle kettle = 500ml boiling water;

   /* Cook the pasta */  pan1 = kettle + pastaDish;
                        Heat (pan1, high, 2 minutes);
                        Heat (pan1, medium, 10 minutes);

   /* Make the sauce */  pan2 = oilDish + cuminDish;
                        Heat (pan2, high, 2 minutes);

                        pan2  = pan2 + onionDish;
                        Heat (pan2, medium, 5 minutes);

                        pan2  = pan2 + tomatoDish + hummusDish;
                        Heat (pan2, low, 5 minutes);

   /* serve */          Drain (pan1);
                        Serve (pan1, pan2);
}
```