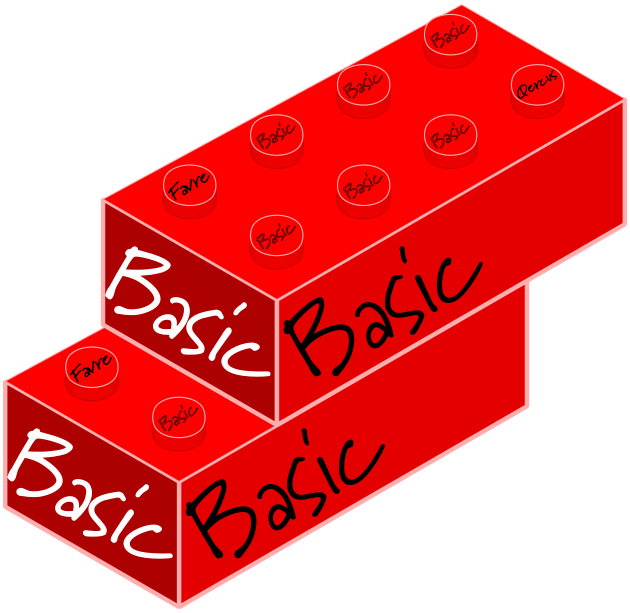

Ray Favre adds a second building blocks in this series on Basic BASIC.

2: Procedures and Functions

Last time we went into some detail on how a Basic program starts life; and left you with some homework. So let's have a look at the answers first:

What happens if you assign a real number to an integer variable?
The number actually assigned is just the integer part of the number eg integer%=6.58 results in integer% holding the value 6. No 'rounding' up takes place.

What happens if you assign an integer number to a real variable?
No change. You can do this.

What is the effect of the brackets in Line 120?
They force the order in which the sub-calculations take place. In this particular case the answer to the sum (73) would be the same with or without the brackets. But the important practical question is whether - without any brackets - you would expect to get the answer 73? Or, putting the question another way, are you confident that you know all Basic's 'precedent rules' for sums like this? If not, then use pairs of brackets freely to ensure you get what you mean to get.

O.K. Let's move on.

This time we are going to introduce Basic 'Procedures' and 'Functions'.

(It is an interesting bit of trivia that in non-Wimp programs PRINT plays a big part and is possibly the most frequently used keyword. Conversely, in Wimp programs, you rarely use PRINT at all. For instance, in the popular Dr Wimp library - comprising over 250 functions - there is only 1 use of PRINT.)

Procedures and Functions

If we refer back to our Prog1a from last time, you'll recall that its complete listing was as shown below left.
Now look at this alternative—2a:

Prog 1a
This is the program from last time

```
10 REM> Prog1a
20 :
30 :PRINT "First Try"
40 :
50 String$ = "Second Try"
60 PRINT String$
70 :
80 Integer% = 8
90 Real = 12.5
100 PRINT Integer% * Real
110 :
120 PRINT 60 + ( 2 * 9 )
- ( 15 / 3 )
130 :
140 END
```

Prog 2a

The program on the right is the new alternative program discussed on the following page.

```
10 REM> Prog2a
20 :
30 String$ = "First Try"
40 PROCprintstring(String$)
50 :
60 PROCprintstring("Second Try")
70 :
80 Integer% = 8
90 Real = 12.5
100 PRINT FNmultiply(Integer%,Real)
110 :
120 A% = 60
130 B = 2 * 9
140 C = 15/3
150 PROCprintaddandminus(A%,B,C)
160 :
170 END
180 REM** End of program run **
190 :
200 DEF PROCprintstring(anystring$)
210 REM** PROC definition. A PROC
carries out some actions. **
220 PRINT anystring$
230 ENDPROC
240 :
250 DEF
FNmultiply(firstnumber,secondnumber)
260 REM** FN definition  An FN carries
out some actions and 'returns' a value.
**
270 =firstnumber*secondnumber
280 :
290 DEF
PROCprintaddandminus(startwith%,add,take
away)
300 PRINT startwith% + add - takeaway
310 ENDPROC
```

Procedures

If you type and run Prog2a it will do exactly the same as Prog1a.

So, what has been done and why is it better?

What we have done is to extract some actions from the main flow of the program and re-located them in custom-built **'Procedures'** and **'Functions'**. In Basic the definition of a particular Procedure is held within a structure commencing with the two keywords **DEF** and **PROC** and ending with the keyword **ENDPROC**. Each procedure is defined by a name, and specific values can be sent to the procedure by attaching **'parameters'** to the definition.

Thus, at Lines 200-230 above the Procedure `PROCprintstring` is defined and we have decided that it shall have one parameter—namely, `anystring$`–held in a pair of brackets immediately following the name. The programmer has a free choice over the number and types of parameter to be used in any PROC—including no parameters at all.

When we want the action defined by a DEF PROC to take place we simply 'call' the PROC—as in Line 30. There is no limit to the number of times we can call a PROC and here we have done it a second time at Line 60.

When we call a PROC we must ensure that the call includes the right number and type of parameter values. With `PROCprintstring` there is only one parameter and it is a string variable—so each call we make to `PROCprintstring` must include a string as a parameter. At Line 30 we have used the already declared string variable `String$` as the parameter—whereas, at Line 60, we have used a direct string as the parameter. Either is acceptable (although, generally speaking, the form at Line 30 is better practice).

`PROCprintstring` is a trivial example: its action is merely to PRINT whatever string is sent to it by the call "First Try" at Line 30 (via String$) and "Second Try" at Line 60. Most PROCs define more complicated actions than this.

Functions

Complementing PROCs are **'Functions'** or FNs. This time the definition is held in a construction which starts with the Basic keywords **DEF** and **FN**, followed by a name and any parameters as before. However, the end of a Function definition is signified by a *sometimes-confusing-to-beginners* Basic statement beginning with the = character. Lines 250-270 show the structure of a Function definition—repeated below:

```
250 DEF
FNmultiply(firstnumber,secondnumber)

260 REM** FN definition  An FN
carries out some actions and
'returns' a value. **

270 =firstnumber*secondnumber
```

Function defined

Note that line 250 is typed in as one line from the number 250 DEF through to the closing bracket following the word secondnumber

FNs are called in a different way to PROCs. Line 100 shows the difference: here we have a statement which requires the program to PRINT something—and that something is whatever FNmultiply 'returns'. We could have equally used:

```
Integer% = 8
Real = 12.5
result=FNmultiply(Integer
%,Real)
PRINT result
```

Either way, a Function always 'returns' a value—which might be a real or integer number or a string: it is the programmer's decision.

Accordingly, the call to the FN can be made in any way that a direct value (of the same type as the return value) could be used. In the listing at Line 100 we have substituted the FN for a value to acted on directly by PRINT—and in the alternative shown just above we have substituted the FN for a value assigned to a variable. (Had the FN returned a string, then it could be used as a direct substitute in any Basic statement where a string could legitimately appear.)

Our final replacement action is in `PROCprintaddandminus(startwith%,add,takeaway)` which deliberately demonstrates that different types of parameters can be used at will: here, one integer number and two real numbers—and we could have included string parameters as well if required.

Note that when more than one parameter is used in a DEF PROC/FN they are all held within just the one pair of brackets and each is separated by a comma. If there were no parameters then there would be no bracket at the end of the PROC/FN name.

It is also worth mentioning at this stage that a DEF PROC/FN can call other defined PROC/FNs—indeed this is usually to be encouraged, see below.

PROCs & FNs in practice

So what has this done for use (apart from making the listing longer!)?

Firstly, please bear in mind that in real life a DEF PROC/FN is unlikely to be as trivial as in our listing. For instance, a common FN might take (as input parameters) a real number and an integer number of decimal places and return, maybe as a string, the number formatted to the specified number of decimal places. Or a PROC might take x/y positions and a colour as input parameters and plot a filled square of a certain size in the specified colour at the specified position.

Whatever their defined actions, the important point is that the programmer only has to design the

Qercus contains a mix of tutorials, reviews, comment and news.

This issue has articles on AppBASIC, Genealogy, RISC OS update, ArtWorks as CAD, A9 preview, Printers, Graphics File Examination, StarInfo, Pluto, Developers Corner, and much more.

definition once and then it can be used repeatedly throughout the program. Generally, therefore, PROC/FNs reduce the size of a listing significantly (despite what our trivial example might suggest). And don't forget, a DEF PROC/FN lovingly crafted in one program is then capable of being lifted into another without having to 're-invent the wheel'.

Just as importantly, PROC/FNs simplify and improve the structure of the program. The main flow of the listing is easier to follow because it is not interrupted by too much detail and it is broken up into logical sections held in the DEF PROC/FNs—which may themselves call other PROC/FNs.

Also, by breaking the program up into smaller lumps, if an action needs modification it is usually much easier and safer for the programmer to address this when the action is limited and confined to a DEF PROC/FN.

The naming of the PROC/FNs should not be considered as unimportant. Names ought to tell you, the programmer, broadly what the PROC/FN is doing—as in Prog2a. It is best to adopt a naming policy early in your programming 'career'. Some people like to use names like
`PROCprintAddAndMinus`
ie using case changes to make it easier to see the meaning. Others prefer variations of
`PROC_print_add_and_minus`
ie using the underscore character to make the name easier to read. There is no need to fear long names: there are many utilities you can use at the end of your programming to reduce the name lengths automatically if required. The important thing when writing a program is that it should remain very easy for you to read and understand—particularly when you come back to it after a few months.

Starting Basic was introduced in issue 271 of Qercus and continues monthly from this issue.

Cheapo

a RiscOS short story from Harriet Bazley

We don't think the following story is connected to a real Qercus competition or real RISC OS computers...

...but short-story writers have been known to stumble across the truth...

Hi. Phil Bexter. I'm not one of your Risc-nut subscribers, but I saw your competition – **"Most unusual power source..."** and reckoned I'd write in. Anyone can enter—right?

I'll just bet some clever-dick already sent in a piece about his new bio-potential rig. 'Harness the Power Differential Across the Body' and all that bio-pot malarkey. Oh, it works—if you don't mind strolling around toffed up from head to toe in a monkey-suit with wires in. Not much good when all you want is to strip off to your skinnies and sun-cream with your girl and a bit of music, now, is it?

Anyhow, I got a story as'll top that, and anything else you can come up with.

See, some hard-sell merchant pushed off one of your el-cheapo Risc machines on me – fits in your pocket, won't fry your balls, goes for ever on one battery. So he said.

Course, it was only when I got it home I worked out as it wouldn't run the *Mini-Windows* kit off my old metalware, either. The Boss—that's Ellie, my partner, only her old man's some kind of high-up Korean diplomat, and her family reckon they're way above my level – well, the Boss, who's the real chiphead out of us two, reckoned maybe that's a plus not a minus and gave me this anti-Windows lecture. But all I wanted was to play my MGT files and run *ZigZagPro* off the *MyNetwork* server, so I wasn't that thrilled.

And it turns out you can't short out a *Compu-cab* with that little battery, either. Well, how was I to know? On all my mates' gear, you can fry an egg with the charge—but there we were, trying to fiddle the cab meter to get home, and the battery just went dead on me, with the cab's little electronic brain screaming for help across the network.

So we legged it. Straight across the cemetery. Well, we had to—wouldn't you?

Only about halfway across, there was this beep as an e-mail came in, and I found el-cheapo Digital Assistant was still working. With no battery.

Ellie said it was residual charge, but it kept going right out the front gates. Then it went dead, like it should. But going down Church Lane, just before we got to the corner, it picked up again. Guess where we were? right opposite the graveyard. Start to see a pattern here?

Ellie was proper spooked, and started on about 'the ancestors'. Couldn't be my ancestors, I told her, and sure as chop-suey they weren't hers. It was when she let me get away with the Chinese crack I knew she was serious.

So we did some tests, next day and the day after. After all, five years back, bio-pot was all pie-in-the-sky – right?

The Boss talks about leylines and nitrogen decay. Me, I don't give a toss. I just say I got the only D.A. that takes so little power it runs on ectoplasm....