SPIGOT – Producing Digits of π one at a time

Abstract

SPIGOT is a proof-of-concept BASIC program written in 1996 for the SHARP PC-1350/1360 pocket computers and compatibles to produce digits of π one at a time using a so-called spigot algorithm.

Keywords: *spigot algorithm, Pi, π, one digit at a time, SHARP PC-1350/1360, pocket computer, BASIC*

1. Introduction

SPIGOT is a very short *(7 lines, 279 bytes) BASIC* program that I wrote in 1996 for the *SHARP PC-1350/1360* pocket computers and compatibles as a proof-of-concept example of implementing a *spigot algorithm* to produce digits of π one at a time. It will also run in most any *BASIC* version with minimal changes (see ***Note 2*** below).

The algorithm produces N of digits of π one by one using just integer arithmetic on reasonably small integer values. None of the previous digits are needed once computed and no floating-point operations are needed at all whether single-precision or high-precision.

On the other hand, the number N of digits needs to be specified in advance (so it's not possible to add more digits to the previously produced ones without having to restart the whole computation) and an array with some 10N/3 integer elements must be dimensioned to generate N digits, which is significantly more memory than needed by other multiprecision algorithms and also much slower. Thus, this algorithm and the resulting program featured here aren't intended to be competitive and are best considered as a proof-of-concept example.

The spigot algorithm is explained in detail in the reference given below. Basically, we start from the series:

$$\frac{\pi}{2} = \sum_{i=0}^{\infty} \frac{i!}{(2i+1)!!} = 1 + \frac{1}{3} + \frac{1.2}{3.5} + \frac{1.2.3}{3.5.7} + \ \ldots = 1 + \frac{1}{3}\left(1 + \frac{2}{5}\left(1 + \frac{3}{7}\left(1 + \frac{4}{9}\,(1 + \cdots)\right)\right)\right)$$

which can be considered a *mixed-radix base* (1/3, 2/5, 3/7, 4/9, ...) representation for π/2, so the digits of π itself in this base would all be **2**. The workings of the algorithm can be seen in the table below, taken from the reference. The array is initialized with the digits of π in the mixed-radix base (all 2) and then the operations described in the leftmost column are performed, starting from the rightmost column and going left, column by column and row by row, from top to bottom, producing one digit of π per row (**3**, **1**, **4**, **1**). We needed [10*__4__/3] = 13 columns (array elements) to get **4** digits.

	Digits of π	1/3	2/5	3/7	4/9	5/11	6/13	7/15	8/17	9/19	10/21	11/23	12/25
Initialize		2	2	2	2	2	2	2	2	2	2	2	2
× 10		20	20	20	20	20	20	20	20	20	20	20	20
Carry	3	+10	+12	+12	+12	+10	+12	+7	+8	+9	+0	+0	+0	=
Remainders		30	32	32	32	30	32	27	28	29	20	20	20	20
		0	2	2	4	3	10	1	13	12	1	20	20	20
× 10		0	20	20	40	30	100	10	130	120	10	200	200	200
Carry	1	+13	+20	+33	+40	+65	+48	+98	+88	+72	+150	+132	+96	=
Remainders		13	40	53	80	95	148	108	218	192	160	332	296	200
		3	1	3	3	5	5	4	8	5	8	17	20	0
× 10		30	10	30	30	50	50	40	80	50	80	170	200	0
Carry	4	+11	+24	+30	+40	+40	+42	+63	+64	+90	+120	+88	+0	=
Remainders		41	34	60	70	90	92	103	144	140	200	258	200	0
		1	1	0	0	0	4	12	9	4	10	6	16	0
× 10		10	10	0	0	0	40	120	90	40	100	60	160	0
Carry	1	+4	+2	+9	+24	+55	+84	+63	+48	+72	+60	+66	+0	=
		14	12	9	24	55	124	183	138	112	160	126	160	0

2. Program Listing

```
1: "A" CLEAR: INPUT N: L=INT(10*N/3): DIM A(255): Z$="000000",T$="999999": WAIT 0
2: FOR I=1 TO L: A(I)=2: NEXT I: M=0, P=0: FOR J=1 TO N: Q=0, K=2*L+1
3: FOR I=L TO 1 STEP -1: K=K-2, X=10*A(I)+Q*I, Q=INT(X/K), A(I)=X-Q*K: NEXT I
4: Y =INT(Q/10), A(1)=Q-10*Y, Q=Y: IF Q=9 THEN LET M=M+1: GOTO 7
5: IF Q=10 THEN PRINT STR$(P+1);LEFT$(Z$,M);: P=0, M=0: GOTO 7
6: PRINT STR$ P;LEFT$(T$,M);: P=Q, M=0
7: NEXT J: PRINT STR$ P: BEEP 2
```

3. Usage Instructions

See the worked example to understand how to use the program.

4. Examples

The following example can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example

Produce the first ***24 digits*** of π, then the first ***76 digits***.

In **RUN** Mode, proceed as follows:

DEF **A** **?_** 24 **ENTER** *0* ***3141592653589793238462643*** *(all digits are Ok, ~7')*

DEF **A** **?_** 76 **ENTER** *0* ***3141592653589793238462643383279502884***
1971693993751058209749445923078164062862 *(all Ok, ~63')*

Notes

1. The *BASIC* version of the *SHARP PC-1350/1360* has a max. limit of **255** for indexing array elements, so the maximum number of digits produced is limited to [255*3/10] = **76 digits**. Other *BASIC* versions may not have this limitation.

2. This program will run in most any *BASIC* version by simply *removing SHARP*-specific statements: **"A"**, **CLEAR**, and **WAIT**. Also, the **DIM A(255)** may be generalized to **DIM(L)** if the *BASIC* version allows for dinamic (re)dimensioning. Further, all variables used in the program (including the **A** array) can be declared as *integer* for maximum speed.

3. Any digit-producing algorithm for normal numbers (which π probably is) has the (unlikely) problem that the last digit(s) may be incorrect if there's a terminating string of 9's, and even with no 9s the very last digit might be wrong. See 50d below.

4. Some timings *(SHARP PC-1350):* 20d in 4'30" *(last 5: 32384)*, 40d in 18' *(84197)*, 50d in 28' *(9375**0**)*, 76d in 63' *(06286)*

References

S. Rabinowitz and S. Wagonrancis (1995). *A Spigot Algorithm for the Digits of π.*
The American Mathematical Monthly, Vol. 102, No. 3

Copyrights