# あとがき

　2章においてプログラミングの神髄は

 i = i + 2

であると書きました。「変数」という値を蓄える箱を用意し、それを更新しながら作業を進めていくのが、プログラミング的発想です。

　数学とプログラミングは、一見似ているように見えます。数学の知識がプログラミングに必要なこともあるでしょう（例えば、sin, cos, tanを使うプログラムなど）。しかし、問題を解くためのアプローチ（発想）は全く異なると筆者は思います。たとえば、「1～nまでの和を求めよ」という問題に対して、

$$\frac{\left(1+n\right)n}{2}$$

という一気に解くためのエレガントな式を導出するのが数学的発想であり、

 sum = 0

 For i = 1 To n

 sum = sum + i

 Next i

のように、素直な手順をプログラムで表現するのが、プログラミング的発想です。

　微分方程式を解くときも同様です。例えば、

$$\frac{dy}{dx}=-2y$$

$$初期条件は x=0 のとき y=1$$

を解きなさい、という問題が与えられたとします。微分方程式を「解く」ことは「この式を満たす関数を求める」ことです。解析的に解いて、一般解

$$y=Ce^{-2x}$$

を導出し、初期条件より*C* = 1を求め、

$$y=e^{-2x}$$

を得るのが数学的発想です。プログラミング的発想は、図Aのようになります。



図A　プログラミング的発想

　x=0のときy=1なので、x=0における傾きdy/dxはy=1を代入して -2です。傾き-2でx=0からx=0.1まで関数を描きます。x=0.1におけるyの値は1-2×0.1=0.8です。x=1における傾きはdy/dx = -2yにy=0.8を代入して-2×0.8=-1.6です。傾き-1.6でxを0.1進めます。これを繰り返すことで、微分方程式を解いてゆきます。

　プログラミング的発想と数学的発想は全く異なります。プログラミング的発想に共通するのは、

変数を設定し、それを更新しながら作業を進める

という点です。エレガントなプログラムを組むには、

何を変数として設定するか

が大切です。ここで誤ってしまうと、どう頑張ってもエレガントなプログラムにはなりません。

　プログラミングは「習うものではない」という考え方もあります。筆者らの世代でプログラミングを「習った」人はいません。独学で色々な雑多な知識を吸収し、それの寄せ集めと、コーディングの実際に行って、身につけてきました。

　皆さんは、Excel VBAの最低限＋αの知識を本書で身につけました。あとは、本やWebがあれば独学可能であると思います。この本をきっかけとして、読者が楽しいプログラミングライフを送ることを願ってやみません。