よいプログラムを書くために

文章に分かりやすい（よい）文章と理解しづらい（悪い）文章があるように、プログラムにもよいプログラムと悪いプログラムがある。ここでは、よいプログラムを書くためのヒントを説明する。

よいプログラムって何だろう

よいプログラムとは、簡単にいうと、「目的を達成するプログラム」のことである。ただし、目的は人によって、またはそのときどきによって違ってくる。ここでは、次の2つの場合を考えてみる。

(a) 決められたプログラムを作る

授業で指示された課題のプログラムを作ったり、仕事で注文を受けてプログラムを作ったりする場合が相当する。「このようなプログラムを作ってください」という、はっきりした要求が与えられる。それを満たすプログラムを作れば、目的を達成したことになる。

(b) 自由に考えてプログラムを作る

授業で自由作品の課題プログラムを作ったり、自分で目的を決めてプログラムを作ったりする場合が相当する。どんなプログラムを作るかはある程度自由に決められる。自分で作りたいと思ったプログラムを作れれば、目的を達成したことになる。

よいプログラムの作り方

プログラムは自由に作ることができるが、いくつかのコツを知っていると、目的のプログラムを確実に素早く作ることができる。ここでは、その中でももっとも基本的なコツを選んで伝授していく。

（1）作るプログラムをイメージする

そもそもどんなプログラムを作ればよいかが分かっていないと、どこから作り始めてよいかも分からない。「完成したら、どのような画面になって、どのような動きをするのか」ということを考えてみよう。このとき、できるだけ具体的にイメージすることが大切である。頭の中で、そのプログラムが動く様子が生き生きとイメージできたり、動いている様子をノートなどに絵で書けるように考えてみよう。

（2）どんな機能が必要かを考える

最初は簡単に作れそうに思ったプログラムでも、書いているうちにどんどん長くなってしまうことがある。長くなってから全体を理解しようとしても、何十行もあると、自分が書いたプログラムなのに理解できない。

そこで、大まかに3〜5個くらいで、作りたいプログラムに必要な機能を考えてみるとよい。機能というのは、そのプログラムが行える動作のこと。最初は簡単でないかもしれないが、どのような順番でプログラムを作っていくかを考えることは重要である。

どのような機能を考えればよいかは、**part□game** で紹介したいくつかのゲームが参考になる。たとえ

ば、**ch□drive** の「宝物拾いゲーム」は、次の4個のステップで作られていた。そして、いちどに全体を作るのではなく、ステップ1から順に小さなプログラムのまとまりごとに作っていくことで、最終的にゲームを作り上げることができた。プログラムのまとまりの先頭には、何をする機能かを**コメント**（「／／」）で書いておくとよい。

- （ステップ1） タートルを操作する 最初は、画面の上に目に見えるオブジェクトを置くプログラムから作るのがよい。このゲームでは、画面に主役のタートルとボタンを作った。プログラムの結果が画面に現れるので、自分のプログラムが正しく動いていることを確認できる。
  - （ステップ2） タートルを前進させる
    - 次に、画面のオブジェクトに動きを付ける。ステップ2では、主役のタートルを前進させていた。ドライブゲームとして遊ぶことができる。
  - （ステップ3） 宝物を画面に置く
    - 続くステップでは、脇役のオブジェクトを登場させたり、衝突したときの動作を定義したりして、プログラム全体を完成させていく。ステップ3では、脇役である宝物のオブジェクトを画面に置いていた。
  - （ステップ4） 宝物を拾う
    - 最後のステップでは足りない機能を作り、プログラムを完成する。ステップ4では宝物とタートルが衝突したときの動作を定義した。

（3）1行ずつ実行しながら作っていく

プログラムがうまく動かないときに、長いプログラムの中でどこに原因があるのかを調べるのは簡単ではない。たとえば、いちどに10行のプログラムを入力して実行すると、何らかのエラーが表示されるか、思ったような動きをしてくれないことが多い。

しかし、「この行に問題がある」ということが分かれば、原因を調べることはそう難しくない。そこで、プログラミングに慣れるまでは、プログラムは基本的に1行入力するごとに実行して動作を確かめながら進めることが望ましい。

実行しながらプログラムを書いていくことは、プログラムが正しいことを確認できることに加えて、「少しずつ形になっている」ことを実感できるので、プログラムを作る作業を楽しくする効果がある。プログラミングは本来楽しい作業なので、それを早いうちに体験できると速く上達することにもつながる。

もし実行してエラーが表示された場合には、そのメッセージを手がかりに問題の箇所を探すことになる[1)]。こまめに実行して確認していれば、前回実行した部分までは正しく動いていたので、問題は直前に入力した数行にあることが分かる。必要に応じて、問題のありそうな行の先頭に「／／」を書いてコメントにすることで、その行にエラーがあるのかを調べることもできる。「この行までは動く」「この行を加えると動かない」という切り分けができれば、問題はずっと見つけやすくなる。

（4）目標を達成できたか確認しよう

エラーがなくなって動くようになったら、プログラムが完成したのかどうか、そして足りない部分があればどうすればよいかを考えよう。確認すべき点はプログラムごとに異なるが、次の点はほぼ共通に求められていると考えることができる。

```
*（目的を達成） 与えられた課題を達成しているか
 *作りたかったものができているかを確認する。
*（正しく動く） 実行すると正しく動くか
 *実行するとエラーになったり、途中で動きがおかしくなることはないか確認する。
*（読みやすい） 他の人が読んで理解できるか
 *一週間後の自分が読んでも分かるように、そして先生など他の人が読んでも理解できるように、プログラムを整理しておく。機能ごとにまとまりを作ったり、機能を説明するコメントを書いておくことは重要である。
```

自由課題の作品や、自分で目的を決めて作るプログラムでは、次の点も確認するとよい。

* （面白さがある） 見る人に楽しんでもらえるか
 * 他の人に見せたり使ってもらうプログラムでは重要になることがある。
* （独自の工夫点） 自分なりの工夫点があるか
 * 図形の形やボタンの配置など、ちょっとしたことでもいいので工夫した点があると達成感につながる。
* （画面デザイン） 見た目も大切
 * 他の人に見せたり使ってもらうプログラムでは、見栄えや使いやすいデザインも重要になる。せっかく作るのだから、使いやすくかっこいい作品に仕上げよう。

[1)]

エラーメッセージとデバッグについてはch□graphics のコラムを参照。