弱い動機のプログラム
―1 画面程度のプログラム―

吉川 宏之

はじめに

単純なプログラムを作成することで，作業を効率化できる場合がある．プログラムの作成に至らない理由として，「難しそう」，「できるかどうか分からない」など，計画にも上がらない場合も多い．反面，個人やグループ内で利用するために作られた，単純な機能のツールプログラムが便利に利用されているところもある．

新しい学習指導要領では、小学校では2020年度，中学校では2021年度、高校では2022年度から順次プログラミング教育が始まっている．内容は，身近な生活でコンピュータが活用されていることやコンピュータの働きを、生かそうとする態度を育むことを重視している．そのため，プログラミングは必修とされていない．

この教育課程を終えることで，「できるかどうか分からない」が「できる」と判断できたとき，そこからプロクラムの完成までにどの程度のコストが必要になるのかを実際のプログラム作成を例に示す．

1 対象とするプログラム

1.1 プログラムを作成する弱い動機

コンピュータを使って同様の作業を繰り返すとき，以下に示す(i)から順に進んでいくものと考える．

(i) 作業方法を見つける
(ii) 何度か繰り返すうちに手順を覚える
(iii) 操作を記録したマクロ，コマンドなどを組み合わせて簡略化
(iv) プログラムを作成して自動化する．

ここで，(ii)から(iii)へは，作業を記録する「マクロ」やコマンドなどの知識が，(iv)にはプログラムの作成が必要になる．

(iv)で作成するプログラムが

- 現状はマウス，キーボードの操作で行っている．
- あると手間が省略できて便利になる．
- 無くても何とかなっている．
- コストをかけて依頼してプログラムを作るまでのものではない．
- 分担するほどの作業量はない．

といったような「ちょっとしたプログラム」の場合，プログラム作成にかかる作業量（コスト）と，手作業で行う場合の作業量（コスト）の差が，作成するかどうかの大きな要因となる．

このような「ちょっとしたプログラム」では，多くの場合，利用者は作成自身もしくは少数のグループ．作成は分担するほどの量ではないため個人作成となる場合が多い．

1.2 作成作業・能力の分類

プログラムの作成過程で行われる作業，及び必要となる能力について，大きく(1)～(5)に分類して考える．

(1) 考え方，アルゴリズム
(2) プログラミング言語の知識
(3) プログラミングスキル・経験
(4) 開発環境・OSの知識
(5) ライブラリ等の追加機能

(1)については高校までのプログラミング教育で行われる基礎的な部分を含む，プログラミング言語などに依存しない基本的な考え方．(3)については，プログラミングの経験により得られるスキル・経験として分類する．

プログラムの作成コストの概略を図1に示す．図の左に，プログラム完成までに必要なものを表す．

1.1で述べたような場合，作成にかかる時間は作業者の経験･能力による違いが大きく影響する．

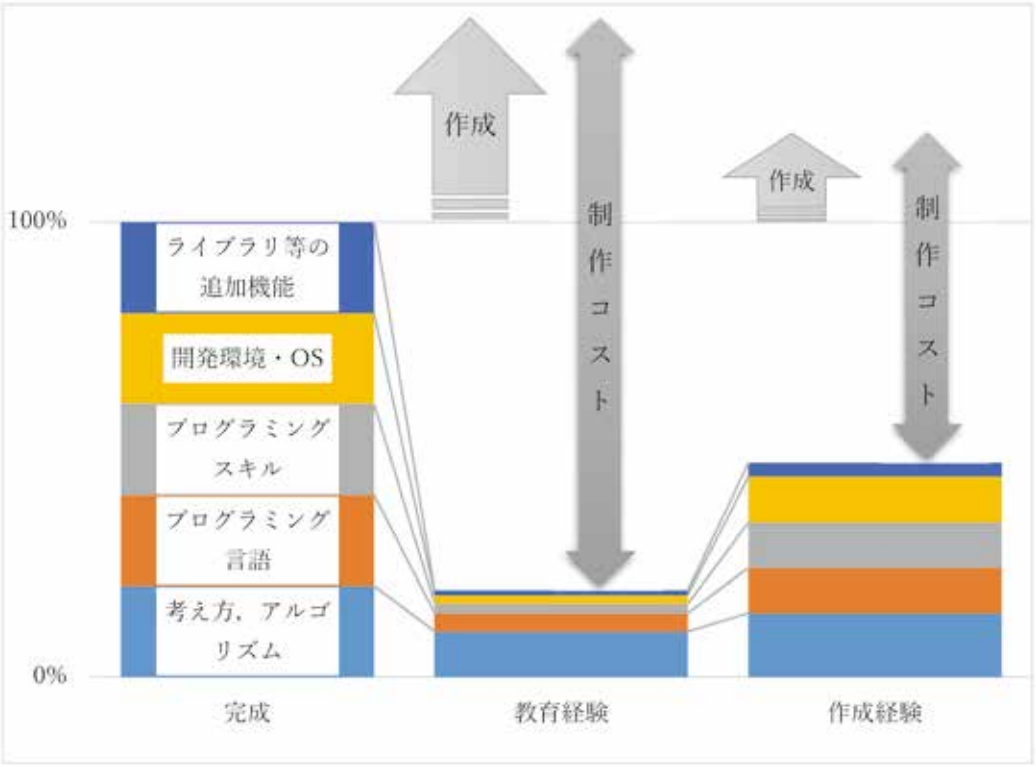

図 1　プログラムの作成コスト（概略）

2　作成例

複数の PDF ファイルを 1 つにまとめるプログラムを例とする．1 画面（ダイアログ 1 個）程度．作業の内容はプログラミング教育では取り上げられないものがほとんどとなる．

- 機能
  - 複数の PDF ファイルをまとめて，1 つのファイルとして出力．
- 操作
  - マウス，一部，キーボードによる操作．
- 基本動作
  - 入力元フォルダを指定．
  - フォルダ内の PDF ファイルを結合して出力．
- 開発・実行環境
  - PC のローカル環境で動作する．開発用の PC とは別の PC で使用する．
- PC 上で動作．スマートフォン，タブレットでの利用予定はなし．
- その他
  - 操作画面を表示する．
  - 操作の簡略化．
  - 指定項目を少なくする．
  - 設定内容の保存する．
  - 大規模な拡張予定はない．
  - 多くの機能を盛り込まない．

2.1　開発・実行環境

プログラミン教育では，プログラミング言語として Scratch，Java，JavaScript，Python などが使用されている．

プログラミング言語により，作成にかかる作業量が大きく変わってくる．例えば，PDF ファイルの操作を Scratch で実現することは，「不可能ではないが，かなり難しい」ものとなり，「あると便利」程度のプログラムの作成には見合わない．

2.1.1　プログラミング言語

言語選択の要因

- 要求される動作（PDF ファイルの結合）を行うことができる．（重要）
- 開発環境の構築が容易，または不要．
- 実行 PC へのインストールが容易，また

は不要.
- 実行 PC でプログラムの修正可能.（重要ではない）
- 他の人が修正する可能性もあるため，一般に使われているプログラミング言語.
- OSS などを利用し，費用はできるだけ抑える.

C++, Java, Javascript, PHP, Python を比較し，実行 PC での修正も可能なこと，速度を重視しないことから Python を選択した.

Python を選択した理由
- 標準インストーラーを利用すれば，動作環境の設定が容易.（開発環境，実行 PC の両方）
- ライブラリの追加は Python から行うことができる.（開発環境，実行 PC の両方）
- インタープリタのため，コマンドから動作確認を行うことができる.（開発環境）
- プログラムの簡単な修正ならばメモ帳でも可.（実行 PC）
- 処理速度は期待できない.（実行 PC）

2.1.2 開発環境の準備

Python のインストール

Anaconda など，Python 本体と Python でよく使うライブラリがひとまとめになったパッケージもあるが, 開発に使用する PC（開発環境）と，実際に利用する PC（実行環境）が異なるため，標準の Python を基本とすることとした.

2.2 プログラミング言語の記述以外に必要な知識

プログラミング教育では，あまり取り上げられない内容を下記に示す．プログラミング言語に追加される部分で，複数の方法があるため，プログラミング教育の内容としてはそぐわないが，プログラム作成では必要になることが多い.

・ユーザインターフェースの追加
・テキストボックス
・リストボックス
・チェックボックス
・ファイル操作
・フォルダ選択
・ファイル一覧取得
・ライブラリ（機能）の追加
・設定の保存，呼び出し

2.3 プログラムの画面

作成プログラムの画面を図 2 に示す.

当初の予定に無かったが，下記の機能を追加した.
- 連番（ページ番号）の付加.
- 見出し（ヘッダ）の付加.

下記の URL で公開している.

公開 URL

https://github.com/hyoshi-lab/mergePDF

図 2　シンプルな画面

3 作成したプログラムの公開

3.1 利用者により増える作業内容

プログラムの利用者が，個人での使用と，グループ内で使用する場合について，作業内容を比較する.

個人での利用

- ユーザインターフェース
  - 簡略化，省略することもできる.
- マニュアル.
  - あってもよい. メモで足りることもある.
  - プログラム内のコメント.

グループでの使用

- ユーザインターフェース
  - 何らかのものが必要.
- マニュアル.
  - 画面の説明.
  - 操作の説明.
- 保守
  - 要請により対応
- インストール作業が必要なこともある.

概略を図 3 に示す.

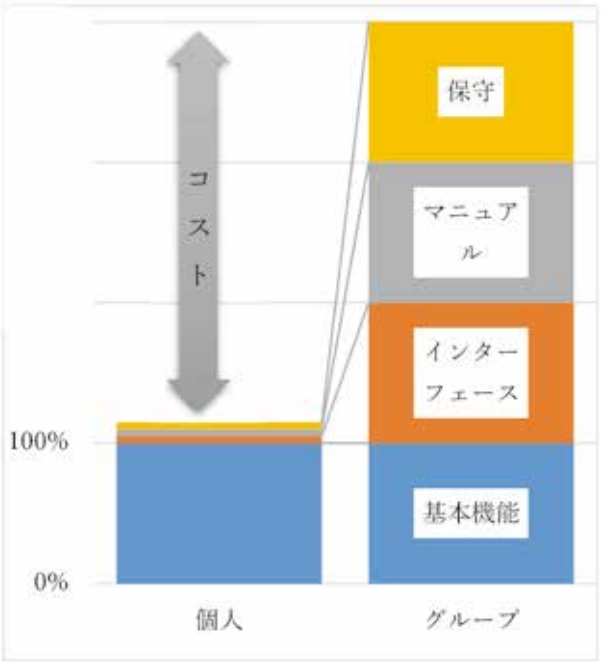

図 3　利用者による違い（概略）

3.2　作成したプログラムの公開

グループで利用するプログラムであれば（著作権関係を考えなければ），公開場所の確保と，ソースプログラムの整理，マニュアルの修正程度の追加作業となる.

公開しない理由としては，そもそも「公開するつもりで作っていない」「たいしたものではない」「使い捨てに近いもの」「はずかしい」などが考えられる.

「自分のメモ代わり」程度の軽い気持ちで公開できると，使う方も気軽に再利用，流用が進むのではないか.

例として，Scratch のギャラリーや，p5.js Editor, OpenProcessing などがあげられる．ソースプログラムが公開されており，改変して利用することができる．プログラミング教育で利用されることが多いことも付け加えておく.

4　まとめ

実際にプログラムを作成するために必要な内容を 1.2 で下記のように 5 つの項目に分類した.

(1) 考え方，アルゴリズム
(2) プログラミング言語の知識
(3) プログラミングスキル・経験
(4) 開発環境・OS の知識
(5) ライブラリ等の追加機能

この中で，プログラミング教育で取り上げられる部分は(1)が中心となり，一部が(2)のプログラミング言語を学ぶことになる．3 で例に示したように，プログラムを作成するときには(3)，(4)，(5)が必要になる．これらは，利用環境などの状況により変わる．プログラミング教育の内容としてはそぐわないが，プログラムの作成には必要となる部分である. ユーザインターフェースの作成方法などは，同一の言語でも実現方法が複数あり，修得しても別の場面では直接利用できないこともあり，避けては通れない部分であるが，教育内容としてはそぐわない. ブラウザ上で動作する JavaScript などであれば問題にならないのかもしれない.

ソースコード付きで公開され，そのまま利用，または改変して利用となるサイクルが出てくると，プログラム作成のコストが減少することが期待できる．これは，OOS（Open Source Sofware）の考え方である.

参考 URL

- https://www.python.org/ Python Software Foundation
- https://pythonhosted.org/PyPDF2/ PyPDF2 Documentation
- https://github.com/pymupdf/PyMuPDF PyMuPDF
- https://pypi.org/project/natsort/ natsort
- https://github.com/ GitHub
- https://scratch.mit.edu/ Scratch
- https://p5js.org/ p5.js
- https://openprocessing.org/ OpenProcessing
- https://code.visualstudio.com/ Visual Studio Code
- https://docs.microsoft.com/ja-jp/windows/win32/api/shlwapi/nf-shlwapi-strcmplogicalw Microsoft Docs『StrCmpLogicalW function（shlwapi.h)』
- https://ja.wikipedia.org/wiki/ 自然順 Wikipedia 『自然順』（2021 年 9 月閲覧）
- https://www.mext.go.jp/a_menu/shotou/zyouhou/index.htm 文部科学省 『教育の情報化の推進』
- https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf 文部科学省 『小学校プログラミング教育の手引（第三版)』
- https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r03/html/nd112490.html 総務省 令和 3 年情報通信白書 『デジタル・トランスフォーメーションにおける課題』
- https://www.soumu.go.jp/johotsusintokei//linkdata/r03_02_houkoku.pdf 総務省 情報流通行政局情報通信政策課情報通信経済室 株式会社 情報通信総合研究所 『デジタル・トランスフォーメーションによる経済へのインパクトに関する調査研究』
- https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1369603.htm 文部科学省 『教育の情報化の推進』