授業用プログラミング環境の開発

Development of the Programming Environment for Lessons

坂田 圭司[*1]
Keiji SAKATA[*1]
[*1] 東海大学情報教育センター
[*1]ICT Education Center, Tokai University
Email: kgsakata@tsc.u-tokai.ac.jp

あらまし：プログラミング学習の基礎段階においては経験値を高めるために，プログラミング言語の記述に注力して，なるべく多くのソースコードを記述することが求められる．既存の統合開発環境は操作が煩雑で習熟まで時間がかかりやすい．本研究では，上記の問題を解決し，授業に適したプログラミング環境を構築することを目的として，Markdown による文書記述機能を備えた Web 上のプログラミング環境を開発したので，その内容と学習効果について発表する．
キーワード：プログラミング学習，開発環境，授業支援システム，電子テキスト，Markdown

1. はじめに

プログラミング学習における重要な目標の一つは，処理の流れを言語で記述して，試行錯誤しながら動作するように作り上げることにある．

一般的なプログラミングの初学者にとって，プログラミング言語の修得と，複雑なソフトウェア開発環境の操作を習熟することを両立しなければならない状況は，学習意欲を低下させる原因の一つとなっている．

本研究では，上記の問題を解決し，授業での活用に適したプログラミング環境を構築することを第一の目標とした．

第二の目標として，電子テキストとプログラミング環境の融合を目指した．オンライン文書中にソースコード編集および実行環境を埋め込み，この文書記述に Markdown [(1)] を採用することで，授業中におけるライブコーディングに対応できるようにした．

2. プログラミング環境の検討

大学のプログラミング授業での開発環境では，Microsoft Visual Studio や Eclipse などの統合開発環境（IDE），またはテキストエディタとコマンドライン環境（CLI）が多く利用されている．これらの環境は，プログラミング初学者にとって，次の問題点を生じやすい．

・プロジェクト作成・管理
　：記述するまでの煩雑な作業
・編集，ビルド，実行手順
　：過剰な入力サポート，手順ミス
・ファイル管理
　：保存し忘れ，保存場所の指定ミス

これらの操作を習熟することは，研究や業務においては必須であるが，基礎レベルの学習では，アルゴリズムの理解や言語記述へ注力したい場合に，阻害要因となりうる．

本研究では，上記の問題点を解決して，授業利用に適したプログラミング環境を構築するために，次の機能を持たせることにした．

(1) Web サーバでのシステム運用
　・Git を用いて，マルチユーザ，ソースコード保管・履歴管理を可能
　・Atom エディタを元にした編集しやすい環境
(2) Web ベースのクライアント環境
　・インストール作業できない環境で利用可能
　・学習者がどこからでも利用可能

これらの機能によって授業担当者は，Web サーバ上にシステムを設置して，学習者に対して URL を伝えてユーザー登録させるだけで利用できるようになる．本プログラミング環境と，既存の IDE，CLI との機能比較を表 1 に示す．

表 1 プログラミング環境の比較

	IDE	CLI	本環境
プロジェクト管理	複雑	無し	簡単
コード編集	機能過剰	別途	あり
ビルド・実行環境	複雑	要入力	簡単
ファイル管理	手動	手動	自動
マルチユーザー	未対応	未対応	対応

3. 電子テキストの記述とソースコード提示

プログラミング授業において，従来はテキストを紙または電子的に提供して，それを見ながら解説を聞いてノートを取り，開発環境で実習を進めるスタイルがとられてきた（図 1）．

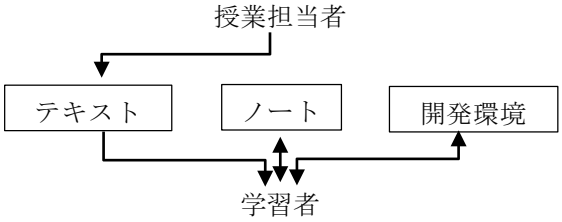

図 1　既存の授業形態

このスタイルでは,実際に作成したプログラムと,テキストおよびノートの内容は離れて存在しており,学習者と授業担当者のやりとりは,LMS など別の手段を必要とする.

本システムでは，電子テキストの記述に，軽量マークアップ言語 Markdown を用いることで，HTML よりも簡単な文法で書け，準リアルタイムでレンダリングした文書を表示可能になる.

```
# 最初の一歩
まず，c++言語で「Hello!」と画面表示するプログラムを作ってみましょう.
- 文字は，英数字モードで入力して下さい.
- 入力完了したら，*保存*，*コンパイル*，*実行*の順番で作業を進めましょう.

```cpp
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello!" << endl;

  return 0;
}
```
input	output	io
入力	出力	入出力

```

⇩ レンダリング

最初の一歩

まず，c++言語で「Hello!」と画面表示するプログラムを作ってみましょう.

- 文字は，英数字モードで入力して下さい.
- 入力完了したら，*保存*, *コンパイル*, *実行*の順番で作業を進めましょう.

```cpp
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello!" << endl;

  return 0;
}
```

input	output	io
入力	出力	入出力

図 2　Markdown 文書とレンダリング出力

これによって固定した内容に固執せず，動的な解説を授業内で実施できる．さらにコメント機能により，教材単位での教員と学習者の双方向性を持たせている.

また，Markdown コード中に，プログラムのソースコードを埋め込み，ビルドと実行を可能にした.

4.　期待される効果

授業担当者の立場から見ると，Markdown によるシンプルな文書記述を用いることで，教材の共有と変更がしやすくなる．さらに準リアルタイムでのレンダリングが可能なことから，解説しながら説明文やソースコードサンプルの追加を行い，ライブ感のある授業進行ができる.

学習者の立場から見ると，テキストとソースコードが一体化していて，その中でビルドと実行できるので,解説と実践のつながりを実感しやすい.また,Markdown の文法は簡単なので，自ら文書を改編してノートとしての利用も可能である.

両者共通の利点としては,授業毎に Web サイトを用意した場合，ファイル管理が容易で，実習結果やコメントの共有がしやすい点がある.

上記の関係を図 3 に示す.

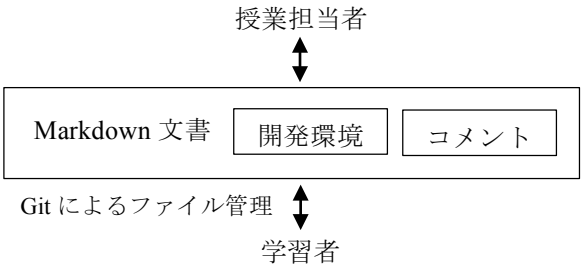

図 3　本システムでの授業形態

これらの特徴から，本システムをプログラミング基礎レベルの授業で利用することで，アルゴリズムやソースコードの理解に注力した学習を期待できると考える.

参考文献

(1)　Official Markdown project at Daring Fireball: http://daringfireball.net/projects/markdown/（2016）