プログラミングに関する 授業実践報告

埼玉県高等学校情報教育研究会
宮崎 万希子

会誌第13号及び教材→埼玉県高等学校情報教育研究会Web
http://www2.spec.ed.jp/krk/jyoho/comm2/?page_id=245

埼玉県高等学校情報教育研究会 研究委員会について

- * 教科「情報」で採用されたメンバーで構成
- * 毎年共通のテーマを決めて指導案を研究
- * 一昨年は「プログラミング」教えるにあたってのアンケートと授業案の検討
- * 昨年はアンケートの継続と授業実践

今回の発表では授業案と実践報告を行います

前半の発表 アンケート結果より

- ほぼ半数がプログラミングの経験がない
- プログラミングは難しいという先入観
- 日常との結びつきが薄い

* プログラミングは難しいという先入観を払しょくし、

プログラミングへ興味を持たせる授業をしたい！

code.orgを利用した授業案の作成

前半の発表 アンケート結果より

- ほぼ半数がプログラミングの経験がない
- プログラミングは難しいという先入観
- 日常との結びつきが薄い

身の回りにはプログラムで動いているものが多くあるのに、それを認識していない生徒が多い。

「身近なところでプログラムが動いている」ことを生徒に実感してもらおう授業(追加目標)

code.org

・ブロックでアルゴリズムを確認

・コード表示で実際に動いているプログラム (JavaScript)を確認

The screenshot shows the code.org interface for a puzzle. On the left, a character stands on a snowflake-shaped path. A yellow arrow labeled '3' points to the character. Below the path is a 'リセット' (Reset) button with a yellow arrow labeled '2' pointing to it. In the center, a message box says 'ゆきのけっしょうは、それぞれちがうかたちなのってる？ "くりかえす"ブロックをつかぎにまげながら、あたらしいゆきのけっしょうをつくりましょう。' with a yellow arrow labeled '1' pointing to the 'くりかえす' (Repeat) block in the code editor. The code editor shows a loop with '前に動く' (Move forward) and '右に向く' (Turn right) blocks. A yellow arrow labeled '4' points to the JavaScript code view on the right, which displays the following code:

```
for (var count2 = 0; count2 < 4; count2++) {  
  for (var count1 = 0; count1 < 2; count1++) {  
    moveForward(100);  
    turnRight(60);  
    moveForward(100);  
    turnRight(120);  
  }  
  turnRight(90);  
}
```

code.studioを用いた 授業案と実践報告

1. **ブロック** → フローチャート → プログラム確認
2. フローチャート → **ブロック** → プログラム確認
3. **ブロック** → フローチャート → プログラム確認
4. **ブロック** → フローチャート → 問題解決
5. **ブロック** → プログラム確認 → プログラミング
6. **ブロック** → プログラム確認 → コーディング

code.studioを用いた 授業案と実践報告

1. ドリル形式のワープロソフト教材でフローチャートを作成

1. ブロック→フローチャート→プログラム確認

実行した時
繰り返し 5回
してください
まえにすすむ

▶ 実行

ブロックを2つだけ使い、緑のブロックにたどりつく方法がありません。わかりますか？

②フローチャートを書いてみよう！
Stage6~Stage13 解答編
年 級 番 氏名

ミッション
code.org の迷路を解いてそれをフローチャートで表してみよう！

必要な記号

- 開始 (開始)
- 終了 (終了)
- 前に進む (処理記号)

新しい必要な記号

- 繰り返しの記号 (繰り返し記号)

流れ線
記号同士を結び処理の流れを示す。水平や上から下へ向かうときは矢印を使う。

Point!

コンピュータは繰り返すのが得意！
間違えずに何度でも素早く繰り返す事ができる。

実施内容と振り返り

プログラムの構造の理解＋フローチャートの書き方の習得＋ワープロソフトの技能習得
楽しみながらフローチャートの書き方、論理的な考え方や構造の理解ができた。
ゲーム感覚になりすぎずに段階的に学習ができた。

「繰り返すブロックが欲しい」等の解答が
得られた。

生徒自ら、ワープロソフト教材で作業することで、反
復構造の必要性や構造を理解することができたと
考察できる。

フローチャート問題 正答率14%→48%へアップ

code.studioを用いた 授業案と実践報告

2. フローチャートとビジュアルプログラミング

2. フローチャート→ブロック→プログラム確認

Code Studio interface showing a character on an ice field with a square drawn around it. The interface includes a 'C O D E' header, a progress indicator, and a 'ブロック達' (Blocks) panel on the right with visual programming blocks like '前に動く' (Move Forward) and '右に向く' (Turn Right). A 'ワークスペース' (Workspace) panel on the far right shows a list of executed blocks.

四角を描いてみる

～順次処理～

実行した時

```
前に動く 100 ピクセル  
右に向く 90 角度  
前に動く 100 ピクセル
```

```
moveForward(100);  
turnRight(90);  
moveForward(100);  
turnRight(90);  
moveForward(100);  
turnRight(90);  
moveForward(100);
```

しかくをかくのにはんぶんのところまでできみたいだわ。さあ、しかくをかくために、4ほんのせんをひきましょう！

おめでとうございます！あなたはパズルを3完了しました。
あなたはたった今5行のコードを書きました！
すべての時間の合計: 13ラインのコード
▼コードを表示します。

アメリカのトップの大学(例えば、**Berkeley Harvard**)でもブロックベースのプログラミングを教えています。あなたが組み合わせたブロックは実際のプログラム言語でどのように行っているか確認できるようにJavaScriptで表示できます。JavaScriptは世界中で最も広く使われているプログラム言語です。

```
moveForward(100);  
turnRight(90);  
moveForward(100);  
turnRight(90);  
moveForward(100);  
turnRight(90);  
moveForward(100);
```

実施内容と振り返り

ICカード乗車券やATMなど
身近な情報システムの仕組みを
フローチャートにすることができた

フローチャートを作成を通じて、あらゆるコンピュータやシステムがプログラムによって動いていることを理解させた。

フローチャートで問題整理ができた
自分達で考えアレンジするグループもあった

グループで課題解決にはフォローが必要
課題Aは変数の概念を理解させる必要あり

- 課題A: 暗証番号の認証
- 課題B: 店頭での電子マネーの支払い
- 課題C: 自動改札機の入場

code.studioを用いた 授業案と実践報告

3. ブラックボックスからホワイトボックスへ～挙動を知る

3. ブロック→フローチャート→プログラム確認

(3) 反復 (繰り返し、ループ) □その2. □
例題1 □.

```
while(notFinished){  
  moveForward();  
}
```

(反復)

ジェイソンをプログラミング (反復)

「反復」という概念を、ジェイソンをプログラミングして動かすことで理解します。

実施内容と振り返り

※Webを利用する教材は、
つながらない場合に備えて
別教材の準備が必要

フローチャートの解説とランダム関数の理解

表計算ソフト、総合開発環境monacaを使用し、ガチャゲームの一部改造

+ 追加機能について期末で出題、出てきた回答をもとに意見交換をさせた

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <script src="plugins/plugin-loader.js"></script>
6   <link rel="stylesheet" href="plugins/plugin-loader.css">
7
8   <script>
9     // Set virtual screen width size to 768 pixels (横幅640pxに設定)
10    monaca.viewport({width: 768});
11  </script>
12
13  <!-- noUISlider -->
14  <script src="./nouislider3/jquery.nouislider.min.js"></script>
15  <link href="./nouislider3/nouislider.fox.css" rel="stylesheet">
16  <style>
17    /* enlarge round noUISlider's handle */
18    .noUISlider.horizontal div {}
19    margin: -13px 0 0 -9px;
20    width: 34px;
21    height: 34px;
22    -webkit-border-radius: 17px;
23  </style>
24 </script>
25
26  <!-- jqplot -->
27  <script type="text/javascript" src="./jqplot/jquery.jqplot.min.js"></script>
28  <link rel="stylesheet" type="text/css" href="./jqplot/jquery.jqplot.min.css" />
29  <script type="text/javascript" src="./jqplot/jqplot.barRenderer.min.js"></script>
30
31  <script type="text/javascript" src="./rr01.js"></script>
32 </head>
33
34 <body>
35 <h1>NoUISlider3 Test</h1>
36 <div class="SliderLabel">999</div>
37 <div class="noUISlider Slider1" style="width:500px"></div>
38
39 <div id="chart0" style="height:500px;width:400px;"></div>
40
41 </body>
42 </html>
```

【情報ノート】
No.1 | 2016.10

ゲームで遊びゴコロを足せる目【ランダム関数】

Excelで → ...rand()
JavaScriptでは → ...Math.random()

【ランダム関数】は0 - 1の間のデシマーを返す

プログラムの一部を解説してみよう。
(おまけにゴロゲーム)

```
var no = Math.random() * 10;
// 0から9までの数字を生成する
var mass_name = [
  { mass_name: "daijichon": [
    { no: 4 }
  ] },
  { mass_name: "chukichon": [
    { no: 3 }
  ] },
  { mass_name: "shukichon": [
    { no: 2 }
  ] },
  { mass_name: "suekichon": [
    { no: 1 }
  ] },
  { mass_name: "kouuns": [
    { no: 1 }
  ] }
];
```

※ゲームとしてどうア...
改善した方がよい点は?

【解決すべき問題】

左のようなアプリを開くと、ボタンをクリックすると、カードの画像が出てくるゲーム。今、「超レアカード」、「レアカード」、「はずれ」の3つの画像を用意した。【ランダム関数】を使って、ゲームを面白くすることは、どのような方法で?

トコニシガチャ
ガチャをクリック!
1回1000コイン

```
var luckyNumber = Math.random() * 10000;
# luckyNumber > 22222 {
  document.getElementById("result-max").innerHTML = "超レアカードゲット!";
  document.getElementById("result-max").src = "images/box_treasure2.png";
}
else # luckyNumber > 2222 {
  document.getElementById("result-max").innerHTML = "レアカードゲット!";
  document.getElementById("result-max").src = "images/box_treasure.png";
}
else {
  document.getElementById("result-max").innerHTML = "はずれはずれです。";
  document.getElementById("result-max").src = "images/box_emboss.png";
}
```

【解決すべき問題】その2
さらに、このゲームを面白くするにはどのようなアイデアが考えられるか?

code.studioを用いた 授業案と実践報告

4. アルゴリズム学習から問題解決実習へ 4. ブロック→フローチャート→問題解決

1限～2限 次のコードは何をあらわしているか

```
for (var count2 = 0; count2 < 4; count2++) {  
  for (var count = 0; count < 2; count++) {  
    moveForward(100);  
    turnRight(60);  
    moveForward(100);  
    turnRight(120);  
  }  
  turnRight(90);  
}
```

フローチャートを書いてみよう
例) 遠足のカレー

作業内容(順番を考えよう)

- 野菜を切る
- 肉を切る
- 材料を炒める
- 鍋に水を入れる
- ルーを入れて煮込む
- 米を炊く
- 火をおこす
- 使った皿や鍋を洗う
- 調理器具を用意する
- 使った調理器具を洗う
- 全員で「いただきます」
- ごみを捨て、流しを掃除する

- テーブルを拭く
- お皿に米とカレーをよそう
- 皿をテーブルまで運ぶ
- 味見する
- 米を研ぐ
- 煮込む
- あくをとる

確認事項

- 材料に火は通ったか
- 全員分のカレーはそろったか
- 片づけは終わ

実施内容と振り返り

機械に伝えるコーディングではなく、
人に伝えるフローチャートを作り、
問題解決へ利用する。

作業をリスト化し、
流れをフローチャートにすることで、
パターン化できる部分を共有できる。
フローチャートをもとに話し合い、
今までできなかった部分に意識を向け、
新しい工夫を生み出すことができた。

課題
発表用資料の作成に時間がかかる
フローチャート図が残らなかった

課題：文化祭のクラス企画を振り返る

- 文化祭準備～当日までの期間に、何をしたか？
- いつ、どんな作業を、誰が担当していたか？
- 自分がした仕事、係りで担当した仕事、クラス全体でした仕事、誰もやらなかった仕事…
- 無駄な部分はどこか、失敗した点はあるか、それに対する改善案はあるか…

できるだけ具体的に書き出してみよう!!

code.studioを用いた 授業案と実践報告

5. VBScriptとPythonを使ったプログラミング

5. ブロック→プログラム確認→プログラミング

- code.orgの実行
プリントでの復習(1・2限目)
- VBScriptまたはPythonを利用した
プログラミング学習(3~6限目)

```
プログラム4 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
input1 = input('tanka: ')
input2 = input('kosuu: ')
goukei = int(input1) * int(input2)
print('goukei:', goukei)
otsuri = 1000 - int(goukei)
print('otsuri:', otsuri)
kaemasu = int(otsuri) / int(input1)
print('ato', int(kaemasu), 'ko kaemasu')

import time
time.sleep(3)
```

```
tanka: 50
kosuu: 11
goukei: 550
otsuri: 450
ato 9 ko kaemasu
```

```
プログラムテスト - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
input1 = input('kazu: ')
amari = int(input1) % 2
if int(amari) == 1:
    print('kisuu')
else:
    print('guusuu')
import time
time.sleep(3)
```

入力

1からいくつまでを足しますか?

OK

キャンセル

実施内容と振り返り

Code.orgはプログラミングの3つの基本構造の理解につながる

- プログラムを**ヒント無しで、自力で作れる生徒は1～2割程度**
- 原因は、**変数の宣言の理解に多く時間を取れなかったことや、**
剰余の計算等の数学的思考力が不足していたため、
時間内にコーディングまでたどり着けなかったと考えられる。
- 今後の課題
変数の扱い方の学習方法
エラーの発見をしやすくするエディタ
について研究を深めるといったことがあげられる。

本授業実践ではVBScriptの方が適切だと感じた。

code.studioを用いた 授業案と実践報告

6. 身近なプログラムのロジックを知る

6. ブロック→プログラム確認→コーディング

おめでとうございます！あなたはパズルを
11 完了しました。

あなたはたった今 3 行のコードを書きました。
すべての時間の合計: 7 フラインのコード
▼コードを表示します。

アメリカのトップの大学例えば、Berkeley Harvardでもブロックベースのプログラミングを教
えています。あなたが組み合わせたブロックは実際のプログラム言語でどのように行っているか
確認できるようにJavaScriptで表示されます。JavaScriptは世界中で最も広く使われているプロ
グラム言語です。

```
for (var count2 = 0; count2 < 4; count2++) {  
  for (var count = 0; count < 2; count++) {  
    moveForward(180);  
    turnRight(90);  
    moveForward(180);  
    turnRight(120);  
  }  
  turnRight(90);  
}
```

JavaScript でプログラムを組んでみよう！

メモ帳で作る

保存するときには拡張子.html (すべてのファイル)

保存先は3000→プログラム (新しいフォルダを作る)

実行はweb ファイルで開く

課題 1 パスワードのチェック
ファイル名 pw.html
4桁数字のパスワードを設定しておき、入力されたパスワードが正しいかどうかの判定をする (例 ログインできました ログインに失敗)

課題 2 おみくじ
ファイル名 kuji.html
任意の数字にコメントを用意しておく
数字が入力されるとコメントが表示される

課題 3 奇数・偶数判定
ファイル名 odd.html
好きな整数を入力すると奇数か偶数かを判定する

課題 4 1 から N までの総和を求める
ファイル名 sum.html
好きな整数を入力すると 1 からその数字までのすべての整数を足した合計を計算する
例 5 を入力すると 1 + 2 + 3 + 4 + 5 の答えを計算して出力する

課題 5 素数判定
ファイル名 prime.html
好きな整数を入力するとその数字が素数かどうかを判定する

実施内容と振り返り

フローチャート → プログラミング だった流れを
Code.org → プログラミング へ変えた。
興味を持たせやすく処理手順の言語化がしやすい

課題解決に数学的理解が必要 (課題の内容要検討)

自力でできた生徒は1割程度

↓追加

今年は1学期に実施し、実施時間を増やしたところ、
全体の理解度がかなり高まった。

プログラミング実習にはある程度の時間をかけ、
慣れさせることが必要と感じた

プログラムしてみよう！！

<命令文> | これさえわかればプログラムはできる！！

キーボードからの入力を使 す	<code>prompt("あなたの好きな数字を 入力してください");</code>	" " (ダブルクォーテーション)で囲まれ た文字をディスプレイに表示する。 入力された数字を <code>num</code> に格納(代入) する。
出力する	<code>document.write("あなたの好きな数字 ですか?");</code>	<code>num</code> に格納されている数字と" の数字列をディスプレイに表示する。
条件分岐	<code>if(num%2==0){ document.write("偶数"); } else{ document.write("奇数"); }</code>	もし <code>num</code> という変数が0ならば 偶数とディスプレイに表示する。 そうでなければ 奇数と表示する
場合分け	<code>switch(num){ case 1: ...alert("●"); ...break; case 2: ...alert("×"); ...break; case 3: ...alert("▽"); ...break; case 4: ...alert("■"); ...break; default: ...alert("□"); ...break; }</code>	もし <code>num</code> に格納されている値が 1ならば ● を表示する (そしてこのルーチンを出る) 2ならば × を表示する (そしてこのルーチンを出る) 3ならば ▽ を表示する (そしてこのルーチンを出る) 4ならば ■ を表示する (そしてこのルーチンを出る) それ以外ならば □ を表示する (そしてこのルーチンを出る)
繰り返し	<code>for(i=0;i<10;i++){ 処理1 }</code>	<code>i=0</code> のときから <code>i<10</code> までの間 (10回) 処理1行ずつ 処理1行の回数に <code>i</code> を1ずつ増やす
ある条件をクリアしたら for 文(繰り返し)から抜ける	<code>for (初期化式 条件式 変化式) ...if (条件式) ...break; ...}</code>	
代入する	<code>a=1</code>	変数 <code>a</code> に1を代入する
比較する	<code>a==1</code>	<code>a</code> は1である

実践報告を終えて全体を振り返ると

- * 2年同じテーマで研究を続けることで、
落ち着いて計画的な授業実践ができた
- * どの授業実践もcode.orgは1～2時間かけて実施している。
 - ✓ 導入に使うことによって、生徒の興味関心を引くことができた。
 - ✓ ハードルが低い・生徒が自分のペースで取り組める・理解が早い
 - ✓ 「アナ雪」と「古典的な迷路」など生徒の実情に合わせて教材が選べる
 - ✓ フローチャート・プログラミングそれぞれに移行しやすい
- * 導入にcode.orgは有用 → そのあとの教材が重要！

実践報告を終えて全体を振り返ると

- * 生徒の実情に応じて、何を、どこまで教えるかは様々。
- * ゴールはどこにすればいいのだろうか？

- ✓ Wordのスキルアップ（反復の必要性）
- ✓ フローチャートに落とし込む
- ✓ 身近なシステムを考えさせる
- ✓ プログラムの改善案を考えさせる
- ✓ プログラムを書かせる
- ✓ フローチャートから問題解決

**プログラミング的な考え方
(論理的思考)と
問題解決能力**

実践報告を終えて全体を振り返ると

* 今回の授業実践は4～9コマかけている。

* プログラミング学習にはある程度の時間が必要。

✓「反復」の理解が難しい

✓変数の概念・数学的知識

✓フローチャートの書き方

✓プログラミング言語の読み方

✓数学的思考力が必要な課題になってしまう

✓グループで行う話し合い

・・・どれも時間がかかる

より効率よく、深い学習ができないか？

おわりに

目的 「論理的思考」「問題解決能力」を身につける

手段 「プログラミング学習」

目標

**楽しく、効率よく、深い学び
ができる授業案を作っていこう**

会誌第13号及び教材→埼玉県高等学校情報教育研究会Web
http://www2.spec.ed.jp/krk/jyoho/comm2/?page_id=245

ご清聴ありがとうございました。