専門学科情報科における生徒の実態把握と アルゴリズムの学習におけるプログラミング言語の影響

藤木　　卓*　倉田　　伸**

（平成23年10月31日受理）

The survey about Informatics Education on Special Course of Senior High School and Influence of Programming language in Computer Algorithm Instruction

Takashi FUJIKI*　Shin KURATA**

(Received October 31, 2011)

１．はじめに

平成15年度から高等学校において教科「情報」（文部科学省1999）が必須科目となり，全ての高等学校において情報教育がなされている。教科「情報」は，普通教科「情報」（平成25年度からは共通教科「情報」となる）と専門教科「情報」から構成されるが，専門教科「情報」を実施している学校は少ない。ところで，専門教科「情報」は，「システムの設計・管理分野」と「情報コンテンツの制作・発信分野」の2分野が大きな柱となっており，情報をデザインする力が求められている。「システムの設計・管理分野」では，主にソフトウェアの開発（プログラミング）能力の育成が必須であり，そのためにアルゴリズムの理解が欠かせない。

長崎県では，平成19年度に専門学科である情報科（以下，「情報科」）がＩ高等学校（以下，Ｉ高校）に新設された。そこで行われている専門教科「情報」の授業では，スペシャリストとして必要な情報に関する基礎的・基本的な知識・技術を習得させることが意図されている。そこで，設立から５年目となる「情報科」の課題を明確にして今後の教育に活かすために，学習の実態把握を意図した。またＩ高校では，これまでアルゴリズムの学習で用いるプログラミング言語をＣ言語としていたが，平成23年度からは，Ｃ言語だけでなくイベント駆動型 BASIC を加えた形で授業を展開している。そのため，プログラミングの学習におけるつまずきに関する研究（安達ら1995）を参考にしながら，この使用言語の変更による生徒の理解への影響についても調べることとした。

以上のような背景から，本研究では，Ｉ高校の「情報科」における，質問紙による生徒の実態把握と，アルゴリズム学習の際に使用するプログラミング言語の違いが生徒の理解に与える影響について調査することを目的とした。

*長崎大学教育学部　**長崎県立諫早商業高等学校

２.「情報科」における学習の現状

２.１　調査の方法

「情報科」における学習の実態調査は，専門教科「情報」の基礎的な学習内容全般についての意識調査と，プログラミングの基礎についての意識調査に分けて実施した。

前者の基礎的な学習内容全般についての調査では，「情報科」に所属する２年生40名を対象に質問紙により行った。これは，「情報科」では，専門教科「情報」の基礎的な学習内容を２年生までに網羅するようなカリキュラム構成をとっているためであり，調査時点で対象となる生徒は該当する基礎的な学習内容をほぼ終えていたことがその理由である。授業で扱われている学習内容は，情報処理技術者試験の IT パスポート試験程度のレベルである。ここでは，内容のまとまりから学習内容を開発技術，情報科学の基礎理論，コンピュータシステムの構成要素，ソフトウェア，メディアとデータベース，ネットワーク，セキュリティの７つに区分した。そして，それぞれの学習分野について，「興味が湧く（興味)」「理解しやすい（理解)」「イメージが湧きやすい（イメージ)」「日常生活に結び付く（日常)」の４観点で，４件法（４：非常にそう思う，３：そう思う，２：あまり思わない，１：全然思わない）による主観評価を実施した。

後者のプログラミングの基礎についての調査では，「情報科」に所属する１年生～３年生でプログラミングの基礎科目である「アルゴリズム」を履修した110名を対象に質問紙により行った。調査項目は，プログラミングの理解や印象に関する22項目とした（表２参照)。そして，それぞれの項目について，４件法（４：非常にそう思う，３：そう思う，２：あまり思わない，１：全然思わない）による主観評価を実施した。

２.２　調査の結果及び考察

基礎的な学習内容全般についての調査結果を，表１に示す。表中，3.0以上の評価を得た項目については太字で，また最も低い評価を得た項目については斜体で示した。

表から分るように，興味の観点で生徒が高く評価したのがセキュリティの学習についてであった。理解の観点では，コンピュータシステムの構成要素が最も高く，次に情報科学の基礎理論とネットワーク，セキュリティが高い値を示した。イメージの観点では，コンピュータシステムの構成要素が高い値を示した。日常の観点では，ネットワークが最も高く，次がコンピュータシステムの構成要素であった。また，開発技術を除く分野では評価の中央の値である2.5を全て上回る結果を示しており，基礎的な学習内容全般については

表１　学習分野と生徒の主観評価結果（※数値は，評価平均値）

学習分野	興味	理解	イメージ	日常
開発技術	*2.2*	2.6	*2.4*	*2.0*
情報科学の基礎理論	2.7	**3.0**	2.9	2.6
コンピュータシステムの構成要素	2.9	**3.2**	**3.1**	**3.0**
ソフトウェア	2.8	2.9	2.7	2.7
メディアとデータベース	2.6	2.8	2.7	2.6
ネットワーク	2.9	**3.0**	2.9	**3.1**
セキュリティ	**3.0**	**3.0**	2.8	2.9

ほぼ理解できているものと考えられる。一方，開発技術では４つの評価観点すべてで最下位であり，かつ理解の観点を除いて2.5以下の評価であった。開発技術の分野は，専門教科「情報」の大きな柱の１つである「システムの設計・管理分野」に関連する分野である。そのため，この分野の学習に関する今後の対策が必要であると言える。

次に，プログラミングの基礎についての調査結果を，表２に示す。表中の評価欄は，評価平均値を示している。評価平均値のうち，評価の中央の値である2.5を上回るものは太字で，評価が１点台のものは斜体で示した。

表２　プログラミングの基礎に関する調査結果

項　　目	評価
01：前の授業で理解できないと，次の授業内容も理解できない	**2.8**
02：プログラムに関する演習問題は，他の教科に比べ解きやすい	2.2
03：出来ればプログラミングはしたくない	2.2
04：検定の問題を解けるようになれば，自分でソフトウェア開発が出来ると思う	*1.8*
05：「オブジェクト指向プログラミング」に興味がある	2.1
06：検定のプログラミングの内容は，頭にイメージが湧きやすい	2.3
07：プログラミングの授業で，自分は演習などが早く終わるので時間がもったいないと感じている	*1.8*
08：プログラミングの内容が把握できないまま，次の内容へ進むことがある	2.5
09：実際自分でプログラミングして，ソフトウェアを開発するよりも，検定の点数を上げるほうが良いと思う	2.4
10：自分でソフトウェア開発がしたいと思うが，どうすれば良いかが分からない	2.5
11：フローチャートを見て，プログラム言語に変更して作ることができる	*1.9*
12：プログラミングの問題を解くにあたり，架空のデータよりも実際の身近なデータの方がイメージしやすい	**3.4**
13：プログラミングを自分で好きなようにやってみたい	**2.6**
14：プログラミングを高校卒業してからも続けていきたい	2.4
15：Javaなどの「オブジェクト指向プログラミング」をやってみたい	2.5
16：プログラミングの授業よりも，他の情報分野を勉強したい	2.5
17：自分のレベルに合った授業展開をしてほしい	**3.0**
18：自分のレベルより上の内容を授業で実施し，それに頑張ってついていきたい	2.3
19：自分のレベルより下の内容を少しずつ完璧にしながらやっていきたい	**2.7**
20：プログラミングにアニメーション機能があると，理解できなかった内容が理解できるようになれると思う	**2.8**
21：もっと図（イラスト）が出てくるようなプログラミングの授業を受けたい	**2.8**
22：プログラミングと日常の生活は結びついていると感じる	2.5

表から分るように，評価の中央の値である2.5を上回る肯定的な評価を示したのは，「01:前の授業で理解できないと，次の授業内容も理解できない」「12:プログラミングの問題を解くにあたり，架空のデータよりも実際の身近なデータのほうがイメージしやすい」「13:プログラミングを自分で好きなようにやってみたい」「17:自分のレベルに合った授業展開をしてほしい」「19:自分のレベルより下の内容を少しずつ完璧にしながらやっていきたい」「20:プログラミングにアニメーション機能があると，理解できなかった内容が理解できるようになれると思う」「21:もっと図（イラスト）が出てくるようなプログラミングの授業を受けたい」の7項目であった。このうち，「01:前の授業で理解できないと，次の授業

内容も理解できない」の問いにあるように，プログラミングは積み上げの授業だと理解しているからこそ，「17:自分のレベルに合った授業展開をしてほしい」や「19:自分のレベルより下の内容を少しずつ完璧にしながらやっていきたい」のような授業への願いや自分自身への自覚を意識したと考えられる。また，よりよく学びたいからこそ，「12:プログラミングの問題を解くにあたり，架空のデータよりも実際の身近なデータのほうがイメージしやすい」や「20:プログラミングにアニメーション機能があると，理解できなかった内容が理解できるようになれると思う」「21:もっと図（イラスト）が出てくるようなプログラミングの授業を受けたい」のような，分り易く学ぶための工夫を望んでいることが伺える。

一方,「04:検定の問題を解けるようになれば，自分でソフトウェア開発が出来ると思う」「07:プログラミングの授業で，自分は演習などが早く終わるので時間がもったいないと感じている」「フローチャートを見て，プログラム言語に変更して作ることができる」の３項目については，評価が１点台と低い値を示していることが分かる。「04:検定の問題を解けるようになれば，自分でソフトウェア開発が出来ると思う」の問いでは，検定の問題が解けてもソフトウェアの開発が楽にできる訳ではないというプログラミングの奥深さに気付いていると考えることができる。また,「07:プログラミングの授業で，自分は演習などが早く終わるので時間がもったいないと感じている」と「フローチャートを見て，プログラム言語に変更して作ることができる」については，プログラミングの能力に関する自己評価と見ることができ，それがなかなか身に付いていないことを実感させられる。さらに,「11:フローチャートを見て，プログラム言語に変更して作ることができる」の評価の低さは，学習が済んでいるにも関わらずフローチャートとプログラム言語の対応ができていない生徒が多い実態を示しており，何らかの対策が必要であろう。

３．アルゴリズムの学習におけるプログラミング言語の影響

３.１　調査の方法

アルゴリズムの学習において，使用するプログラミング言語の影響を調べるために，イベント駆動型 BASIC を学習した情報科の１年生38名と，C 言語を学習した２年生40名を対象に調査を行った。調査の対象とする学習内容を，表３に示す。

表３　「アルゴリズム」にいて調査対象とする学習内容

フローチャート	プログラミング言語（イベント駆動型 BASIC or C 言語）
順次構造基本形	同左
分岐の基本形	If 文（C 言語では括弧がつけない）
分岐の else 節省略	If 文
複合文の形式	If 文（C 言語では括弧を必ずつける）
他分岐選択制御	If 文（else　if文）
（分岐箇所：２個，３個）	If 文（ネスト）
反復構造	While 文
反復構造（ネスト数:１，２）	While 文（ネスト）
反復構造（ネスト数:１，２）& 結合子	

表３に示す学習内容は，構造化プログラミングの基本的な論理構造である「順次」「分岐」「反復」に関するものである。「分岐」については，条件が正しい場合と誤っている場合にそれぞれ何らかの処理を行うパターン（以下，分岐の基本形）と，条件が正しい場合のみ何らかの処理を行うパターン（以下，分岐の else 節省略）に分類した。また，分岐の基本形の中で，何らかの処理を行う部分が複数行（または複数個）あるパターン（以下，複合文の形式）も設定した。「反復」については，フローチャートの反復処理は継続条件で繰り返す（「…の間」）の形式とし，プログラミング言語では，While（イベント駆動型 BASIC，C 言語の両方とも）文を使用した。そしてネスト（入れ子）をしないパターン，ネストを１つ含むパターン，ネストを２つ含むパターンに分類した。更にネストを含んだフローチャートにおいて，結合子を含むパターンと含まないパターンに分類した。

調査は，ペーパーテストの形式とし，各論理構造のパターンで表記されているフローチャートとプログラミング言語の一部を左右に並記し，理解しやすい方を選択するといった二者択一方式とした。質問項目は14種類を準備し，構造化プログラミングの基本的な論理構造を万遍なく調べることとした。１年生には，フローチャートとイベント駆動型 BASIC の比較を行わせ，２年生にはフローチャートと C 言語の比較を行わせた。また，フローチャートとプログラミング言語の全般的な理解のし易さの項目も追加した。調査用紙の例を図１に示す。

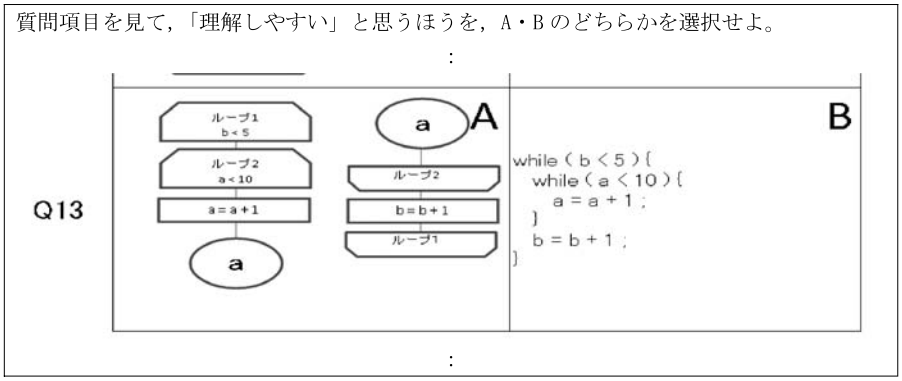


図１　調査した内容の例（フローチャートと C 言語の比較調査の一部）

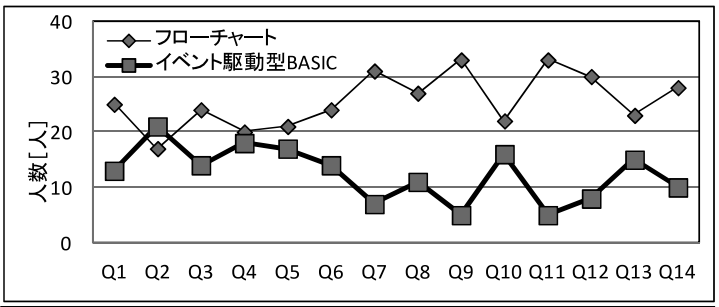
３.２　調査の結果及び考察

＜フローチャートとイベント駆動型 BASIC の比較＞

フローチャートとイベント駆動型 BASIC を比較した１年生の結果を，図２に示す。

図から分かるように，Q２以外のパターンではイベント駆動型 BASIC よりもフローチャートの方が理解し易いという結果を示した。その理由としては，「図（線を含む）が文字よりも見やすい」という自由記述回答が最も多かった。全般的に見た場合，処理の量が多くなるにつれて，フローチャートのほうが理解しやすいという傾向が示唆される。

Q２で，イベント駆動型 BASIC のほうがフローチャートよりも理解しやすいと回答されたのは，設問に含まれる情報の量が影響していたと考えられる。すなわち，イベント駆動型 BASIC の行数は1行であったが，フローチャートでは図が表記されている分多くの



Q1：全般的な理解しやすさ
Q2：順次構造基本形
Q3：分岐の else 節省略形
Q4：分岐の基本形
Q5：複合分の形式
Q6：他分岐選択制御
（分岐 2 箇所：else if 使用）
Q7：他分岐選択制御2
（分岐 2 箇所：ネスト使用）
Q8：他分岐選択制御3
（分岐 3 箇所：else if 使用）
Q9：他分岐選択制御4
（分岐 3 箇所：ネスト使用）
Q10：反復構造
Q11：反復構造（ネスト 1 箇所）
Q12：反復構造（ネスト 2 箇所）
Q13：反復構造（ネスト 1 箇所：結合子使用）
Q14：反復構造（ネスト 2 箇所：結合子使用）

図２　フローチャートとイベント駆動型 BASIC の比較

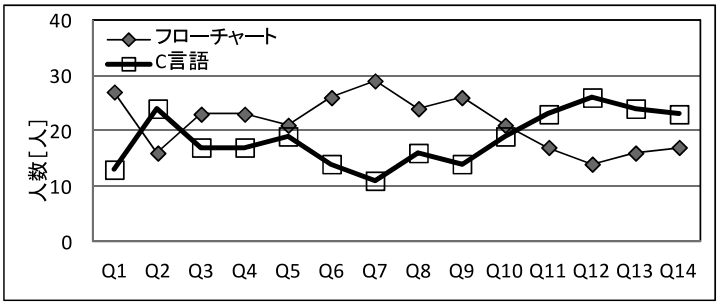


図３　フローチャートと C 言語の比較（設問 Q１～Q14は，図２と共通）

情報が含まれていたことによると考えられる。

Q６とQ７は同じく分岐選択制御の問題であるが，Q７でフローチャートを選択した生徒が多い。これは，ソースコードの行数と字下げの段数が関係すると考えられる。すなわち，Q６のイベント駆動型 BASIC 表記は行数が７行で字下げが１段のみであるのに対して，Q７のイベント駆動型 BASIC 表記は行数が９行で字下げが２段まである。このことから，プログラミング言語を学習し始めた学習者は，ソースコードの行数と字下げが多くなると，理解困難になると考えられる。

Q11からQ14の結果を見てみると，フローチャートにおいて結合子を使用せずに１列に表記した場合と，結合子を用いて２列に表記した場合では，結合子を用いて２列に表記した場合のほうが理解しやすいという結果になった。このことから，アルゴリズムを学習し始めた学習者は，フローチャートの１列あたりの処理の量が多くなると理解困難になると考えられる。

＜フローチャートとC言語の比較＞

フローチャートとC言語を比較した2年生の結果を，図3に示す。

図から分かるように，他分岐選択制御に関する設問であるQ6～Q9においてフローチャートがC言語よりも理解しやすく，ネストを含んだ反復構造に関する設問であるQ11～Q14においてC言語がフローチャートよりも理解しやすいという結果を示している。他分岐選択制御に対するフローチャートの優位性は，イベント駆動型BASICにおける考察と同様に，処理の量が多くなるとフローチャートのほうが理解しやすいためであると考えられる。ところが，ネストを含んだ反復構造に関しては，イベント駆動型BASICとは逆の傾向を示すことが明らかである。そこで，次にイベント駆動型BASICとC言語の比較を試みる。

＜イベント駆動型BASICとC言語の比較＞

前述の，フローチャートとイベント駆動型BASIC及び，C言語との比較に関する考察から，ネストを含んだ反復構造は，フローチャートやイベント駆動型BASICよりもC言語のほうが理解しやすいといえる。両言語の表記方法を比べてみると，C言語は反復処理を行う部分を括弧で囲むのに対し，イベント駆動型BASICは括弧を使用しない。このことから，ネストを含む複雑な反復処理において括弧をつけることは，学習者の理解を容易にすると考えられる。

4．まとめ

本研究では，専門学科情報科における実態把握とアルゴリズムの学習におけるプログラミング言語の影響を調査することを目的として実験を行った。その結果，以下のことが明らかになった。

- ●基礎的な学習内容全般については，開発技術を除いてほぼ理解できている。
- ●プログラミングの基礎については，難しさの自覚から，より分り易く学ぶための工夫を望んでいる。
- ●アルゴリズムの学習におけるプログラミング言語の影響については，ネストを含む反復構造の学習において言語の差が表れている。

参考文献

安達一寿，中尾茂子（1995）　プログラミング学習におけるつまずき箇所の分析．日本教育情報学会学会誌，10(4)，11-20，1995

文部科学省（1999）　高等学校学習指導要領（平成11年版）