プログラム学習における専門的概念導入に用いられる言語表現の調査

竹内　和広*　　越前　拓真**

Investigation of natural languge usage in introduction of programming concepts

Kazuhiro Takeuchi　　Takuma Koshimae

キーワード：プログラム作成技術、概念表現、テキストマイニング、共起ネットワーク

英語要約

The use of natural language in explaining programming ideas such as algorithms might be different from how daily language is used to communicate ideas. Novice programmers that are not accustomed to such linguistic expression may have difficulty in understanding documents for programming. In this study, we employ a text mining methodology to investigate how natural language expressions are used in descriptions for explaining programs. In particular, we analyze the specialized concepts in programming required for sophisticated IT engineering using past examination questions for the certification of information technology engineers. Based on the investigation, we discuss how initially learning the C language contributes as a basis for gradually learning more sophisticated programming.

１．はじめに

プログラム作成は情報技術における専門技能として重要な位置づけにある。独立行政法人情報処理推進機構（以下IPA）が実施する情報処理技術者試験は、情報処理技術者としての専門技能であるプログラム作成能力についても、一つの基準となる存在である。特に本稿では、情報処理技術者試験の試験区分のうち基本情報技術者試験の合格対象者像が「高度ＩＴ人材となるために必要な基本的知識・技能をもち、実践的な活用能力を身に付けた者」とされる点に着眼する。具体的には、プログラム初学者が習得した基礎概念を発展させ、高度ＩＴ人材に必要な専門的概念を段階的に学習する指針を、基本情報技術者試験の過去問題にテキストマイニング手法を適用す

* 大阪電気通信大学　情報通信工学部
** 大阪電気通信大学　大学院工学研究科

ることによって検討したい。ここで、テキストマイニング手法を導入するのは、アルゴリズムの理解と言語表現には密接な関係があると考えるからである。

プログラム作成の前提には、対象となる問題を解くための手順を定式化した形で表現する必要がある。このような手続きはアルゴリズムと呼ばれ、コンピュータが情報を処理するための基礎となる。アルゴリズムには様々なレベルがあると同時に、それを説明するための様々な記法がある。例えば、自然言語、擬似コード、フローチャート、プログラミング言語といった記法である。そのような記法にはそれぞれに特徴がある。例えば、プログラミング言語は、自然言語のような曖昧性を持たないため、明確にアルゴリズムを記述できるものの、プログラミング言語そのものを学習しておく必要がある。それに対して、フローチャートや擬似コードは、工夫をすれば、プログラム言語を習得することなく、直感的にアルゴリズムの構造を説明できる可能性がある。自然言語は誰もが日常的に用いているものであり、アルゴリズムを学ぶ事前学習の必要性から考えると、プログラムを体験していないプログラム初学者と教授者との間の共通言語としての役割が重要と言える。

しかし、自然言語のみで複雑なアルゴリズムを説明することは困難であることも事実である。そのため、多くの教科書や解説書、そして基本情報技術者試験の問題でも、アルゴリズムを擬似コードやフローチャート、処理内容を示す図等の記法と併用して自然言語による説明を行うことが一般的である。また、自然言語は、プログラミング言語で記述されたプログラム中にも使用される。例えば、プログラム中の部分要素がどのような働きを持つかを説明するコメント、変数や関数名などの識別子等の命名に自然言語が用いられる。このように、プログラミング言語で記述されるアルゴリズムは、たとえプログラミング言語を学習済であったとしても、自然言語を介在せずに説明することが困難なことを示している。

専門的な技術であるプログラム技術、特にアルゴリズムを説明する上での自然言語は、日常的な言語使用とは異なる語用を持ち、そのような言語表現に不慣れなプログラム初学者は戸惑いを覚える可能性がある。例えば、プログラムに対する説明文書が煩雑で長く、通常の言語使用とは異なる意味をもっているために、その読み手が文章の意味を誤解したり、理解が困難であったりする恐れがある。

以上のような背景から、本稿では、プログラム説明における、言語使用を調査するため、まず、大阪電気通信大学情報工学科１年生で利用している教科書を題材にプログラムの基礎概念がどのように表現されるかを概観する。次に、プログラム初学者と実際のプログラム経験を積んだプログラム作成者が、それぞれ、どのような自然言語の表現を用いて、自らのプログラム理解を表明・説明するかを調査する。そして、上記のようなプログラム説明における自然言語の使用に関する整理を前提に、基本情報技術者試験のプログラム作成に関わる過去問題を対象として、高度ＩＴ人材に要求される専門的概念を、言語表現を手掛かりに分析したい。特に、プログラム初学者によるプログラミング言語であるＣ言語の学習が、より高度なプログラム作成および専門的概念の段階的学習の基礎となりうるかを検討したい。

２．プログラム教科書における基礎概念の記述

２．１教科書表現の共起ネットワーク

プログラム作成を学ぶために、実際にプログラム実行をコンピュータで試すことができるプログラミング言語を学ぶことは、実践的な学習の導入となる。大阪電気通信大学情報通信工学部情

報工学科では、プログラム入門教科書のベストセラーとして知られている教科書[1]をプログラム教育に利用している。

この教科書の、出版社からの紹介としては、「経験がなく、はじめてプログラミングをはじめる人にも、無理なくプログラミングの基本から学習してもらえる」ように、次の工夫がなされているとされる。また、第五版は、全国学校図書館協議会選定図書となっている。

- 読みやすい解説でスラスラ読み進められる。
- 豊富なイラスト図解で、概念をイメージでわかる。
- たくさんのサンプルプログラムで、試して理解できる。

これらの特徴を客観的に見るために、同書のまえがき、目次、索引、プログラムリスト、図といった部分を除く、本文の自然言語による文章記述をコンピュータに入力し、出現語の共起ネットワークを作成したものを**図 1** に示す。共起ネットワーク図は、テキストマイニングの典型的な分析手法の一つである。なお、この作業はテキストマイニングのオープンソフトウェアであるKH Coder[2]を用いて行った。KH Coderは自由記述アンケートの分析[3]等に柔軟に対応できるため、広く用いられているツールである。

テキストマイニングは、日本語に対する自然言語処理を共通前処理として行う。具体的には、日本語の文を形態素に分割する、形態素解析と呼ばれる処理を行う。形態素とは、意味を持つ最小単位である。直感的に言えば、意味を持つ最小単位は語であると考えられるが、例えば大阪電気通信大学という語は、大阪、電気、通信、大学という語の結合により構成されている。このような語の認定、例えば大阪電気通信大学を固有名詞として認定したいという要求は、分析者に依存して定義される度合いが高い。そのため、どのような分析要求であっても、広く対応できる単位として形態素という単位が導入されている。

なお、共起ネットワークに示された、語が形態素という言語単位であることは、前述した通りであるが、ネットワークのノードとして、どの語を選択するかも一定の基準があるわけではなく、分析者の分析主眼によって選択する必要がある。**図 1** の語の選択は、教科書の自然言語記述に使われた内容語になりうる語の上位150を基準に出現回数11回以上の語をノードの対象として形式的に選択した。

共起ネットワークは、語の概念を表現する関連語ネットワークの一種である。関連語ネットワークは、特定の語の概念を、他の語との関係により表現する手法である。語間の関係は、特定の文や段落、あるいは文章といった言語単位に同時に生起するかを基準に計量することが多い。そのような共起の計量手法には、Jaccard係数、Dice係数、Simpson係数といったものが存在する。**図 1** では、Jaccard係数が0.1以上であるものを関係性ありとし、ノード間のアークとしてネットワークに表現している。

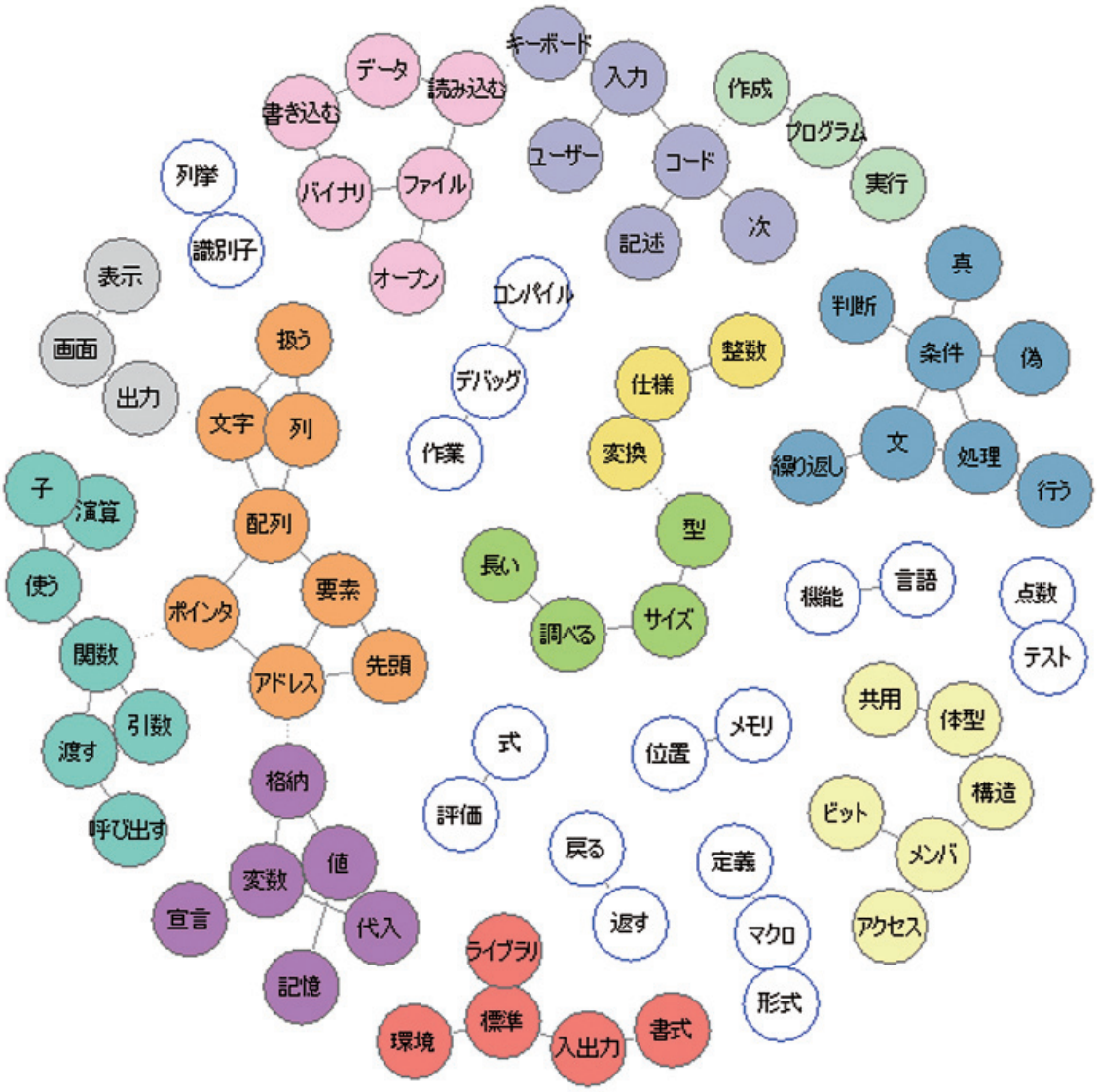

図１. 調査したプログラム教科書における共起ネットワーク

２. ２ プログラムの基礎概念

プログラムの基本構造そのものは専門家でなくとも体験的に理解できるとされる。例えば、ＧＵＩプログラミング言語であるScratch[4]は、情報を専門としない中学生や高校生でも条件分岐や繰り返しといったプログラムの基本構造の学習に効果的とされている。

図１を見ると、プログラムの基本構造である、条件分岐や繰り返しだけではなく、Ｃ言語というプログラミング言語の特徴である、配列、変数、ファイル入出力、構造体といった概念を良く表現できている。もちろん、このような概念表現の記述力は限定的である。例えば、図１のネットワークで、「繰り返し」に関連付けがなされている「文」は、実際の教科書では、次のように記述されている。

> *Ｃ言語では、１つの小さな処理（「仕事」）の単位を文（statement）と呼び、最後に;（セミコロン）という記号をつけます。そして、この「文」が、原則として先頭から順番に、１文ずつ処理されるということになっています。(p.18)*

上記の定義は、記憶装置にプログラムを内蔵し、それを先頭から順番に「１つずつ」逐次的に処理するノイマン型コンピュータの概念的な本質を分かりやすく説明しているが、そのような詳細にわたる概念は共起ネットワークには十全に表現されていない。しかし、特定分野の概念を形式的に示す上では有益な表現方法と考えることができ、本稿では概念の記法として、共起ネットワークを基に議論を進める。

他方、プログラム作成は、基礎概念の習得だけで十分に行われるわけではない。高度なプログラム作成には、習得した基礎概念の基盤の上に、さらに高度なプログラム作成の専門的概念の習得が段階的に必要であるものと考えられるが、**図１**に表現されたような基礎概念が、どのような形で高度化されるかについては明らかではない。

３．プログラム理解に関わる言語表現

３．１プログラム初学者によるプログラム説明

プログラムの高度化は、プログラムの基本命令を組み合わせて、より高度な目的を達成することが本質であるが、基本命令の組み合わせ方・高度化についてプログラム初心者がどう自覚的にとらえているかを言語表現的に調査する。

具体的には、プログラムの基本を学んだ情報系学科の大学生が、プログラムの基本構造をどのような自然言語で他人に説明するかを記述させた。調査対象とした学生は半期15回講義・演習でＣプログラミングの基礎を学んだ150名の大学１年生である。

調査のうち、**図２**のようなプログラム例を説明させた試みの結果について端的に紹介する。**図２**の２つのプログラムは、調査項目の一部である。調査対象者は全員が授業内でこのプログラムを学んでいる。

図２の調査の主眼は、プログラムが果たすべき目的を調査対象者が理解し、説明文の中にその理解をどのような言葉を用いて記述するかを知ることにある。例えば、**図２**のプログラム例②は､(整列されていない）配列に含まれる要素のうち、最大の数を見つけ出すプログラムである。配列に含まれる全ての要素を調べる必要があるが、Ｃ言語の基本的な命令では、一度に調べることができる要素は１つだけであるため、繰り返しが必要となる。このアルゴリズムを、日本語で記述すると、例えば次のようになる。

> *最初の要素（数）が最大と仮定する。残りのリスト上の数を順に見ていき、最大の数よりも大きい数が見つかったら、それを新たな最大として記憶する。最後に記憶した数がそのリストでの最大の数であり、これで処理が完了する。*

もし、調査対象者がプログラム例②の目的を理解できず、プログラムの説明をするならば、プログラム各行の処理を逐次記述するのみで､「最大」や「最大値」といった表現は使わない可能性が高い。

実験結果は次の通りである。例が前後するが、プログラム例②では、有効回答101件と少なくはあったが、有効回答のうち68%が「最大値」、あるいは「最大の値」といった表現を使い、最大値を求めるプログラムという説明を書き、複数行に渡るプログラムの目的や機能を説明することができた。

それに対して、何の制御構造も含まれていないにも関わらず、プログラム例①の説明では、有効回答124件中の16%にしか、「値の交換」や「入れ替え」という表現は見られなかった。大多数の説明例は、「ｃにａの値を代入、ａにｂの値を代入、ｂにｃの値を代入する。」といった、プログラムの各行の説明を、単に時系列で日本語にしただけの説明であった。

この結果は、プログラムの各行が示す役割の理解や、プログラム例②の配列や繰り返しといった基礎概念の理解があっても、より高度なプログラム作成を理解するためには、複数行に渡るプログラムが組みになって果たす目的や機能である「交換」といった概念が学習者に導入される必要があることを示唆している。

```
c = a ;
a = b ;
b = c ;
```

プログラム例①

```
/* 配列 a の各要素には整数値が入っている*/
max=a[0];
for(i=1; i<10; i++){
if(a[i]>max) max=a[i];
}
```

プログラム例②

図２．プログラム初学者が説明する説明対象の例

３．２プログラム言語内の言語表現

前節ではプログラム初学者がプログラムの部分要素の理解をどのように言語表現を用いて説明するかを調査した。ここでは、前節とは対照的に、実際にプログラムを記述する経験を積んだプログラム作成者が、プログラムの部分要素が果たす機能をどのような言語表現を使うかを調査する。しかし、本稿では実際のプログラム作成者に前節で調査対象者に実施したような調査実験を行うのではなく、プログラミング言語によるプログラム記述中に用いられる表現に着眼した。

プログラミング言語は人工言語であるが、その記述において変数や関数名といった識別子に自然言語の語が用いられる。例えば２．２節の調査結果において、調査対象者が、プログラム例②が最大値を求めるプログラムと説明できた一因には、maxという最大値を想起させる語が使われていることも考えられる。

本節の調査を行う上で、対象とするプログラム例はプログラミング言語Java（以降、単にJavaと記す）で記述されたものとした。理由はJavaのプログラム記述において識別子の命名制約がＣ言語に比べて、慣例的に厳しいと思われたからである。

Javaは、サン・マイクロシステムズにより、Ｃ++／Ｃ言語の代替となるプログラミング言語として開発された[5]。Javaの特徴は様々なものがあるが、オブジェクト指向を採用していることがあげられる。Javaでは、Ｃ言語にオブジェクト指向を取り入れて拡張したＣ++言語にも類似し、Ｃ言語の関数に相当する部分は、クラスに対してメソッドとして定義する機構になっている。

C言語の関数の命名に関しては、２．１節で調査した教科書でも、「関数のしくみを知る」として次のような記述があり、このような処理のまとまりに、例えば「引き出し処理」という自然言語の名前をつけ、関数により構造的にプログラムを記述することを説明している。

> *私たちは日常生活の中で、一定のまとまった処理を何度も繰り返して行うことがあります。たとえば、毎月自分の貯金からお金を引き出す場合について考えてみてください。このとき、預金を引き出すたびに次のような処理をおこなっていますね。*
>
> *通帳を自動現金支払機に入れる*
> *暗証番号を入力する*
> *金額を指定する*
> *お金を受けとる*
> *お金と通帳を確認する*
>
> *C言語でも、複雑なコードを書くようになってくると、たびたび行わなければならない一定の処理が出てくる場合があります。このような処理を、そのたびに何度も記述していくのはとても面倒です。そこでC言語では、一定の処理をまとめて記述する関数（function）という機能を用意しています。(p.216)*

さらに、オブジェクト指向では、処理の対象となる文字列や数値といった抽象物をオブジェクトととらえ、その抽象物を処理するメソッドを定義する。例えば、整数をオブジェクトとするなら、整数というオブジェクトの値を変えたり、値に他の整数値との加減乗除の演算を加えたりする処理をメソッドと呼ぶ。

このようなオブジェクトとメソッドの関係は、名詞に対する動詞の関係と密接な関係を持つ。例えば、ポチという犬クラスのオブジェクトに、「ポチを走らせる」、「ポチに食べ物を食べさせる」という処理を行わせることをrun(ポチ), eat(ポチ, 食べ物)と記述する、自然言語の述語項構造の記述特性をより反映した記述が求められる。

以上のような背景から、Javaで記述された数理プログラムの実装において、処理を定義するメソッド名に用いられる語を調査することを考えた。調査対象は、初中級の数理プログラム例題としてよく実装される、決定木とニューラルネットワーク（以下ＮＮ）のオープンソースプロジェクトを対象とした。具体的には、決定木、ＮＮの実装として、それぞれ６プロジェクトのJava言語で書かれたプログラムを選んだ。

ただし、Javaのプログラムであっても、プログラム作成者がクラス名、メソッド名に選ぶ語に自由度が存在するため、単なる識別子比較では検討が難しいことが実情である。これに対して、著者が所属する研究室では、プログラムソースコード向けのマイニングツール[6]を開発し、クラス名が異なっていても同一の役割を持つオブジェクト例の抽出を行ってきた。今回は、そのようなツールを用い、特定のオブジェクトに対して、類似性の高い機能をもつ部分プログラムをあらかじめ抽出した。そして、決定木およびＮＮをそれぞれ実装した６つのプロジェクトのうち、２プロジェクト以上で類似性の高い部分プログラムに同一のメソッド名が命名されているものを抽出した。抽出したメソッド名の一部を**表１**に示す。

表１中の語は、決定木およびＮＮのそれぞれの実装プロジェクトにおいて、複数のプロジェクトで使用されていたメソッド名にさらに制約を加えて抽出したものである。具体的には、メソッ

ド名が意味をもたない偶然の命名が一致してしまう可能性を排除し、より標準的な命名であることを担保するために、Javaのプログラム例が多数掲載された書籍ＡまたはＢの掲載プログラム例で使われていた語に絞った。具体的には、表中の書籍Ａの欄に掲載された語は、デザインパターンに関する教科書[7]のプログラム例でも使用されていた語である。同様に、書籍Ｂの欄の語はアルゴリズムに関する辞典[8]のサンプル例でも使用されていた語である。「書籍ＡおよびＢ」の欄は、複数プロジェクトで左記教科書の両方ともに出現するメソッド名である。

表１が示すように、教科書や辞典のプログラムサンプルの識別子に出現する語は、決定木とＮＮという数理的な機械学習アルゴリズムであっても、それぞれの実装特徴を反映して選択がなされる。そして、それらの語は開発者が異なっていても、共通して使用される語が存在する。

以上の知見は、**表１**のような、決定木やＮＮの実装という特定の目的のプログラム作成には、必要な専門的概念として共通に認識される語が存在するということである。すなわち、決定木プログラムの実装を説明するような自然言語の文章を、**図１**のような共起ネットワークで分析すると、「label」「split」「merge」「new」「key」「reduce」といったノードから構成される部分木は、それ以外の語に対応するノードから構成される部分木が示す概念より、決定木の概念を示している可能性が高いといえる。なぜなら、書籍のプログラム例に出現するような一般的な語であり、かつ特定のアルゴリズムのメソッド名として複数のプロジェクトで共通認識される自然言語の語が、プログラム中で、恣意的な意味として使われる可能性は低いと考えるからである。

すなわち、**表１**の語は、初学者向け教科書が導入するプログラムの基礎概念に対応する**図１**のグラフに、プログラム学習が進行するとともに付け加わっていく存在であると考える。そこで、次節では、プログラム学習が一定程度進んだ学習者に対してプログラムを説明する文章を**図１**のような共起ネットワークで分析し、そのグラフと**図１**とを対照させることにより、**図１**のグラフ中で表現された基礎概念をどのような位置づけとしてとらえるべきかを検討する。

表１．複数プロジェクトおよび教科書で命名されたメソッド名の例

	決定木	ＮＮ
書籍Ａのみ	label, iterator, factory	trace, iterator, factory
書籍Ｂのみ	split, merge, new, key, reduce	index, weight, error, verify
書籍ＡおよびＢ	set, value, count, add	list, array, line, input, next, get

表２．調査対象とした基本情報技術者試験問題の内容

試験期			疑似言語問題	Ｃ言語問題
2009	H21	春	図形の塗替え	絶対パスへの変換
2009	H21	秋	数値計算と計算誤差	多倍長整数の加算
2010	H22	春	マージソート	英文テキストの整形出力
2010	H22	秋	符号付き２進整数の乗算	バスの到着待ち時間
2011	H23	特別	組合せ	劇場の空き座席の確認
2011	H23	秋	代入文の処理	循環小数の出力
2012	H24	春	ビットの検査	会議時間の調整
2012	H24	秋	駅間の最短距離を求める	くじ番号当選番号確認
2013	H25	春	食料品店の値引き処理	放送サービスの料金計算

2013	H25	秋	文字列の圧縮	辞書順での文字列比較
2014	H26	春	空き領域の管理	英文テキストの編集
2014	H26	秋	編集距離の算出	利用者 ID の管理状況の確認
2015	H27	春	選択アルゴリズム	暗号文生成
2015	H27	秋	BM 法による文字列検索	入退室状況管理
2016	H28	春	簡易メモ帳のメモリ管理	フラクタル図形の描画
2016	H28	秋	数値の編集	開発文書最適化

4．基本情報技術者試験問題の調査

3．1 節で得た、プログラム作成の専門的概念の理解達成を語の使用を観点に調査した知見、また、3．2 節で得たプログラム中で使用される語に関する知見から、基本情報技術者試験の過去問題において、どのような専門概念が求められているかを、語の使用を観点に検討したい。

調査対象とするのは、基本情報技術者試験の2009年春期から2016年秋期まで 8 年間16回実施分の過去問とした。2009年春期からを対象としたのは、現行試験に至る制度改定後初めての試験となり、2016年秋期まで、ほぼ一貫した出題傾向がみられるからである。基本情報技術者試験は試験が午前の部、午後の部に分けられる。調査対象としたのは、プログラムの実例に関して出題がなされる午後問題のうち、全受験者の必須問題となる特定のプログラミング言語によらない疑似言語を使った出題（疑似言語問題と呼ぶ）、選択となるＣ言語を用いた出題（Ｃ言語問題と呼ぶ）とした。分析対象とした試験の内容を**表 2** に示す。

調査対象の試験問題の傾向を、2．1 節で得られた**図 1** と比較・対照するため、2．1 節と同様の手法により、プログラム言語のリストや、図・表を省いた自然言語で記述された文章に対して共起ネットワークを作成したものを、疑似言語問題およびＣ言語問題について、それぞれ**図 3** と**図 4** に示す。

図 1 と**図 3** を比較すると、プログラミング言語によらない疑似言語問題では、2．1 節で調査したＣ言語の教科書による導入された基礎概念を基本に、新たな概念が導入されていることがわかる。具体的には、**図 1** の概念に加え、**図 3** 中にはリスト、グラフ、探索、データベースといった概念と対応すると考えられる部分グラフが、基礎概念に加えて共起ネットワークを構成していることが伺われる。

そのような知見が得られる理由付けは、3．2 節のメソッド名に相当する語が共起ネットワーク内の部分グラフを構成していることに求められる。例えば、**図 3** における、「隣接」「経路」「最短」「距離」というノードは、**表 1** の決定木やＮＮのメソッド名に相当するため、それらのノードが関連付けられて構成する部分グラフは、プログラム技術としては典型的なデータ構造のグラフ構造の概念を表現していることがわかる。

むろん、このような部分グラフがプログラム技術のどのような概念と対応しているかを形式的に分析するためには、プログラム知識の計算辞書化という基盤資源開発が必要となるため、現状では主観分析に頼らざるを得ない。しかし、このような問題文に出現する言語表現に基づく問題分析は、**表 2** のような問題名の記述からだけでは明確に導きだせるものではないことに注意が必要である。

例えば、専門概念を、言語表現を基準に使って扱うこのような問題分析が行えなければ、3．2 節の調査資料として使ったアルゴリズム事典のような網羅的な学習資料を漠然と学習する他ない。すなわち、導入教材の言語表現に基づく分析が、基礎概念から何を指針に学習を行っていけばよいか、その道筋の明確化に役立つ可能性を示している。

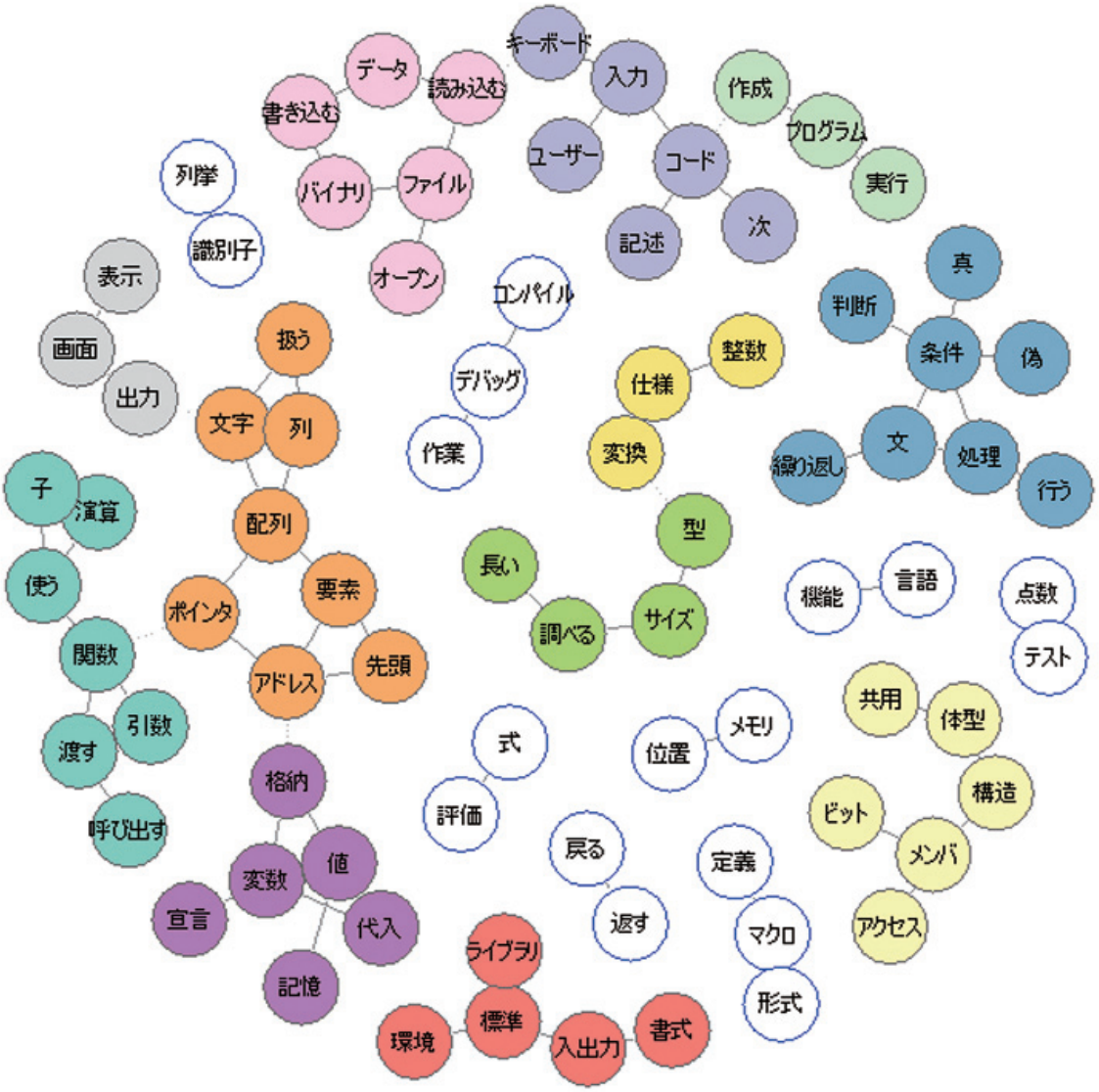

図 3．疑似言語問題に関する共起ネットワーク

図 3 と図 4 を比較すると、端的には基本情報技術者試験の出題傾向を見て取れる。具体的には、C 言語問題の共起ネットワーク図 4 では、疑似言語問題の図 3 に比べて、プログラムの目的が、表 2 の出題内容に関連する表現である「バス停」や「座席」、「回文」、「英文」といったノードが見られる。すなわち、C 言語問題が、疑似言語問題のような抽象的・一般的な問題ではなく、具体的な応用に根差した問題傾向を持つことが、共起ネットワークから見て取れる。なお、それらの語は 3．2 節で説明したようにオブジェクト指向のプログラミング言語では、オブジェクト名に採用される傾向がある語である。C 言語はオブジェクト指向を採用してはいないが、プログラムの自然言語説明上で使われた具体的対象や目的をもつ語が共起ネットワーク上に出現することが確認できる。

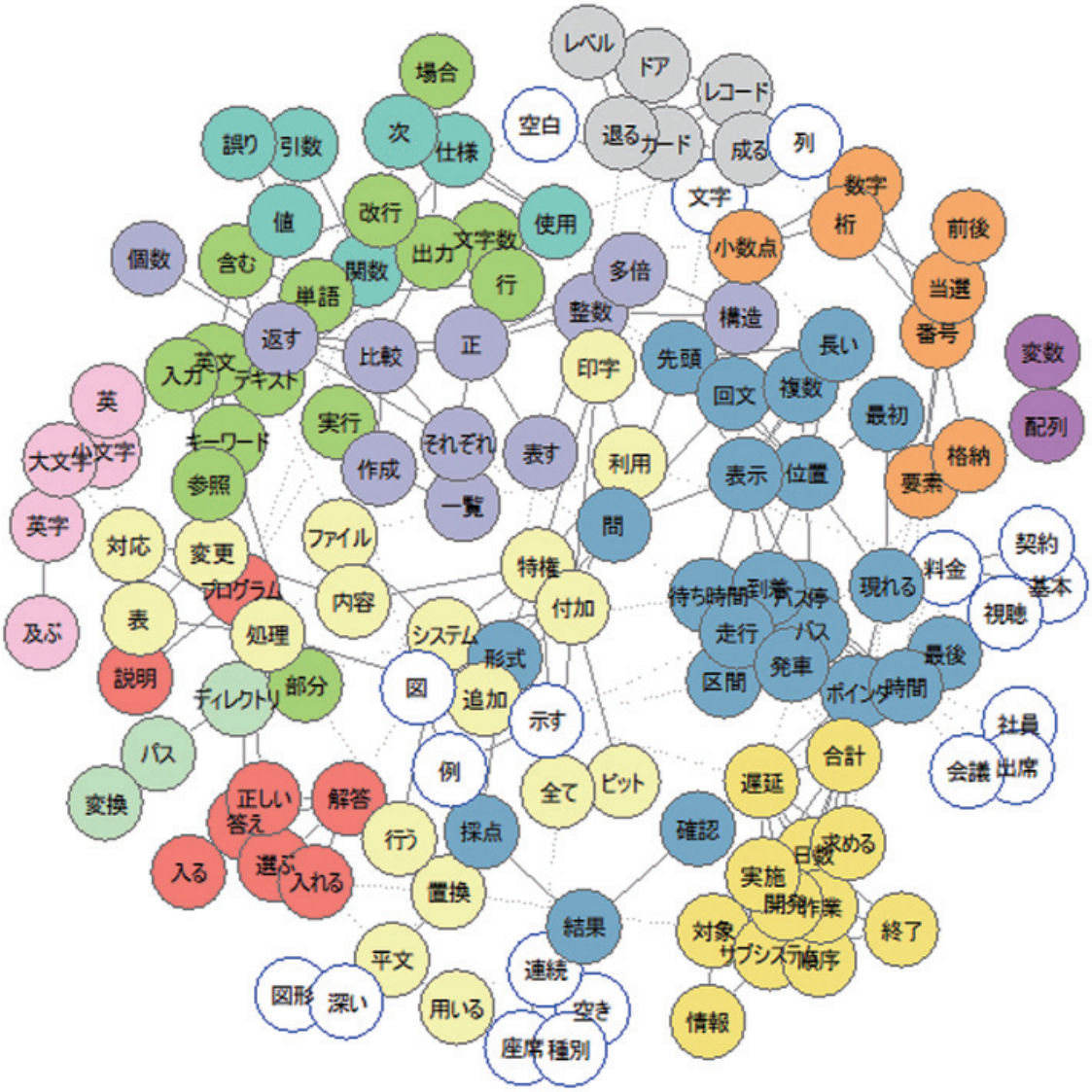

図４．C 言語問題に関する共起ネットワーク

他方、**図 4** のノードは、３．２節で述べたメソッド名相当語とクラス名相当語が混在しているため、疑似言語問題の共起ネットワークに比べて、図中の隣接ノードで色分けした、部分グラフの役割を認定しにくいという問題がある。そこで、基礎概念を共起ネットワークで示した**図 1** において、文字列に関係するように思われた、「文字」「配列」「要素」「先頭」「ポインタ」といったノードで構成される部分グラフを、C 言語問題文章の制約として加え、それで得られた自然言語の文章の共起ネットワークを示したものが**図 5** となる。具体的には、問題文章のうち、「文字」「配列」「要素」「先頭」「ポインタ」という表現を含む段落のみを対象文章として、共起ネットワークを作成した。なお、**図 5** のノードに選択した語の出現数は 7 回以上とした。

図 5 を見ると、段落単位の出現語を使った分析対象文章に対する制限は、プログラム作成の目的となる対象をどのようにモデル化・実装したかを知る手がかりになると考えられる。すなわち、基礎概念として学習した、**図 1** の共起ネットワークにおいて「文字」「配列」「要素」「先頭」「ポインタ」といった部分グラフを形成するプログラム技法が、その具体的な「テキスト」や「英文」あるいは「回文」という対象や目的をもつ問題で、応用されていることが示されていると考えられる。

共起ネットワークの部分グラフが、アルゴリズム、モデル化あるいはデータ構造の概念とどのように関わるかの分析は今後の課題であるが、図 5 で示される関連語ネットワークで表現される専門概念は、単純であった図 1 の基礎概念が段階的に発展した一形式であると考えている。

なお、図 3 および図 4 それぞれの整理・詳細化とさらなる分析、また、図 3 と図 4 の詳細な対照分析は今後の研究課題である。

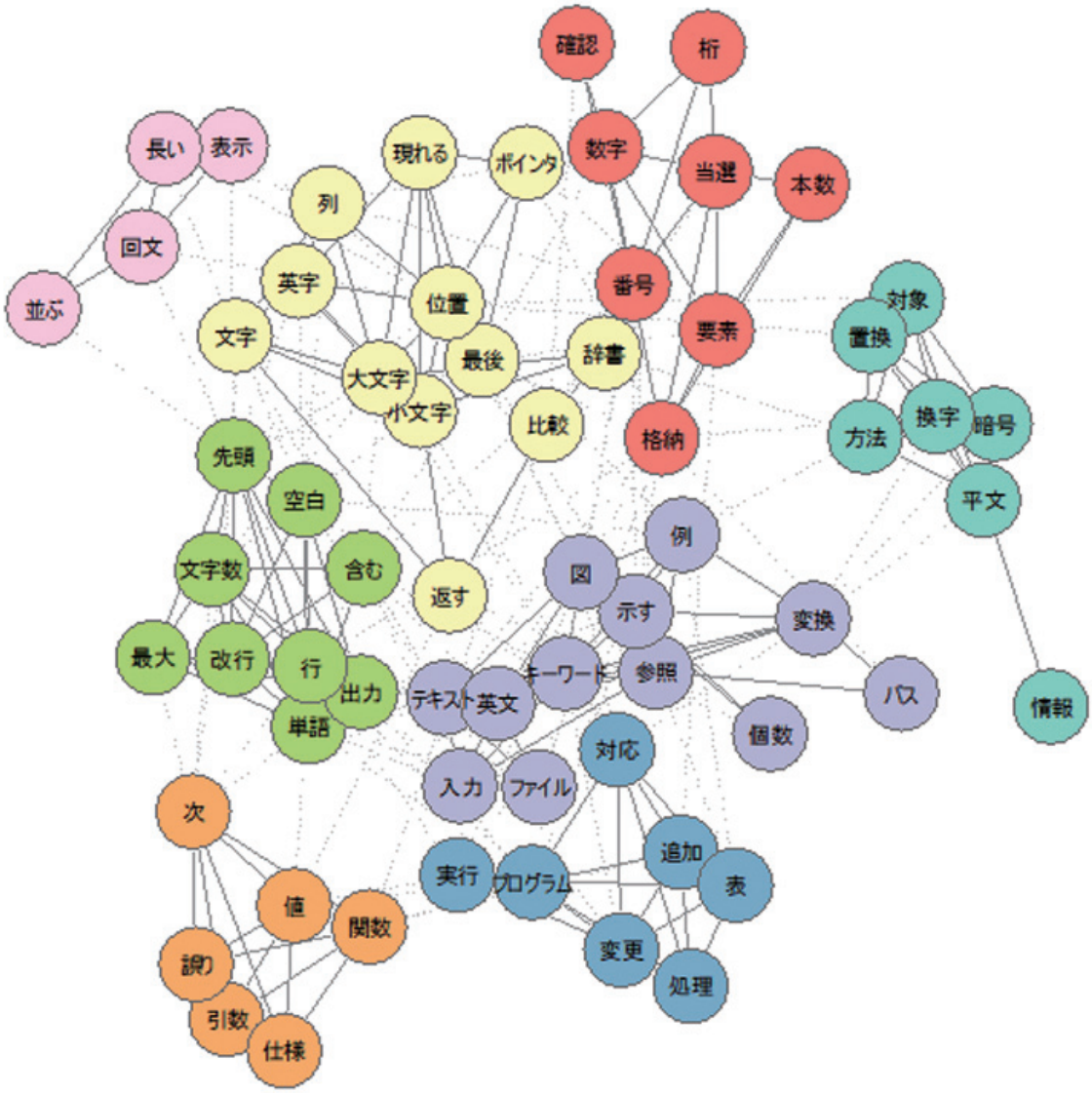

図 5．C 言語問題をテキスト関連問題に限定した共起ネットワーク

5．おわりに

本稿では、テキストマイニングの 1 手法である共起ネットワークを概念表現として導入し、まず、大阪電気通信大学情報工学科 1 年生で利用している教科書によりプログラムの基礎概念がどのように表現されるかを、自然言語の共起ネットワークを使って概観した。次に、プログラム初学者と実際のプログラム経験を積んだプログラム作成者が、それぞれ、どのような自然言語の表現を用いて、自らのプログラム理解を表明・説明するかを調査した。

さらに本稿では、そのような知見を前提に、基本情報技術者試験のプログラム作成に関わる過

去問題を対象として、言語表現を手掛かりに分析を試みた。その結果、プログラム初学者によるプログラミング言語であるＣ言語の学習が、プログラム作成における専門的概念の高度化に対応していく指針の基礎となる可能性を示した。

専門的なプログラム学習は、プログラミング言語の学習だけに留まらず、数多くのプログラム実例に触れ実践的に学ぶべきという意見がある。本稿の調査は、多様なプログラム事例のうちのどれが、特定の学習段階にある学習者が次の学習段階に進む上で適切かを判断する基準になりうると考えている。そのような学習者の学習段階をより個別かつ精緻に判定し、学習教材として有効なプログラム例を提示する仕組みを構築していくことが今後の課題である。

謝辞

本研究はJSPS科研費(基盤(C)課題番号15K01100)の助成を受けたものです。本原稿を完成にするにあたり、丁寧に有益なコメントを提供して下さった査読者諸氏に感謝します。

参考文献

[1] 高橋麻奈：『やさしいＣ第２版』、ソフトバンククリエイティブ、2003
[2] 樋口耕一：今日から始めるテキストマイニング ―計量テキスト分析の環境『KH Coder』―、石田基広・金明哲編著『コーパスとテキストマイニング』、共立出版、pp. 204-209、2012.
[3] 阪口祐介・樋口耕一：「震災後の高校生を脱原発へと向かわせるもの ―自由回答データの計量テキスト分析から―」、友枝敏雄編『リスク社会を生きる若者たち ―高校生の意識調査から―』、大阪大学出版会、pp.186-203、2015
[4] M. Resnick et al.: Scratch: Program for All, Communication of ACM, 52-11, pp.60-68, 2009
[5] J.Gosling et al.: The Java language specification, third edition, Addison-Wesley, 2005.
[6] 山下大貴•竹内和広：WordNet および Wikipedia と連携するソースコード上の関連語マイニングツール、Software Engineering Symposium (SES2014)予稿集、pp194-195、2014
[7] 結城 浩：『増補改訂版 Java 言語で学ぶデザインパターン入門』. 技術評論社、2008
[8] 奥村晴彦他：『Java によるアルゴリズム事典』. 技術評論社、2003