共通テスト「情報Ⅰ」プログラミング問題の解決に必要な知識
—初めての試験を終えて—

寺尾敦[*1]・秋岡敬子[*1]・中村優花[*1]・青木梨奈[*1]・井上真緒[*1]
Email: atsushi@si.aoyama.ac.jp

*1：青山学院大学社会情報学部社会情報学科

©Key Words　大学入学共通テスト，プログラミング，認知的課題分析，発話プロトコル

1. はじめに

大学入学共通テストでは，令和７年度より，「情報Ⅰ」の試験が新たに始まった[(1)]。この新たな試験での出題について具体的イメージを共有するため，大学入試センターは，2021 年 3 月 24 日に「サンプル問題」を，2022 年 11 月 9 日に「試作問題」を公表した[(2)]。2025 年 6 月時点での「過去問」は，追・再試験問題を別とすれば，これら３セットである。

大学入学共通テストは「深い理解を伴った知識の質を問う問題や、知識・技能を活用し思考力・判断力・表現力等を発揮して解くことが求められる問題を重視した問題作成」を行うとされている[(3)]。しかし，思考力，判断力，表現力といった「力」のメタファーはマジックワードであり，教育や学習において有効に機能しない[(4)]。生徒から，大学入学共通テストでよい点が取れるようにするためにどうしたらよいかと相談されたとき，「思考力を身につけよう」では，生徒はどうしたらよいのかわからない。どのような知識が必要なのかを具体的に示す必要がある。

本研究では，共通テスト「情報Ⅰ」のプログラミング問題に焦点を当て，過去問（サンプル問題，試作問題，令和７年度本試験）の認知的課題分析と，この問題を解決した大学生の発話プロトコル分析により，要求されている知識を明らかにする。

2. 方法

３セットの過去問（サンプル問題，試作問題，令和７年度本試験）それぞれについて，以下の３ステップで，解決に必要な知識を明らかにした。ただし，令和７年度本試験の問題については，ステップ２と３が未実施である。

ステップ１：認知科学を専門とする大学教員１名（本論文の第１著者）が，PC 画面に提示した問題を発話思考で（つまり，意識内容を発話しながら）解決した。問題解決中の PC 画面と発話は動画配信のためのソフトウェアで収録された。動画は YouTube で公開されている[(5-7)]。

ステップ２：次に，大学教員の発話プロトコルをもとに，認知的課題分析を行った。この分析では，この問題の解決に使用されたと考えられる知識を，プロダクション・ルールとして記述した。プロダクション・ルールは，手続き的知識の表現であり，IF-THEN 形式のルールである。条件（IF）節は問題の状態を，行為節（THEN）はその状態で行うべき操作を表す。

ステップ３：最後に，認知的課題分析の妥当性を確認するため，大学生の実験参加者に問題を発話思考で解決することを求めた。解答に誤りが含まれていた場合には，誤りのあった設問を指摘し，もう一度解決を求めた。それでもまだ正解を与えることができなかった場合には，認知的課題分析で得られたプロダクション・ルールに基づいて作成したヒントを提示した。大学生の発話が認知的課題分析によって提示されたプロダクション・ルールの使用を示唆するものであり，解決に躓いたときにヒントが有効に機能すれば，認知的課題分析の妥当性を支持する証拠となる。

3. 試作問題の分析

「試作問題」でのプログラミング問題の解決に必要な知識については，2024PC カンファレンスで分析結果を報告したので[(8)]，ここでは得られた知見の概要を述べる。

大学教員１名がこのプログラミング問題を解決したときの発話プロトコルから，問題解決に使われたと推測される知識として，日常生活に由来するルール，プログラミングでの領域固有のルール，汎用的な推論ルール，という３つのカテゴリのルールが見いだされた。

日常生活に由来するルールとは，日常生活で獲得されると考えられるルールである。試作問題では，買い物で客と店が交換する硬貨の枚数の最小値である「最小交換硬貨枚数」を求めるプログラムを作成する。ここで利用可能な，ある金額を支払うときにどの硬貨を何枚使うかという知識は，日常生活で獲得されると考えられる。

プログラミングでの領域固有のルールとは，配列を用いた繰り返し処理のように，プログラミングの学習で獲得されるルールである。

汎用的な推論ルールとは，類推のような，どの問題領域でも使用可能なルールである。試作問題を解決した大学教員は，この問題での重要な数量関係を最後に問われたとき，それよりも前に同じ関係が問われていることに気づいて，解答の整合性を確認していた。

この問題の解決を行った 10 名の大学生の発話プロトコルは，汎用的な推論ルールが利用されなかったことを除けば，認知的課題分析で得られたプロダクション・ルールによる問題解決と一致していた。このことは，認知的課題分析が妥当であったことを示している。

これらプロダクション・ルールを見ると，共通テストが測定すると称している「思考力」「判断力」「表現力」は，

それほど複雑ではない手続き的知識の適用であることが示唆された。これらプロダクション・ルールは，いずれも日常生活あるいは授業で獲得が期待できるものであり，獲得が非常に困難と思われるものはなかった。

4. サンプル問題の分析

4.1 認知的課題分析（ステップ１および２）

「サンプル問題」のプログラミング問題では，比例代表選挙での各政党の当選者数を，政党の得票数に応じて決める。出題文では，M さんと K さんの２人が，先生の支援を受けながらプログラムを改良していく。最初に，各政党の得票数を「基準得票数」（得票総数 / 議席数）で割って当選者数を決めるプログラムを書く（問１）。このプログラムでは各政党の当選者数が整数値にならず，当選者数をうまく決められない。そこで先生に相談すると，ドント方式を教えてもらえる。この手順を確認したのち（問２），プログラムを書く（問３）。最初は各政党の候補者数が十分に足りる（どの政党も，当選者数以上の候補者がいる）という前提でプログラムを書き，次に候補者数が足りなくなる場合にも対応したプログラムを完成させる。

大学教員１名の発話プロトコルから，この問題の解決に使われたと考えられる知識をプロダクション・ルールで表した。「試作問題」とは異なり，ルールはすべてプログラミングでの領域固有のルールであった。もちろん，問題文を理解するためには選挙について日常で得る知識が使われているはずであるが，ひとたび問題文が理解されれば，プログラミングに固有の知識だけで問題解決が可能であると考えられた。

図１は，問題に登場する生徒２人が最初に書く，各政党の得票数を「基準得票数」（得票総数 / 議席数）で割って当選者数を決めるプログラムである。空欄［ア］を考えているときの，大学教員の発話を表１に示す。各政党の得票数が配列 Tokuhyo に記録されているので，これらを合計すれば得票総数となる。発話からは，配列の要素をひとつずつ取り出すループで変数 sousuu の値を更新すること，変数 sousuu の初期値は 0 であること，最初の要素を 0 番とすれば最後の要素の番号は要素数より 1 小さいことに気がついていることがわかる。

表 1 での問題解決に用いられたと考えられるプロダクション・ルール（P1）を表２に示す。これは配列に記録されている数値を合計する典型的な方法であり，プログラミングでの配列の学習で獲得されると考えられる。図１の空欄［イ］および［ウ］も配列の要素の算術演算で解答できる。この処理を行うルール（P2）も表２に示す。

```
(01) Tomei = ["A 党","B 党","C 党","D 党"]
(02) Tokuhyo = [1200,660,1440,180]
(03) sousuu = 0
(04) giseki = 6
(05) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(06)     sousuu = sousuu + Tokuhyo[m]
(07) kizyunsuu = sousuu / giseki
(08) 表示する("基準得票数：",kizyunsuu )
(09) 表示する("比例配分")
(10) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(11)     表示する(Tomei[m],"：",[イ] / [ウ] )
```

図１　得票に比例した当選者数を計算するプログラム（「サンプル問題」問１）

表１　図１の問題を解く大学教員の発話

政党名の文字の配列それから得票数の配列があって，sousuu って書いてありますね。（中略） えーと「m を何とかまで増やしながら」。これ配列の操作かな。配列の操作だとたぶん 0 から 3 なんでしょうけれども。（中略） sousuu 最初 0 ですね。sousuu に Tokuhyo の m ですか。だから Tokuhyo の 0 番目を，A 党ですか。Tokuhyo の A 党の 1200 ってやつが sousuu に入るわけですよね。（中略） これループの外ですから，ループでやってるのは単純に全部足し算するだけの話ですね。えーとだからこれは 1200 から最後の 180 までを全部足すだけのことですよね。これ確かに「0 から 3 まで増やしながら」がよくて，だから [ア] は答えはとりあえず (3) でだいじょうぶですね。

表２　配列の要素に算術演算を行うプロダクション・ルール

P1：配列の要素の合計を求める IF 　目標が数値の合計を求めて変数 T に代入することであり， 　変数 T には初期値 0 が入力されており， 　合計される数値は配列 A の k 番目までに記録されている THEN 　ループ変数 i の値を 0 から k - 1 まで変化させて，T の値を T＝T＋A[i] と更新せよ。
P2：配列の要素を相対量にする IF 　目標が X を基準量とした相対量を求めることであり， 　相対量を求めたい数量が配列 A の k 番目までに記録されている THEN 　ループ変数 i の値を 0 から k - 1 まで変化させて， 　A[i] / X を計算せよ。

```
(01) Tomei = ["A 党","B 党","C 党","D 党"]
(02) Tokuhyo = [1200,660,1440,180]
(03) Tosen = [0,0,0,0]
(04) tosenkei = 0
(05) giseki = 6
(06) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(07)     Hikaku[m] = Tokuhyo[m]
(08) [セ] < giseki の間繰り返す:
(09)     max = 0
(10)     i を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(11)         もし max < Hikaku[i]ならば:
(12)             [ソ]
(13)             maxi = i
(14)     Tosen[maxi] = Tosen[maxi] + 1
(15)     tosenkei = tosenkei + 1
(16)     Hikaku[maxi] = 切り捨て( [タ] / [チ] )
(17) k を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(18)     表示する(Tomei[k],"：",Tosen[k],"名")
```

図２　改良されたプログラム（「サンプル問題」問３）

表３　図２の問題でのループの継続条件を考える大学教員の発話

tosenkei，合計が 0。giseki が 6 になっていて，（中略） 「何とかの間を繰り返す」ですから，そうですね，これだから，giseki，tousenkei，当選の合計ですよね。 tosenkei が giseki に届かない間繰り返せばいいってことでしょ。だから 6 になっていればこれ止まりますもんね。それがね。はい。だから [セ] は tosenkei。

表４　継続条件の判定で始まり変数の値更新で終わるループを構成するプロダクション・ルール

P3：ループのタイプを決める IF 目標はループを構成することであり， 更新される変数の値についてループの継続条件 C がある THEN 条件 C が満たされている限り処理を繰り返すループを構成せよ
P4：継続条件のあるループを構成する IF 目標は条件 C が満たされている限り処理を繰り返すループを構成することであり， 条件 C は変数 X の値が変数 Y の値より小さいことであり， 変数 X の値はループのたびに増加する THEN 最初に X<Y であることを確認し， 最後に X の値を更新するループを構成せよ

図 1 のプログラムでは各政党の当選者数が整数値にならず，当選者数をうまく決められない。この問題点を解決して図２のプログラムを作成する（問３）。

プログラムの 8 行目から，空欄 [セ] に入る値が変数 giseki よりも小さい限り処理を繰り返すループが始まる。空欄 [セ] の解答を考えているときの大学教員の発話を表３に示す。変数 giseki の値は 6 に固定されていて，選挙での議席定員数を表している。発話からは，変数 tosenkei の値はループのたびに 1 ずつ増えること，この値が giseki より小さい間繰り返しを行うこと，tosenkei の値が giseki の値である 6 と等しくなったらループが終わることに気づいていることがわかる。

表３での問題解決に用いられたと考えられるプロダクション・ルールを表４に示す。ルール P3 では，ループの継続条件があることに気づいて，この条件の判定を伴うループ（多くのプログラミング言語では while ループ）を選ぶ。続いて，継続条件の判定で比較される２変数が認識されると，ルール P4 により，ループの最初（継続条件の判定）と最後（変数の値の更新）での処理が決められる。ループの最初に条件判定を行い，条件判定に用いられる変数の値をループの最後に更新するのは，典型的なループ構造のひとつである。こうしたループを構成するプロダクション・ルールはプログラミングの学習で獲得されると考えられる。

表５　図２の問題で最大値を持つ配列の要素を見つける大学教員の発話

max を 0 にして，（中略） 「i を 0 から 3 まで 1 ずつ増やしながら」えーと，Hikaku[i]が，えー，max あー，要はあれですね，いつもの手ですよね。あの max っていうのはどこが大きいかを探すってやつですよ。 とりあえず最初 0 番にしておいて，Hikaku のですね i 番がもし max よりもでっかければ，（中略） [ソ] をやって，maxi を i にするんですよね。（中略） i 番目のところがいま一番でかいですよとやって，このループが終わると一番でかい数が入っている配列の要素がわかるってことですよね。

表６　配列中の最大値とその位置を見つけるプロダクション・ルール

P5：配列中の最大値とその位置を見つける IF 目標は，要素数 k の配列 A に記録されている数値の最大値と，その位置 INDEX を見つけることであり， 最大値を代入する変数 MAX は初期値 0 が入力されている THEN ループ変数 i の値を 0 から k - 1 まで変化させて， MAX ＜ A[i] ならば， MAX＝A[i] INDEX＝i と更新せよ

プログラムの 10 行目から 13 行目では，配列 Hikaku に保存されている数値の最大値とその位置を探している。プログラムのこの部分に取り組んでいるときの大学教員の発話を表５に示す。「いつもの手」と述べていることから，配列での最大値とその位置を探す手法になじみがあることがわかる。すなわち，変数 max にその時点での最大値を保持し，それと配列中の値を順に比較して，より大きな数値が見つかれば max の値を更新する。同時に，その位置を変数 maxi に保存する。

表４での問題解決に用いられたと考えられるプロダクション・ルールを表６に示す。これは配列中での最大値とその位置を探す典型的な手法である。

4.2　認知的課題分析の検証（ステップ３）

大学教員１名の発話プロトコルに基づいて行った認知的課題分析の妥当性を検討するため，大学生 11 名から，「サンプル問題」を発話思考で解決した発話プロトコルを得た。課題分析で同定したプロダクション・ルールと学生の発話に整合性があるか，プロダクション・ルールに基づいて作成したヒントは解決に躓いたときの助けになるかを検討した。

11 名の学生の問題解決は認知的課題分析で同定したプロダクション・ルールと整合的であった。ただし，選択肢から正解を選ぶという共通テストの出題形式を利用して正解をしぼることがあった。この場合，解決プロセスは認知的課題分析で想定したものとは異なる。

表７　図２の問題で最大値を持つ配列の要素を見つける学生の発話

プロダクション・ルールに基づく解決
つまり max が今の比較の中で一番でかくなるまで繰り返すってことか。今の比較の中で一番でかいやつが出てきたらそれに置き換えるって意味か。てことは今の max よりもでかい比較があったらそのでかい比較を max にしろって意味だから (2) 番だ (2) 番。
出題形式を利用した解決
max に何か足していくから [ソ] は max に何か入れんだよな。最初 0 からスタートで，じゃないかな。だから左側に max が来るのを選べばいいはず。うーん，max が Hikaku[i]より，Hikaku[i] 入れればいいのかな。

図２のプログラムの 10 行目から 13 行目で，空欄 [ソ] の正解を選ぶことのできた２人の学生の発話を，表７に示す。大学教員による問題解決では表５の発話と対応する部分である。表７上段の発話は課題分析で同定したプロダクション・ルールと整合性のある発話である。下のプロトコルは，多肢選択という出題形式を利用して，選択肢をしぼっている。

誤答があった場合，問題全体を解き終わった後で，学生はどこで誤っているかを指摘され，もう一度考えた。ここでも正解を得られなかったとき，プロダクション・ルールに基づいて作成したヒントが出された。ヒントはおおむね有効に機能した。たとえば，図１のプログラムでの空欄 [ア] がわからなかった学生に，得票数の合計を求めて変数 sousuu に代入しようとしていることと（表２のプロダクション・ルール P1 の IF 節），配列の要素に記録されている数値を合計するにはループを使えること（同じく THEN 節）を教示したところ，正解することができた。

この学生はプログラムの５行目と６行目で得票数の総数を求めようとしていることが理解できていなかった。本研究ではプログラムで実行しようとしていることを理解するためのプロダクション・ルールをまだ明らかにできていない。同定したプロダクション・ルールの IF 節はこの理解を反映しているため，プログラムが理解できなければ，ルールを使うことができない。文章題の解決過程の認知心理学的研究では，この解決は理解過程と解決過程から構成されると考える [9]。本研究で明らかにしたプロダクション・ルールは解決過程を導くものであり，理解過程で働く知識については今後の検討が必要である。

5.　令和７年度本試験でのプログラミング問題

令和７年度に初めて行われた共通テスト「情報 I」でのプログラミング問題については，大学教員１名（第１著者）による発話思考プロトコルは得られているものの，認知的課題分析に着手したばかりである。

認知的課題分析のためにこの問題を解いた印象としては，「試作問題」のように日常生活に由来するルールは必要なく，プログラミングでの領域固有のルールだけで解答可能である。問題の文脈は日常的（複数の工芸品を複数人で分担して製作するときの，各工芸品の担当者を決める）だから，問題とプログラムの理解過程では日常生活に由来する知識が必要となるかもしれない。しかし，問題の解決過程では，プログラミングで学習する知識だけで解答が可能なように思われる。

6.　おわりに：教育への示唆

大学入学共通テストは，思考力，判断力，表現力を測定するとされている。しかし，こうした「力」のメタファーはマジックワードであり，教育や学習において有効に機能しない。認知心理学的に考えれば，問題解決を支えているのはけっきょく知識である。

認知心理学では，辞書的，事実的な知識である宣言的知識と，物事のやり方についての手続き的知識を区別する。認知心理学での ACT-R 理論 [10] によれば，学習者は最初に，テキストや教師の説明によって，宣言的知識を獲得する。宣言的知識の検索と使用を繰り返すことで，やがて手続き的知識が生成される。プロダクション・ルールは手続き的知識の表現である。手続き的知識は言語化できないことも多い。意識しないでも使える，洗練された手続き的知識の獲得は，熟達化の鍵である。

プログラミングを教える教師は熟達者なので，自分が使っているプロダクション・ルールを明確に言語化できないかもしれない。しかし，問題解決に必要な知識を明らかにして，その獲得プロセスを考えなければ，適切な学習支援はできない。本研究での認知的課題分析は，こうした学習支援に貢献できるだろう。

謝辞

本研究は JSPS 科研費 JP23K02718 の助成を受けたものです。

参考文献

(1) 大学入試センター：“令和７年度試験”，https://www.dnc.ac.jp/kyotsu/kako_shiken_jouhou/r7/（2025 年 6 月 30 日アクセス）
(2) 大学入試センター：“（参考）試作問題等令和４年度までの検討状況”，https://www.dnc.ac.jp/kyotsu/shiken_jouhou/r7/r7_kentoujoukyou/（2025 年 6 月 30 日アクセス）
(3) 大学入試センター：“共通テストの仕組み等”，https://www.dnc.ac.jp/kyotsu/shiken_gaiyou/shikumi_unei.html（2025 年 6 月 30 日アクセス）
(4) 鈴木宏昭：“私たちはどう学んでいるのか：創発から見る認知の変化”，筑摩書房（2022）.
(5) 寺尾敦：“大学入学共通テスト「情報 I」サンプル問題（プログラミング）はこうやって考える”，https://youtu.be/LnwpnROYtDQ（2025 年 6 月 30 日アクセス）
(6) 寺尾敦：“大学入学共通テスト「情報 I」試作問題（プログラミング）はこうやって考える”，https://youtu.be/rgxM2KcHQwY（2025 年 6 月 30 日アクセス）
(7) 寺尾敦：“2025 年度大学入学共通テスト「情報 I」第３問（プログラミング）はこうやって考える”，https://youtu.be/pCuh2H6RG9s（2025 年 6 月 30 日アクセス）
(8) 寺尾敦，秋岡敬子，中村優花：“共通テスト「情報」プログラミング問題の解決に必要な知識―発話プロトコル分析―”，2024PC カンファレンス論文集，pp.131-134（2004）.
(9) 吉田甫・多鹿秀継：“認知心理学からみた数の理解”，第５章，北大路書房（1995）.
(10) Anderson, J. R.: “The atomic components of thought”, Lawrence Erlbaum Associates, (1998).