プログラミング学習における学習者の編集過程を考慮した不得意度判定方法の提案

2017SE062 大河 菜々　2017SE070 島津 祐衣
指導教員：蜂巣 吉成

1　はじめに

大学のプログラミング演習では問題を繰り返し解き，多くのプログラムに触れることで理解を深めることができる．しかし，教員がたくさんの演習問題を作り，採点をすることは負担が大きい．そこで，問題作成者の負担を軽減しプログラミング学習を支援する方法として，空欄補充問題の自動生成が提案されている．空欄補充問題の自動作成では問題に学習意図を考慮させることなどの学習者のプログラミングの理解度を高めるような工夫がされている [1][2][3]．これらの研究では学習者の不得意とするプログラミングの箇所についてわからないので，効率よく学習を行うことができない．学習者がプログラミング言語を効率よく学習するには，学習者の不得意とするプログラミングの箇所を把握する必要がある．

本研究では，プログラミング学習における学習者の編集過程を考慮した不得意度測定を行い，学習者が自分の不得意な箇所を把握することで，学習者が効率よく学習できることを目的とする．プログラムを記述する形式の演習問題において，学習者の編集途中のソースコードを取得し，不得意な箇所の判定を行う．本研究で学習者の不得意な箇所を判定し，学習者が自身の不得意な箇所を知ることで，個々の学習者に対して不得意度の箇所を空欄にした空欄補充問題を出題することが可能になり，学習者は効率よく学習を行うことができる．

2　関連研究

加藤ら [1] が提案した，理解状況度に着目した空欄補充問題によるプログラミング学習支援システムは，解答の正誤や解答時間，誤解答数，解答の種類を用いて理解状況度を算出し，その値をもとに問題の難易度を変化させる．学習者のプログラミング学習の理解状況度に対応はできるが，学習者がどの学習項目の問題をわかっていないかを判定することができない．本研究では，学習項目ごとに誤りパターンを作成することで学習者の理解していない学習項目を判断することができる．

杉浦ら [2] が提案したプログラミング学習のための理解度モデルを用いた自動出題システムは，算出した理解度をもとに点数を加算，減算し 100 点に達することを目標として問題を解く．100 点に到達した際学習者の全体的なプログラミング学習の理解度は上昇するが学習項目ごとの理解度には差がでてしまう．本研究でははじめに演習問題を行いその結果から判定することで学習者がもっとも不得意としている学習項目を発見し，効率的な学習を行うことを目指す．

3　不得意度測定方法の提案

3.1　概略

本研究は，学習者のプログラミングにおける不得意な箇所を把握することが目的である．そのため学習者の不得意な箇所を知るための学習項目の設定が必要である．また，学習者の不得意な箇所を客観的に判断するためにどこがどの程度不得意であるのかを数値で表す必要がある．

本研究では学習項目の分類を行い，それをもとに作成したモデルや，そのモデルを利用して項目ごとの不得意度を設定することを提案する．

3.2　学習項目モデル

プログラミングの学習項目の分類を行い，入出力・演算，関数，条件分岐，繰り返し，配列，文字列，ポインタ，構造体を対象とする．分類の決め方について，本研究はプログラミング初心者に対してのツールの提案であり，学習項目とはプログラミング初心者の理解すべき項目であるので，教科書を参考に決めるのが良いと判断し 8 つの項目を抽出し学習項目とした．

分類した学習項目をもとに学習項目モデルを作成した．杉浦ら [2] は理解度モデルを用いた自動出題システムの提案を行っているが，学習者に合った問題を出題するために学習者は空欄問題を解かなくてはならない．本研究では記述式の演習問題から学習者が不得意とする箇所を判定する．杉浦ら [2] が作成した理解度モデルを参考に新しいモデルを作成した．

学習項目モデルは，設定した学習項目を根として，学習者の不得意とする箇所を詳細に判断できるような木構造とした．木のノードは構文と静的意味に大きく分けられる．学習者が間違いのあるソースコードをコンパイルした際，構文の間違いによるコンパイルエラーのとき，その間違いを学習項目モデルの構文のノードに分類する．文法的に正しいが C 言語のソースコードとして，適切な記述でない間違いをしていた場合のコンパイルエラーや用意した誤りパターンにマッチするとき，その間違いを学習項目モデルの静的意味のノードに分類する．静的意味とは，構文的には適切に記述できない構造的な制約を記述したものである．

学習項目モデルの例を図 1，図 2，図 3 に示す．

図 1 は配列の学習項目モデルである．構文，宣言の子の「要素数」は配列の要素数の宣言に関わる誤りに対応する．例えば，配列の要素数を与えずに宣言している誤りなどである．「初期値」は配列の初期化に関わる誤りに対応する．

図 1　学習項目モデル (配列)

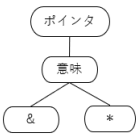

図 2　学習項目モデル (ポインタ)

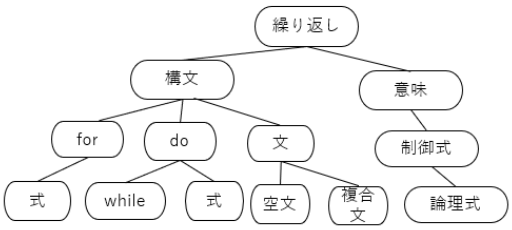

図 3　学習項目モデル (繰り返し)

例えば，{} で囲わずに配列の初期値を設定するなどである．「代入」は配列に要素を代入することに関わる誤りに対応する．例えば，文字配列に文字列リテラルを代入する誤りなどである．意味の子の「初期値」は配列の要素と要素数の関係に関わる誤りに対応する．例えば，配列の要素数を超える数の初期化子を与えている誤りなどである．「代入」は配列の代入に関わる誤りに対応する．例えば，初期化子を代入している誤りなどである．「要素」は配列の要素に関わる誤りに対応する．例えば，要素数が整数定数でない誤りなどである．

図 2 はポインタの学習項目モデルである．意味の子の「&」はポインタの&に関わる誤りに対応する．「*」はポインタの*に関わる誤りに対応する．

図 3 は繰り返しの学習項目モデルである．構文，for の子の「式」は for 文の制御式に関わる構文の誤りに対応する．例えば ; で区切って 3 つの式を記述することがわかっていない誤りなどである．do の子の「while」は do 文における while の記述に関する誤りに対応する．例えば while が無いなどの誤りである．文の子の「空文」は for (式; 式; 式); のように記述している文の誤りに対応する．意味，制御式の子の「式」は do 文における制御式に関わる誤りに対応する．静的意味にあたる制御式の誤りは意味，制御式の子ノード「論理式」に分類される．

3.3　不得意度の測定

不得意度とは，学習者がどの項目が不得意であるのかを数値で表したものとする．あらかじめコンパイルエラーメッセージや間違っているソースコードパターンを用意しておき，学習者が演習問題となるソースコードを記述し，コンパイルした際に用意したパターンとマッチした場合，そのパターンと対応する項目の不得意度を 1 加算する．この仕組みをもとにすべての演習問題を解き終わった際に不得意度の値が大きい項目が学習者の不得意としているものとする．その他の不得意度の決まりは以下に示す．

- コンパイルされた際にパターンマッチングを行う
- 1 つのソースコード内に同じ項目の誤りが複数あった場合，誤っている箇所すべてが加算される
- 誤ったソースコードをコンパイルした後，再度ソースコードをコンパイルした際，誤っていた箇所が修正されていない場合再び対応する項目の不得意度が加算される
- 不得意度の初期値は 0 である
- 不得意度は学習者の中の相対評価であり，他人との比較は行わない
- ツール使用毎に 1 から測定し 1 回ごとのツール使用でリセットされる
- 複数の不得意度が大きい場合，複数が不得意であるとする
- 学習者の解答が不正解であるがパターンにマッチしなかった場合や，学習者が問題を解かなかった場合には不得意度はどの項目にも加算されない

本研究では，学習者の不得意な箇所というのは問題を解くにあたって何度も繰り返し間違えるものということを前提としている．不得意度は加算式で算出し，上限値を設定しない．1 つのソースコード内に同じ項目の誤りが複数あった場合，誤っている箇所全てが加算されるという決まりについて，1 つのソースコード内であれば同じ誤りは 1 つ分の誤りであるとして 1 のみ加算するという考え方もできる．しかし，何度も同じ誤りをするのはその誤りが学習者に定着してしまっていると考えられるので誤りの個数分を不得意度に加算する決まりとする．

学習者の解答が不正解であるがパターンにマッチしなかった場合，ソースコードの中に誤りがあることは間違いないが，設定した学習項目のうちどこに該当するのか判定ができない．また，学習者が問題を解かなかった場合も同様で，どの学習項目がわからなかったのか，問題の解き方の方針そのものがわからなかったのかなどの判定ができない．これらの状況の際は不得意度はどの項目にも加算されない決まりとする．

学習者のプログラミング理解状況によっては，学習者ごとに不得意であると判定された項目の不得意度の値に差がでる場合があると考えられる．しかし，本研究で提案するツールは，ツールを利用する学習者の不得意なものを判定することが目的であるので，他人との比較は不要である．不得意度は学習者の中の相対評価であり他人との比較を行わないことと設定する．また，誤りは複数の学習項目モデルのノードと関連がある場合があるが，最も関連が強いノードと対応づける．

3.3.1 エラーメッセージ

学習者の不得意な箇所を判定する際に使用するエラーメッセージと，それらが対応する学習項目モデルの場所を表 1 に一部示す．エラーメッセージは gcc バージョン 5.4.0 のものである．

表 1 エラーメッセージ

概略	エラーメッセージ	対応する場所
double d[10], i; d[i] = 0.0; のように配列の添字が実数	array subscript is not an integer	配列→意味→宣言→要素
char str[128]; str = ”abc”; のように文字配列に文字列リテラルを代入	assignment to expression with array type	配列→構文→代入
int a[3]=2; のように {} で囲わずに配列の初期値を設定	invalid initializer	配列→構文→宣言→初期値
ポインタ変数の前の*つけ忘れ	assignment makes pointer from integer without a cast	ポインタ→意味→＊
関数呼び出しの実引数の&のつけ忘れ	passing argument 1 of ‘ swap ’ makes pointer from integer without a cast swap(na, nb);	ポインタ→意味→&

3.3.2 コードパターン

コンパイルした際エラーにならない間違いは，エラーメッセージだけでは判定できない．間違いパターンを問題ごとに用意しておき，パターンマッチングにより判定する．パターン例を表 2 に示す．${:ID_VF} は変数名，${:OP} は演算子，${:LIC} は文字リテラル，${:EXPR} は式を表す．

3.4 不得意度の測定例

実際にどのように不得意度が測定されるのかを，2019 年度の加藤らの研究「プログラミング演習における複数の観点を用いた指導が必要な学習者の特定方法の提案」[6] で得られたプログラミングを学習済みの学部 3 年生が実際に記述したソースコードから示す．問題を下記に示す．

問題 1. 整数 x の n 乗を求める関数 power を作成せよ．ただし，n は自然数 (n>) とする．

問題 2. 実数 x の n 乗を求める関数 power を作成せよ．ただし，n は整数とする．

表 2 コードパターン

概略	コードパターン	対応する場所
演算子の間違い	${:ID_VF}==${:ID_VF} ${:OP}${:ID_VF}	入出力・演算→構文→演算
制御文の本体が空文	if({$EXPR)};	条件分岐→構文→文→空文
	for ({$:EXPR}; ${EXPR};${EXPR});	繰り返し→構文→文→空文
文字列でない配列を ‘ \0 ’ と比較	${:ID_VF}[${:ID_VF}] ${:OP}${:LIC}	配列→意味→要素

問題 3. 秒を分, 秒の形へ変換する void 型関数 time(int s, int *min, int *sec) を作成せよ．変数 s は変換前の秒数であり，min，sec は結果を返すポインタである．

問題 4. 自然数 a，b の最大公約数を求める関数 euclid(int b,int a) を「再帰関数」で作成せよ．

問題 5. 文字列 s の中に，文字 c が含まれていれば，その文字（複数含まれる場合は，最も先頭側の文字）へのポインタを返し，含まれていなければ NULL を返す関数 strchr(char *s,char c) を作成せよ．

学習者 A の不得意度を図 4 に示す．問題 2，問題 4 において未定義の変数を使用していたので，入出力・演算→意味→演算の点数が 7 点と高くなっている．問題 5 において，

```
while(s!=\0){
if(*s==c){
return *c;
}
```

などのコードを記述して，ポインタ変数の*の付け間違いが多く，ポインタ→意味→*が 13 点と高くなっている．

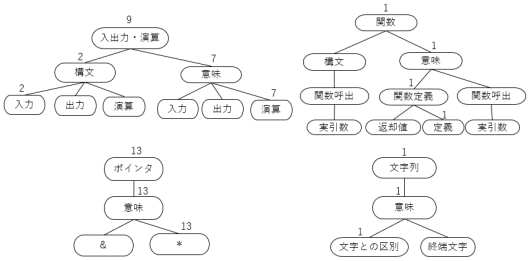

図 4 学習者 A の不得意度

学習者 B 不得意度を図 5 に示す．学習者 B は問題 1，2 で for(s=1，s=n，s++) や for(x=1，n，n++);{ と記述する間違いが多く，繰り返し→構文→ for →式が 12 点，文→空文が 4 点と高くなっている．問題 1，2，5 で関数定義をするときに return を記述しない間違いが多く，関数→意味→関数定義→返却値が 9 点と高くなっている．

実際の演習問題におけるソースコードの編集過程から不得意度を測定できることが確認できた．

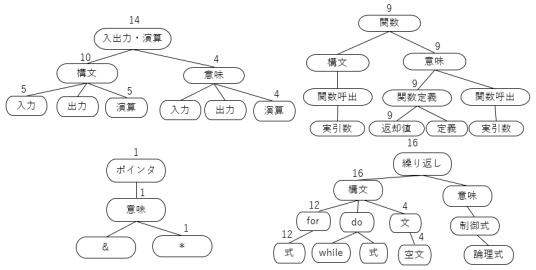

図 5 学習者 B の不得意度

4 考察

不得意度の測定方法の妥当性について考察する．今回提案した方法により，学習者は演習問題の間違えた箇所から，不得意な箇所について把握することができる．本研究では，あらかじめ間違っているソースコードパターンやコンパイルエラーメッセージパターンを用意しておき，学習者が演習問題となるソースコードを記述し，コンパイルした際に用意したパターンとマッチした場合，そのパターンと対応する項目を不得意とした．しかしこれでは，パターンに当てはまらない間違いをした場合，間違えているにもかかわらず不得意と判定されない．問題別に正解パターンも用意し，学習者が本当に正しいソースコードを書けているのかを判定する方法について検討する必要がある．

本研究ではコンパイルエラーメッセージと間違っているソースコードパターンを用意したが，ある問題では間違いであるソースコードパターンであっても別の問題では正解であるソースコードパターンであるという場合がある．間違いであるソースコードパターンをすべての問題共通で使用にすることが難しい．また同じコンパイルエラーメッセージパターンであっても学習項目に分類したときに別の分類を示すコンパイルエラーメッセージパターンのときがある．そこで，問題に依存した間違ったソースコードパターンを作成することや，コンパイルエラーメッセージとソースコードパターンをうまく組み合わせることで不得意な箇所を特定する精度を上げることができると考える．また，記述式の演習問題であるので実現は難しいが，正しいソースコードパターンを用意できれば 3 つのパターンを用いて判定を行うことで，より不得意な箇所を特定する精度を上げることができると考える．

本研究では不得意度の測定の際，加算式を用いている．学習者がコンパイルをした際にパターンマッチングを行い不得意度の加算を行うので，ケアレスミスであるにも関わらず不得意度が大きくなる可能性がある．これらを解決するのに，不得意度の加算が行われたあと再びコンパイルを行い，すぐに間違いが訂正されていたら不得意度を減算する方法や，() や {} のつけ忘れなどの不得意度が一度しか加算されず，ケアレスミスと思われる場合は不得意度を減算する方法についても検討する必要がある．

本研究では演習問題のソースコードを学習項目ごとに分類し，分類した項目が木構造の根となる学習項目モデルの，葉にあたる項目を最終的な不得意な箇所とする項目としている．大まかな初めに分類した木構造の根にあたる学習項目の中からどの学習項目が不得意であるかを知りたい学習者がいる可能性があり，それに対応する必要がある．本研究で提案する不得意度測定方法では演習問題を解き終わったときにそれまでの解答で得た不得意度を項目ごとに加算し最終的な不得意度を算出する．学習項目モデルの根の項目の中から不得意な項目を選びたい場合，それぞれの学習項目の葉の不得意度を加算すれば良い．学習者が演習問題をすべて解き終えたとき，関数の学習項目モデルの葉である構文に分類される実引数のノードに不得意度が 1，返却値のノードに不得意度が 0，意味に分類される実引数のノードに不得意度がであった場合，根である関数のノードの不得意度は葉の値を合算して 3 である．例に示したように，提案する方法を利用して新しい不得意度を算出することができるのでこのような不得意度算出も検討する必要があると考える．

5 おわりに

本研究では，プログラミング学習における編集過程を考慮した学習者の不得意なものの判定方法の提案を行った．不得意な箇所の判定のために学習項目モデルを用い，不得意度を設定した．学習者がソースコードのコンパイルを行う際に行うパターンマッチングや，コンパイルを行った際のエラーメッセーに応じて不得意度が算出される．今後の課題として提案方法に基づくシステムを実現し，実際の演習に用いることがあげられる．

参考文献

[1] 加藤 拓也，森 拓矢，沖 良太：理解状況度に着目した空欄補充問題によるプログラミング学習支援システムの提案, 南山大学 2011 年度卒業論文，(2012).

[2] 杉浦 啓太，寺内 雄基：プログラミング学習のための理解度モデルを用いた自動出題システムの提案, 南山大学 2013 年度卒業論文，(2014).

[3] 有安 浩平，池田 絵里，岡本 辰夫，國島 丈生，横田 一正：「学習者に合わせた C 言語演習穴埋め問題の自動生成」 第 1 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2009), 5 pages， (2009).

[4] 柴田望洋：新・明解 C 言語入門編，SB クリエイティブ (2014).

[5] 長谷川 靖成, 大竹 諒, 田島 侑典：プログラミング学習における意図を考慮した空欄補充問題の自動生成，南山大学 2012 年度卒業論文, (2013).

[6] 加藤 芳基，加藤 祐樹：プログラミング演習における複数の観点を用いた指導が必要な学習者の特定方法の提案，南山大学 2019 年度卒業論文, (2019).