失敗知識を利用したプログラミング学習環境の構築

知見　邦彦[†]　　櫨山　淳雄[†a)]　　宮寺　庸造[†b)]

A Programming Learning Environment Focusing on Failure Knowledge

Kunihiko CHIKEN[†], Atsuo HAZEYAMA[†a)], and Youzou MIYADERA[†b)]

あらまし　プログラミング技術の習得には学習者の積極的な姿勢や努力が必要である．その中でも自分の失敗を内省することはプログラミング技術を定着させるために重要な過程である．また，教授者が学習者の学習状況を把握することはプログラミング教育を行う上で重要となるが，非常に困難である．そこで本論文では，学習者に内省を行わせるために学習過程における失敗情報を利用した学習環境を構築し，適用実験を行った結果について述べる．

キーワード　プログラミング学習環境，失敗知識，失敗学，内省

1. まえがき

近年，情報教育が盛んに行われている．その中でも学習者が実際にプログラム言語を利用し，プログラミング技術を習得する教育は重要視されてきている．現状のプログラミング教育では，教授者が学習者に対して演習課題を提示し，学習者がその演習課題に取り組む形がほとんどである [15]．プログラミング教育では，プログラム言語の文法やアルゴリズムに関する知識のみでなく，学んだ知識を用いて具体的な演習課題を解決できる能力及び演習の過程で生じる問題を解決できる能力を養成することが重要である．このとき，学習者が参考書やその他の資料などを参照し，解決法などの本質的な理解をすることなく演習課題を完成させてしまうことがある．しかしこれでは学習者が実際にプログラミング技術を習得できたとは言いがたい．教授者の視点では，学習指導や授業設計の参考のため，学習者の学習状況や理解度を把握する必要がある．しかし，プログラミングの演習課題を提示しても，最終的な成果物であるソースプログラムとその実行結果を提出させるだけでは学習状況や理解度を測ることはできない．これらを把握するためには，最終的な成果物よりも問題解決に至るまでのプロセスを把握する方がより重要となるが，教授者がそのプロセスを把握することは非常に困難である．

一方，「失敗から新たな知識を学ぶ」という概念をもつ失敗学 [1] が注目されている．発生した失敗を否定的にとらえるのではなく，肯定的にとらえて同様の失敗を繰り返すことを防ぐために利用するという学問である．失敗学は機械設計の分野を起源としているが，その考え方は多くの分野で適用可能なものである．失敗学では，失敗情報を知識化することにより学習効果が得られると期待されている．また，過去の失敗を振り返るという行為は一般的な学習における内省とみなすことができる．内省とは自らの問題解決過程を振り返る行為であり，課題演習を通した学習において最も貢献する過程である [12]．

学習者は内省を行うことにより，プログラミングの知識を定着させることが期待できる．しかし，内省を行うためには動機付けやメディアが必要であるとされており，学習者が内省を行うことは容易ではない [3]．また，教授者は学習者の内省行為を見ることにより，どのように問題解決を行ってきたかという学習プロセスを把握することができ，授業へのフィードバックが期待できる．これまでに多くのプログラミング教育支援環境が開発されている（例えば [4], [6], [11], [13]）が，以上のような失敗学に基づいた内省促進によるプログラミング教育支援手法は見当たらない．

そこで我々は，プログラミング学習において失敗学

[†] 東京学芸大学，小金井市
Tokyo Gakugei University, 4–1–1 Nukui-kita-machi, Koganei-shi, 184–8501 Japan
a) E-mail: hazeyama@u-gakugei.ac.jp
b) E-mail: miyadera@u-gakugei.ac.jp

に基づいた内省を行うことにより学習効果を向上させることを考えた．本論文では，学習者に，失敗学に基づいた内省を促進させるためのプログラミング学習環境を提案し，失敗学に基づいたプログラミング教育支援の有効性の検証を行うことを目的とする．

2. 関 連 研 究

プログラミング教育を支援するシステムはこれまでに数多く提案されてきた．その中で森らはプログラムの正しさを学習することに焦点を当て，教材作成システムの開発を行っている [6]．教授者が作成した教材をもとに学習者が Web ブラウザを利用してプログラムを作成すると，システムが自動的に正誤判定を付加し，その正誤判定を含めた実行結果を得ることができる．このシステムにより学習者は正しいプログラムについて学習することができるとしている．我々のアプローチはこれとは逆で，自らの失敗を内省させ，問題解決能力の向上を目指そうとしている．

高橋らは C 言語を対象としたプログラミング学習のための電子学習環境を構築している [11]．学習者が自らのプログラムの失敗を訂正し，理解した結果をメモに残す一連の学習を支援している．この学習環境においても本論文と同様，エラー情報を利用している．そしてそのエラー情報をもとに，エラーに対する理解を深めていくことによる学習効果を期待している点は筆者らも同じである．しかし，彼らのシステムが自動取得するエラーはコンパイル時のエラーのみであり，実行時に OS（Operating System）から発生するエラーについては学習者に記入させている．これら二つのエラーに加え，エラー出力はされないが学習者の意図する結果が得られない場合についてもエラーと定義することができるが，このことに関して述べられていない．またこの学習環境では，用意されたボタンをクリックすることで作成したプログラムのコンパイルや実行を行う．しかし，UNIX 上でこれらの行為を行う場合にはコマンドを入力する必要があり，専用のボタンを用意してしまうとプログラミング学習に付随する UNIX コマンドの知識を習得することができなくなってしまう．このことに関しても教育的配慮が必要である．

一方，内省支援を目的とする手法やシステムも考案されている．

認知療法の分野では，自分のもつ極端な考え方から適応的（現実的で合理的）な考え方に変えていくため，七つの項目（状況，気分，自動思考，根拠，反証，適応的思考，心の変化）を提供し，対象者に記述させている [8]．構造化された項目を提供し内省を促すという考え方は筆者らも同じであるが，対象領域や記述項目は異なっている．また，認知療法の分野では計算機による支援は主眼ではない．

平嶋らは失敗を認識しにくい力学を対象とし，学習者にその失敗を認識させるために EBS（Error-Based Simulation）と呼ばれる失敗のシミュレーションシステムを提供している [3]．EBS とは学習者が作成した失敗を含む式をもとにシミュレーションを行い，学習者が予測するものと異なった現象を提示することにより認知的葛藤を発生させるものである．この認知的葛藤を内省の動機付けとさせている．内省の動機付けを提供するという点では本論文と共通するが，どのように内省を行わせるかについては触れられていない．

中原らは，教師が他者との相互作用を通じて，自身の教育実践の内省を促す TET（Teacher Episode Tank）を開発した．内省はリフレクションボードという画面で，タグ付きアイコンとリンクを使用して行う [7]．システムが内省の動機付けを与えるという点では共通するが，本研究とは対象領域や内省支援方式が異なる．

以上に挙げたように，プログラミング教育において学習者支援の研究が多くなされているが，プログラミング教育における問題解決過程での内省プロセスを支援するものは存在していない．また近年，教授者に対する支援も重要視されてきている [2], [5]．学習者に内省のためのメディアを提供することにより，教授者が学習者の内省の様子を観察することができるという教授者支援にもつながる．

本論文では，プログラミング学習における内省に着目し，失敗学 [1] の理論に基づいて失敗情報を知識化することにより内省を行わせる，C 言語を対象としたプログラミング学習環境を提案する．本学習環境では，内省の動機付けとなる失敗情報には論理エラーを含む C 言語におけるすべてのエラーを利用し，プログラムのコンパイルや実行は教育的配慮から UNIX のコマンドベースで行わせる．

3. 学習環境の提案

3.1 支 援 対 象

本論文では，プログラミング学習を行う学習者を対象とし，教授者によって提示された演習課題に取り組む中で学習者に内省を促進させる学習環境を利用する

表 1　失敗知識の属性
Table 1　The attributes of failure knowledge.

属性	失敗学による定義	本論文によるプログラミング教育における定義
事象	どのようなことがあったか	発生したエラー
背景	事象が起きた全体の背景	演習課題の内容
経過	どのように進行したか	エラーが発生したときのソースプログラム
原因	推定した原因と確定した原因	修正前に推定した原因と修正後に確定した原因
対処	失敗に対してどのように行動したか	発生したエラーに対して行ったこと
総括	全体のまとめ	このエラーから学んだ教訓の記述

ことを想定する．内省の軌跡を残すことで学習効果を上げるだけでなく，教授者がその軌跡を見ることができる．このように，学習者支援と教授者支援の両面に対する支援を行う．

3.2 内　　省

内省は一般的な学習において重要な過程であり [12]，プログラミング学習においても有効である [11]．しかしながら，学習者が内省を行うことは必ずしも容易ではない．内省を行うにはその動機付けとして自分の問題解決過程を振り返り，過去の失敗を認識する必要があり，内省のためのメディアも必要となる [3]．内省を行わなければ，たとえ失敗が解消されたとしても再び同じ失敗を犯す可能性が高いといわれている [9], [14]．しかし，現状のプログラミング学習では学習者がどのような失敗を経験したかを把握することは困難である．そこで，学習者に対して内省を行う学習環境を提供することで，プログラミング学習の支援を行う必要がある．本論文における内省は，失敗学 [1] の理論に基づき学習過程において発生した自らの失敗情報をもとにして，失敗知識を記述すること（失敗の知識化）により実現する．

3.3 失敗情報と失敗知識

本論文ではプログラミング教育における失敗情報を，学習者が演習課題を行う際に発生したエラーと定義する．更に，演習課題やエラー発生時のソースプログラムも失敗情報と定義する．失敗学では失敗知識を「事象」，「背景」，「経過」，「原因」，「対処」，「総括」という六つの属性に構造化したものであるとしている [1]．本論文では，この失敗知識の属性をプログラミング教育に適応させる．その定義を失敗学による定義とともに，表 1 に示す．本研究では，これらの 6 項目にプログラミング演習教育という対象領域の特徴を踏まえた意味付けを行う．そして，学習者に対して，プログラミング演習において発生するエラーごとに，これらの属性を記述させることにより内省を促進させることを目的とする．一つの演習課題を終えるまでに複数のエラーが発生することも珍しくなく，エラーが発生するつど失敗知識を記録していくものとする．

内省の契機は課題遂行中に発生するエラーであり，これは失敗学の定義と照らし，「事象」に当てはめた．「背景」は演習課題の内容と定義した．失敗学において「経過」は失敗がどのように進行したかという定義となっている．プログラミング学習においてエラーが発生するつどプログラムを修正し，コンパイル，実行を行う．そこで，コンパイル，実行時のソースプログラムを保存し，それを「経過」と定義した．

一方，学習者は発生したエラー（問題）を解決し，それを「原因」，「対処」，「総括」属性に記述する．問題解決においては，一般にエラーの原因究明と，エラーへの対処（本研究ではプログラムの修正）が必要となる [16]．これはそのまま失敗学の「原因」及び「対処」属性に当てはまる．最後に，問題解決から学んだ教訓を「総括」属性として記述する．問題解決過程を振り返り，そこから自ら学んだ教訓を記述する行為は原因とそれへの対処から更に深く内省を進める行為であるため，我々はこの「総括」属性を非常に重要なものととらえ，この属性の記述を必須とする．

失敗情報を内省のための動機付けとし，失敗学で定義されている失敗知識の各属性を学習者に記述させることで学習者自身の内省を促進する．プログラミング教育において行われる演習では一般に課題文，ソースプログラム，エラーメッセージは電子化文書として存在する．それらを学習環境が学習者に提供することにより，学習の効率化を図ることが可能となる．またエラーに対する問題解決過程を電子化文書として外化することにより，学習者に問題解決過程や自身の陥りやすい誤りを意識させるとともに，これらを記録として残すことは自身の陥りやすい誤りを再確認することに役立つ．また，教師が閲覧することができ，指導に活用することが期待できる．そこで本論文では，学習者に対して学習者自身の失敗情報を提供し，学習者が失敗知識を記述することを促進する学習環境を提供する．

3.4 学習環境の要件

（1） 失敗情報の収集

本論文ではプログラミング教育の対象言語を C 言語とし，失敗情報として次のようなエラーを収集する．コンパイル時に gcc コンパイラより出力されるエラーをコンパイルエラー，実行時に OS より出力されるエラーを実行エラーと定義し，これらを収集する．また，プログラムが文法的に正しくコンパイラや OS からエラーが出力されないが，実行結果が作成者の意図するものとは異なる場合のエラーを論理エラーと定義する．これらのエラーの分類を表 2 に示す．

コンパイルエラーと実行エラーはシステムにより自動収集するものとし，論理エラーは学習者の判断により入力するものとする．学習者は，自動収集または自己入力により得られた失敗情報をもとにして失敗知識の属性を記述する．

（2） 失敗知識の記入と閲覧

失敗情報であるエラー内容，演習課題，ソースプログラムを学習者に提示し，失敗知識の属性である「原因」,「対処」,「総括」を記述するための入力フィールドを提供する．学習者にとって失敗知識の属性を記述することは手間と時間がかかるが，内省のためには欠かせない作業となるため，実行エラーと論理エラーに関する内省は義務づけることとする．コンパイルエラーに関しては学習者の判断により内省を行わせる．また，入力された失敗知識は学習者，教授者とも閲覧することを可能にする．これにより学習者は過去の自分の失敗知識を閲覧することができ，教授者は学習者がどのような過程を経てプログラムを作成したかを把握することができる．

（3） 付随的な学習効果の維持

一般に，C 言語の学習を行う際にはターミナルクライアント等を利用して UNIX 系サーバにアクセスしサーバ上でプログラムを作成，コンパイル，実行という流れで行う．このとき学習者は，C 言語の知識のほか，UNIX の知識やコンパイルや実行の手順に関する知識も習得することができる．本研究で提案する学習環境においても，これらの知識を習得させられるようにする必要がある．また，エラーの自動収集に関しても，学習者がプログラムをコンパイルする際や実行する際に，学習者に対して特別な仕掛けを用意することなく収集されることが望ましい．これも，ソースプログラム作成，コンパイル，実行という一連の作業を行う上で得られる知識を失わないようにするためである．

表 2 エラーの分類
Table 2 The classification of errors.

エラー名	エラー定義
コンパイルエラー	コンパイル時に発生するエラー
実行エラー	実行時に発生するエラー
論理エラー	意図する実行結果が得られないエラー

4. 学習環境の構築

4.1 学習環境の概要

前章で議論した要件を受け，以下のような機能をもつ学習環境を構築した．C 言語のプログラミング環境としては，ターミナルクライアント等を利用し，UNIX 系サーバにアクセスしてプログラミングを行うことが一般的である．しかし，本論文では失敗情報として表 2 で挙げたエラーを取得する必要があることと，内省促進の機能をもたせることを考慮し，標準的な Web ブラウザで動作する Web アプリケーションとして構築した．付随的な学習効果も考慮し，コンパイルや実行は UNIX のコマンドベースで行う．構築した学習環境の構成を図 1 に示す．詳細は次節以降で説明する．

4.2 プログラミング環境

学習者に C 言語のプログラミングを行う環境を与える．学習者は Web ブラウザ上のプログラム作成インタフェースから HTTP（HyperText Transfer Protocol）を利用しシステムサーバ上のサーバ接続部へアクセスする．そのサーバ接続部から FTP（File Transfer Protocol）を利用して学習者が Web ブラウザ上で作成したソースプログラムを UNIX 系サーバへ転送する．これにより UNIX 系サーバの上で仮想的にプログラミングを行うことができる．また，**3.4**（3）の要件を考慮し，コンパイルや実行の際に必要なコマンドを Web ブラウザ上から SSH（Secure Shell）を利用して転送する機能も提供する．本学習環境では，サーバへ送るコマンドを入力するためのフィールドと実行

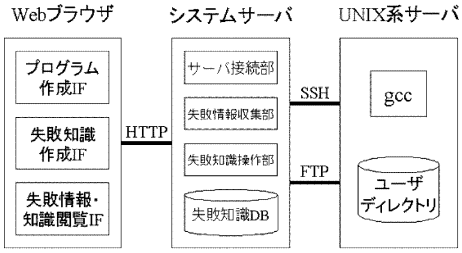

図 1 学習環境の構成
Fig. 1 The architecture of a learning environment.

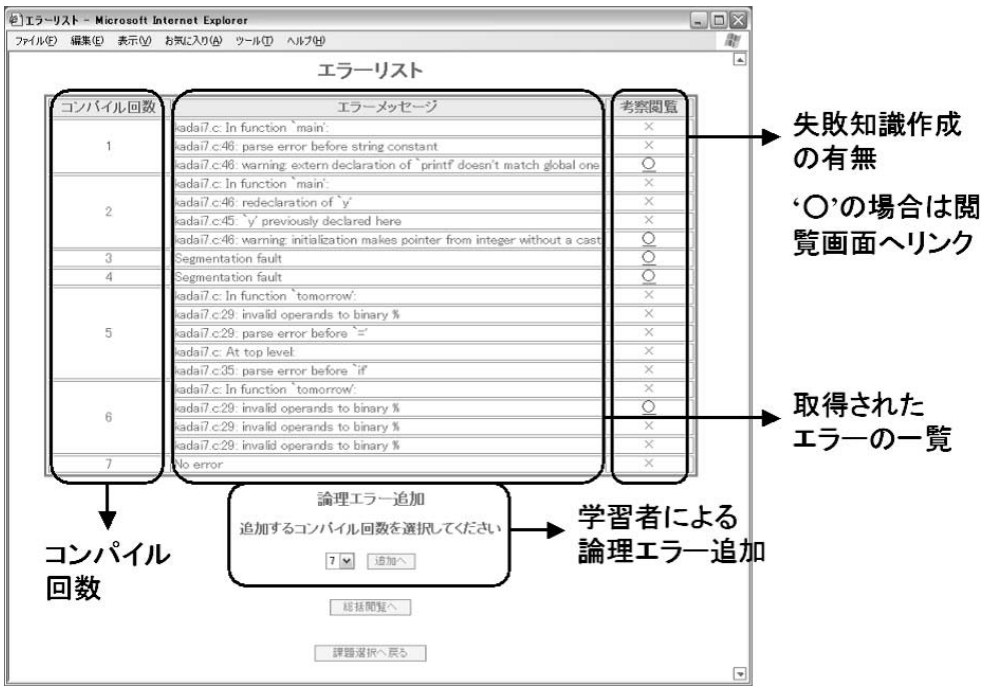

図 2　エラーリスト表示画面
Fig. 2　The screen shot of the error list.

結果を表示するためのフィールドを用意する．**4.3** に示す失敗知識の記述を終了させると，学習者は各演習課題に対する総括を記述し，これにより演習課題の提出とする．このように課題全体を通した総括を記述することで，より高い内省効果が期待できる．課題全体の総括を記述するフィールドは課題ごとに提供する．

4.3　失敗知識作成

3.3 で定義した失敗知識の属性を記述させるために以下の機能を提供する．

4.3.1　失敗情報収集

本学習環境では，**3.4** の要件（1）より，失敗情報として表 2 で示した各エラーを収集する．システムサーバが自動で収集する対象はコンパイルエラーと実行エラーである．コンパイルの際に発生するコンパイルエラーはコンパイラである gcc から出力されるエラーメッセージを取得し，システムサーバ内の失敗情報収集部を介して失敗知識データベースへ格納する．実行エラーは実行時に gcc がインストールされている UNIX 系サーバの OS から出力される実行コードを取得し，そのコードに対応したエラーメッセージをシステムサーバ内の失敗情報収集部を介して失敗知識データベースへ取得する．論理エラーはシステムによる自動収集が不可能であるため，学習者の判断により入力させるものとする．システム上には，コンパイルの回数とそれぞれのコンパイル回数に発生したエラーが表示される．これをエラーリストと定義する．コンパイルエラーや実行エラーがなかった場合，エラーリストには「No error」と表示される．学習者が論理エラーを入力する場合には，エラーリスト上でコンパイル回数を選択することで取得することができる．エラーリストの画面を図 2 に示す．

4.3.2　各属性の記述

3.4 の要件（2）より，**4.3.1** で収集したエラーを **3.3** で定義した失敗知識の属性である「事象」に，また「背景」には演習課題を，「経過」にはエラーが発生した際のソースプログラムをそれぞれ対応させ，「原因」，「対処」，「総括」の属性を学習者に記述させることで失敗情報の知識化を行い，内省を支援する．「事象」にはコンパイルエラーや実行エラーの場合はそれぞれ収集したエラーメッセージを表示させ，学習者が入力した論理エラーの場合は「Logical error」と表示させる．「事象」には自動取得された失敗情報が反映されるほか，それぞれどのようなエラーであったかを学習者に記述させるフィールドも別に用意する．「背

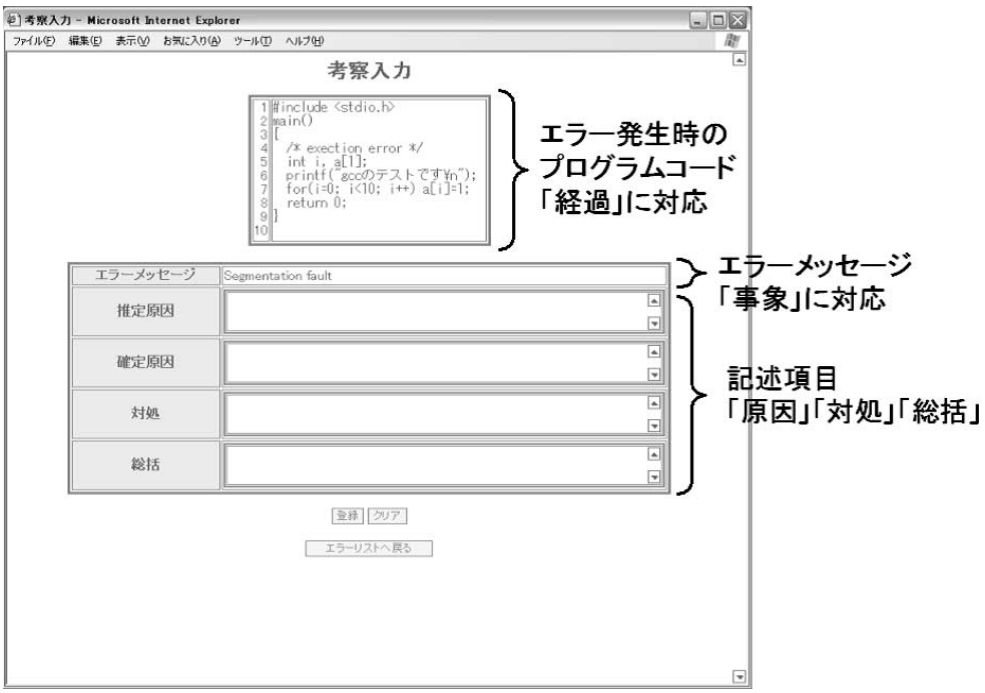

図 3　失敗知識記述画面
Fig. 3　The screen shot of the failure knowledge description.

景」は演習課題の内容，「経過」はソースプログラムと定義しているので，これらについてもエラーが発生した時点で自動的に収集する．学習者が記述する属性である「原因」，「対処」，「総括」については記述用のフィールドを提供する．これらのフィールドは，各失敗情報からリンクされている．以上の属性を Web ブラウザ上の失敗知識作成インタフェースで作成し，作成された失敗知識はシステムサーバ上の失敗知識操作部を介して失敗知識データベースに格納される．失敗知識作成画面を図 3 に示す．

4.4　失敗知識閲覧

3.4 の要件（2）より，**4.3** で作成され，失敗知識データベースに格納された失敗知識をシステムサーバの失敗知識操作部が加工し，Web ブラウザ上の失敗情報・知識閲覧インタフェースに表示させる．このとき，プログラミングにおける失敗情報は完成までに繰り返し発生することを考慮し，時系列に表示させる．各失敗情報から作成された失敗知識へのリンクが存在する．このリンクは失敗知識の記述が済んでいなければ作成用のインタフェースへ貼られ，記述が済んでいれば閲覧用のインタフェースへ貼られる．学習者は自分のエラーリストを演習課題ごとに閲覧することができ，教授者は学習者の失敗知識を横断的に閲覧することができる．

5.　実験による評価

本章では構築した学習環境の有効性を検証するために行った実験について，方法並びに結果を述べる．

5.1　適 用 方 法

東京学芸大学教育学部情報教育専攻の 2 年生及び 3 年生計 21 名に対して適用実験を行った．本学 2 年生は，C 言語の学習をひと通り行っており，またデータ構造とアルゴリズムについても学習している．3 年生は 2 年次に履修している．実験は，三つの部分により構成した．

（1）　事前テストの実施：被験者全員に C 言語のテストを実施し（以降，事前テストと呼ぶ），その正答率の平均と分散の値が均等になるように 2 グループに分け，学習環境適用群と非適用群とした．適用群は 10 人，非適用群は 11 人で構成した．

（2）　演習課題の実施：両群被験者各人に同じ演習課題を与え，同一期間（1 週間）内に各自で学習させた．演習課題は 7 問から構成されており，いずれも C 言語を利用したプログラミングの課題である．演習課

表 3　学習前後の各群における正答率の変化
Table 3　The result of pre-test and post-test by both groups.

	適用群（被験者：10 人）	非適用群（被験者：11 人）	検定結果
事前テスト	平均：57.3%	平均：56.7%	有意差なし
	標準偏差：4.65	標準偏差：6.34	F 検定（$\alpha = 0.05$）で有意差なし
事後テスト	平均：53.6%	平均：48.9%	平均値の差の検定（$\alpha = 0.05$）で有意差なし
	標準偏差：8.06	標準偏差：17.50	F 検定（$\alpha = 0.05$）で有意差あり

題の内容は，初歩的な加減乗除の計算から配列の使い方，ポインタを利用した問題などである．学習環境適用群には学習環境を提供し，失敗知識を記述することを求めた．

一方，学習環境非適用群には，本専攻のプログラム言語の授業で使用している環境で演習を行うよう依頼した（ソースプログラムの作成，コンパイル，実行は端末エミュレータから UNIX サーバにログインして行い，最終結果をレポート提出システム [10] にて提出する）．

（3） 事後テストの実施：課題の提出直後に演習課題の内容と類似した問題を用意しテストを実施した（以降，事後テストと呼ぶ）．テストの結果から被験者の正答率を適用群と非適用群で比較し，本学習環境の有効性を検証した．また，適用群には本学習環境についてのアンケートも実施し，評価を行った．

5.2　学習者の正答率による評価

事前テストと事後テストの結果を表 3 に示す．

事後テストでは事前テストよりも適用群と非適用群間の，平均値の格差は広がったが，平均値の差の検定（$\alpha = 0.05$）で有意差が出るに至らなかった．有意差は出なかったものの，学習環境適用群の方が平均値が高いという結果であった．標準偏差に関していうと，事前テストでは両群に有意差はなかったが，事後テストでは有意差があった．

5.3　アンケートによる評価

学習環境適用群の被験者に対して，本学習環境により内省を自主的に進め，その効果が得られたかについてアンケート調査を行った．アンケートは 4 段階評価（4：「十分，内省を進めることができた」，3：「まあ，内省を進めることができた」，2：「あまり，内省を進めることができなかった」，1：「全く，内省を進めることができなかった」）と，4 段階評価の理由を自由記述で求めた．

図 4 にその結果を示す．10 名のうち無効回答 1 名を除いた 9 名の中で 7 名がやや肯定的な評価で，2 名がやや否定的な評価で，平均では 2.78 であった．肯定的な評価を下した 7 名についてその理由を見てみると，「エラーとそれに対する考察を書くことにより，エラーが出たときも容易に対処することができた」，「頻繁に出るエラーを知ることができ気をつけるようになった」，「最初はまた同じ間違いをしたが，何回か繰り返すうちにしなくなった」等同種の記述が 7 人中 6 人からあった．これはまさに本研究で意図した内省による学習効果を示している．一方，これはコンパイラが出すエラーメッセージの問題であるが，「エラーに対して多くのメッセージが出て，どれがどのエラーを示しているのかの判断が大変だった」，「タイプミスのようなケアレスミスのときには役に立たなかった」というコメントがあったが，これらは本学習環境に対してやや否定的な評価を下した 2 名も全く同じコメントをその理由に挙げていた．更には，課題が簡単だったのであまり内省を行う場面がなかったというコメントもあった．

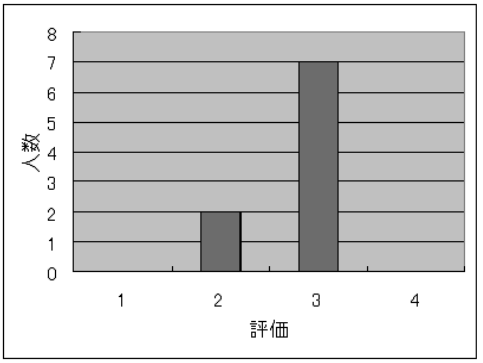

図 4　学習環境による内省支援の有効性についてのアンケート結果
Fig. 4　The result of the questionnaire on effectiveness of reflection support by the environment.

5.4　学習環境に蓄積された内省記述の分析

本節では，実験によって学習環境に蓄積された失敗知識の記述を分析した結果を示す．

今回の実験では，発生したエラーに対する内省記述の割合は，コンパイルエラーについては 30%，実行エ

表 4　失敗知識の記述例
Table 4　Samples of the failure knowledge descriptions.

項番	エラー種別	推定原因	確定原因	対処	総括
1	実行時エラー (segmentation fault)	malloc の使い方を間違えて記述してしまった？	malloc と free がおかしかった？	malloc と free の部分の記述を削除した.	動的メモリ割り当てを行わないプログラムに作りかえた.
2	実行時エラー (segmentation fault)	基本的な間違え.	分からない.	もう一度コンパイルした.	特になし
3	実行時エラー (segmentation fault)	基本的な間違え.	分からない.	スペースを消した.	特になし
4	実行時エラー (segmentation fault)	基本的な間違え.	変数 c を初期化していなかった.	宣言に c=0 を追加した.	Segmentation fault が出てしまい，なかなか原因を見つけるのが難しかったが，よく考えれば分かるエラーでした.
5	論理エラー	配列を 0 から数えて計算していたのが，途中で 1 から数えて計算してしまったため.	同左.	計算をやりなおした.	配列で図形を描く際，よく ±1 ずつずれてしまうので計算に気を付けたいと思いました．また，論理エラーは簡単に直してしまえるので，自分が間違えたところを忘れがちになってしまい，同じようなミスをしてしまうのだなと思いました.
6	論理エラー	for 文の制御変数の範囲が不適切.	使用する変数名の誤り.	正しい変数に置き換えた.	使用する変数の勘違いであった.
7	論理エラー	while 文が適切でない.	while 文の条件文が不十分.	while(ptr!=a) の a を a-1 にした.	while 文の条件式がきちんと考えれていなかった.
8	論理エラー	ポインタ変数の使い方で，printf のとき%c に対応するものは ptr だと思っていた.	ポインタ変数の使い方で，printf のとき%c に対応するものは*ptr だった.	printf(“%c￥n,” ptr) の ptr →*ptr に書き直した.	ポインタ変数の使い方を忘れていた. -ポインタ変数の定義→*つける ポインタ変数を使う式： -アドレスを参照 → * つけない -値を参照 → * つける は覚えていたが，printf のときポインタ変数 ptr に * をつけるのを忘れていた.
9	論理エラー	*x++;	同左	*x++:→*x=*x+1;	*x=*x ＋ 1；でポインタの示すところの値に 1 足せるということを学んだ.

ラーと，論理エラーについては，すべて記述がなされていた.

コンパイルエラーについては，内省記述を必須にしていなかったため，30%と低い割合であった．内容を見てみると，変数名のタイプミスが最も多く，ピリオドとコンマの記述ミス，「)」,「}」,「“」のような組で意味をなすものの漏れといった単純なものが多かった.

実行時エラーと論理エラーについては，そのすべてに対して内省行為を行い，結果を記述するようになっていたので 100%であった．表 4 にその一例を示す.

表 4 の項番 2〜4 はある学習者が一つの実行エラーに対して，プログラムの 3 回の修正とコンパイル，実行を行ったものである．項番 2，3 では確定原因が分からず，総括を記入することができなかったが，項番 4 で解決策を見出した事例である．項番 5 の総括は本システムが提案した原因，対処，総括を記述させることによりエラーを意識させることの有効性を裏づける記述となっている．更にこの学習者は別の類似問題の総括において「前回のを踏まえ，今回は配列の数え方 $(0, 1, 2, \cdots)$ と自然数の数え方 $(1, 2, \cdots)$ を分けて考えることができたと思います.」と記述しており，内省の有効性を裏づけるものといえる．項番 9 でも，誤りからそれを修正する過程を通して，正解の意味付けを具体的に記述している．項番 5 並びに項番 9 を記述した 2 名は，事前テストから事後テストへの成績上昇率を見てみると，学習環境適用群の被験者 10 名の中で最も大きかった学習者と 2 番目に大きかった学習者であった．今回の実験から，失敗に対する内省結果を的確に記述した学習者は結果として学習効果を上げていたことが分かった.

6. むすび

本論文では，一般の学習において有効であるとされている内省行為をプログラミング教育において実現することを目的とし，学習者自身の失敗情報をもとにして内省を行う環境を提供した．適用の結果から，以下が明らかになった．

- 事後テストにおける正答率の平均値の差が事前テストよりも大きく，学習環境適用群の方が学習環境非適用群よりも平均値が高かった
- 事後テストにおいて，母分散について学習環境適用群と非適用群では有意差が見られた
- 定性的なアンケートから，失敗知識を記述させることによる学習効果を示唆するコメントが見られた
- 内省の結果を的確に記述した学習者は事前テストから事後テストへの成績上昇率が大きい，すなわち，学習効果を上げていたことが分かった

これらから失敗情報に基づく内省行為がプログラミング教育において有効であることを確認できた．また，それを支援する学習環境が内省を自主的に促進することに有用であることも示された．学習者の学習軌跡を教授者が把握することも本論文の目的として挙げたが，今回は学習者支援に焦点を当て評価を行った．今後はプログラミングの授業において長期にわたる大規模な適用から評価を行うとともに，教授者の視点から蓄積されたデータを活用した指導方法に関する検討を行う必要がある．また，今回はエラーの分類を現象面から行ったが，誤りの原因で分類し記述させることも検討していきたい．

謝辞 本研究の一部は，東京学芸大学重点研究費 (C)（代表櫨山淳雄），科学研究費補助金基盤研究 (B)(1)（課題番号：14380073，代表宮寺庸造）の助成を受けて行いました．ここに記して謝意を表します．また，認知療法に関する情報を御提供下さいました大阪大学産業科学研究所稲葉晶子先生に感謝致します．最後に，本論文に対して貴重なコメントを下さった査読者の方々に謝意を表します．

文　献

[1] 畑村洋太郎，失敗学のすすめ，講談社，2000.
[2] 服部徳秀，石井直宏，“プログラミング演習の評価サポートシステムの構築，” 教育システム情報学会誌，vol.14, pp.21–28, 1997.
[3] 平嶋　宗，堀口知也，小出　誠，柏原昭博，豊田順一，“力学教育のためのリフレクション環境，” 人工知能学会知的教育システム研究会（第 13 回），SIG-IES-9503, pp.9–16, 1996.
[4] W.L. Johnson and E. Soloway, “PROUST: Knowledge-based program understanding,” Proc. ICSE 84, pp.369–380, 1984.
[5] 小西達裕，鈴木浩之，伊東幸宏，“プログラミング教育における教師支援のためのプログラム評価機構，” 信学論（D-I），vol.J83-D-I, no.6, pp.682–692, June 2000.
[6] 森　正，角川祐次，阿江　忠，“プログラムの正しさの理解を目的とした教材作成システム，” 情処学コンピュータと教育研報，no.061-006, pp.31–38, 2001.
[7] 中原　淳，西森年寿，杉本圭優，堀田龍也，永岡慶三，“教師の学習共同体としての CSCL 環境の開発と質的評価，” 日本教育工学会論文誌，vol.24, no.3, pp.161–171, 2000.
[8] 大野　裕，こころが晴れるノート，創元社，2003.
[9] L.B. Resnick, “A development theory of number understanding,” in The Development of Mathematical Thinking, ed. H.B. Ginsberg, Academic Press, 1982.
[10] 諏訪正則，倉澤邦美，鈴木惠介，森本康彦，横山節雄，佐々木整，宮寺庸造，“プログラミング教育における学習履歴取得システムの開発，” 情報科学技術フォーラム講演論文集（FIT2003），pp.583–584, 2003.
[11] 高橋参吉，松永公廣，“プログラミング学習のための電子学習環境の構築，” 日本教育工学会論文誌，vol.23, no.3, pp.155–165, 1999.
[12] M. Thuering, J. Hannemann, and J.M. Haake, “Hypermedia and cognition: Designing for comprehension,” Commun. ACM, vol.38, no.8, pp.57–66, ACM Press, 1995.
[13] H. Ueno, “A program normalization to improve flexibility of knowledge-based program understander,” IEICE Trans. Inf. & Syst., vol.E81-D, no.12, pp.1323–1329, Dec. 1998.
[14] K. VanLehn, “Toward a theory of impasse-driven learning,” in Learning Issues for Intelligent Tutoring Systems, ed. H. Mandl and A. Lesgold, pp.19–41, Springer-Verlag, 1988.
[15] 渡辺博芳，荒井正之，武井惠雄，“初等アセンブラプログラミング評価支援のための事例ベース構築法，” 情処学論，vol.44, no.2, pp.496–505, 2003.
[16] ウィリアム・J. アルティエ，木村　充（訳），問題解決と意思決定のツールボックス，東洋経済新報社，2000.

（平成 16 年 3 月 31 日受付，8 月 5 日再受付）

知見　邦彦

2003 東京学芸大・教育・情報環境科学課程教育情報科学専攻卒．現在，同大大学院教育学研究科数学教育専攻数学・情報教育講座情報科学分野在学中．ソフトウェア工学，CSCW，教育工学に関する研究に従事．

櫨山　淳雄　（正員）

1985 早大・理工・工業経営卒．1987 電気通信大大学院計算機科学専攻修士課程了．1987 日本電気（株）入社．1999 東京学芸大学教育学部数学・情報科学科助教授．2004 同技術・情報科学講座助教授．現在に至る．博士（工学）．1996 から 1998 まで本会知能ソフトウェア工学研究専門委員会幹事．1999 から 2003 まで本会和文論文誌 D 編集委員会委員．2003 より情報処理学会論文誌編集委員．協調ソフトウェア開発支援に関する研究に従事．IEEE Computer Society，ACM，情報処理学会，人工知能学会，教育システム情報学会各会員．

宮寺　庸造　（正員）

1984 東京電機大・理工・数理卒．1986 同大大学院理工学研究科数理学専攻修士課程了．同年同大理工・情報科学・助手．1999 東京学芸大・教育・数学情報科学・講師．2001 同大・助教授，現在に至る．博士（理学）．2001 より 2002 まで文部科学省メディア教育開発センター客員助教授併任．プログラム言語処理系，アルゴリズム，情報視覚化，教育工学等の研究に従事．1998 から 2002 まで本会教育工学研究会幹事，現在同和文論文誌 D 編集委員．IEEE Computer Society，ACM，情報処理学会，日本ソフトウェア科学会，人工知能学会，教育システム情報学会，日本教育工学会各会員．