プログラミング学習のための可視化対話環境

川﨑雄登[†1]　平井佑樹[†1]　金子敬一[†1]

C 言語は情報システムの構築のため企業などで広く用いられており，情報系技術者として必須のスキルである．そのため，大学の情報系学科では，C 言語を教育している．しかしながら，学習者にとって C 言語の習得は困難である．そのため，ソースコード中の変数などの可視化を行い，学習者の理解を促すシステムがいくつか提案されている．これらのシステムでは，学習者の作成したプログラムが本人の意図した動作をしない場合，学習者自らがソースコードの修正を行う必要がある．しかしながら，これらのシステムは，コード修正の支援を行わないため，初学者に大きな負担を強いる可能性がある．そこで，本研究では，学習者の意図する動作を実現するためのコード候補を提示する対話的教育システムの開発を目的とする．

A Visual Interactive Environment for Programming Learning

YUTO KAWASAKI[†1]　YUKI HIRAI[†1]　KEIICHI KANEKO[†1]

Nowadays, the programming language C is widely used in industry and its skill has become very important with every IT engineer. Therefore, it is taught in almost all IT-related departments of universities. However, the programming language C is very difficult for learners to understand all of its skills. Hence, there are some systems that support learners by visualizing operations with variables in a sources code. In these systems, the learners have to modify their source codes by themselves when the behaviors of the codes are not correct. These systems do not support learners' code modification, which may be arduous for novice learners. Therefore, in this study, we aim to develop a visual interactive educational system that has a function to suggest code candidates that reflect the operations intended by learners.

1. はじめに

現代の情報化社会においてプログラミング能力の需要は増し，大学などにおいてもプログラミングを学ぶ学生が増えてきている．プログラミングを行う際に使用されるプログラミング言語には,様々なものが存在する.この中で，日本の大学に設置された情報系学科の多くにおいて，おもに C 言語[1][2][3]が教育に使用されている[4]．これは，C 言語が比較的低レベルな言語のため，計算機そのものを理解するのに役立つという点と，汎用性の高さによるものである．

そのように，C 言語プログラミング能力は日本の情報系学科の学生にとって重要なスキルである．しかし，初学者が学習する際には大きな困難を要することが多い[5]．これは，学習者がその概念を上手くイメージができていないからであると考えられる[6].

これらの問題を解消するための研究として，変数やデータ構造の可視化を用いたいくつかのシステムが提案されている[4][7]．これらは，学習者がプログラムを作成し，その実行に応じて変数やデータ構造の内容を表示することにより,学習者の理解を促すというものである.しかしながら，これらのシステムでは，作成したプログラムが，学習者の意図する動作をしない場合，学習者はソースコードを闇雲に修正することになる．これでは学習者に大きな負担となり，学習の効率性という観点からも好ましくない．

そのため，学習者の作成したプログラムが，本人の意図する動作をしない場合，意図する動作を実現するためのコード候補を提示する環境が，プログラミングの学習に有効であると考えられる．

そこで，本研究では，学習者による可視化領域の操作に応じて，学習者の意図する動作をさせるためのコード候補を提示するシステムを構築し，対話的な学習支援システムを開発することを目的とする．

以下，本論文は，次のように構成される．まず，2 章では，既存のプログラミング学習ソフトウェアおよびその問題点を述べる．次に，3 章では，本システムの実現について述べる．その後，4 章では，システムの評価実験について述べ，5 章では評価実験の結果を示す．さらに，6 章では実験結果の考察を述べ，7 章で本研究をまとめる．

2. 関連研究

1 章で述べたように，学習者が変数やデータ構造をイメージし易くすることで，プログラミング学習の支援を行ういくつかの教育ソフトウェアが提案されている．ここでは，近年開発された 2 つのソフトウェアを取り上げ，その特徴と問題点を述べる．

2.1 SuZMe

SuZMe(Suitable Zooming for Memory)は，図 1 に示すように，プログラムのメモリ空間を可視化し，プログラミング学習の支援を行うシステムである[4].

SuZMe は，初学者がプログラミングを学習するための教育システムである．学習者は，まず，ソースファイルを作

†1 東京農工大学大学院工学府
Graduate School of Engineering, Tokyo University of Agriculture and Technology.

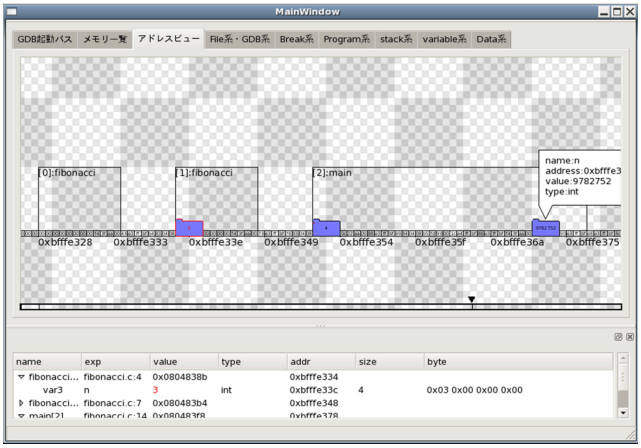

図 1　SuZMe によるメモリ空間の可視化[4]

Figure 1　Visualization of a memory space in SuZMe [4].

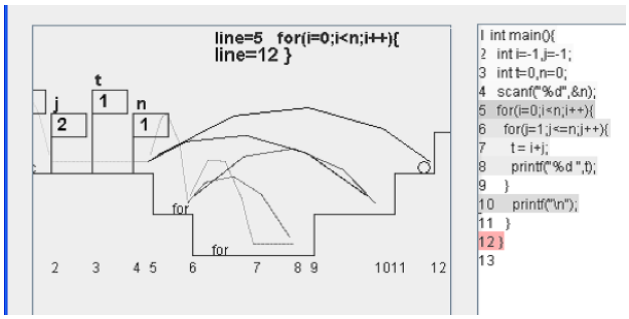

図 2　今泉らのシステムによるブロック構造の可視化[7]

Figure 2　Visualization of a block structure in the system by Imaizumi et al. [7].

成し，それにデバッグ情報を付加してコンパイルを行う．次に，学習者は，システムを起動し，そこからコンパイルをしたファイルを実行する．すると，システムは，そのプログラムのメモリ空間を平面上の位置に対応づけ，アニメーションによってグラフィカルに表示する．これにより，学習者は，プログラムの実行状況を知ることができる．さらに，このシステムでは，空間のパンニングやズーミングなどの直感的な操作が可能である．よりハードウェアに近い具体的な可視化をすることにより，学習者は，プログラムに対する深い理解を得ることが可能となる．

SuZMe は，変数などの情報を可視化する．その際，可視化すべき対象数が多くなると煩雑となり，可視化領域が見づらくなるという欠点がある．これは，よりハードウェアに近い可視化のために，アドレス空間を直線上に配置していることに由来する．さらに，このシステムでは，ソースファイルの作成，コンパイルという一連の作業をシステム内で実施することができない．そのため，学習者が別のツールを用いてソースファイルの作成およびコンパイルを自ら行わなければならず，体系的な学習をすることが難しくなっている．

2.2　今泉らのシステム

今泉らは，図 2 に示すようにプログラムのブロック構造を可視化し，プログラミング学習の支援を行うシステムを提唱している[7]．

今泉らのシステムは，初学者がプログラミングを学習するための教育システムである．学習者が，プログラムの制御構造の動作をイメージし易いよう，以下の 3 つの機能を備える．

- プログラムのブロック構造の表示
- プログラムのステップ実行
- ブロック構造上でのプログラムの実行過程の可視化

まず 1 つ目のブロック構造は，プログラムのネストに対応する動作イメージを階層的な形で表示を行う．これにより，学習者がプログラムの静的な構造を理解することを支援する．

2 つ目の，あるステップ実行の機能は，プログラムの実行過程を学習者が認識する手助けをする．

さらに 3 つ目の実行過程の可視化は，ステップ実行によって取得したプログラムの実行位置に対応するブロック構造を表示する．これにより，プログラムの動作イメージを直感的に理解することができる．

これら 3 つの機能を 1 つの開発環境として提供することで，学習者がソースコード，ソースコードの静的な側面，プログラムの動的な側面という 3 つを対応づけて学習をすることが可能である．

今泉らのシステムは，プログラムのブロック構造を可視化するため，その全体の動作を学習者がイメージをし易くなっている．しかしながら，ポインタや関数といった項目に対する可視化支援を行わないため，これらの項目の学習には適していない．

また，学習者の作成したプログラムが，本人の意図した動作をしない場合，自らソースコードの修正を行う必要がある．しかし，システムは，その支援を行わない．このことは，学習者に大きな負担を強いることとなる．

3.　VIE システム

本研究では，対話的な学習支援を目的とした VIE(Visual Interactive Educational)システムを開発した．

3.1　システムの概要

VIE システムは，初学者にプログラミングを教育するためのシステムである．学習者は，システム内で，ソースファイルを作成し，これをコンパイルして，実行する．システムは，可視化領域にプログラムの変数や関数を可視化し，学習者の理解を助ける．

VIE システムは，以下に示す 3 つの主機能を持つ．

- プログラムのステップ実行およびステップ実行の取消し

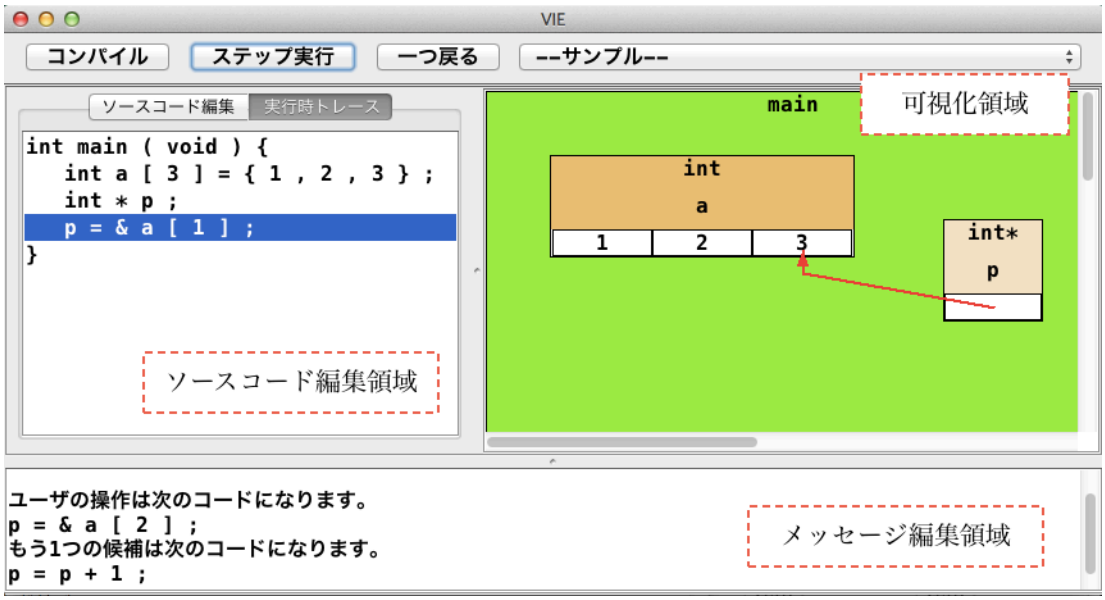

図 3　VIE システムの画面構成

Figure 3　Window layout of the VIE system.

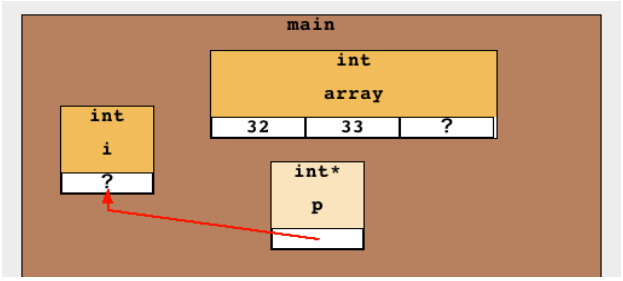

図 4　描画された変数オブジェクト

Figure 4　Drawn objects of variables.

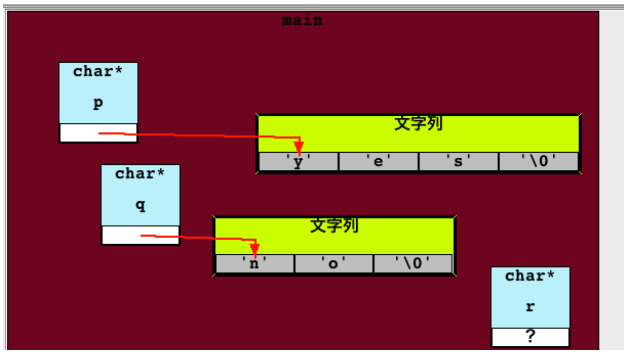

図 5　描画された文字列オブジェクト

Figure 5　Drawn objects of strings.

- 可視化領域に作成されたオブジェクトに対する自由なマウス操作
- 可視化領域に対するマウス操作に応じた候補コードの表示

1 つ目の機能により，学習者は，入力したソースコードを 1 行ずつ実行したり，初期状態に戻るまで実行を取り消したりすることができ，学習者が自分自身の理解に応じてプログラムの実行を制御することができる．

2 つ目および 3 つ目の機能により，学習者は，可視化領域を直接マウスで操作することができ，システムは，その操作に対応するソースコードの候補を表示する．これにより，プログラムが学習者の意図する動作をしなかった場合に，可視化領域を操作すれば，妥当なソースコードの例を得ることができ，システムと学習者とで対話的な学習を進めることが可能となる．

3.2　VIE システムの構成

VIE システムは，以下に示す副システムによって構成されている．

- 学習者がソースコードを編集するエディタ
- ソースコードのコンパイルを行うコンパイラ
- コンパイルされたプログラムを実行する仮想機械
- プログラムの実行に応じた可視化
- エラーや候補コードなどのメッセージ表示

VIE システムの画面構成を図 3 に示す．以下で各構成要素について説明する．

3.2.1 ソースコード編集領域

学習者が，プログラムのソースコードの入力および編集を行う領域である．この領域は，テキストエディタとして必要最低限の機能であるコピー・切取り・貼付けを提供する．

さらに，「実行時トレース」タブに切り替えることにより，プログラムの実行箇所を表示することができる．これにより，学習者は，ステップ実行をしている際に，ソースコードのどの行を実行中かが容易に分かる．

なお，VIE システムは，エラー表示が出た際にもソースコード編集領域が表示されたままとなるようにしている．それにより，学習者は，どの行を間違えて記述したかを容易に理解することができる．

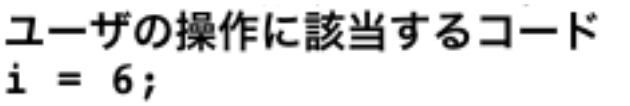

図 6　ソースコードの候補の表示

Figure 6　Provision of a code candidate.

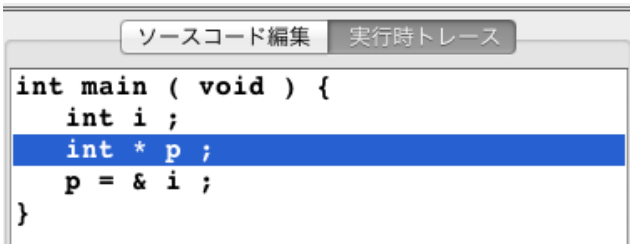

図 7　強調表示された実行中の行

Figure 7　A highlighted statement being executed.

3.2.2 可視化領域

学習者が作成したプログラムの実行状況を可視化する領域である．学習者が，ステップ実行を行うと，システムは，実行中のソースコードに対応する変数および関数オブジェクトを描画する．さらに，変数に関しては，値が更新されるたびにオブジェクトもそれに応じて更新される．図 4，図 5 にオブジェクト描画の例を示す．

VIE システムは，変数を箱として表現し，変数名・変数型・値を表示する．配列は，複数の値を表示でき，ポインタは，矢線によって，保持元と指す先を表現する．文字列は値変更ができないことを明示するために，輪郭線が太く，背景色が灰色の箱として表現した．関数は，色のついた矩形で表現し，関数名を表示する．変数オブジェクトおよび文字列オブジェクトは，宣言された関数に対応する矩形内に描画する．

さらに，マウスとキーボードからの入力により，学習者が，これらのオブジェクトに対して，位置の移動，オブジェクトの大きさの変更などの操作を自由に行うことができる．また，変数オブジェクトに対しては，値の変更，他の変数の代入などの操作も行うことができる．

3.2.3 メッセージ領域

システムが，学習者に対してメッセージを表示する領域である．メッセージには，大きく分けて次の 2 つがある．

- エラーメッセージ
- 可視化領域の操作に対応するソースコードの候補

1 つ目のメッセージは，コンパイルなどの際にエラーの内容と，ソースコードのエラーの原因となった行番号を表示する．これにより，学習者が入力を誤った箇所を的確に把握することができる．

2 つ目のメッセージは，可視化領域のオブジェクトのマウス操作に応じたソースコードの候補を表示するためである．具体的には表 1 に示すソースコードの候補を提示する．

表 1　ユーザの操作とコードの提示

Table 1　Codes generated by user's action.

ユーザの操作内容		コード
変数への操作	変数 x に，整数 k を入れる	x = k;
	char 型変数 x に，文字 A を入れる	x = 'A';
	変数 a に変数 b の値を代入する	a = b;
	変数 a の値を変数 b に代入する	b = a;
	配列 a の要素 i に，整数 k を入れる	a[i] = k;
配列への操作	char 型配列 a の要素 i に，文字 A を入れる	a[i] = 'A';
	配列 a の要素 i に，変数 x の値を代入	a[i] = x;
	配列 a の要素 i の値を，変数 x に代入	x = a[i];
	配列 a の要素 i に，配列 b の要素 j の値を代入	a[i] = a[j];
	配列 a の要素 i の値を，配列 b の要素 j に代入	a[j] = a[i];
ポインタへの操作	ポインタ変数 p が配列 a の先頭要素を指す	p = a;
	ポインタ変数 p が配列 a の要素 i を指す	p = &a[i];
	配列 a の要素 i を指しているポンタ変数 p が，要素 i+j を指す	p = p + i;
	ポインタ変数 p が，何も指さない	p = NULL;

例えば，学習者が，マウスにより，既に宣言されている int 型の変数 i に 6 を代入する操作を可視化領域で行ったとする．このとき，VIE システムは，図 6 のように，その操作を実現するコード候補を提示する．これにより，プログラムが学習者の意図する動作をしない場合，可視化領域の操作を行うことで，妥当なコード候補を得ることができる．

また，表 1 で示したマウスでの操作内容において，複数の項目に該当する場合は，それぞれのコード候補を全て提示する．例えば，図 3 では，配列 a の 2 番目の要素 a[1]を指しているポインタ変数 p を，3 番目の要素 a[2]を指すようにマウス操作した場合を示している．この場合，学習者が即値として a[2]を指したい場合と，現在ポインタ p が指している次の要素を指したい場合の，二種類が考えられるため，このような場合にはどちらの候補も提示されるようにしている．

3.2.4 コンパイルボタン

学習者が，ソースコードの入力後，それをコンパイルするためのボタンである．学習者が，このボタンを押すと，システムは，ソースコードに対して，字句解析，構文解析を行い，中間コードを生成する．この際，ソースコードに文法上の誤りがあった場合には，メッセージ領域にエラー表示を行う．

表 2　コンパイラの仕様

Table 2　Specifications of our compiler.

項目	仕様
全般	ライブラリをリンクすることはできない
	四則演算ができる．しかし，型の異なる変数同士の演算はできない
変数	変数型は，int 型、chat 型のみである
	ポインタ変数を扱うことができる
	変数名の長さは 16 文字までである
	変数の前に ‘&’を付けることで，変数のアドレスを求めることができる
	変数の前に ‘*’を付けることで，ポインタが指している先の変数の値を求めることができる
関数	ユーザ定義の関数は 1 つのみである
	関数の引数は，1 つのみである
条件分岐	条件文は if 文，while 文のみで，条件式は，比較式でなければならない
	扱うことのできる比較演算子は，‘==’, ‘!=’, ‘>’, ‘>=’, ‘<’, ‘<=’のみである
配列	配列変数を宣言できる．しかし，添え字の省略はできない
	宣言可能な配列の要素数は 10 である
	波括弧の初期化子を用いることにより，配列に初期値を与えることができる
	配列名のみを記載することで，配列の先頭要素を指すポインタを生成することができる
	文字列リテラルを用いて，文字配列に初期値を与えることができる
	文字列リテラルを用いて，文字列における最初の要素へのポインタを生成することができる

さらに，仮想機械の初期化を行い，プログラムの実行および可視化のための準備を行う．そのため，コンパイルボタンを押すと，可視化領域に描画されていたオブジェクトは，すべて消去される．

3.2.5 ステップ実行ボタン

学習者が，ステップ実行を行うためのボタンである．学習者が，このボタンを押すと，システムは，ソースコード 1 行分に相当する中間コードの実行を行い，それに応じた描画を可視化領域で行う．その際，図 7 のようにソースコード編集領域の実行時トレースタブにおいて，実行中の行を青ラインで強調する．これにより，ソースコードにおける現在実行中の行を学習者が明確に知ることができる．

3.2.6 コンパイラ

コンパイラは，学習者が入力したソースコードを中間コードに変換する．なお，本研究では，C 言語から初学者に不要な機能を削り，学習に適した仕様としたものを対象とした．その主な仕様を，表 2 に示す．

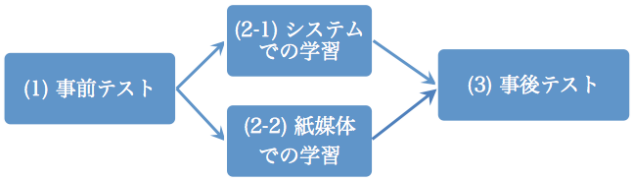

図 8　評価実験の流れ

Figure 8　Procedure of the evaluation experiment.

4.　評価実験

4.1　実験目的

本実験の目的は，VIE システムを用いることによる，C 言語プログラミングの学習効果について，紙媒体を用いる学習と比較することである．VIE システムでは，C 言語プログラミングの学習範囲全ては扱っていない．そこで，本実験では VIE システムで扱っているポインタと配列の分野を主な学習対象として比較する．

4.2　実験概要

以下に，実験期間，所要時間，実験参加者，実験環境についてまとめる．

- 実験期間: 2014 年 1 月 9 日から 2014 年 1 月 27 日
- 所要時間: 約 60 分
- 実験参加者: 情報系学科の大学生 16 人
- 実験環境: Windows 7

4.3　実験の流れ

評価実験の流れを図 8 に示す．評価実験は，(1) 事前テスト，(2-1) システムでの学習，(2-2) 紙媒体での学習，(3) 事後テストからなる．それぞれについて，以下で説明する．

（1）事前テスト

実験参加者 16 名に対して，C 言語のポインタと配列に関する内容[8][9]の事前テストを実施した．問題数は計 10 問で，そのうち 6 問がポインタ操作に対するコードを答えさせる問題（問題番号 4~9），残りの 4 問がポインタの指す先や変数の値などを答えさせる問題（問題番号 1~3, 10）である．事前テストは 10 点満点であり，各問題に完答すると 1 点が与えられる．実験参加者 16 名のうち 9 名をシステムで学習するグループ A とし，残りの 7 名を紙媒体で学習するグループ B とした．

（2-1）システムでの学習

グループ A の実験参加者に対して，まず，システムの使用方法を 10 分間かけて説明した．その後，システムを用いた学習を行うように指示した．ここでの学習時間は，授業の 1 コマ内に収めることを考慮し，30 分間と設定した．この際，実験参加者に事前テストの問題を配布し，システムを実際に操作しながら問題の復習をする形で学習するように指示した．

（2-2）紙媒体での学習

グループ B の実験参加者に対しては，事前テストの問題

の内容を網羅した書籍[10]による学習を，システムでの学習時間と同じ 30 分間で行うように指示した．この際，学習者に事前テストの問題を配布し，書籍で調べながら問題を復習する形で学習するように指示した．

ここで使用した書籍は，各章ごとに一つのテーマを取り上げ，多くのサンプルコードと解説を提示することにより，学習者が C 言語プログラミングを独習できるように工夫されている．一方 VIE システムと異なり，本実験のように，学習者が自らの意図する動作をする，正しいコードを学習したい場合，学習者は提示されている多くのサンプルコード内から最適なものを見つけ出し，自らのプログラムに適用させる必要がある．

（3）事後テスト

(2-1)および(2-2)で学習を終了した実験参加者に対し，事後テストを実施した（付録 A）．事後テストの内容は，事前テストで用いたプログラム内の値を変えた問題で，事前テストと事後テストとの間に難易度の差が出ないように考慮した．事前テストと同様に事後テストも採点した．

5. 実験結果

5.1 テスト結果

5.1.1 総合得点

グループ A，B の各実験参加者における事前テスト，事後テストの結果について，表 3 および表 4 に示す．

5.1.2 分野別の得点

学習者の理解した範囲を分析するため，グループ A，B における，(1) ポインタ操作に対するコードを答えさせる問題（問題番号 4~9）の採点結果を，表 5 と表 6 に，(2) ポインタの指す先や変数の値などを答えさせる問題（問題番号 1~3,10）の採点結果を，表 7 と表 8 に，それぞれ示す．

6. 考察

6.1 テストの総合得点に対する考察

本実験においては実験参加者をグループ A とグループ B とに分けた．グループ A および B の事前テストにおける平均点は，それぞれ 4.78 点と 5.71 点であった．

ここで，グループ A とグループ B における事前テストの平均点に有意差があるか否かを調べる．まず，グループ A とグループ B における事前テストの得点について，有意水準 α=0.05 として，それぞれ「母集団が正規性を持つ」という帰無仮説に対して，Shapiro-Wilk 検定を行った（以降の検定において特に断りがない限り，有意水準 α=0.05 とする）．すると，グループ A，グループ B ともに帰無仮説を棄却することができない結果となり，両グループの得点分布に正規性があると仮定した．

さらに，グループ A とグループ B における事前テストの得点について，「母集団が等分散である」という帰無仮説に対して，F 検定を行った．すると，帰無仮説を棄却することができない結果となり，両グループにおける得点分布は等分散であると仮定した．

以上から，グループ A とグループ B における事前テストの平均点に有意差があるかどうかを調べるため，これらについて，「両グループの平均点に差がない」という帰無仮説に対して，t 検定を行った．すると，帰無仮説を棄却する

表 3　グループ A に対するテスト結果
Table 3　Pre- and post-test results of group A.

	事前テスト	事後テスト	伸び
平均	4.78	9.00	4.22
標準偏差	2.94	0.47	2.97

表 4　グループ B に対するテスト結果
Table 4　Pre- and post-test results of group B.

	事前テスト	事後テスト	伸び
平均	5.71	7.86	2.14
標準偏差	2.60	0.83	2.42

表 5　ポインタ操作に対するコードを答えさせる問題の採点結果（グループ A）
Table 5　Pre- and post-test results of group A on pointer operations.

	事前テスト	事後テスト	伸び
平均	2.11	5.11	3.00
標準偏差	1.79	0.31	1.94

表 6　ポインタ操作に対するコードを答えさせる問題の採点結果（グループ B）
Table 6　Pre- and post-test results of group B on pointer operations.

	事前テスト	事後テスト	伸び
平均	3.29	4.29	1.00
標準偏差	1.67	1.03	1.41

表 7　ポインタの指す先や変数の値などを答えさせる問題の採点結果（グループ A）
Table 7　Pre- and post-test results of group A on values of pointer variables.

	事前テスト	事後テスト	伸び
平均	2.67	3.89	1.22
標準偏差	1.33	0.31	1.23

表 8　ポインタの指す先や変数の値などを答えさせる問題の採点結果（グループ B）
Table 8　Pre- and post-test results of group A on values of pointer variables.

	事前テスト	事後テスト	伸び
平均	2.43	3.57	1.14
標準偏差	1.05	0.49	1.25

ことができない結果となった. よって, グループ A とグループ B における事前テストの平均点に関して有意差は認められず，両群は比較対象となりえることを確認した.

そのため, グループ A とグループ B における事前テストと事後テスト間の点数の上昇点（以後,「点数の伸び」と示す）の平均値に有意差があるかどうかを調べる．まず、Shapiro-Wilk 検定を行い，グループ A とグループ B の点数の伸び分布の母集団は正規性を持つと仮定した．さらに，F 検定を行い，グループ A とグループ B の点数の伸び分布の母集団は等分散であると仮定した.

以上から, グループ A とグループ B における点数の伸びの平均値に有意差があるかどうかを調べるため，これらについて,「両グループの点数の伸びの平均値に差がない」という帰無仮説に対して，t 検定を行った．その結果，帰無仮説を棄却することができなかった (t (14) = 1.44, p = 0.17). よって, グループ A とグループ B における点数の伸びの平均値に関して有意差は認められず，両グループ間において学習効果に有意な差があるとはいえなかった.

6.2 テストの分野別得点に対する考察

問題の分野別に集計した結果を表 5 から表 8 で示した．この結果をもとに，それぞれの分野別にグループ A とグループ B における点数の伸びの平均値に有意差があるか否かを調べる.

6.2.1 操作に対するコードを答えさせる問題

操作に対するコードを答えさせる問題（問題番号 4~9）の集計結果について解析を行う.

まず, グループ A とグループ B における事前テストの成績について，6.1 節と同様に Shapiro-Wilk 検定，F 検定，t 検定を行った．すると，両群は比較対象となりえる結果となった.

そのため, グループ A とグループ B における点数の伸びの平均値に有意差があるかどうかを調べる．まず，グループ A とグループ B における点数の伸びについて，6.1 節と同様に Shapiro-Wilk 検定, F 検定を行った. すると，グループ A とグループ B の点数の伸びの母集団はそれぞれ正規性を仮定でき，等分散であると仮定した.

以上から, グループ A とグループ B における点数の伸びの平均値に有意差があるかどうかを調べるため，これらについて,「両グループの点数の伸びの平均値に差がない」という帰無仮説に対して，t 検定を行った．その結果，帰無仮説は棄却され（t (14) = 2.23, p = 0.04），グループ A とグループ B における点数の伸びの平均値に関して有意差が認められた. よって,「操作に対するコードを答えさせる問題」については両グループ間において学習効果に有意な差があるといえる.

本節で解析した「操作に対するコードを答えさせる問題」は，本システムの特徴であるコード候補を返す機能によって学習することのできる範囲である．よって，学習者はこのコード候補を返す機能を用いることにより，書籍を用いるよりも効果的な学習を行うことができたといえる.

6.3 ポインタの指す先や変数の値などを答えさせる問題

ポインタの指す先や変数の値などを答えさせる問題（問題番号 1~3,10）の集計結果について分析を行う.

まず, グループ A とグループ B における事前テストの得点について，それぞれ「母集団が正規性を持つ」という帰無仮説に対して，Shapiro-Wilk 検定を行った．すると，グループ A，グループ B 共に帰無仮説が棄却される結果となった．よって，グループ A とグループ B における事前テスト得点分布の母集団は正規性を持つと仮定できない.

よって，t 検定を用いることができないため,「両グループの成績の順位和に差がない」という帰無仮説に対して，マン･ホイットニーの U 検定を用いて, グループ A とグループ B における事前テスト得点の順位和に有意差があるかどうかを調べた．すると，帰無仮説を棄却することができない結果となった．よって，グループ A とグループ B における事前テストの成績の順位和に関して有意差は認められないといえ，両群は比較対象となりえる.

そのため, グループ A とグループ B における点数の伸びの平均値に有意差があるかどうかを調べる．前述と同様に Shapiro-Wilk 検定を行った．すると，グループ A は帰無仮説が棄却され，グループ B は棄却されない結果となった.

よって，t 検定を用いることができないため,「両グループの点数の伸びの順位和に差がない」という帰無仮説に対して，マン・ホイットニーの U 検定を用いて，グループ A とグループ B における点数の伸びの順位和に有意差があるかどうかを調べた．その結果，帰無仮説を棄却することができなかった（U = 32.0, p = 0.99）．よって，グループ A とグループ B における点数の伸びの順位和に関して有意差は認められず，両グループ間において学習効果に有意な差があるとはいえなかった.

本節で解析した「ポインタの指す先や変数の値などを答えさせる問題」は，本システムのコードを可視化する機能によって学習することのできる範囲である．そのため，学習者は，この機能を用いて学習を行ったにもかかわらず，書籍を用いた場合との学習効果に差が生じたとはいえないことになる．しかしながら，グループ A は事後テストにおける満点者が 8 人中 7 人であるのに対し，グループ B では 7 人中 4 人と割合が低くなっている．これらから，システムに対する学習効果の期待が得られるといえよう.

7. おわりに

本研究では，学習者による可視化領域の操作に応じて，学習者の意図する動作をさせるための正しいコードを表示することによる，対話的な学習を支援するシステムを開発した.

書籍による学習とシステムによる学習における学習効果の違いを検証するため，16 人の実験参加者を 2 グループに分けて，評価実験を行った．その結果，ポインタ操作に対するコードを答える問題に対してのみ，有意差を得た．また，ポインタの指す先や変数の値などを答えさせる問題に対して，学習効果を期待することができる結果となった．

今回は，約 60 分間という短い時間内での評価実験であったため，狭い範囲の短時間での学習効果の検証にとどまった．また，実験参加者に事前テストの問題を配付し，復習を行わせることで学習を行わせたが，この方法では本質的な学習支援効果を測る上では多少の疑わしさを残している．今後は，広範囲の長期間にわたる繰り返しの学習を行わせるなど，より良い評価実験の方法について検討していきたい．また，可視化領域の同じマウス操作であっても，学習者が意図する内容が異なる場合がある．その場合の，コードの提示方法などの改善策についても検討していきたい．

今後は，より多くの C 言語プログラミングの初学者にシステムを利用してもらい，本システムを改良して，対話的な学習を本質的に支援する環境の開発を実現したい．

謝辞

本研究に協力いただいた，鈴木岳大氏，吉田英樹氏に感謝の意を表す．本研究の一部は，科学研究費補助金若手研究（B）25870207 による．

参考文献

1) 林晴比古: 新訂 新 C 言語入門 シニア編, ソフトバンククリエイティブ (2004).
2) 田口景介: C プログラミング講座, アスキー (2008).
3) 林晴比古: 明快入門コンパイラ・インタプリタ開発 C 処理系を作りながら学ぶ, ソフトバンククリエイティブ (2010).
4) 小池伸弥, 郷健太郎: プログラム実行時のメモリ空間可視化におけるインタラクションの提案, 情報処理学会インタラクション, pp. 665-670 (2012).
5) 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光: 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援, 情報処理学会論文誌, Vol. 48, No. 2, pp.958-968 (2007).
6) 小池伸弥, 郷健太郎: 空間的なアドレス表現を用いたプログラム理解支援ツール, ヒューマンコンピュータインタラクション, pp. 518-526 (2012).
7) 今泉俊幸, 橋浦弘明, 松浦佐江子, 古宮誠一: ブロック構造の可視化環境によるプログラミング学習支援, 電子情報通信学会技術研究報告, Vol. 109, No. 193, pp. 45-50 (2009).
8) 前橋和弥: C 言語ポインタ完全制覇, 技術評論社 (2001).
9) スティーブ・サミット: C プログラミング FAQ—C プログラミングのよく尋ねられる質問, 新紀元社 (2004).
10) ハーバート・シルト: 第 4 版 独習 C, 翔泳社 (2007).

付録

付録 A　事後テスト問題

プログラミング言語 C で記述された次のプログラムについて，下記（1）～（10）に答えよ．ただし，文中の「配列の 1 つ目の要素」とは，配列の最初の要素のことを指す．

```
 1  #include<stdio.h>
 2
 3  int main(void){
 4      char *p = "blue";
 5      char *q;
 6      char word;
 7      char color1[  <い>  ]="red";
 8      char color2[6] = {'g','r','e','y'};
 9      p = p + 1;
10          <ろ>
11          <は>
12          <に>
13          <ほ>
14          <へ>
15          <と>
16      word = *p;
17      return 0;
18  }
```

（1）　4 行目のポインタ p は何を指しているか？簡潔に答えよ．

（2）　空欄＜い＞に入れることのできる最小の数字は何か？

（3）　9 行目の後，ポインタ p は何を指しているか？指している先の値を答えよ．

（4）　空欄＜ろ＞に一文加え，配列 color2 の 1 つ目の要素（最初の要素）に配列 color2 の 3 つ目の要素を代入せよ．

（5）　空欄＜は＞に一文加え，変数 word に配列 color2 の 4 つ目の要素を代入せよ．

（6）　空欄＜に＞に一文加え，ポインタ p が 4 行目の文字列リテラル"blue"の 4 文字目'e'を指すようにせよ．

（7）　ポインタ q が配列 color2 の先頭要素を指すよう，空欄＜ほ＞に一文加えたい．次の中から，正しいものの記号を全て書け（答えが 1 つの場合もある）．

ア：q = color2;　エ：q = color2[0];
イ：q = &color2;　オ：q = &color2[0];
ウ：q = *color2;　カ：q = *color2[0];

（8）　さらにポインタ q が配列 color2 の 3 つ目の要素'e'を指すように，空欄＜へ＞に一文加えたい．次の中から，正しいものの記号を全て書け（答えが 1 つの場合もある）．

ア：q = color1[2];　オ：q = *q+2;
イ：q = &color1[2];　カ：q = q+3;
ウ：q = *color1[2];　キ：q = *q+3;
エ：q = q+2;

（9）　さらにポインタ p が配列 color1 の 3 つ目の要素'd'を指すように，空欄＜と＞に一文加えよ．

（10）　全ての空欄が埋まっている状態でプログラムを実行した場合，16 行目の後に変数 word に入っている値を答えよ．