JSiSE Research Report
vol.31,no.7(2017-3)

遠隔プログラミング教育環境の構築

那須靖弘[*1]
*1 大阪滋慶学園

Development of programming language educational environment for distance leaning

Yasuhiro NASU[*1]
*1 Osaka Jikei Colledge

In distance programing education via the Internet, learner uses various computer environments, so it needs to be examined that how students can set up the program development environment for learning. Therefore, we developed a C language processing system which operates on the browser (including smartphone) using the bytecode interpreter. Interactive programing exercises using console I/O and graphics exercises are possible using it. In addition, we constructed a C programming learning material as the Moodle plugin.

キーワード: プログラミング教育，C 言語，バイトコード・インタープリタ，遠隔教育，モバイル端末

1. はじめに

遠隔教育において，学習者の利用するコンピュータ環境はさまざまであり，インターネットカフェ等における学習やモバイル端末による学習など，開発実行環境の導入が難しいケースも想定される．このため，Web ブラウザ上で動作するプログラム開発実行環境があれば大変都合がよい．

現在，ブラウザ上で各種のプログラミング言語を簡単に実行できるサイトは多数存在する(1)．また，高等教育機関等においても授業での利用を目的とし，同様の機能を持った学習システムが開発されている．これらのサイトは，サーバ側でプログラムを実行して結果を表示するもの(2,3,4)と，ブラウザ上でプログラムを実行しているもの(5,6,7)，それらのハイブリッド(8)に分類できる．サーバ上でプログラムを実行させる方式は，豊富な既存の言語処理系が利用できる点が利点であるが，①悪意のあるプログラムの実行等によるセキュリティ上のリスクがある．また，ブラウザ上で実行する方式に比べて，②サーバ側に大きな CPU リソースが必要となりサーバコストの上昇を招く．③実行時に多くの通信が必要であり，通信帯域や通信コストに配慮すると，アニメーション表示を行う課題などは実施が難しいという問題がある．

一方，ブラウザ上でプログラムを実行させる方式は，セキュリティ上の問題がなく，運営側にとってメリットのある方式といえる．しかし，JavaScript はブラウザで直接実行可能な唯一の言語であるが，①静的型付言語でないためデバッグに時間を要するケースが多い，②シングルスレッドモデルで動作するため処理時間の長いプログラムを実行するとブラウザがフリーズする，③プログラムの実行を中断する機能がないなどの点において初学者がプログラミングを学ぶために適した言語であるとはいえない．

そこで，筆者は，ブラウザ上で動作する C 言語処理系を開発し，それを利用したプログラミング教育システムを構築した(9)．本処理系はソースプログラムをコンパイラでバイトコードに翻訳した後，バイトコード・インタープリタによって実行を行う方式を採用しており，素数計算等の処理に時間の掛かるプログラムであってもブラウザはフリーズせず，さまざまな種類の課題による演習を実施することができる．

本稿は，開発したブラウザ上で動作する C 言語処理系と，それを利用した遠隔プログラミング学習環境に

ついて述べるものである．

2. プログラミング教育環境

2.1 プログラミング教育環境

筆者らが開発した C 言語処理系は JavaScript により C コンパイラ，バイトコード・インタープリタを記述したもので，ブラウザ単体で動作するものであり，学習者の端末に別途プログラムを導入する必要がないという遠隔教育用として優れた特徴を有している．この C 言語処理系とオープンソースの Web エディタ（Ajax Cloud9 Editor，以下 Ace という）を組み合わせて遠隔教育用のプログラミング学習教材を構築した（図 1）．

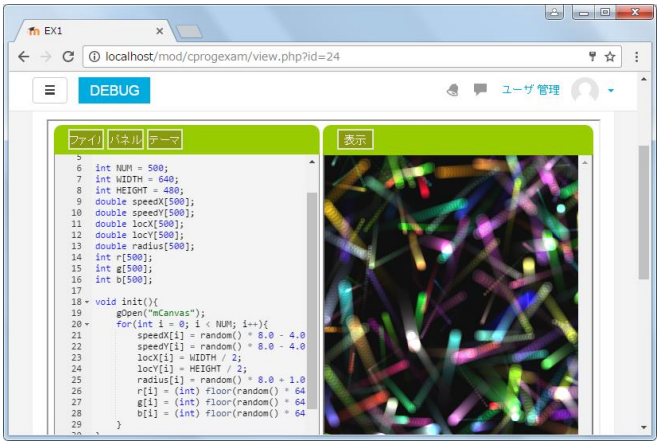

図 1 プログラミング演習環境[1]

本システムの特徴は以下の通りである．

(1) キー入力や画面表示を行う対話処理課題からグラフィック課題まで実施可能

(2) エラーメッセージをクリックすることによりエディタ上の該当する箇所にカーソルが移動

(3) スロー・ステップ実行機能や動作説明機能などの動作モードを持ち，プログラムが実行される様子の確認や，デバッグに活用できる

(4) スマートフォンなどモバイル端末でも動作可能

2.2 TinyC

TinyC とは，開発した遠隔プログラミング演習向け開発実行環境用の初学者向けの C 言語処理系であり，標準的な C コンパイラから演習で利用しないものを削除した，C 言語のサブセットとなっている．

標準ライブラリ関数は数が多く，すべてを実装することは現実的ではない．do-while 文や switch-case 文は実装していないが，他の構文で代用可能であり，初学者向きプログラミング教育において扱う必要はないと考えている．goto 文はないが，その代わりにラベル付き break 文を実装している．

共用体を実装するためには，メモリのさらに詳細なエミュレーションが必要であり，実用的な速度で動作する環境の構築は難しいと判断して除外した．また，文字列は JavaScript のオブジェクトをそのまま利用している．

一方，配列は C 言語においてどのように実装されるか内部形式を確認して学ぶことができるように，JavaScript の配列は使用せず，C 言語と同じ形式でスタック上に展開している．

ところで，JavaScript は実数と整数の区別はなく数はすべて Number 型で表現されている．このため，本システムでは，整数の演算は Number 型で演算した後，整数に変換しているが，sizeof 演算子はスタック上のデータの大きさを表すこととして，int 型，double 型ともに 1 としている．

TinyC で実装していない主な機能を表 1 に示す．

表 1 TinyC が実装していない機能

プリプロセッサ	なし
ポインタ	文法上なし
標準ライブラリ関数	ほとんどなし
基本型	int，double のみ
制御構文	if，for，while のみ
goto 文	なし
共用体	なし
文字列	JavaScript オブジェクト

一方，TinyC では学習者が学びやすいように C 言語の仕様を変更している部分もある．C 言語ではグローバル変数とローカル変数，あるいはスコープの異なるローカル変数に同じ名前をつけることができる．この名前空間の分離は，大規模なシステム開発等で名前の衝突を避ける場合に役立つが，初学者向きのプログラミング演習において異なる変数に同じ名前をつける必

[1] http://yoppa.org/taumedia10/2065.html に掲載のプログラムを C 言語用に改変

然性はない．同じ名前が使われているとすれば，間違って付けている可能性が高く，ミスによりこのような記述がなされた場合，コンパイルは正常終了するが意図した動作とならず，原因の究明に時間が掛かる．

実は, Java では同じ名前のローカル変数を用いると文法エラーとなる．そこで，本コンパイラではグローバル変数とローカル変数，ローカル変数同士に同じ名前を付けた場合，警告を出すようにしている．

さらに，配列の初期化を行う場合，C 言語では配列のサイズと初期値の数が一致しなくてもよい. しかし，これも入力ミスの可能性が高いと考え，警告を出すようにしている．

2.3 同期コンソール出力機能

JavaScript は排他制御等のオーバヘッドを避けるためシングルスレッドモデルを採用しており，ユーザプログラムが処理を行っている間は，他の UI に関する処理は停止する．そのおかげで，JavaScript プログラムは自由に Web ページの書き換えが可能となるが，その一方で短い時間で処理を終了することが求められる．これは，JavaScript が動きのある Web ページを作成するためのツールであることを考えれば当然の仕様である．しかし，この仕様は初学者向きプログラミング演習においては問題となる．

例えば，学習者が繰返しの終了条件を誤り，無限ループとしてしまうと，ブラウザは簡単にフリーズしてしまう．ブラウザがフリーズ状態となると強制終了するしか対処法はなく，最悪の場合その時点までの学習記録が消失し，学習者のモチベーションの低下を引き起こす可能性がある．

さらに, JavaScript は画面更新時のチラつきを防ぐため，プログラムから行った画面変更はプログラムの実行終了後にブラウザに反映される仕様となっている. しかしその結果，プログラムの実行途中でメッセージを表示することはできなくなっている．また，これらの仕様とも関連するがプログラムの実行をキャンセルする仕組みが提供されていない点もプログラミング演習では問題となる．

JavaScript でこれらの問題点を解決する手法には，Web ワーカー，継続渡し形式（CPS），yield 等が利用できる．

Web ワーカーとは JavaScript のマルチスレッド機構である．Web ワーカーを用いると，ワーカースレッドが無限ループとなっていても UI スレッドはフリーズせず，プログラムの処理と同期した画面表示が可能であり，さらにプログラムの実行をキャンセルすることができる．しかしながら，JavaScript には Wait 機構がないため，処理の途中でキー入力を受け取るプログラムを実現することは難しい．

ところで, 公開されている Web 上のプログラム開発環境の多くが，プログラム実行前に処理に必要なデータを入力し，実行結果は一括して表示する汎用機のバッチ処理に似た実行方式となっている．実行時にキー入力を受け取る必要がなく，ステップ実行等の機能も必要ないのであれば，Web ワーカーを用いるのが最も簡単な方法であろう．

継続渡し形式（CPS）への変換を行い，setTimeout によってプログラムを分割する手法は，JavaScript においてプログラムの実行途中で画面表示を行うためのテクニックとしてよく利用される．CPS への変換を行えば，元のプログラムにおいて無限ループとなっているプログラムを動作させてもブラウザがフリーズすることはなく，プログラムの実行停止機能を追加することもできる．

しかし，CPS へ変換されたプログラムは構造が大きく変化するため，元のプログラムと同じ動作をするプログラムであると判読することは難しい．このため，期待通りの動作をしなかった場合，デバッグ作業は困難となり，プログラミング演習で用いるには，元のコードとの対応関係を自動的にとりながらデバッグできるようなツールが必要であろう．

ECMAScript6 で導入された yield は関数の処理を中断させる機能である．このため，事前にプログラムの適切な箇所に yield を挟み込んでおけば，無限ループとなっていても実行を停止できる．yield は，元のプログラム中の任意の場所に単純に挿入するだけでよく，CPS と比べればプログラムの対応関係をとりやすい形式である．

しかし，yield は IE11 ならびに，Android4, iOS9 のモバイル端末のブラウザは対応しておらず(10)，現状では不特定の端末からの利用を前提とするシステムで利用するには不安が残る．

本処理系はバイトコード・インタープリタを用いており，画面表示処理の後でプログラムの実行を一時停止することによりブラウザに画面更新の機会を与えている．このことによって，for ループ等の繰返しの中で文字を表示するようなプログラムにおいても，処理の進行に合わせ滑らかに文字が表示されるようになっている．また，プログラムの実行を数十ミリ秒ごとに停止してブラウザに制御を返しており，画面上に配置している「停止」ボタンによってプログラムの実行停止が可能となっている．

なお，CPS，yeild，バイトコード・インタープリタは，いずれも setTimeout によって処理を分割するが，本稿では，分割された各実行単位のことをタイムスライスと呼ぶこととする．

2.4 性能評価

図 2 に示すプログラムの本システムによる処理時間を表 2 に示す. 評価 PC は東芝 dynabook B552/G, CPU : Core i5-3320M, 2.6GHz, RAM : 4GB である．

```
int p = 3456793;
int r = 1;
for(int i = 2; i < p; i++){
  if((p % i) == 0){
      r = 0;
      break;
  }
}
```

図 2 時間計測用素数判定プログラム

表 2 本システムの素数判定プログラムの実行時間

実行環境	Elapse Time
Firefox ver.48	1.09sec
chrome ver.52	1.12sec
IE ver.11	4.14sec
Safari ver.5（参考）	10.44sec
Visual-C（最適化 O2）	0.011sec

処理時間はブラウザによって大きく異なり，最も速い Firefox と最も遅い IE では 3.8 倍の開きがある．Visual-C と比較すると，最も速い Firefox でも 99 倍遅くなっている．なお，Safari は Windows 版の開発が止まっているため参考とした．

つぎに，図 2 と同等の素数判定プログラムを JavaScript で実行した処理時間を表 3 に示す．

表 3 CPS による素数判定プログラムの処理時間

実行方式	Elapse Time
CPS(Firefox)	13852.65sec
ConcurrentThread(Firefox)	6.17sec
JavaScript(Firefox)	0.032sec

for 文による繰返しを CPS に変換する場合, ループ 1 回分の処理を継続として，それを setTimeout でつなぐ形となる．今回の素数判定プログラムではタイムスライス中の処理は if 文が一つだけとなり，処理時間が短く効率が悪い．ConcurrentThread とは牧(11)らが開発した CPS を用いる JavaScript のマルチスレッドフレームワークであるが，setTimeout による中断を最適化する実行制御を行うことにより高速化を実現している．一方，素数判定プログラムを JavaScript でそのまま実行した場合の処理時間は 0.032 秒であり，Visual-C の処理時間の 2.9 倍と非常に健闘している．これは，最近の JavaScript 実行環境がプログラム実行時に JIT コンパイラによる最適化を行うためである．

最後に，yield を使用してタイムスライスを実現する場合の処理時間の計測結果を表 4 に示す. CPS の場合と同様に，yield 呼び出しによる関数の中断に対して，必ずしも setTimeout によって実行を停止する必要はない. このため, この表は何回の yield に対して setTimeout を実行するかをパラメータとして変化させ，処理時間を計測したものとなっている．

表 4 yield による素数判定プログラムの処理時間

yield の回数	Elapse Time	Time Slice
1000000	0.232sec	67msec
100000	0.386sec	11msec
10000	2.044sec	8.7msec

表より，タイムスライスの処理時間を数十ミリ秒となるように制御を行った場合，処理時間は 0.3 秒程度になると見積もることができる．

バイトコード・インタープリタはバイトコードの命

令単位で分割が可能であり，文単位で分割を行う CPS や yield と比較すると粒度が細かくなる．このため，バイトコード・インタープリタは実行制御処理に要するオーバヘッドがより大きくなる．そこで，本システムではタイムスライスを構成するためのループはステップ数のみで制御し，時間の計測はタイムスライス終了時点で行い，次回のタイムスライスで実行するステップ数に反映するようにしている．その結果，各タイムスライスの処理時間は変動することになるが実用上問題はない．

CPS と yield は分割の粒度は同じであるが，この実験による比較では CPS が最も遅いという結果となった．これは，CPS で用いる継続を構成するために関数オブジェクトの生成が必要であり，その処理に時間がかかっているためであると考えられる．

つぎに，Chrome のデバッガを使い，本システムにおける図 2 に示すプログラム実行時のタイムラインを取得し本システムの動作を詳細に分析した．

本システムは図 2 のプログラムの構文解析に 15.10msec，バイトコード生成に 5.00msec 要している．また，実行に関しては，タイムスライスごとに処理時間が変動するが，下表の通りであった．

表 5　実行時各タイムスライスの処理時間

Time Slice	Elapsed time	Remarks
No.1	6.79 msec	print 呼出
2	69.59 msec	
3	46.92 msec	
4	60.65 msec	
5	45.41 msec	
6	7.95 msec	JIT 最適化
7	15.78 msec	
8	31.43 msec	
以下省略		

図 2 では省略しているが，プログラムの先頭で print 関数を使いメッセージを表示しており，タイムスライスの No.1 は print 関数の呼び出しのため途中で中断し処理時間が短くなっている．

No.2 から No.5 は 45～69msec であるが，No.6 で 7.95msec と処理時間が短くなっている．この理由は，この時点で Chrome の JIT による最適化が完了したためであると推測される．その後，No.8 にかけて処理時間が 2 倍ずつ増加している．これは，本システムがタイムスライスの処理時間が短すぎると判断し，次回のタイムスライスあたりの処理ステップ数を 2 倍に増やしているためである．なお，No.9 以降の処理時間は 30～34msec でほぼ一定であった．

2.5　スロー・ステップ実行機能

スロー・ステップ実行機能とは，プログラムを一行ずつ実行し，エディタ上で対応する文をハイライト表示するもので，多くのデバッガが有する機能である．このスロー・ステップ実行を用いると，繰り返しや条件分岐といった制御構構造におけるプログラムの動作を視覚的に伝えることが可能である．プログラミングの学習において条件分岐や繰返しといった制御構造の単元で躓く学生は多いが，スロー実行やステップ実行でどのようにプログラムが動作するのかを確認しながらプログラムを実行することで理解を深めることができる(12).

2.6　プログラム動作説明機能

この機能は，プログラムを実行しながら，どのような処理が行われているかを説明する機能であり，関数呼び出し，代入，if 文，for 文，while 文，return 文が実行された時点でコンソールに説明を表示している．特に，代入や条件判断では，その根拠となる計算過程を表示するようにしており，どのように，コンピュータが処理を進めているのかを確認することができる．

```
年=2016
/// atoi("2016") 呼出し
/// y = 2016 : 2016
/// input("月=") 呼出し
月=11
/// atoi("11") 呼出し
/// m = 11 : 11
/// if 文を実行します
/// 偽 : (11==2)
/// if 文を実行します
/// 真 : ((((11==4)||(11==6))||(11==9))||
/// dm = 30 : 30
/// if 文を実行します
/// 偽 : ((11==1)||(11==2))
```

図 3 動作説明機能の実行画面による実行例

植野(13)らはトレース課題における学習者の解答によって能力を推定し，適応的にヒントを出すシステムを提案している．この手法では「足場かけ」に用いるヒントとして，主にプログラムの動作説明のアニメーションを用いている．この場合の，動作説明アニメーションは実行するステップ数によってレベル分けされており,「足場かけ」レベルの高いヒントは低いヒントを完全に包含する関係にある．

このため，本システムの動作説明機能を用いて，できるだけ少ないヒントで回答するように指示し，何ステップまで動作説明を表示させたかにより学習者の能力推定を行うことにより同様の学習効果が期待できる．

2.7 バイトコード表示機能

この機能は，コンパイルされたバイトコードの表示ならびに，プログラム実行中の，スタックやレジスタの状態を表示する機能である．これによって C 言語の各文が, CPU によりどのように実行されるのかを確認することができる．実行画面の例を図 4 に示す．

```
/// 関数実行
: FRAME 1
  [PC:6 Top:3 FP:2 AD:0] STACK:[0,2,0,0]
/// for 文を実行します
/// 変数 i 定義
: PUSHI 0
  [PC:12 Top:4 FP:2 AD:0] STACK:[0,2,0,0,0]
: ADDRL 0
  [PC:14 Top:4 FP:2 AD:3] STACK:[0,2,0,0,0]
: ST    1
  [PC:15 Top:4 FP:2 AD:3] STACK:[0,2,0,0,0]
```

図 4 バイトコード表示機能による実行例

この機能は上級者向けのものであり，初学者はバイトコードの詳細について理解する必要はないが，初学者に対しても高級言語の利点やコンパイラの役割，コンピュータの仕組みについて学ばせる教材として使用することができる．

3. Moodle 活動モジュール

学習者が利用するコンピュータ環境によって，学習者が作成したプログラムをローカル端末に保管することが難しいケースもあり，ファイルをサーバ上に保管できると便利である．さらに，課題の提示やレポート提出といった作業を簡単に行えることが望ましい．

e-learning の標準規格である SCORM は多くの LMS で採用され広く利用されているが，定型的な e-learning で必要とされる API しか持たず，上記のような機能を提供することは難しい．

そこで，プログラミング演習の授業において必要となる機能をパッケージ化し，Moodle の活動モジュールを構築した．

本教材では学習者は，サーバ上にファイル保管領域を持ち, 作成したプログラムはサーバ上に保管される．また，共有フォルダの機能があり，学習者全員にファイルを配布することも可能となっている．なお，ファイルシステムは安全性に配慮して，サーバのファイルシステムとは切り離し，Moodle のデータベース上に実装している．

また，本教材は,「エディタ」,「コンソール」,「ファイラー」,「課題提示」,「レポート提出」の 5 つのパネルで構成され, 図 1 に示すように画面を左右に分割してパネルを組み合わせて表示する形式となっている．

学習者は課題の説明を見ながらプログラムの作成を行い, 実行結果を確認しながらデバッグを行うなど，それぞれの演習フェーズにおいて使用する画面を変更しながら学習を進めることができるようになっている．

4. 関連研究

三浦(6)は，Processing による Web アプリケーション開発ための教育環境を Web 上に構築し, 教育を実践し評価を行っている．その中で Processing は JavaScript に変換されて実行されるため，無限ループを作ってしまうとブラウザがフリーズしてしまう点を問題点として挙げている．本システムは一定の時間間隔でバイトコード・インタープリタを停止しているため, 無限ループでブラウザはフリーズすることはなく，「停止」ボタンによって実行をキャンセルすることができる．

(14)は，JavaScript で動作する BASIC インタープリタであり，文単位で実行制御を行うことでブラウザがフリーズしないようにしており，言語が BASIC である点を除けば，授業での利用も可能なほど完成度が高い．この BASIC は初期の PC 環境の再現を目指し

たもので，性能は重視していないと推察されるが，図2と同等のプログラムの実行に724秒必要であった．

本システムはソースプログラムをコンパイルした後，バイトコード・インタープリタで実行し，さらに，実行制御の工夫によって高速化することにより，処理に時間のかかる課題など幅広い内容を扱うことができる．

Chansilp(15)らは，初学者がプログラムの動作に対して持つ概念の間違いが理解の妨げになっているとし，プログラムの動作と連動しながらコンピュータ内部でどのような処理が行われているかを表す動作説明アニメーションを提示することで，プログラムの動作に対する正しい概念を形成することが可能となると指摘している．また， Byckling(16)は変数の役割が，初級レベルのプログラミングでは「固定値」,「カウンタ」など 10 種類に収まるとし，それぞれの役割を表すアイコンを用いたアニメーションによって学ばせる手法を提案している．これらは，あらかじめ作製したアニメーションを提示して学習を行うものであり，学生が作成したプログラムを自動的にアニメーション表示することはできない．

変数の役割はコンパイラで判断できないため，別途ラベル付けを行う必要があるが，本システムの動作説明機能は，学生の作成したプログラムであっても実行可能であり，内容として上記のアニメーションと同等のものを文字として出力している．アニメーション表示については，その必要性も含めて今後検討して行く．

プログラミング演習環境を Moodle のプラグインとして構築しているものには，Juan(3)，布施(4)などがある．Juan(3)らは，VPL というプログラミング教育用の Moodle プラグインを開発し GitHub において公開している．これは，Jail サーバというプログラム実行専用のサーバを別途用意することで Moodle サーバのセキュリティを確保するものであり，ブラウザ上でのプログラム開発実行が可能となっている．また，Test ベースの提出課題の自動評価機能，さらには不正コピーチェック機能等を備えている．布施(4)らは，Ruby の学習環境として，ファイルマネージャをインターフェースの中心としたシステムを構築している．このシステムは，各ユーザがサーバ上にホームディレクトリを持ち，アイコンの操作によって作業を行う GUI 型の演習環境となっている点に特徴がある．

本システムも，Moodle のプラグインであり，同様の演習環境を提供するものであるが，プログラムがクライアント側で実行されるため，コンソール入出力を用いた課題からグラフィックスを利用する課題まで幅広い題材を扱うことが可能である．また本システムでは，スロー・ステップ実行や動作説明を用いた，初学者に対する木目の細かい演習が可能となっている．

なお，本システムは提出課題の自動評価の仕組みを持たないが，管理画面において提出課題を選択するだけで自動的に提出プログラムの実行が行われるなど，教員の採点作業をサポートする仕組みを提供している．

5. おわりに

Processing，Brython など，他の言語を JavaScript に変換しブラウザ上で実行する言語処理系は各種存在する．これらの，トランスパイラ言語は処理速度の面では有利であるが，処理時間の長いプログラムを実行するとブラウザがフリーズする等の，JavaScript の実行方式の問題点を引き継いでいる．

本システムでは，バイトコード・インタープリタを採用し，画面更新処理後にその動作を停止することで，プログラム実行中に画面が更新可能となっている．また，一定の間隔で動作を停止してブラウザに処理を返しており実行停止ボタンによって，プログラムの実行を途中でキャンセルすることも可能となっている．さらに，スロー・ステップ実行，動作説明機能，バイトコード表示機能など初学者向きプログラミング学習のための有用な機能を備えている．

バイトコード・インタープリタは，トランスパイラと比較して実行速度が遅くなるのが問題点であるが，単純に CPU クロックだけで比較すると，現在の標準的なモバイル端末の CPU クロックが 1GHz であるとすれば，本システムは 10MHz（≒1GHz ÷99）程度のコンピュータと同等の性能があることになり，モバイル端末でのプログラミング演習用の環境として十分な性能を有しているといえる．さらに，グラフィックス処理などブラウザにまとまった処理を依頼する部分は高速に動作し，描画を繰り返してアニメーションを表現するような課題も実施することができる．

さらに，コンパイル済みのオブジェクト形式でのプログラムの配布が可能である点も利点と言うことができる．例えば，完成したプログラムをヒントとして学生に配布する場合，バイトコードに変換したプログラムというのは，解読困難であり解答そのものが学習者に伝わる心配はない．

本システムは，プログラミングの初学者を対象とし，コンピュータの動作を深く理解させるために行う演習での利用を想定しており，簡単なアルゴリズムの実装を演習として取り上げる予定である．しかし，C 言語は高級アセンブラとしての側面を持ち，現在でも組込システム開発の分野で使用されているが，当該分野では，プログラムがどのような機械語に翻訳され実行されるかを意識する必要がある．プログラミング言語には，その言語の持つ文化のようなものがあり，その文化的側面についてふれることも重要である．その意味では C 言語教育にバイトコード・インタープリタを利用する方式は理に適っているといえる．

今回，独自の C 言語処理系を開発したが，独自コンパイラは，学習者のレベルに合わせてエラーメッセージを変更することも可能である．また，言語仕様も教育目的に合わせて変更することができるため，授業の目的が特定の言語を学ぶことではない場合，独自コンパイラを用いるプログラミング教育も意義があると考えている．

参 考 文 献

(1) アサインナビ: “ブラウザ上で簡単にプログラミングができる”, http://room.assign-navi.jp/column/1519/,（参照 2016.5.30）

(2) 平田克己，Ahmad Z:“Web ブラウザ上で動作する C プログラミングシステムの開発”，小山工業高等専門学校研究紀要, No.42, pp.123-128（2010）

(3) Juan C.R, Enrique R, Zenón J.H: ”A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features”, Proceedings of EEE'12, (2012)

(4) 布施泉，中原敬広，岡部成玄:”授業での利用を前提とした初学者プログラミング学習環境の開発－Ruby 言語の Moodle 環境での学習支援－”，大学 ICT 推進協議会 2016 年度年次大会論文集, FE22, 京都(2016)

(5) 那須野薫，上野山勝也，松尾豊:“次世代プログラミング学習サイト構築の試み”，2013 年度人工知能学会全国大会（第 27 回）富山（2013）

(6) 三浦元善:”Web 技術を活用したインタラクティブな情報教育環境の構築と実践”，信学技報 116(85), pp.19-24, 2016

(7) 兼宗 進，本多 佑希，林康平，他: “オンラインで利用可能なプログラミング学習環境の提案”，日本情報科教育学会 第 9 回全国大会講演論文集, pp.89-90, 2016

(8) 宇野健, 畝傍みなみ:“C 言語学習支援のための Web 上でのプログラミング環境の開発(2)”，県立広島大学経営情報学部論集，第 6 号, pp.35-41 （2014）

(9) 那須靖弘:”遠隔教育のためのプログラミン言語教育環境の構築”，2016 年度教育システム情報学会全国大会，宇都宮(2016)

(10) ECMAScript compatibility table , https://kangax.github.io/compat-table/es6/, （参照 2016. 12. 14)

(11) 牧 大介，岩崎 英哉: “非同期処理のための JavaScript マルチスレッドフレームワーク”，情報処理学会論文誌. プログラミング vol.48, no.12, pp.1-18, 2007

(12) 西田智博， 原田章，中村亮太，他 “初学者用プログラミング学習環境 PEN の実装と評価”，情報処理学会論文誌, Vol.48, No.8, pp.2736-2746 , 2007.

(13) 植野真臣，松尾淳哉，“項目反応理論を用いて適応的ヒントを提示する足場かけシステム”，信学論 D, VolJ98-D, No.1, pp.17-29, 2015

(14) “ブラウザ版 N88-BASIC 互換 BASIC のようなもの”, http://uso800-basic.appspot.com/, （参照 2017.2.3）

(15) K. Chansilp, R. Oliver:“Using multimedhia to develop students’ programing concepts”, proceedings of EDU-COM 2002, pp.91-101, 2002

(16) P. Byckling, J. Sajaniemi:"Roles of variables and programming skills improvement", Proceedings of the 37th SIGCSE technical symposium on Computer science education, SIGCSE '06, pp.413-417