コンピュータと教育　51－2
（1999．2．23）

初心者入門用言語“若葉”によるプログラミング学習環境の設計と実現

吉良智樹*、並木美太郎*、岩崎英哉**
*東京農工大学工学部
**東京大学工学部

プログラミング教育の重要性が増す一方であるのに対し、プログラミング入門用の言語が初心者にとっては習得が難しく、その役割を果たしきれていないという現状がある。そこでアンケートなどの結果を参考に、習得が容易な新しいプログラミング言語「若葉」を設計した。一つの制御構文には一つの記述方法、変数の型を数値型と文字列型に限定、などといった言語仕様の簡略化が、プログラミング技術の習得を容易にする。また、若葉によってプログラミングの学習を行う際に、Java との関連によるポータビリティの保持、ネットワークへの対応といった特徴がある。本論文では、若葉の言語仕様と若葉コンパイラを中心とした学習環境の設計について述べる。

Design and Implement of Learning Environment for Programming with Programming Language "WAKABA" for Novice Users

Tomoki KIRA*, Mitaro NAMIKI*, Hideya IWASAKI**
Faculty of Technology, Tokyo University of Agriculture and Technology*
Faculty of Technology, Tokyo University**

Although importance of educating programming is increasing at present, choice of programming language for education is difficult for beginners. So, this paper describes about a new programming language "WAKABA" that is designed for entry course of programming and its system. This programming language has following advantages for programming education: (1)The simplification of the language specifications, that the one way of describing in one control construction, the patterns of the variable are limited to the numerical value type and the string type, makes learning of the programming technique easy. (2)An effective environment has portability and correspondence to the network by the relation with Java. This paper is stated about the language specifications of WAKABA and the design of the learning environment with WAKABA compiler .

1. はじめに

コンピュータの需要は年とともに増え続け、コンピュータを使っていない分野、業種はないといっても過言ではない。また、学術機関における実験や研究に関しても、プログラミングを必要とする分野はより一層増加した。プログラミングによるシステムの開発を行う研究はもちろんのこと、実験のデータ収集ののためにプログラミングをしたり、シミュレーションをプログラミングによって行っている。さらには非理工学系や小中高等学校でのコンピュータ利用教育、およびプログラミング教育が重要視されつつある。

この背景を受けて、プログラミングをする人はまず、プログラミングの技術を習得するための勉強をするのだが、はじめから言語 C のようなシステム開発向きのプログラミング言語に手を出してしまうと、ポインタや構造体、インクルード機能など高度な技術でも参照しなければならず、覚えることが増

えてしまうため、理解するのが困難であることも多いので、プログラミング技術習得に向いた言語が開発されるようになった。一般的には BASIC[1]や PASCAL[2]といった言語が、初心者入門用に使用されることが多い。東京農工大学の電子情報工学科においても、プログラミング序論(以降「プロ序」と表記する)というプログラミング入門講義で、BASIC の進化形態といわれる TrueBASIC[3]が入門用言語として使用されている。

しかし現実問題として、初心者入門向けのプログラミング言語を使用しても、プログラミングの技術を習得するのはなかなか難しい。実際にプロ序の講義でアンケートを実施(以降「プロ序アンケート」と表記する)した結果、プログラミング学習に苦労したという声が多く得られた。様々な理由はあるのだろうが、やはり初心者入門用のプログラミング言語が、その役割を果たし切れていないというのが現状である。そこで「システム開発にも向いた言語」ではなく、徹底的に初心者入門用にこだわった言語があってもよいのではないだろうか、と考え、新しい初心者入門用言語の設計を行うことにした。

本研究における「初心者入門用言語」とは、プログラミングの知識がなくても簡単にプログラムを作れるというものではなく、これからプログラミングを勉強しようという人が最初に習得しやすく、かつ他のプログラミング言語にも応用ができるような言語を意味する。「初心者」とは、今後もプログラミングを実験や研究などに使うであろうと思われる人である。以後この論文中において、初心者という表現は基本的にこのような人を指すこととする。しかし、必ずしも将来計算機科学や情報工学の専門に向かう人だけを対象としているわけではなく、大学の理工学系の初年度に相当する人を対象として想定している。

初心者入門用のプログラミング言語に求められるのは、習得が容易であること以外に、プログラミングの煩わしさをそれほど感じさせず、初心者が興味を持てるような題材のプログラミングが簡単にできるようにすることである。そこで本研究では、簡潔な文法、簡略化された機能、Java によるネットワーク指向、機種非依存といった特徴を持つプログラミング言語「若葉(WAKABA)」とそのプログラミング環境を設計した。若葉という名前は、初心者入門用のプログラミング言語ということもあり、自動車の初心運転者用のマークである「若葉マーク」からとったものである。そのシステムは、若葉コンパイラを中心に、プログラミング技術習得のための環境を提供する。

2. 初心者入門用プログラミング言語の調査

若葉の言語仕様は、初心者入門を目指した仕様として設計する必要がある。そこでまず、プロ序を受けた学生 52 名にアンケートを実施した(図 1)。このアンケートでは、TrueBASIC の習得度を調査し、TrueBASIC の良かった点、悪かった点を挙げてもらうことで初心者入門用言語の問題点を浮き彫りにさせる目的があった。また、初心者入門用の言語に求めることを答えてもらうことで、言語仕様設計の参考とした。その後、有志 8 名に簡単なレポートを提出してもらい、より具体的な意見を集約した。

2.1 アンケート結果から見られる傾向

アンケート結果より、初心者が入門用のプログラミング言語に求めることとして一番多かったのは、「簡単でわかりやすい書式」であった(図 1.3)。特に、グラフィックスなどが簡単にできることが望まれている。最初にプログラミング言語というものに触れる際、複雑で難しいものであればそれだけで学習する意欲がそがれてしまい、グラフィックスなどの直感的に興味に結びつくようなことが簡単にできれば学習する意欲が沸いてくる、という傾向が見られた。また、プロ序で TrueBASIC に触れる前に他の言語を学習したか、どうかを調査した(図 1.1)ところ、約 3 割の学生が以前に他の言語を学習した経験があったのだが、その学生たちは「TrueBASIC は簡単に習得できた」と答える傾向にあり、逆にまったくの初心者には「TrueBASIC は難しかった」と答える傾向にあった(図 1.2)。つまり、一度プログラミング言語について学習してしまえば、たとえ異なった言語に移行してもその言語を習得するのには、

それほどの苦労を必要としない。これはプログラミング言語の記述方法だけを習得するのではなく、データ構造、制御構造などのプログラミングについての概念を習得することができているからだと言える。

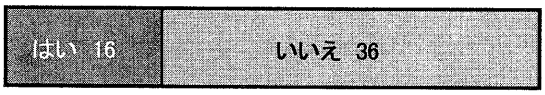

図 1.1 以前にプログラミング経験はありますか？

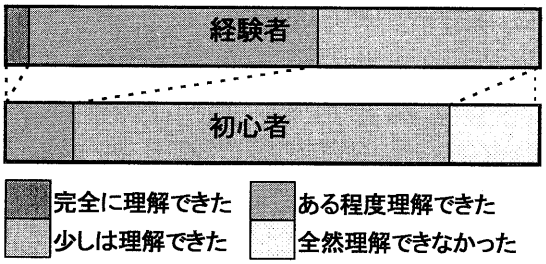

図 1.2 TrueBASIC は理解できましたか？

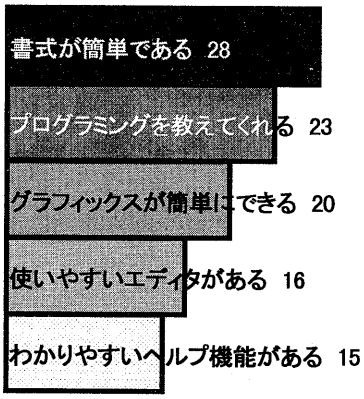

図 1.3 入門用言語に求めるものは？

図 1 プロ序アンケートの結果(抜粋)

2.2 レポートの結果から見られる傾向

レポートの結果からは、C 言語などの汎用的な言語では逆に覚えることが多くて大変である、TrueBASIC は単純でわかりやすかった、などの意見が得られた。つまり初心者には、より多くの事ができる柔軟さよりも、単純で深く考える必要のない仕様がよいという傾向があった。

また、エディタなどの統合環境の使いにくさ、マニュアルのわかりにくさ、グラフィック機能の面倒さ、などのプログラミング環境や学習環境に関する不満が多く寄せられた。

他に、学校などでプログラミングの学習を受けるだけでなく、家にある計算機で自分なりにプログラミングの勉強をしたいという人も多い。その際、家では環境が違うのでできないということや、自分が作ったプログラムを実際に動かそうと思っても家ではうまく動かない、などといった問題があった。

3. 本研究の目標

前章での問題点の結果をまとめると、

(1)覚える事柄が多く、労力が割かれてしまう

既存の言語では、構造体や分割コンパイルなどの初心者には不要と思われる機能や複雑な機能が多く、それら全てを覚えるのに苦労している。

(2)グラフィック機能などが使いにくい

グラフィックスの記述は初心者にとって楽しそうなのだが、その記述方法には苦労している。

(3)いろいろな環境で、問題なく学習したい

家や学校などの異なる環境で学習する際に、プログラムをそのままでは移植できないなどの問題が発生している。

(4)マニュアルがわかりにくい

自分でマニュアルや教科書を読んで学習しようにも、専門用語の羅列であったり、記述内容が具体的でなくわかりにくい。

(5)統合環境が使いにくい

プログラミングの環境が不便であり、思い通りにプログラミング学習が進まない。

などといった点が重要であると考えた。そこで、本研究では次の目標を定める。

(1)簡潔で理解の容易な言語仕様

同じ制御(例えばループ構造など)をするのにいくつもの記述方法があることや、数値データにいくつもの種類があること、分割コンパイルや構造体などの様々な機能があることなどは、初心者にとっては混乱の元となる。そこで、仕様を簡潔にして言語自体の習得を容易にし、プログラミング概念を習得することを目的とする。4 章にて詳細を述べる。

(2)グラフィックス機能の重視

初心者が興味を持ちやすいという意味で重要なグラフィックス機能について、出力の統一やツールの提供などによるサポートを行う。5 章にて詳細を述べる。

(3)ネットワーク指向のプログラミング言語と環境

コンパイラそのものと若葉プログラムを JavaVM[4]上で実行することにより、システムや若葉プログラムのポータビリティやネットワーク指向[5]など、初心者がどこでも学習できるプログラミング環境を提供することができる。5 章にて詳細を述べる。

(4)適切なヘルプや教材などを提供する

専門用語の羅列や、抽象的な記述などを行わず、実例を添えたり、関連個所へのリンク、検索機能などを搭載することで、わかりやすいヘルプ機能やマニュアルを整備するとともに、GUI による使いやすい統合環境の構築を行う。5 章にて詳細を述べる。

4. 若葉の言語仕様

若葉を用いたプログラミングでは、次のような技術の習得を目標としている。

(1)プログラミング構造の理解

データ構造、制御構造といったプログラミングに関する基本的な技術を習得させる。これらの知識があれば、他の言語に移行したときにも苦労が少なく、容易に習得できることがアンケート結果より示されている。

(2)アルゴリズムの概念の理解

プログラミングとアルゴリズムの関わりについての学習をさせる。プログラミングによってどのようなことが実現できるかということを理解する。四則演算、数学の公式や定理などの実装、物理シミュレーション、文字列処理、ソーティングや格納、検索などのデータ処理、などが考えられる。大規模なシステム構築や、メモリ操作などは対象としていない。

この 2 点を容易に達成するために、使える機能を限定し、初心者の混乱を押さえる簡潔な仕様とする必要がある。まず、データ構造や制御構造などに関して「一つの内容は一つの機能」という方針で設計した。また、言語モデルは手続き型言語とした。手続きを実行する順に書いていく、というトップダウン形式は処理内容を直感的に表現することができ、標準入出力、代入などを手軽に扱える点が初心者入門に適している。その他、構造体やクラス、分割コンパイルやプリプロセッサ等といった概念の排除による機能の簡潔化を行った。

次に、図 2 を参考にしながら主な仕様を述べる。

・データ型

C 言語や Java などでは豊富なデータ型を持ち、初心者の習得を妨げる原因となっている。若葉ではデータの型を次に示す二つに限定し、変数の概念の簡潔化を行うことでその理解を容易にした。

変数は使用前に宣言をすることにより、その種類を明らかにする(図 2-(5))。宣言はプログラムの先頭とそれぞれのブロックの先頭に記述でき、基本的に局所的に扱われる。例外として、プログラム先頭で宣言された変数は大域変数となる。

(1)数値データ型

全ての数値を倍精度浮動小数点数(64 ビット)に統一することで、整数や浮動小数について考える必要がなくなり、初心者が容易に数値データを扱えるようになる(図 2-(6)(7)(8))。

(2)文字列データ型

文字列データの扱いは重要なので、文字列データ型を特別に用意した。配列やポインタを使うことなく、文字列を変数に格納して扱うことができ、初心者にも扱いやすい。

また、変数をまとめて扱うことができる配列は、プログラミングにおいて重要な要素である。そこで若葉では数値データ用配列と文字列用配列を用意した。C 言語ではポインタや構造体などの技術があるが、これらはバグの原因や理解の妨げになりやすい。これらは配列によって実現可能な技術でもあるので、
若葉では扱わないことにする。

```
func abs(num x){                          //(1)関数の定義
  if(x<0) x = -x;                         //(2)条件分岐  真のとき
  else x = x;                             //(3)          偽のとき
  exit x;                                 //(4)関数からの脱出
}
num a,b,c;                                //(5)数値変数の宣言
a = 2;                                    //(6)整数も数値型
b = 1.5;                                  //(7)固定小数も数値型
c = 1e-10;                                //(8)浮動小数も数値型
loop{                                     //(9)無限ループ開始
  if((abs(b*b-a) / a) < c) exit;          //(10)ループ脱出条件
  b = ( b + a /  b) / 2;                  //(11)数値計算の結果は数値型変数
}
print(b);                                 //結果の表示
```

図 2 若葉のプログラム例（ニュートン法による平方根の近似）

・制御構造

TrueBASIC や C 言語などの既存の言語では、ループなど制御を、用途によって使い分ける。そのため、同じ制御なのに複数の記述方法が存在する。このことは、初心者に覚えることを増大させ、混乱の原因にもなる。若葉では次に示すように、一つの制御に一つのプリミティブな構文を用意することで混乱を防ぎ、習得が容易となる。また、プリミティブな構文を組み合わせてプログラムを構築する必要があり、言語に依存しない技術の習得が可能となる。

(1)条件分岐

if-else 式だけを用意した(図 2-(2)(3))。C 言語などにある switch 文に相当するものは、if-else の羅列により実現する。

(2)繰返し

無限ループだけを用意した(図 2-(9))。脱出条件をループ内の任意の場所に if 式を用いて設定する(図 2-(10))ことで、for 文や while 文のような特別な文法を覚える必要がない。

(3)脱出構文

ループ、条件分岐、関数などすべてに共通なブロック脱出構文を用意することで、ブロックからの脱出をすべて同じ概念として捉えることができる。マルチレベルブレークはなく、一番内側のブロックから脱出する(図 2-(4)(10))。

・関数

関数についても、次に示すような概念の統一化を行った。

(1)引数は参照渡し

C 言語や Java のように変数を値渡しにすると、配列が参照渡しなので概念の統一が取れず、混乱の原因となる。すべて参照渡しにすることで、変数と配列を同じように扱うことができる。 また、C 言語のようにポインタによって引数を渡す必要もなく、複数の変数を関数によって直接変更でき、直観的にもわかりやすくすることができる。

(2)返り値は数値型データ

型宣言の必要性をなくし(図 2-(1))記述を容易にすることができる。また、動的に割り当てられた文字列変数や配列を返すといった処理にも制限をかけられる。

・要素の式化

若葉において、プログラム中の要素はすべて式と

して扱う。例えば、図3に示すようにブロックが値を持つことにもなる。このことにより、関数だけでなくすべての要素がブロック脱出構文によって値を持つことが可能、と概念を統一できる。

```
x = {
  num y;
  y = getnum();
  if(y >= 0)exit y;
    //ブロック脱出構文exitが値を持って脱出
  else      exit 0;
};
y = max(50,100);
    //ブロックも関数も同じように代入が可能
```

図3 若葉のプログラム例（ブロックの式化）

5. プログラミング学習環境の構築

2章の結果を受けて、若葉によるプログラミング学習に効果的な環境を構築する。全体の構成を図4に示す。若葉コンパイラを中心に、エディタ、グラフィックス機能、ヘルプなどの統合環境を提供する。

3章で述べたように、若葉のシステムはJavaと深く結びついている。この特徴を生かし、ネットワーク指向のプログラミング環境を提供する。例えば若葉を用いたプログラミング学習の際に、学校と自宅の計算機ではOSが異なるなど計算機環境が異なった場合でも、同じシステムを用いてプログラミングを行うことができる。また、学校などで全ての計算機が同じ環境ではないといった場合にも、同じシステムによる学習を行うことができる。

またJavaのGUI構築機能により、使いやすくわかりやすいGUIをもったプログラミング環境を提供することができる。コンパイラとエディタ、そのほかデバッグ機能などを融合した統合環境も、初心者に使いやすくまとめることができる。

5.1 Javaバイトコードを生成するコンパイラ

若葉は、コンパイラによって若葉のソースコードからJavaのバイトコード[6]を生成し、JavaVM上で実行する(図5)。このことにより、計算機の環境ごとに異なったプログラミングをする必要がなくなり、学校では動作したのに家では動作しないということや、教科書どおりに記述したのに動作しない、といった弊害が起こらない。リファレンスマニュアルなども、機種やOSごとに書きかえる必要がない。

また、自分が作ったプログラムをホームページ上で公開したいという場合、若葉で作られたJavaのバイトコードによる配布を行えば、実行するOSの種類や若葉のシステムの有無に関わらず、そのプロ

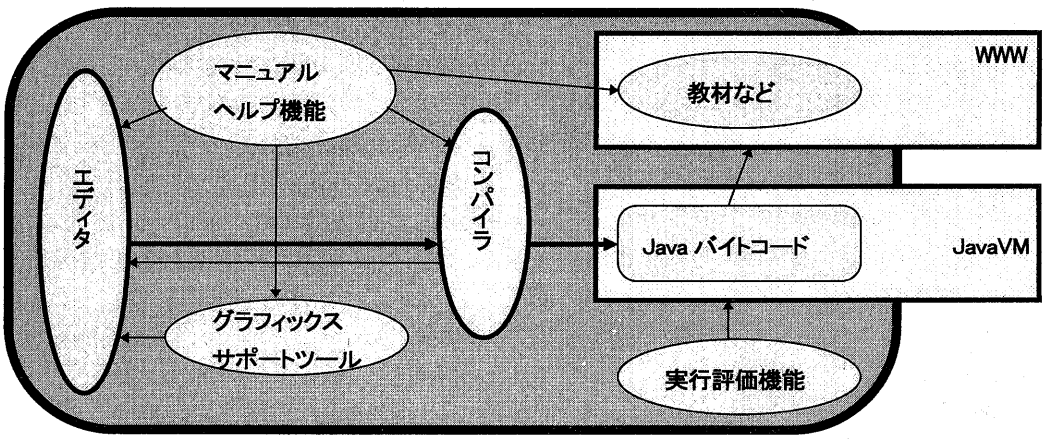

図4 若葉プログラム環境全体構成図

グラムを使ってもらうことができる。このことは、初心者のプログラミング意欲を誘うことにもつながる。

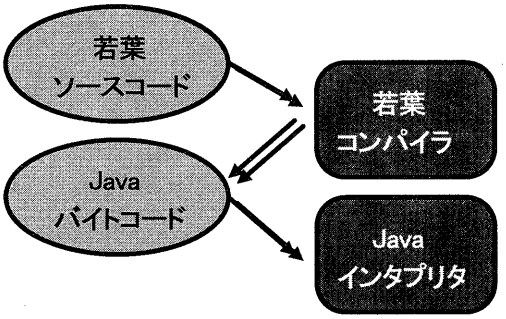

図 5 若葉によるプログラムの実行の流れ

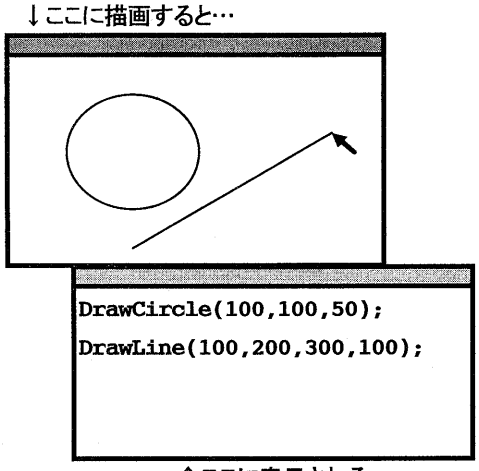

図 6 グラフィック記述サポートツールのイメージ図

5.2 教材のネットワーク上への提供

現在、教育の現場でもインターネットを用いた学習を行うことも珍しくはない。そこで、ネットワーク上に教科書があれば、教科書を容易に手にいれることができるうえに、教科書の改訂もリアルタイムで行うことができ、プログラミング学習の共有もできる。最初はプロ序の教科書を若葉用に書き換えたものを www 上に公開し、将来的には著名な教科書を参考に作り上げていく予定である。

5.3 その他設計中の環境

次に、現在設計中の環境について述べる。

(1)グラフィックスサポート機能

基本的にグラフィックスライブラリを使用した関数による記述で行うが、二つの方法でこのサポートを行う。一つ目は、標準出力とグラフィック出力の統一である。グラフィックスも標準出力も同じウインドウに出力することにより初心者の混乱をなくす。二つ目は、グラフィックス記述サポートツールの提供である。簡単にグラフィックス記述が行えるようにするために、実際に描かれた図からコードを生成することで、プログラミング例を示すというアプローチを行うツールを提供する(図 6)。

(2)マニュアル、ヘルプ

関数サーチなどの機能を取り入れ、わかりやすく使いやすいものを提供する。

(3)コンパイル作業の軽減

初心者は特にエラー修正という作業が多くなりがちなので、コンパイル作業の手間を省くために、表面上はコンパイルという作業をユーザには見せないで、編集、保存、実行というステップを提供する。

(4)エラーメッセージ

いくつものメッセージを一気に表示するのではなく、エラーが見つかった時点でメッセージを出して止めるようにする。エラーを一つ一つ解決していくほうが、初心者にはわかりやすい。またエディタと関連させて、エラー個所を指し示す機能を持たせる。

(5)プログラム評価機能

計算量や実行時間などの測定による評価のためのサポートを行う。

6.実現と評価

前章まで、若葉についての設計を述べてきた。現在、クラスライブラリやグラフィックス機能など一部未実装ではあるが、コンパイラ部までを実装しており、基本的な制御構造を用いたプログラミングが可能である。(現時点でファイル数 7 個、プログラム行数 1700 行程度。)

定性的評価のためにプロ序の教科書内のプログ

ラムを若葉の文法に書き直したところ、n 人のインディアン(入出力・ループによるプログラム)、最大公約数(代入と繰り返し・判定)、ニュートン法(図2参照)、関数・サブルーチン定義のプログラム、簡単な統計処理(配列・データ・ファイル入出力)など、グラフィックス処理やTrueBASIC独自のデータ処理を除いたプログラムを記述することができた。プログラム行数は各々15 行から 25 行程度(統計処理プログラムは53行)で、計150行程度である。このことは、若葉が初心者入門用の講義での使用に十分な言語仕様であることが言える。また、1998年11月に東京農工大学にて開催された科学技術展では動態展示を行い、若葉のプログラムと Java のプログラムを比較実行し、若葉による記述が容易であることが確認できた。

次に、システムの定量的評価を行うために、若葉コンパイラがソースコードからバイトコードを生成するための時間と、そのコードを実行するのに要した時間を測定した。プログラムは、図2に示したものを使用した。ただし機種依存による影響をなくすために、標準出力部分は削除した。比較のために同様のプログラムを Java で記述し、Java コンパイラのコンパイル時間と生成コードの実行時間を測定した。

測定環境は表1に、測定結果は表2に示す。

表1 測定環境

機種	Gateway G6-200
CPU	Pentium Pro 200MHz
メモリ	32MByte
OS	Microsoft Windows 95
Java	JDK 1.1.5

表2 測定結果

	コンパイル時間	実行時間
若葉	2.34s	2.13s
Java	5.08s	2.04s

若葉コンパイラは Java コンパイラに比べ、最適化を行わないなど機能を限定しているためか、コンパイルに要する時間が大幅に短くなった。実行時間に関しては、若葉のほうが若干速度は劣るものの、ほぼ同程度の速度のものが生成されたといえる。

コンパイル時間、実行時間ともに、プログラミング学習に使用する際には十分な速度であるといえる。

7.おわりに

本論文では、プログラミング学習における初心者の現状と要求を調査し、その結果を参考にした初心者入門用に適したプログラミング言語若葉の仕様設計と、若葉を使用した学習に効果的なプログラミング環境の設計について述べた。プログラミング環境についての実装、及びその性能評価が今後の課題となる。性能評価に際しては、実際にプログラミング教育の現場で使用してもらうことを考えている。

参考文献

[1]Thomas E. Kurtz: BASIC, ACM SIGPLAN NOTES, Vol.13, No.8, pp.103-118 (1978).

[2]N. Wirth: Recollections about the Development of Pascal, ACM SIGPLAN NOTES, Vol.28, No.3, pp.333-342 (1993).

[3]John G. Kemeny & Thomas E. Kurtz (株式会社フェムテック 訳): TrueBASIC リファレンスマニュアル,啓学出版 (1991).

[4]Tim Lindholm & Frank Yellin: The Java virtual machine specification, Addison-Wesley (1996).

[5]日本サンマイクロシステムズ株式会社 マーケティング本部 製品企画部 監訳:JAVA 言語環境 A White Paper,日本サンマイクロシステムズ株式会社(1996).

[6]James Gosling: Java Intermediate Bytecodes ACM SIGPLAN NOTES,Vol.30, No.3, pp.111-118 (1995).