3P-3

初心者用プログラミング言語「ビギン」の開発

平野　恵一†　　藤平　祥吾†　　二見　瑞樹†　　丸山　真佐夫†

木更津工業高等専門学校情報工学科†

1. はじめに

現在プログラミングを学ぶための言語として Pascal，C，Basic など様々なプログラミング言語がある．我々，木更津高専のプログラミング教育は Linux 上での C 言語を使った演習が中心である．だが，中学校を卒業したばかりの一年生がいきなり C 言語から勉強を始めるのはとても困難だ．なぜなら，入学したての一年生には，プログラミングの基礎とも言えるアルゴリズムを考える力（論理的思考力）は十分備わっていないからだ．例えば，繰り返しや配列を扱う上で，数列の知識はとても重要である．しかし，数列は高専一年生で学ぶ内容であり，プログラミングと並行して学んでいる．つまり，大多数の学生にはそれらをうまく扱うことを期待できない．

我々は，思考力向上を目的とするプログラミング学習システムを開発中である．本報告では，その中核となるプログラミング言語について述べる．

2. 設計の方針

我々は教育用の言語に必要なのは「思考力を向上させること」と考える．思考力の向上のために必要なことは学ぶ知識を限定させて，覚えることを減らし，アルゴリズムの学習に時間と労力を当てることが必要だと考えた．木更津高専の例で言うと，C 言語で繰り返しの概念を学ぶためには while の文法を覚え，等式の概念を理解する必要がある．ここに数列の問題がでて，配列を使うことになったら覚えることの多さに初心者は混乱してしまう．これを避けるために学ぶ知識を一度に見せず，段階的な学習をしていく．

また，プログラミング上の様々な制限は学習にとっても制限となる．例えば，変数の型である．変数の型は代入できる値を制限し，変数の用途を制限してしまう．これはプログラミングを学んでいく上で必要な知識であるが，プログラムが書きにくくなる．初心者にはプログラミングの幅を狭くする知識よりも思考力の向上が必要である．そこでこのような制限を取り払う．

つまり覚えることを減らし，それぞれの概念を段階的に学べるシステムを開発し，そこで学習していくことで思考力の向上が図れる．

Development of Begin：a programming language for beginner

†Kisarazu National College of Technology

```
program sample;
var 年齢;
var 氏名:array;
begin
  年齢 := 18;
  氏名[1] := “hirano”;
  氏名[2] := “keiichi”;
  print(“年齢は”，年齢);
  print(“名前は”，氏名[1],氏名[2]);
end
```

図 1　プログラム例

教育用の言語として，ドリトル[1]などがある．ドリトルはオブジェクト指向を学ぶ目的で作られている．これに対して，我々はアルゴリズムの学習に重点を置いた言語を開発する．

3. 言語仕様の設計

本研究では「思考力の向上」と「どのような言語が初心者にとって学びやすいのか」ということを第一に考え，思考力の向上を目的としたシンプルな言語を設計する．

3.1 変数の型

プログラミング言語では変数を定義するには型の宣言が必要になる．しかし，型の宣言はユーザのためというよりは機械側の都合である．また，型は変数に代入できる値を制限してしまうため，不便である．思考力の向上のための学習では型はない方が学習をしやすいと考えた．

本言語では変数の型宣言をなくして，変数にはどんな値でも代入できるようにする．

3.2 文字列と配列

C 言語では配列を用いる場合，配列変数か，もしくはポインタ変数を宣言しなくてはならない．そして変数のサイズの指定をする必要がある．このサイズの指定はわずらわしい．これは「メモリの無駄遣いをしない」という機械側の都合である．また，配列のサイズをあらかじめ決めるというのは配列に格納できるデータ数を制限してユーザの学習を妨げる．また，配列のサイズを間違えるエラーは初心者にはありがちである．

そこで本言語では図 1 で示したように，配列は使用時にサイズを自動的に確保してくれるようにする．しかし，コンピュータのメモリについてもいずれは学習をしなければいけない．そこで，サ

イズ指定もできるようにしてユーザが学ぶポイントによって選択できるようにする．

また，本言語の特徴である型無しの機能は配列型変数でも導入し，要素一つ一つに違う型の値を代入できるようにする．文字列は配列でなくふつうの変数に代入できるようにする．

3.3 繰り返し

初心者にとって，繰り返しは難易度が高い概念である．C 言語では繰り返しの記述に for 文と while 文の二つがあり，どちらを使っても同じ繰り返しを表すことができる．「どちらを使ってもよい」というのは，初心者に混乱をまねく．

本言語では繰り返しの概念のハードルを低くするために，教育用言語 Pascal を参考にして，for では繰り返しの基本である「指定した回数繰り返す」という動作を学べ，while では繰り返しの応用の「指定した状態になるまで繰り返す」という動作を学べるようにした．

しかし，段階的に学べても初心者には繰り返しの動作を理解すること難しい．そこで，言語仕様ではないが，繰り返しの動作を理解するためにプログラムの動作を GUI で表示するシステムを開発する．これにより，繰り返しの動作を目で見ながら学習できる．つまり，変数の変化を目で見ることができて初心者の理解を助ける．

4. 学習システムの構成

4.1 GUI

初心者に理解しやすい言語を作ることも大事だが，どんなにわかりやすい言語を作っても初心者にとってプログラミングの抽象的な目に見えない部分は理解しがたい．たとえば「繰り返しを用いて変数 num の値を 0 から 100 まで 1 ずつ増やしていく」という課題がある．すでにプログラミングを知っている人にとって，このプログラムを頭の中で考えるのは容易である．しかし初心者がこのような抽象的なことを頭の中で考えるのはとても難しい．常に変数の値を確認できて，動作を見ることのできるシステムがあれば初心者のプログラミングの理解が助かる．

そこで，本研究では上記で述べた初心者用言語と GUI を連結し，変数の値を常に確認できる環境を開発し，自分の行った命令と，それにともなって実行される動作を可視化する．また，図 2 のように GUI を操作してソースプログラムを作り出すことができ，動作からプログラムを学ぶこともできる．

4.2 実行処理

本言語はプログラムが不完全でも実行できデバッグのしやすいインタプリタ型の言語を開発する．

4.3 変数の履歴

数多あるプログラムのエラーの中に変数への代入ミスがある．このミスによってプログラムが動かなかったり，思った通りに動かなかったりすることがある．変数の履歴を残しておけば変数にいつ，どんな値を代入したかわかりその種のエラーを簡単に直すことができる．

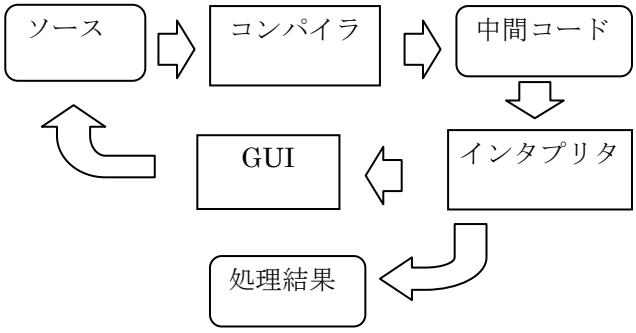

図 2 システム構成図

4.4 デバッグ

どのようなレベルのプログラマであろうと，プログラミングにデバッグは必要である．本言語では 4.2 節でも述べたようにデバッグのしやすいインタプリタ型の言語を開発する．

本言語では初心者でも簡単に理解できるような丁寧な日本語のエラーメッセージを表示させる．

しかし，文法上は正しくても意味が正しくない場合がある．このようなエラーは変数の値がユーザの思ったとおりの値になっていないのがほとんどである．これは 4.1 節で述べた GUI と 4.3 節で述べた変数の履歴を用いることで解決できる．

4.5 学習課題

思考力の向上には，その目的にあった言語とシステムが必要である．しかし，それだけでは思考力は向上しない．思考力の向上には 2 節の設計の方針でも述べたような段階的な学習が必要である．そのために，本研究では繰り返しや分岐などの概念を理解するための段階を踏んだ適切な課題をいくつか用意し，初心者は課題を解くことで自分で考える事を学び，思考力を向上させる．

例えば，繰り返しの概念を理解するには，まず文法を覚え，次に条件式を理解する．配列を用いるなら配列の概念を理解しないといけない．それらを段階的に学べる課題を用意する．

5. まとめ

今回の研究では言語の実装と GUI の実装を行った．今後の課題は，言語と GUI の連結である．

参考文献

[1] 兼宗進， 御手洗理英， 中谷多哉子， 福井眞吾，久野靖：学校教育用オブジェクト指向言語「ドリトル」の設計と実装，情報処理学会論文誌，Vol.42， No.SIG11， pp.78－90， 2001.