コンピュータと教育　28−4
（１９９３．７．１６）

アルゴリズム教育を目的としたチャート型言語システム

山本義一，辻野嘉宏，都倉信樹

大阪大学基礎工学部情報工学科

一般情報処理教育においても，計算機科学の考え方を教えることを重視するようになってきた．そのひとつであるアルゴリズムについて，従来のプログラミング言語を用いた教育では，教育を始めるまえに，計算機システムの操作方法や言語の細かな構文など本質的でない知識から教えなければならない．この問題は，計算機科学の専門教育の初期教育においても同様である．もし，アルゴリズム教育専用のシステムがあれば教育をスムーズに行なうことができる．そこで，チャートを用いたアルゴリズム教育のための言語システムを試作した．このシステムはＸウィンドウ上で動作する．本稿では，そのチャート型言語システムの設計と実現について述べる．

A chart style language system for algorithm education

Yoshikazu Yamamoto, Yoshihiro Tsujino, Nobuki Tokura

Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University

We come to attach importance to concepts of computer science in general education of data processing. Concerning algorithm, so far as traditional methods using a programing language, we cannot teach the concept of algorithm without nonessential knowledge such as commands and syntax of a particular system or language. This issue is equally true in the first step of technical education of computer science. If there is an exclusive system for algorithm education, we can teach the concept of algorithm more effectively. So we tried to develop a language system using charts for the algorithm education, which is running on X Window system. This paper presents the design and implementation of the language system.

1 はじめに

アルゴリズムの教育を行なうために，Pascal のようなプログラミング言語を用いた場合，アルゴリズムの教育を行なう以前にかなりの時間を費やす必要がある．これは，アルゴリズムを教えるためには，エディタの使い方，コンパイルの仕方，実行の仕方のような計算機システムの基礎概念，および，言語の構文が理解されていることが必要不可欠であり，初心者を対象にした場合，これらのことを一から教育しなければならないからである．つまり，計算機システムの基礎概念を教え，その後，アルゴリズムの教育をするという形になる．もちろん，学生によっては，計算機システムの基礎概念やプログラミング言語を教えることも必要である．しかし，最初からアルゴリズムの教育ができれば，計算機システムの基礎概念とアルゴリズムを並行して教えることが可能である．

また，一般に計算機言語で書かれたプログラムでは入出力処理の部分が含まれることになり，これは，アルゴリズム教育を目的とした場合，好ましいことではない．例えば，ソーティングアルゴリズムを教える場合を考えてみると，おそらくプログラム例は，(1) 配列にデータを読み込む，(2) ソーティングする，(3) 配列の内容を表示する，という流れになる．このうち，入出力処理である (1), (3) の部分は重要ではなく，かえってアルゴリズムをわかりにくくしている．

アルゴリズム教育をより良くするためには，アルゴリズム教育を目的とした専用の言語を開発するという方法が考えられる．その言語は，

- 言語仕様が簡単であり理解しやすい．
- 本質でない部分はプログラムに現われない．
- 実行の流れがわかりやすい．

という性質をもっていることが望まれる．

初心者にとってテキストよりも図的表現の方がわかりやすいと思われるので，その実現方法としては，プログラムをチャート化するのが適している．プログラムのチャート化の例として NS チャートがある (図 1)．チャート型の言語であれば言語仕様の理解が容易であり，また，実行している文をハイライトすれば実行の流れもわかりやすくなる．

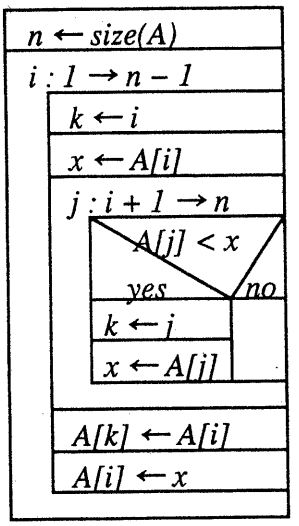

図 1: NS チャートの例

さらに，変数の値を常時表示すれば，アルゴリズムの流れの理解を助けることになるし，出力処理を行なう必要がなくなる．その上，変数の初期値を自由に変えられるようにすれば，入力処理も必要ではなくなり，入出力部分が完全にプログラムから消えることになる．

そこで，初心者を対象としたアルゴリズム教育のためのチャート型の言語システムを開発するという試みを行なった．

2 言語仕様の設計

この節では，チャート型言語システムの言語仕様の設計について考える．システムの作成を容易にするために，アルゴリズム教育に必要最小限のものを揃えた．ここでとりあげる仕様は，型，文である．

2.1 型

型の種類には大きく分けて，単純な型，構造をもった型，抽象データ型がある．

最もよく使われる単純な型は，整数型，実数型，論理型であり，少なくとも，この三種類の型は扱える必要がある．これら以外の単純な型としては，文字型，文字列型が考えられ，これらも使えた方が教育可能なアルゴリズムの幅が広がる．そこで単純な型として，整数型，実数型，論理型，文字型，文字列型を使えるようにする．

構造をもった型としては，配列型，レコード型などが挙げられる．配列型は使用用途が広く，アルゴリズム教育を行なう上で必ず扱うソーティングアルゴリズムでは必要不可欠である．

抽象データ型は，アルゴリズム教育において重要な概念である．抽象データ型としては，スタック，キュー，木，ファイルなどが考えられるが，なかでも，スタックとキューはよく使われる抽象データ型である．よって，スタックとキューを抽象データ型として扱えるようにする．なお，システムの作成を容易にするために，レコード型やポインタ型，新たな抽象データ型を作る機能は，初期バージョンでは用意しないが，データ構造などの教育まで考えるときには必要である．

2.2 文

一般に文の種類には，代入文，手続き呼び出し文，条件文，繰り返し文があり，さらに，これらの文をいくつかまとめた複文がある．

このシステムでも，これらすべての種類の文が使えるようにするが，このうち，繰り返し文はさらにいくつかの種類に分かれるのが普通である．例えば，Pascal では，while 文，for 文，repeat~until 文が用意されている．while 文があれば for 文，repeat~until 文を模擬できるので，while 文のみ用意すれば十分である．しかし，for 文は使用頻度が高い上に，while 文で模擬した場合，プログラムが見づらくなるので，for 文もあるほうがよい．一方，repeat~until 文は while 文，for 文に比較して使用する頻度は低い．このような理由から繰り返し文としては，while 文，for 文に相当する文の二種類を用意する．以上から，このシステムが提供する文の種類は，

- 代入文
- 手続き呼び出し文
- 条件文
- 繰り返し文 (while)
- 繰り返し文 (for)

の五種類である．この五種類の文が使用可能であれば不自由なく基本的なアルゴリズムを表現できると思われる．これらの文のチャート表現を図 2に示す．

3 システムの設計

ここでは，チャート型言語システムの画面構成，メニューの内容についての考察を行なう．

チャート型言語システムで表示しなければならないのは，チャート，グローバル変数，ローカル変数である．1 つのウィンドウを 3 つに分割してそれぞれを表示する方法が考えられるが，それよりもそれぞれに 1 つずつウィンドウを割り当て 3 つのウィンドウを使う方がより良い．別のウィンドウであればそれぞれの表示領域を拡大縮小することが容易であるし，見たくないウィンドウはアイコン化することも可能である．

操作方法として，メニュー選択方式を採用するので，メニューを表示する領域が必要である．その領域は 3 つのウィンドウのいずれかの中に確保することになるが，チャートを表示しているウィンドウでの作業が主となることからチャート表示用のウィンドウの最上部にメニューを表示することにする．メニューの方式としては，メニューボタンを押すとメニュー内容が表示され選択したい項目までマウスカーソルを移動させた後マウスボタンをはなす，というプルダウン方式を採用する．メニューは必要な機能をファイル，手続き・関数，編集，変数，実行，終了の七種類にわけて用意した．

プログラムが長くなるとチャートを表示するために必要な領域のサイズが大きくなる．また，変数の数が多くなるあるいは大きなサイズの配列変数を使用したりすると変数を表示するために必要

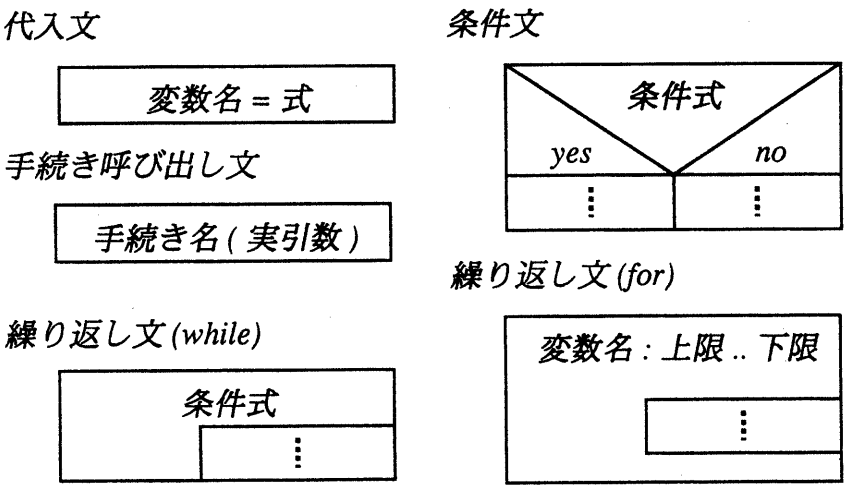

図 2: 文のチャート表現

な領域のサイズが大きくなる．そのような場合に，ウィンドウのサイズを大きくして対処する方法もあるが，ディスプレイの大きさという物理的な制限があるためいくらでも大きくするというわけにはいかない．そこで，スクロールさせて表示部分を変更できるようにする必要がある．一般にスクロールさせる方法としてスクロールバーを利用することが多い．ところが，チャート表示領域，変数表示領域ともに水平，垂直方向の二方向にスクロールする必要があり，スクロールバーを使用した場合二つのスクロールバーを使うことになる．このため，利用者はまず垂直 (水平) 方向，次に水平 (垂直) 方向をあわせるという二段階の作業をする必要がある．これは，好ましいことではない．

そこで，このシステムでは “パナー” と呼ばれるものを利用する．パナーは広い領域の一部を四角に区切って領域を指示するものである (図 3)．表示領域を変更するにはキャンバス上のスライダーを移動させればよく，目的の領域を表示する作業がスクロールバーに比べて簡単である．

以上のことから，最終的な画面構成は図 4のようになる．

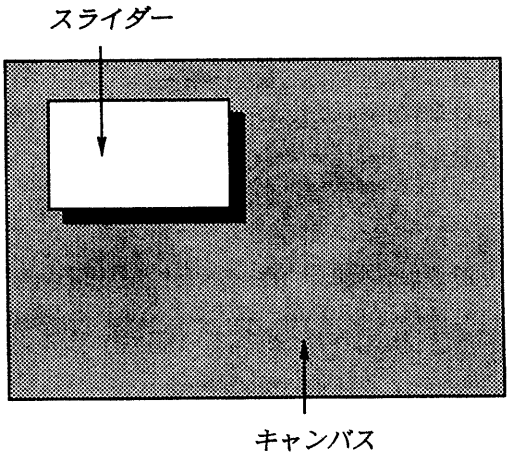

図 3: パナー

4 実現

プロトタイプの作成は，X ウィンドウ上で Athena Widget Set を用いて行なった．プログラムの行数は 1 万行強である．システムの主要な部分はチャート表示部，変数表示部，エディタ部，実行部に分けられる．これら各部分の実現方法を考える前に，システム内部でのプログラムの表現法を考えなければならない．まず考えられるのがプログラムを

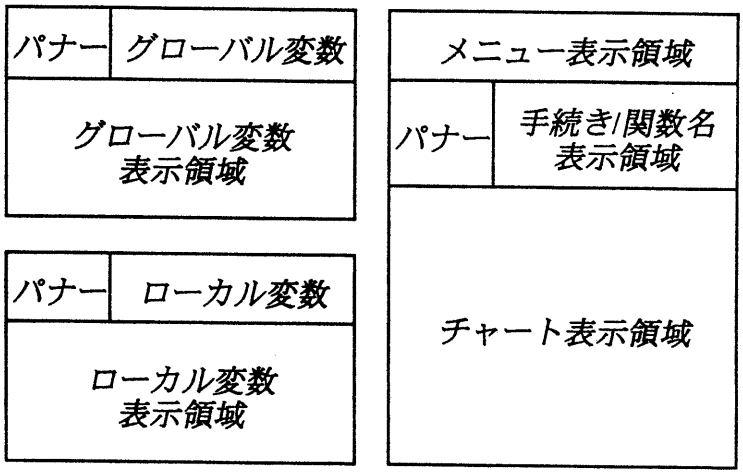

図 4: 画面構成

構造のない一次元列で扱う方法である (図 5). しかし，この方法では複文がどこからどこまでなのかを調べるのに手間がかかり適した方法ではない. 内部表現として木構造を使うと挿入，削除等の処理が簡単に行なうことができるので，本システムではプログラムを文の階層構造を表した木の形で扱うことにする. 実際にはプログラムはいくつかの手続きおよび関数からなるので，木表現は各手続き・関数ごとに作成する. 図 1を木表現で表したものを図6 に示す.

以下，チャート表示部，変数表示部，エディタ部，実行部の各部分についてその実現方法を考える.

4.1 チャート表示部

チャート表示部は，手続き (関数) の木表現からチャートを表示する部分である.

木から得られる情報は手続き (関数) の構造のみであり，チャートの図形的情報は一切得られない. つまり，チャートを表示する際に必要になる座標情報はこのチャート表示部内で計算により求めなければならない. この座標計算のアルゴリズムがこのチャート表示部の核となる.

座標計算アルゴリズムは，まず，内側に入れる箱の中で最も大きいものにあわせて箱の大きさを決め，次に，外側の箱の大きさにあわせて内側の箱を拡大していく，という二段階の処理となる.

```
assign n = size(A)
for i : i + 1 .. n
    assign k = i
    assign x = A[i]
    for j : i + 1 .. n
        if A[j] < x
            assign k = j
        else
            assign x = A[j]
        endif
    endfor
    assign A[k] = A[i]
    assign A[i] = x
endfor
```

図 5: 一次元列でのプログラムの表現例

一段階目の処理では，深さ優先探索で木を辿りながら各文を表示するのに必要な矩形の幅と高さを求める. 以下に，各文の大きさを求めるための式の概略を表 1に示す.

二段階目の処理では，一段階目で求めた各文のサイズを利用して，外側の箱の位置と大きさを決

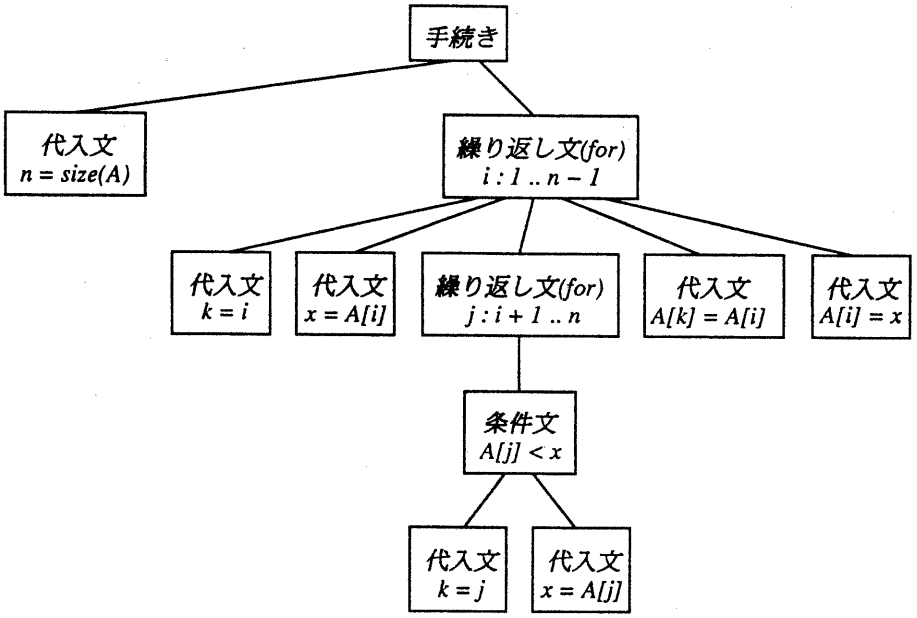

図 6: 木表現の例

め，内側の箱の位置と大きさを再帰的に決めていく．このとき，内側の箱は外側の箱に接するように拡大する．

以上のように座標計算をすることによりチャートを表示することができる．

4.2 変数表示部

変数表示部は，変数の表示処理を行なう．

単純変数のもつ属性には，変数名，型名，値があるので単純変数は図 7のように表示する．

変数名
型名
値

v
integer
1234

図 7: 単純変数の表示方法とその例

配列変数の場合には，添字と値を組にして表示する必要がある．そこで，図 8のように表示する．

抽象データ型に関しては，変数名と型名のみを

変数名	
型名	
添字	値
添字	値
添字	値
添字	値

A	
array of integer	
1	23
2	87
3	46
4	18
5	93

図 8: 配列変数の表示方法とその例

表示する．もちろん，スタックあるいはキューの内容を見たいという要求が考えられるが，情報隠蔽を強調するために内容の表示は行なわないことにする．図 9に抽象データ型変数の表示方法を示す．

各変数の表示に必要な領域サイズは異なるので変数の配置アルゴリズムを考える必要がある．ここで採用したアルゴリズムは，変数の表示領域の横幅の最大値を決めておき可能な限り横に並べ，横

手続き (関数) の大きさ: ⟨ 手続き (関数) の大きさ ⟩ := ⟨ 手続き (関数) 内の複文の大きさ ⟩.
複文の大きさ: ⟨ 複文の高さ ⟩ := ⟨ 複文内の各文の高さの和 ⟩. ⟨ 複文の幅 ⟩ := max(複文内の各文の幅).
代入文の大きさ: ⟨ 代入文の大きさ ⟩ := ⟨ その文字列の大きさ ⟩.
手続き呼び出し文の大きさ: ⟨ 手続き呼び出し文の大きさ ⟩ := ⟨ その文字列の大きさ ⟩.
条件文の大きさ: ⟨true 部の大きさ ⟩ := ⟨true 部の複文の大きさ ⟩. ⟨false 部の大きさ ⟩ := ⟨false 部の複文の大きさ ⟩. ⟨ 条件文の高さ ⟩ := ⟨ 条件式の高さ ⟩+max(⟨true 部の高さ ⟩,⟨false 部の高さ ⟩). ⟨ 条件文の幅 ⟩ := max(⟨ 条件式の幅 ⟩,⟨true 部の幅 ⟩+⟨false 部の幅 ⟩).
繰り返し文 (while) の大きさ: ⟨ 複文の大きさ ⟩ := ⟨while 内の複文の大きさ ⟩. ⟨while 文の高さ ⟩ := ⟨ 条件式の高さ ⟩+⟨ 複文の高さ ⟩. ⟨while 文の幅 ⟩ := max(⟨ 条件式の幅 ⟩,⟨ 複文の幅 ⟩).
繰り返し文 (for) の大きさ: ⟨ 複文の大きさ ⟩ := ⟨for 内の複文の大きさ ⟩. ⟨for 文の高さ ⟩ := ⟨ 条件式の高さ ⟩+⟨ 複文の高さ ⟩. ⟨for 文の幅 ⟩ := max(⟨ 条件式の幅 ⟩,⟨ 複文の幅 ⟩).

表 1: 各文の大きさの計算式

図 9: 抽象データ型変数の表示方法とその例

幅を越えるとその行に並べた変数の中で最も高さの大きい変数の高さ分だけ下に移動してから，左端から順に横に並べていく，という方法である．この方法の利点はアルゴリズムが簡単であり実現が容易であるということである．その反面，配列変数があると他の種類の変数との高さの比が大きいため間に余白ができるという欠点がある．

4.3 エディタ部

エディタ部は，手続き (関数) に文を挿入，削除を行なう部分である．

複文の最後に文を追加できるように，複文の最後にはダミーの箱を表示する．

文を挿入するときに文を入力するが，その文が文法に合っているか否かは入力時に調べることができる．よって，文法チェックはこのエディタ部で行なうことにする．

手続き・関数は構文木の形で扱っているので，挿入，削除の作業は非常に簡単に行なうことができる．文を挿入するには，適当なノードの子として追加すればよい．逆に削除を行なうには，削除する文のノードをその親から切り離せばよい．

4.4 実行部

実行部は，プログラムの実行処理を行なう部分である．

プログラムの実行時には，実行している文と参照あるいは代入された変数が一目でわかるようにする必要がある．そこで，実行している文と参照あるいは代入された変数をハイライトする (図 10)．ハイライトは，次に実行する文をハイライトさせ，その後，参照，代入された変数をハイライトさせるというふうに二段階で行なう．ステップ実行時には，それぞれのハイライトのあとで一時停止をする．

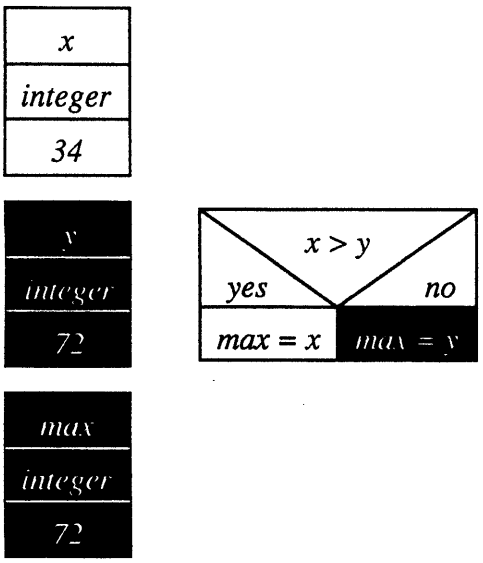

図 10: 文および変数のハイライト

実行は，手続き main の先頭から順に解析，実行していく．

実行時には，実行時エラーのチェックを行なう必要がある．主な実行時のエラーは，

- 未代入の変数を参照した．
- 未定義の手続き・関数を呼び出した．
- 被演算数の型が不適切である．
- 実引数の個数が不適切である．

などである．これらのエラーをチェックし，エラー発生時にはエラーを表示して実行を中止する．

プログラム実行中に実行を中止したいこともよくあるので，実行の中止が出来るようにする．

5 評価

作成したプロトタイプを実際に使用してみると次のような問題点があることが判明した．

- 編集機能が弱いためにチャートを少し変更するだけでも多くの手間がかかる．
- 編集時はダミーの箱を表示しているために，チャートが見づらくなってしまう．
- ダミーの箱が編集時には表示され実行時には表示されないため，チャートが編集時と実行時で変わってしまう．

これらの問題点を解消するためには，

- 編集機能を強化する．
- ダミーの箱を使わないようにする．

ようにすればよい．

編集機能の強化については，挿入，削除に加えて，削除した部分のペースト，文の内容の修正，文の種類の変更ができれば良いと思う．

ダミーの箱を使わないようにするためには，2 つのアプローチが考えられる．まず，挿入位置が一意に決まらない時は，ユーザに候補の中から選択させるという方法である．また，チャートの表現を構造の内側の箱が外側の箱に接しないように変更するという方法がある (図 11)．

6 エディタについての検討

前述のように，ダミーの箱を使わない方がよいと思われた．ダミーの箱を使わないようにするにはエディタの操作方法を変える必要がある．そこで，エディタの操作方法について検討してみた．

ダミーの箱を表示しないようにすると，候補が一意に決まらないときがある．そのときには，クリックするごとに候補を切り替えることにする．

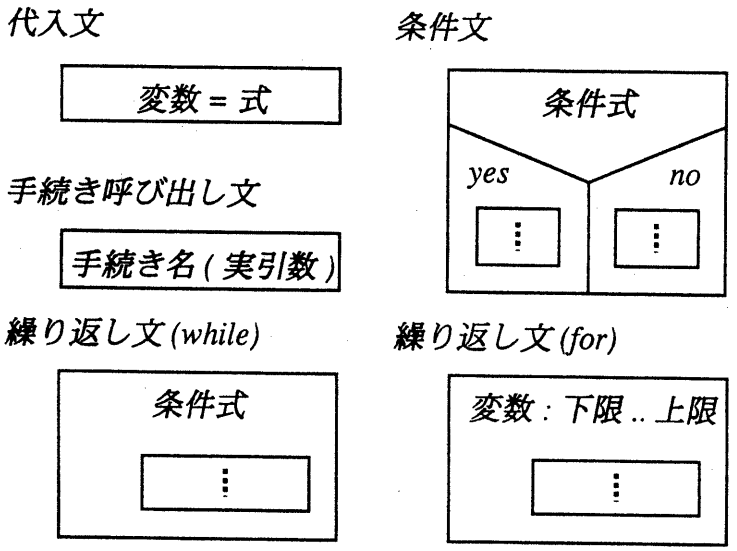

図 11: 変更した文のチャート表現

編集機能には大きく分けて，削除のように文に対する**編集**と挿入のように文間に対する**編集**がある．そこで，文の指定と文間の指定を切り替える必要がある．その実現方法として以下の 7 つの方法を考えてみた．

1. **編集**モードを切り替えることにより，文の指定方法をかえる．
2. マウスポインタの位置により，文/文間のどちらを指定するかを自動的に切り替える．その際，どちらのモードであるかがわかるようにマウスカーソルをかえる．
3. **修飾**キーを押すことにより，モードを切り替える．
4. 文／文間のどちらを指定するかをメニューから選択することで切り替える．
5. クリックされた位置からある範囲内にある文(文間) を候補として，その中から選択させる．
6. 文間のみを指定できるようにし，文に対する機能の場合には，指定した文間のすぐ上の文に対して操作をする．
7. 文のみを指定できるようにし，文間に対する機能の場合は，指定した文のすぐ上の文間に対して操作をする．ただし，一連の文の最後のみ文間を指定できるようにする．

どの方法が優れているかを判断するためにそれぞれの方法について必要な操作を比較した．まず，操作の表記法を導入する．

M	...	マウスカーソルを移動する
C_l	...	左ボタンをクリックする
P_l	...	左ボタンを押す
R_l	...	左ボタンを放す
P_{mod}	...	修飾キーを押す
R_{mod}	...	修飾キーを放す
D	...	確認する

文の挿入を 5 回，修正を 2 回行なうような簡単な編集 (図 12) を行なったときに，各方法についてどの操作が何回行なわれるかを考えると，表 2のようになった．ただし，挿入時や修正時に文字列を入力するのに必要な操作はどの方法でも同じであるので含まれていない．なお，C_lにあらわれる+は候補が複数のときにクリック回数がそれ以上になることを示す．

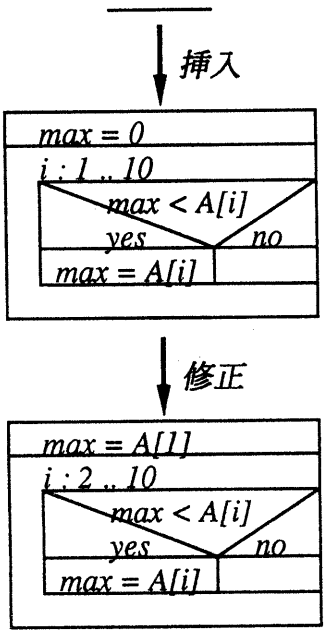

図 12: 簡単な編集例

この結果から，必要な操作が少ないのは方法5~7であることがわかる．この3つの方法についてそれぞれの問題点を考えてみて方法を決めることにした．

方法5の問題点は，候補数が多くなる場合があり指定したい文(文間)を選択するのが面倒になることがある，ということである．これは操作数が多くなることがあるということであり，好ましいことではない．

方法	M	C_l	P_l	R_l	P_{mod}	R_{mod}	D
1	18	6+	6	6			6
2	17	5+	6	6			11
3	17	5+	6	6	2	2	6
4	21	5+	8	8			8
5	17	5+	6	6			6
6	17	5+	6	6			6
7	17	5+	6	6			6

表 2: 各方法での操作コスト

方法6の問題点は，文に対する操作の場合にどの文が操作対象になっているかがわかりづらい，という点である．逆にいうと，どこを指定すれば希望する文に対する操作ができるかがわかりづらいということであり，操作性が悪くなると思われる．

方法7の問題点は，一連の文の最後のみが例外処理になることである．これは，実現の際に例外処理になるだけでユーザにとってはそれほど気になることではないと思われる．

以上から，方法7が最も適していると思われる．

7 まとめ

本稿では，初心者を対象としたアルゴリズム教育のためのチャート型言語システムの設計と実現について述べた．

このシステムは初心者でも手軽にアルゴリズムの学習が出来るように，言語仕様，システム仕様の両面において注意を払って設計を行なった．特に，メニュー形式を採用し，ウィンドウを複数使用したことにより操作がしやすいシステムになった．

また，アルゴリズムを教える立場の人がアルゴリズムの提示手段として利用することも十分に可能なシステムとなっている．

しかし，システムはプロトタイプレベルのもので使いづらいところがあり，とくに編集機能は挿入と削除という必要最低限の機能しか提供していない．そこで，ペースト，文の修正，文の種類の変更などを追加し，編集機能を充実させることが今後の課題である．さらに，変数の値をグラフ表示するなどユーザインタフェースの強化も望まれる．

また，言語仕様の面でもレコード型を導入し抽象データ型の種類を増やすことで，このシステムを使用して教育ができるアルゴリズムの幅を広げることができる．その際，それらのデータをどのように表示するかが問題となる．