

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4140980号
(P4140980)

(45) 発行日 平成20年8月27日 (2008. 8. 27)

(24) 登録日 平成20年6月20日 (2008. 6. 20)

(51) Int. Cl. F I
G O 6 F 9/06 (2006. 01) G O 6 F 9/06 5 3 0 J

請求項の数 17 (全 26 頁)

(21) 出願番号	特願平6-267118	(73) 特許権者	500046438
(22) 出願日	平成6年10月31日 (1994. 10. 31)		マイクロソフト コーポレーション
(65) 公開番号	特開平7-182147		アメリカ合衆国 ワシントン州 9805
(43) 公開日	平成7年7月21日 (1995. 7. 21)		2-6399 レッドモンド ワン マイ
審査請求日	平成13年8月21日 (2001. 8. 21)		クロソフト ウェイ
審査番号	不服2004-24908 (P2004-24908/J1)	(74) 代理人	100059959
審査請求日	平成16年12月6日 (2004. 12. 6)		弁理士 中村 稔
(31) 優先権主張番号	08/145689	(74) 代理人	100067013
(32) 優先日	平成5年10月29日 (1993. 10. 29)		弁理士 大塚 文昭
(33) 優先権主張国	米国 (US)	(74) 代理人	100065189
			弁理士 穴戸 嘉一
		(74) 代理人	100074228
			弁理士 今城 俊夫
		(74) 代理人	100084009
			弁理士 小川 信夫

最終頁に続く

(54) 【発明の名称】 プログラムツリーを表示するコンピュータプログラムの構文依存しない表示方法

(57) 【特許請求の範囲】

【請求項 1】

コンピュータプログラムの構文規則に依存しないプログラムツリーを基に指定されたプログラミング言語に対応するディスプレイ表示を発生するコンピュータシステムにおける方法であって、当該コンピュータシステムは、1以上のプロセッサとメモリ素子と表示装置と入力手段とを有するものであり、上記プロセッサは、

ユーザが入力したコンピュータコードの各ステートメントを構文解析して、宣言ポインタ及び別のノードに対するオペランドポインタが指定されている複数のノードから構成されるプログラムツリーを作成又は更新し、

上記宣言ポインタ及び上記オペランドポインタが示す上記プログラムツリーのノード間の連結関係のみを抽出することで、コンピュータプログラムの構文規則に依存しないプログラムツリーのノード及びノードの相互接続を上記メモリ素子に記憶し、

上記ユーザからの指示にตอบสนองして、上記メモリ素子から上記プログラムツリーのノードを読み出し、

ユーザ選択可能な入力としてのディスプレイ表示フォーマットを上記入力手段を介して受信すると、上記コンピュータプログラムの構文規則に依存しない上記プログラムツリーから、当該ディスプレイ表示フォーマットに従うディスプレイ表示を発生するために、

a) 非拡張であるか拡張であるかのいずれかであるリスト項目を含むディスプレイリストと称するデータ構造体を、プログラムツリーノードの内容を定義する上記宣言ポインタ及び上記オペランドポインタの識別から作成し、且つ上記ディスプレイリストの各エント

10

20

りの内容インジケータを、所定のストリングテーブルに記憶される対応のストリングノードとリンク付ける処理と、

b)上記ディスプレイリストに上記非拡張のリスト項目が含まれていれば、その非拡張リスト項目をエントリに一致するプログラムツリーノードの子ノードを含んで表現されるように、上記ディスプレイ表示フォーマットの規則に従って上記ディスプレイリストを展開し、そして展開された各エントリの内容を上記ストリングノードへ書き込むこと及び上記ディスプレイリストと上記ストリングテーブルのリンク付ける処理と、
をディスプレイ表示発生器に実行するよう指示し、

上記プログラムツリーを所定の順序でトラバースしながら、上記非拡張のリスト項目が無くなるまで上記b)を繰り返した後、上記ストリングテーブルに記憶されているデータを用いて、上記選択されたディスプレイ表示フォーマットのコンピュータプログラムを上記表示装置に表示する、ことを含む方法。

10

【請求項2】

上記ディスプレイ表示発生器は、上記ディスプレイリストを作成するために、上記プログラムツリーの1以上のノードを上記宣言ポインタ及び上記オペランドポインタを基に上記メモリ素子から検索する請求項1に記載の方法。

【請求項3】

上記宣言ポインタは、論理及び演算の演算子、定数、for構造を含む流れ制御、割当演算子、及びタイプ定義を含む計算構造を定義または記述するために用いられる請求項1に記載の方法。

20

【請求項4】

上記ユーザからの指示は、上記表示装置に表示されるプログラムツリーを構成する任意のノードを操作する1以上のコマンドによって実行される請求項1に記載の方法。

【請求項5】

上記1以上のユーザ選択可能なコマンドは、上記コンピュータシステムのユーザからの上記入力手段を介したメニュー選択入力として受け取られる請求項4に記載の方法。

【請求項6】

上記1以上のユーザ選択可能なコマンドは、上記コンピュータシステムのユーザからの入力を受け取るトークナイザの出力としてそれぞれ受け取られる請求項5に記載の方法。

【請求項7】

30

上記ノードを上記メモリ素子に記憶する処理は、演算子に対応する入力ノードのオペランドを表す、前記プログラムツリーの未定義ノードを記憶することを含む請求項1に記載の方法。

【請求項8】

未定義ノードの定義を指定することができるように、上記記憶されたプログラムツリーの複数のノードのリストを表示することを更に含む請求項7に記載の方法。

【請求項9】

上記未定義ノードのリストを表示することは、予め定められた表示で上記未定義ノードを前記表示装置に表示することを含む請求項8に記載の方法。

【請求項10】

40

上記コンピュータプログラムの構文規則に依存しない上記プログラムツリーから、当該ディスプレイ表示フォーマットに従うディスプレイ表示を発生する処理は、上記プログラムツリーのノードに関連したファンクションを呼び出してそのノードに対するディスプレイ表示を発生することを含む請求項1に記載の方法。

【請求項11】

上記ディスプレイ表示発生器を実行するファンクションは、上記ノードの表示をグラフィックベースで発生させる請求項10に記載の方法。

【請求項12】

上記ディスプレイ表示発生器を実行するファンクションは、上記ノード表示をビットマップ表示で発生させる請求項10に記載の方法。

50

【請求項 13】

上記ディスプレイ表示発生器を実行するファンクションは、上記ノードのアイコン表示を発生させる請求項 10 に記載の方法。

【請求項 14】

上記複数のノードは、論理及び演算、定数、for 構造を含む流れ制御、割当演算、及びタイプ定義を含む計算構造を定義または記述するために用いられる宣言ポインタを有するノードを含み、

上記プログラムツリー内の少なくとも 1 つのノードは、対応する宣言ノードに対する 1 以上の参照を含む請求項 1 に記載の方法。

【請求項 15】

上記宣言ノードは、上記宣言ノードに対応し且つ上記ノードによって参照された上記ファンクションを呼び出すノードに関するディスプレイ表示を発生するための対応ファンクションを有することを特徴とする、請求項 14 に記載の方法。

【請求項 16】

上記プログラムツリーのトラバース処理は、上記プログラムツリーを構成するノード間において、或る親ノードから複数の子ノード A、B が連結されている場合は、上記子ノード A に更に孫ノード a が連結していればその孫ノード a へのトラバースが終了した後に上記子ノード B へのトラバースに戻り、その子ノードに更に孫ノード b が連結していれば子ノードのトラバース後に孫ノード b のトラバースを実行するという、ノードの縦関係が優先される、請求項 1 に記載の方法。

【請求項 17】

上記非拡張のリスト項目をトラバースする所定の順序は、上記ディスプレイリストの開始部に最も近い非拡張のリスト項目から行う、請求項 1 に記載の方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】

本発明は、一般に、コンピュータプログラムを発生するコンピュータ方法及びシステムに係り、より詳細には、プログラムツリーを編集するコンピュータ方法及びシステムに係る。

【0002】

【従来の技術】

コンピュータプログラムは、通常、高レベルのプログラミング言語（例えば、パスカル及び C）で書かれる。次いで、コンパイラを用いて、高レベルプログラミング言語の命令をマシン命令に変換し、コンピュータによって実行できるようにする。コンパイルプロセスは、一般に、次の 6 段階に分けられる。

1. 語彙解析
2. 構文解析
3. 意味解析
4. 中間コード発生
5. コード最適化
6. 最終コード発生

語彙解析は、コンピュータプログラムを走査し、そして高レベル言語のコンポーネント又はトークンを確認することを伴う。この段階中に、コンパイラは、コンピュータプログラムを一連のトークンに変換し、これが構文解析中に処理される。例えば、語彙解析の間に、コンパイラは、次のステートメント

c T a b l e = 1 . 0 ;

を変数 (c T a b l e)、演算子 (=)、定数 (1 . 0) 及びセミコロンとして確認する。変数、演算子、定数及びセミコロンは、高レベル言語のトークンである。構文解析中に、コンパイラはトークンを処理し、そしてプログラムを表す構文ツリーを発生する。構文ツリーは、演算子が非リーフノードに記憶されそしてオペランドがリーフノードに記憶

10

20

30

40

50

されたツリー構造である。上記の例では、演算子(=)は、2つのオペランド、即ち変数(c T a b l e)と定数(1. 0)を有する。意味解析中に、コンパイラーは、意味の正しさを確保するように構文ツリーを変更する。例えば、変数(c T a b l e)が整数で、定数(1. 0)が実数である場合は、意味解析中に、実数－整数変換が構文ツリーに加えられる。中間コード発生、コード最適化及び最終コード発生中には、コンパイラーは、構文ツリーによって表されたプログラムを実行するためのマシン命令を発生する。これらマシン命令は、次いで、コンピュータによって実行することができる。

【0003】

【発明の構成】

本発明は、コンピュータプログラムを発生する方法及びシステムを提供する。好ましい実施例において、本発明は、プログラムツリーを直接操作するためのプログラムツリーエディタを提供する。プログラムツリーは、計算構造に対応する複数のノードを備えている。プログラムツリーエディタは、プログラミング言語構文とは独立したユーザからのコマンドを受け取る。又、本発明は、プログラムツリーのディスプレイ表示を発生するためのディスプレイ表示ジェネレータも提供する。このディスプレイ表示ジェネレータは、プログラムツリーからノードを検索し、そしてノードのディスプレイ表示を表示する。本発明のユーザは、表示されたディスプレイ表示に基づいてプログラムツリーエディタと対話するのが好ましい。

【0004】

【実施例】

本発明は、コンピュータプログラムを発生し維持するための方法及びシステムを提供する。好ましい実施例において、ソフトウェア機構(ファシリティ)は、コンピュータプログラムをプログラムツリーとして表す。プログラムツリーは、非リーフノードが演算子を表し、リーフノードがオペランドを表すツリー構造である。ファシリティは、ノードを直接操作するメカニズム、即ちプログラムツリー内においてノードを追加し、削除しそして移動するメカニズムを果たす。

【0005】

ファシリティは、プログラムツリーの部分を選択し、プログラムツリーに挿入点を配置し、そして挿入点に挿入すべきノードのタイプを選択するためのコマンドを与える。ファシリティは、現在選択された部分及び現在の挿入点に対して種々のコマンドを実行できるようにする。例えば、現在選択された部分はコピーすることもできるし、或いはクリップボードに対してカットすることもできる。次いで、クリップボードの内容を、ペーストコマンドを用いてクリップボードから現在挿入点へペーストすることができる。又、ファシリティは、現在挿入点に新たにノードを挿入するための種々のコマンド(例えば、「P a s t e =」)を与える。

【0006】

ファシリティは、好ましくは、プログラムツリーのディスプレイ表示を発生することによりプログラマにプログラムツリーを表示する。ディスプレイ表示フォーマットは、プログラムツリーに挿入できるノードの各タイプのテキスト表示を指定する。ファシリティは、好ましくは、C、パスカル、ベーシック及びリズプのような多数の異なる一般的なプログラミング言語に対しディスプレイ表示フォーマットをサポートする。これは、プログラミングツリーのディスプレイ表示を発生するためにファシリティが使用するディスプレイ表示フォーマットをプログラマがいつでも選択及び変更できるようにする。例えば、あるプログラマは、特定のプログラムツリーをCディスプレイ表示フォーマットで見るように選択することができ、そして別のプログラマは、同じプログラムツリーをリズプのディスプレイ表示フォーマットで見るように選択することができる。又、あるプログラマは、あるプログラムツリーに対し、Cのディスプレイ表示フォーマットと、リズプのディスプレイ表示フォーマットとの間で切り換えを行うことができる。

【0007】

ファシリティは、好ましくは、プログラムツリーの現在選択された部分を、プログラムツ

10

20

30

40

50

リーの対応するディスプレイ表示をハイライト処理することによりプログラマに指示する。同様に、ファシリティは、好ましくは、ディスプレイ表示内に挿入点マーク（例えば、「|」又は「^」）を表示することにより現在挿入点をプログラマに指示する。又、ファシリティは、プログラマが新たな現在部分を選択したり、或いはディスプレイ表示に基づいて挿入点を再位置設定したりできるようにする。

【0008】

図1は、ファシリティが好ましく動作するところの汎用コンピュータシステムの高レベルブロック図である。コンピュータシステム100は、中央処理ユニット（CPU）101と、コンピュータメモリ（メモリ）102と、入力／出力装置103とを備えている。ファシリティのコンピュータプログラムは、メモリに存在し、CPUにおいて実行される。入力／出力装置には、ハードディスクドライブのような記憶装置104と、キーボード105と、マウスのようなポインタ装置106と、モニタのようなディスプレイ装置107とがある。

【0009】

図2は、好ましい実施例におけるファシリティのコンポーネントのブロック図である。ファシリティ201は、プログラムツリーエディタ202と、コマンドウェルエディタ203と、ディスプレイ表示ジェネレータ204と、プログラムツリー変換器205と、コマンドウェル206と、ユーザインターフェイスマップ207と、プログラムツリー208とを備えている。プログラムツリーエディタは、使用可能なコマンドをコマンドウェルから直接又はユーザインターフェイスマップに基づいてコマンドウェルから選択するのを制御し、そしてプログラマによって選択されたコマンドの実行を行う。ユーザインターフェイスマップは、コマンドに対してメニュー項目、ツールバー及びキーをマップする。メニュー項目、ツールバー又はキーが選択されたときには、プログラムツリーエディタは、それに関連したコマンドを実行する。コマンドウェルは、プログラマの選択に使用できるコマンドと、各コマンドの実行に作用するコードとのリストを含んでいる。コマンドウェルエディタは、プログラムツリーエディタによって与えられたユーザインターフェイスをプログラマがカスタマイズできるようにする。ディスプレイ表示ジェネレータは、プログラマが選択したディスプレイ表示フォーマットに基づいてプログラムツリーのディスプレイ表示を発生し表示する。プログラムツリー変換器は、プログラムツリーを実行可能なコードに変換するのを制御する。

【0010】

プログラムツリー

図3は、例示的なプログラムツリーを示す図である。この例示的なプログラムツリーは、次のコンピュータコードに対応する。

```
{
B = 4 ;
A = B + 3 * ( 4 - C ) ;
D = A / 2 ;
}
```

この例示的なプログラムツリーは、コンピュータコードの各ステートメントに対してノード302、303、304を含んでいる。ステートメント302及び304に対するサブツリーは、図3には示されていない。ステートメント「 $A = B + 3 * (4 - C) ;$ 」に対するサブツリーは、ノード304ないし312を備えている。図4は、例示的プログラムツリーのCディスプレイ表示を示している。図5は、例示的プログラムツリーのリスプディスプレイ表示を示している。又、例示的プログラムツリーは、図3のレイアウトと同様のツリーディスプレイ表示で示すこともできる。

【0011】

プログラムツリーは、ツリーエレメント（TE）と称するノードの集合としてメモリに記憶されるのが好ましい。ファシリティは、2つの異なる種類のTE、即ち表現（EXPR）及び宣言（DCL）を維持する。各TEは、次の形態を有する。即ち、（演算子、特性

〔*i*〕、オペランド〔*i*〕)。但し、演算子及び特性は、0又はそれ以上ある。演算子フィールド及びオペランドフィールドは、他のTEに対するポインタである。表現は、プログラムツリーにおける表現を表す一種のTEである(EXPR TE)。例えば、「+」演算子に対応するEXPR TEは、演算子を定義するDCL TEに対するポインタ(演算子)を含むと共に、各オペランドごとにEXPR TEに対するポインタ(演算子〔*i*〕)を含む。オペランドがプログラムツリーのリーフノードに対応する場合には、そのオペランドのEXPR TEは、オペランドを定義する(例えば、特定の整数変数の定義)DCL TEを指す。オペランドフィールドは、DCL TE又はEXPR TEを指すことができる。例えば、図3において、ノード306は、メモリにおいてEXPR TEとして表され、演算子フィールドは、「+」演算子に対するDCL TEを指し、一方のオペランドフィールドは、「B」変数に対するDCL TEを指すEXPR TEを指し、そして他方のオペランドフィールドは、ノード308に対応するEXPR TEを指す。好ましい実施例において、プログラムツリーは、典型的な計算構造を記述するための予め定められたDCL TEを含む。典型的な計算構造は、論理及び演算の演算子、定数、流れ制御(例えば、「for」構造)、割当演算子、及びタイプ定義を含む。

10

【0012】

宣言(declaration)は、エンティティ(例えば、計算構造)の定義を表す一種のTEである(DCL TE)。この定義の一部は、エンティティを記述する情報、例えば、そのプリントネーム(例えば、「+」、それが生じるネームスペース(例えば、モジュール)、その揮発性(例えば、一定又は可変)、及び好ましくは特性として記憶される種々の他のフラグより成る。更に、宣言は、通常、1つ以上の子、即ちそのタイプ及びその初期値を有している。これらの子は、タイプ及び値を指定するDCL TEを指すことによりエンティティのタイプ及び値を定義する。EXPR TEは、オペランドにおける演算子のための宣言を指すので、宣言の特性を変更すると共に、プログラムツリー全体にわたってそれを直ちに反映することができる。例えば、特定の変数が強調される場合には、その変数に対する宣言の特性がハイライトにセットされる。その変数に対応する各EXPR TEは、その変数に対するDCL TEを指すので、プログラムツリーを処理するときに、ハイライト特性を容易にアクセスすることができる。

20

【0013】

リーフノードに対応する演算子又はオペランドのための各EXPR TEは、DCL TEのポインタを含んでいるので、各DCL TEの(特性の)ネームを任意に指定することができる。これらのネームは、ディスプレイのみに使用するのが好ましい。更に、各DCL TEは、独特なネームを必要としない。DCL TEは、非独特なネーム及びDCL TEの他の特性によるか、或いはDCL TEの位置(例えば、モジュールのネームを含む)によってユーザに識別することができる。このように、構造のフィールドとして同じネームをもつ変数を区別することができる。又、典型的なプログラミング言語に通常関連したスコープ規定の制約なしに、オペランドを宣言に関連させることができる。例えば、ネームの競合を生じることなく、異なるライブラリーからのコードを組み合わせることができる。

30

【0014】

好ましい実施例において、各DCL TEは、宣言によって記述された演算子又はオペランドを表示するための対応するファンクション(方法)を有する。以下で詳細に述べるように、ディスプレイ表示ジェネレータは、EXPR TEに対応する表示を発生するときにこのファンクションを呼び出す。EXPR TEは、DCL TEを指す。従って、ディスプレイ表示に影響するDCL TEの特性が変更されるときには、そのディスプレイ表示が次に発生されるときにその変更が反映される。

40

【0015】

リストは、集合を表す一種のTE(リストTE)である。プロジェクトのモジュール、構造体のフィールド、ユニオンのフィールド、ステートメントブロックにおける宣言、手順の公式パラメータ、及び複合ステートメントにおけるステートメントをリストしたものを

50

含む多数の種類のリストが存在する。これらのリストは、0 又はそれ以上の子を有することができる。

【0016】

特性リストは、TEの種々のアトリビュートをセットするのに使用される。例えば、TEは、ディスプレイ特性（例えば、色及びフォント）を指定するためのスタイル特性を含むことができる。又、これら特性は、他のTEから継承されてもよい。例えば、EXPR TEに対するスタイル特性は、演算子フィールドにより指されたDCL TEから継承されてもよい。

【0017】

これらのポインタ相互接続は、全て、任意の複雑さのグラフを形成する。しかしながら、これらの接続が一般的なグラフを形成しても、これら接続のサブセットはプログラムツリーを形成する。更に、プログラムツリーは、プログラムツリーにおける全てのTEをトラバースし処理するための自然の又は規範的な順序に適したスパンツリーである。

10

【0018】

プログラムツリーのルートは、リストTEである。ルートリストの宣言ツリーエンティティ（DCL TE）は、プロジェクトを構成するモジュールに対応する。プロジェクトは、コンピュータプログラムに対応する。

【0019】

プログラムツリーエディタ

プログラムツリーエディタは、プログラムツリーを形成し変更するコマンドを与える。プログラムツリーは、ノード（3つのエレメント）を形成してプログラムツリーに挿入する一連のエディタコマンドを入力することにより発生される。例えば、プログラマは、複合ステートメントに対してあるノードを選択して、子ノードを割当演算子（=）と共に追加すべきであることを指示することにより、複合ステートメントに割当ステートメントを追加する。

20

【0020】

プログラムツリーエディタは、図4のユーザインターフェイスを与える。プログラムツリーエディタは、プログラムツリーの表示をディスプレイ装置に表示するようにディスプレイ表示ジェネレータに指令する。又、プログラムツリーエディタは、コマンドウェル、メニュー、ツールバー又はキーマップから呼び出されたコマンドの性能にも影響を与える。コマンドは、典型的に、プログラムツリーを操作し（追加及び削除ノード）、挿入点を変更し、又は現在選択を変更する。他のコマンドは、プログラムツリーを記憶及び検索できるようにする。プログラムツリーエディタのユーザインターフェイスは、図4に示すように、メニュー401、水平ツールバー402、垂直ツールバー403、プログラムテキストエリア404、及び左のマージンエリア405を含んでいるのが好ましい。

30

【0021】

図6は、プログラムツリーエディタの選択メニューの選択を示している。選択メニューは、プログラムツリーの一部分を選択して挿入点を位置設定するための種々のコマンドをリストするプルダウンメニューである。

【0022】

1つの実施例において、プログラムツリーエディタは、キーボード入力されたテキストを受け取り、トークンを確認し、そしてそのトークンを一連のエディタコマンドに変換するためのトークナイザ(tokenizer)を備えている。プログラムツリーエディタは、プログラムツリーの一部分を選択しそしてプログラムツリーに挿入点を位置設定するためのコマンドを与える。又、プログラムツリーエディタは、現在選択された部分及び現在の挿入点に対して種々のコマンドを実行できるようにする。例えば、「Paste if」コマンドは、プログラムツリーの現在選択されている部分を、条件実行のための計算構造を定義するDCLを指す表現ノードと置き換える。条件実行は、一般に、現在のコンピュータ言語構文において「if」として示される。「if」トークンをタイプすると、それに対応する「Paste if」コマンドが選択に対して実行される。

40

50

【0023】

プログラムツリーエディタは、部分を選択し、挿入点を位置設定し、プログラムツリーにおけるノードを追加及び削除し、そしてプログラムツリーを操作するための種々のコマンドを与える。上記したように、コマンドウェルエディタは、プログラマがコマンドに対するキーのマップをカスタマイズできるようにする。プログラムツリーを操作する典型的なユーザインターフェイスについて以下に説明する。

【0024】

あるツリーノードのディスプレイ表示を指しそして左（マウスボタン）のダブルクリック動作によってあるサブツリーが選択される。そのノードをヘッドとする全サブツリーが選択される。サブツリーのリストは、マウスボタンを下に保持しながらリストを通して線を引くようにすることにより選択される。挿入点の選択及び位置設定は、単一のコマンドで行うことができる。単一のクリックはサブツリーを選択するが、そのクリックがノードのディスプレイ表示の左半分であったか右半分であったかに基づいて挿入点を選択の一端に位置設定する。サブツリーは選択されるが、サブツリーを変更することはできない。むしろ、選択は、挿入点に挿入されるべき演算子のオペランドを定義する。演算子が2進である場合には、選択されたサブツリーは、挿入点の配置に基づいて演算子の右又は左のオペランドとなる。欠落オペランドは、「???」と表示される。アルトクリック(atl-click)は、サブツリー全体を強調せずにサブツリーのヘッドを選択する。これは、オペランドをその位置に残したまま、演算子を置き換えられるようにする。

【0025】

種々の選択及び挿入点位置設定コマンドについて以下に説明する。

【0026】

SELECT NODE（ノードの選択）

これは、特定のノードを選択する。このコマンドを指定するために、プログラマは、表示されたプログラムのトークンの上にマウスポインタを配置しそしてマウスをクリックする。例えば、プログラマがステートメント「 $A = B + 3 * (4 - C)$ 」の中の「=」を選択すると、プログラムツリーエディタは、ノード304を選択する。以下の説明において、プログラムツリーの選択された部分をアンダーラインで指示する。以下にノードの選択を示す。

$A = B + 3 * (4 - C)$

マウスポインタを「=」の上に配置する

コマンド：ノードの選択

$A = B + 3 * (4 - C)$

【0027】

SELECT SUBTREE（サブツリーの選択）

これは、現在選択されているノードの全サブツリーを選択する。例えば、現在選択されている部分がノード304でありそしてプログラマがサブツリー選択コマンドを入力したときには、プログラムツリーエディタは、ノード304ないし312より成るサブツリーを選択する。サブツリー選択コマンドを以下に示す。 $A = B + 3 * (4 - C)$

コマンド：サブツリー選択

$A = B + 3 * (4 - C)$

【0028】

SELECT CROWN（クラウン選択）

これは、現在選択されているサブツリーのヘッドノードのみを選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマがクラウン選択コマンドを入力すると、プログラムエディタはノード304を選択する。サブツリーのクラウンノード選択するコマンドを以下に示す。

$A = B + 3 * (4 - C)$

コマンド：クラウン選択

$A = B + 3 * (4 - C)$

【0029】

SELECT PARENT (親の選択)

これは、現在選択されているサブツリーの親ノードのみを選択するか、或いは単一のノードのみが現在選択されている場合には、その親ノードを選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマが親選択コマンドを入力するときには、プログラムツリーエディタはノード301を選択する。ヘッドノードの親を選択するコマンドを以下に示す。

```
{
B = 4
A = B + 3 * (4 - C)
D = A / 2
}
```

コマンド：親選択

```
{
B = 4
A = B + 3 * (4 - C)
D = A / 2
}
```

10

【0030】

SELECT FIRST CHILD (第1の子の選択)

これは、現在選択されているサブツリー又は現在選択されているノードのヘッドノードの第1の子ノードを選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマが第1の子を選択するコマンドを入力したときは、プログラムツリーエディタは、ノード305を選択する。第1の子を選択するコマンドを以下に示す。

```
A = B + 3 * (4 - C)
コマンド：第1の子を選択する
A = B + 3 * (4 - C)
```

【0031】

SELECT LAST CHILD (最後の子の選択)

これは、現在選択されているサブツリー又は現在選択されているノードのヘッドノードの最後の子ノードを選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマがヘッドノードの最後の子を選択するコマンドを入力したときには、プログラムツリーエディタは、ノード306を選択する。最後の子を選択するコマンドを以下に示す。

```
A = B + 3 * (4 - C)
コマンド：最後の子を選択する
A = B + 3 * (4 - C)
```

【0032】

SELECT PREDECESSOR SIBLING (祖先兄弟の選択) これは、現在選択されているサブツリー又は現在選択されているノードのヘッドノードの祖先の兄弟を選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマが祖先の兄弟を選択するコマンドを入力したときには、プログラムツリーエディタは、ノード302を選択する。別の実施例において、サブツリーが現在選択されている場合には、プログラムツリーエディタは、祖先兄弟の全体を選択する。祖先の兄弟を選択するコマンドを以下に示す。

```
{
B = 4
A = B + 3 * (4 - C)
D = A / 2
```

20

30

40

50

}

コマンド：祖先兄弟を選択する

{

 $B = 4$ $A = B + 3 * (4 - C)$ $D = A / 2$

}

【0033】

SELECT SUCCESSOR SIBLING (子孫兄弟の選択)

これは、現在選択されているサブツリー又は現在選択されているノードのヘッドノードの子孫の兄弟を選択する。例えば、現在選択されている部分がノード304ないし312をもつサブツリーでありそしてプログラマが子孫の兄弟を選択するコマンドを入力したときには、プログラムエディタは、ノード303を選択する。別の実施例において、サブツリーが現在選択されている場合には、プログラムエディタは、子孫兄弟の全体を選択する。子孫の兄弟を選択するコマンドを以下に示す。

{

 $B = 4$ $A = B + 3 * (4 - C)$ $D = A / 2$

}

コマンド：子孫兄弟を選択する

{

 $B = 4$ $A = B + 3 * (4 - C)$ $D = A / 2$

}

【0034】

SELECT NEXT「???」(次のノードの選択)

これは、プログラムツリーの深さ優先の左から右へのトラバースにおいて遭遇する未定義(以下で述べる)のプログラムツリーにおける次のノードを選択する。

【0035】

SELECT ABOVE LEFT (左の上の選択)

これは、挿入点を現在選択されているサブツリー又はノードの上に配置する。多オペランド演算子に対するノードが選択されたときには、現在選択されているノードの左に新たなオペランドノードが挿入される。コマンドは、現在選択されている部分の左へマウスポインタを配置しそしてマウスをクリックすることにより選択されるのが好ましい。例えば、現在選択されている部分がノード308ないし312をもつサブツリーでありそして挿入点が現在ノードの上で左にある場合には、プログラムエディタは、新たなノードを現在選択されているサブツリーの上でその左へ挿入する。挿入点(「|」)を現在選択されているノードの上で左へ配置する場合を以下に述べる。

 $A = B + 3 * (4 - C)$ $A = B + | 3 * (4 - C)$ $A = B + E ??? 3 * (4 - C)$ $A = B + E + (3 * (4 - C))$

選択「 $3 * (4 - C)$ 」は、挿入される2進演算子「+」の右のオペランドを定義することに注意されたい。かっこは、演算子の手順を示すためにディスプレイ表示ジェネレータによって表示される。

【0036】

SELECT ABOVE RIGHT (右の上の選択)

これは、現在選択されているサブツリーノードの上に挿入点を配置する。多オペランドの

10

20

30

40

50

演算子に対するノードが挿入されるときには、新たなオペランドノードは、現在選択されているノードの右に挿入される。コマンドは、現在選択されているノードの右にマウスポインタを配置しそしてマウスをクリックすることにより選択されるのが好ましい。例えば、現在選択されている部分がノード 3 0 8 ないし 3 1 2 をもつサブツリーでありそして挿入点が現在ノードの上で右にある場合には、プログラムツリーエディタは、新たなノードを現在選択されているサブツリーの上でその右へ挿入する。挿入点を現在選択されているノードの上でその右へ配置する場合を以下に示す。

$$A = B + 3 * (4 - C)$$

$$A = B + 3 * (4 - C) \mid$$

$$A = B + 3 * (4 - C) + E$$

選択「 $3 * (4 - C)$ 」は、挿入された 2 進演算子「+」の左のオペランドを定義することに注意されたい。

【0037】

SELECT BEFORE NODE (ノードの前の選択)

これは、挿入点を現在選択されているノード又はサブツリーの左の兄弟として配置する。コマンドは、マウスポインタを現在選択されているサブツリーの左へ配置しそしてマウスをクリックすることにより選択されるのが好ましい。例えば、現在選択されている部分がノード 3 0 4 ないし 3 1 2 をもつサブツリーでありそしてプログラマが挿入点を左の兄弟として配置したときには、プログラムエディタは、その後にペーストされるノードをノード 3 0 2 と 3 0 4 との間に挿入する。ノードの前に選択するコマンドを以下に示す。

```
{
B = 4
>
A = B + 3 * (4 - C)
D = A / 2
}
```

新データ：C = 2

```
{
B = 4
C = 2
>
A = B + 3 * (4 - C)
D = A / 2
}
```

【0038】

SELECT AFTER NODE (ノードの後の選択)

これは、挿入点を現在選択されているノード又はサブツリーの右の兄弟として配置する。コマンドは、マウスポインタを現在選択されているサブツリーの右へ配置しそしてマウスをクリックすることにより入力される。例えば、現在選択されている部分がノード 3 0 4 ないし 3 1 2 をもつサブツリーでありそしてプログラマが挿入点を右の兄弟として配置したときは、プログラムツリーエディタは、その後のノードをノード 3 0 4 と 3 0 3 との間に挿入する。ノードの後に選択するコマンドを以下に示す。

```
{
B = 4
A = B + 3 * (4 - C)
>
D = A / 2
}
```

新データ：C = 2

```
{
```

10

20

30

40

50

```

B = 4
A = B + 3 * (4 - C)
>
C = 2
D = A / 2
}

```

【0039】

ある演算子は、ある数又は最小数のオペランドを必要とする。例えば、「+」演算子は、2つのオペランドを必要とする。「=」演算子が入力されたときは、プログラムツリーエディタは、2つのオペランドノードをもつ「=」ノードを形成する。オペランドがまだ指定されない場合、プログラムツリーエディタは、オペランドノードを「決定されるべき」(TBD)にセットし、これは「未定義」とも称され、「???」と表示される。例えば、ステートメント「 $A = B + 3 * (4 - C)$ 」を入力するときには、TBD識別子「???」が未定義オペランドノードを表すように表示される。ステートメントの入力を以下に示す。

```

A = ???
A = B |
A = B + ???
A = B + 3 |
A = B + 3 * ???
A = B + 3 * 4 |
A = B + 3 * (4 - ???)
A = B + 3 * (4 - C |)

```

上記の例は、プログラムツリーにノードを挿入する間にプログラムツリーエディタによって実行されるノード及び挿入点の好ましいデフォルト選択を示すものである。しかしながら、ユーザがステートメント「 $A = B + (3 * 4) - C$ 」を入力しようとする場合には、ユーザは、次の表示がなされるまで、上記のようにノードを入力する。

```

A = B + 3 * 4 |

```

次いで、ユーザは、「*」演算子によって表されたノードを選択し、現在選択を変更する。これを次に示す。

```

A = B + 3 * 4 |

```

次いで、ユーザは、次に示すように「-C」を入力して操作を進める。

```

A = B + (3 * 4) - ???
A = B + (3 * 4) - C

```

【0040】

プログラムツリーの形成の間には、多数の未定義ノードが存在する。プログラムツリーエディタは、これらのノードを識別する「T o - D o」リストを維持する。この「T o - D o」リストは、未定義ノードの形成順序又は他の順序で維持することができる。プログラムツリーを変換する前に、プログラマは、典型的に「T o - D o」リストの全てのノードが定義されるよう確保する。未定義ノードの使用は、プログラマが常に完全な計算構造を入力するように強制せずに、プログラムツリーが正しく構成されるよう確保する。例えば、プログラマは、次のものに対応するプログラムツリーを形成することができる。

```

i f (???)
{???)
e l s e
{B = 1}

```

「T o - D o」リストは、「i f」ステートメントの条件部分及び「t h e n」部分に対応する未定義ノードを含む。次いで、プログラマは、プログラムツリーの他の部分を編集し、そして後で、これらの未定義ノードを定義することができる。

【0041】

10

20

30

40

50

プログラムエディタは、挿入点においてデータをペーストすることができる。データのソースは、「ソースウェル」、クリップボード又は他の二次的なソースから得られる。ソースウェルは、次のものを含む。

全てのローカル変数

全てのタイプ

全ての計算構造

全てのグローバル

全ての演算子

全ての手順

ソースウェルの内容は、挿入点に適当に挿入できるデータを表すように挿入点が変わるときに変化する。

10

【0042】

プログラムツリーにペーストできるプリミティブノードは、コマンドウェルにおいて個別のコマンドとして得られる。例えば、「Paste+」は、「+」ノードをペーストする。プログラムテキストエリアの左にある垂直ツールバーは、これらコマンドを次の分類に編成してリストするのが好ましい。

— if、for、switch、goto、label等のステートメント

— +、-等の演算の演算子

— =、+=等の割当の演算子

— ==、>等の関係

20

— &、&&、<<等のブール及び論理演算子

— 増加及び減少

— &、*、-、>、[]等のデータ構造の演算子

— {}、a()、特殊化等のグループ分け及びコール

【0043】

キーボードは、プログラムテキストをタイプ入力するのに使用できる。キーストローク及びマウスイベントシーケンスの実質的な部分は、プログラムツリーエディタのコマンドキーストロークマッピングファシリティにより取り扱われる。プログラムツリーエディタは、キーボード入力をトークンに分割する。トークンの終わりに達する前に、バックスペースを用いてトークンを編集し、挿入点においてスクリーン上にトークンテキストを見ることが出来る。トークンは、次のいずれかによって輪郭定めされる。

30

— スペース（1つ又は複数）。

— 「a+」のように特殊な文字がアルファ又は数字に続くとき。

— 「+b」のようにアルファ又は数字が特殊な文字に続くとき。

— 幾つかの文字がそれ自体でトークンを形成し、例えば、セミコロンの前のものを終わらせる。

— アルファ及び特殊文字の両方を含むトークンに対し他の脱出規定がある。

【0044】

次いで、トークンは、種々のキーマップ及び記号リストにおいてルックアップされる。

— 先ず、トークンにキーマップ内のコマンドが直接関連される。例えば、トークン「if」は、「Paste if」コマンドの実行を生じる。

40

— 次いで、トークンは、選択からルートに向かいツリーをトラバースすることにより「ルックアップ」される。トークナイザは、構造宣言においてフィールド選択の右のオペランドをサーチする。又、トークナイザは、手順におけるローカル宣言、モジュールにおけるグローバル宣言及びシステムライブラリーにおける宣言もサーチする。又、プログラムツリーエディタは、種々のDCLENTに関連したネームを含む記号リストを維持するのが好ましい。トークナイザは、このリストを一致する名前に対してサーチすることができる。トークンが独特のネームを表さない場合には、トークナイザは、プログラマに更に別の識別情報を与えて、プログラマが非独特のネームで適当なノードを識別するよう助けることができる。

50

【0045】

従って、プログラムツリーエディタは、典型的にシステムライブラリーにおけるオペレーションのネームとして、例えば、トークン「+」を見つける。このトークンの作用は、「Paste +」コマンドを実行して「+」演算子ノードをプログラムツリーへペーストすることである。

【0046】

又、プログラムエディタのツリーは、プログラムツリーのディスプレイ表示を拡張し崩壊するためのコマンドも与える。プログラムツリーのディスプレイ表示は、アウトライン形態で全体的に又は部分的に拡張及び崩壊することができる。崩壊されるエレメントは、左マージンにおけるグレーバレットによって識別される。次のコマンドは、ディスプレイ表示を拡張及び崩壊するのに使用できる。即ち、`expand all`（全て拡張）、`expand next level`（次のレベルの拡張）、`collapse all`（全て崩壊）及び`collapse next level`（次のレベルの崩壊）。

【0047】

図8は、ファシリティのプログラムツリー変更サブルーチンのフローチャートである。ステップ801ないし807において、ファシリティは、プログラムツリーをロードし、プログラムツリーの表示を発生して表示しそしてプログラマがプログラムツリーにノードを挿入できるようにする。ステップ801において、ファシリティは、プログラマが使用できるディスプレイ表示フォーマットを選択できるようにする。好ましい実施例では、ユーザが後で別のディスプレイ表示フォーマットを選択してもよく、そしてファシリティは、プログラムツリーを表示するときにその新たに選択されたフォーマットを使用する。

【0048】

ステップ802において、ファシリティは、プログラマからのコマンドに応答してプログラムツリーをメモリにロードする。ロードされるプログラムツリーは、プログラムツリー変更サブルーチンの以前の呼び出しによって記憶されているかもしれない。又、プログラムツリーは、従来のソースコードを分解するような他の方法で形成されているかもしれない。ファシリティは、最初にプログラムツリーを構成するためにこのステップを省略するのが好ましい。

【0049】

ステップ803において、ファシリティは、選択された表示フォーマットを用いてプログラムツリーのディスプレイ表示を発生する。ディスプレイ表示を発生するために、ファシリティは、プログラムツリーを深さ優先でトラバースし、プログラムツリーの各ノードを、表示フォーマットで指定されたテキスト又はグラフィックで表す。グラフィック又はビットマップの使用は、演算子、オペランドに画像を使用できるようにし、そして宣言ネーム又はポインタ（指示された弧）を対応する宣言に対して描くことができる。例えば、一定値がIビームカーソルを表す場合には、Iビームビットマップを用いて定数の識別子を表すことができる。又、ネームに対して多数の表示フォーマット（特性として記憶された）がある。例えば、`DCL TE`は、短いネームと長いネームをもつことができる。プログラマは、短いネームを用いてプログラムツリーを表示できるが、プログラマが馴染みのない部分をデバッグするときには、プログラマは、長いネーム（おそらく、より記述的である）を表示することができる。又、別のネームを種々の外国語で記憶することもできる。ディスプレイ表示ジェネレータについては、以下に詳細に述べる。

【0050】

ステップ804において、ファシリティは、ディスプレイ表示をディスプレイ装置に表示する。ファシリティは、挿入点に対応するディスプレイ表示内の位置に挿入点指示を表示する。好ましい実施例において、ファシリティは、長いディスプレイ表示の一部分のみを表示し、ユーザがディスプレイ表示内をスクロールしてファシリティがプログラムツリーの異なる部分を表示できるようにするのが好ましい。ファシリティは、ステップ803において、プログラムツリーの表示部分を形成するのに必要なだけのプログラムツリーのディスプレイ表示を発生するのが好ましい。

【0051】

ステップ805において、プログラマがノードを挿入しようとする場合には、ファシリティは、ステップ806へと続いてノードを挿入するか、さもなくば、サブルーチン復帰となる。ステップ806において、ファシリティは、新たなノードを挿入する。プログラマは、挿入すべきノードのタイプと、ノードを挿入すべきところのプログラムツリー内の位置とを選択することにより新たなノードを選択する。ノードを挿入するプロセスについては、以下で詳細に述べる。

【0052】

ステップ807において、ファシリティは、挿入されたノードを含むプログラムツリーを記憶する。ファシリティは、プログラムツリーをメモリ又は記憶装置或いはその両方に記憶することができる。プログラムツリーが記憶された後の任意の時点で、それがプログラマによって種々の方法で使用される。例えば、プログラマは、記憶されたプログラマツリーを実行可能なコードに変換することもできるし、或いはプログラマは、デバッグユーティリティを呼び出すことによってプログラマツリーをデバッグすることもできる。ステップ807の後に、ファシリティはステップ803へ続き、挿入されたノードを含むプログラムツリーに対するディスプレイ表示を発生し、その発生されたディスプレイ表示を表示し、そしてプログラマが更に別のノードを挿入できるようにする。ステップ808において、挿入ノードコマンド以外のコマンドが受け取られた（例えば、挿入点の選択又は再位置設定）場合に、ファシリティは、ステップ809においてコマンドを処理し、そしてステップ803へループしてディスプレイ装置を更新する。

【0053】

図9は、ステップ803を詳細に示す新たなノード挿入ルーチンのフローチャートである。ステップ901ないし907において、ファシリティは、プログラマが挿入点において指定したタイプのノードを挿入し、その新たなノードによって必要とされる位置保持ノード挿入し、そして新たな挿入点を選択する。

【0054】

ステップ901において、ファシリティは、ユーザがディスプレイ表示を参照して挿入点を選択できるようにする。このステップにおいて、ファシリティは、プログラムツリーのノード又はプログラムツリーのノード間の分離のいずれかに対応するディスプレイ表示の位置をプログラマが挿入点として選択できるようにする。プログラマは、ポインタ装置を用いてディスプレイ表示を指すことにより挿入点を選択することができる。又、プログラマは、キーボードを使用し、カーソルをディスプレイ表示内の挿入点へ移動するカーソル位置設定キーの組み合わせを押すことにより挿入点を選択してもよい。又、プログラマは、ファシリティによって与えられる挿入点位置設定コマンドを使用してもよい。これらのコマンドは、挿入点を絶対的に又は挿入点の現在位置に対して相対的にプログラムツリー内の所与の位置へ移動させる。

【0055】

ステップ903において、ファシリティは、挿入点に挿入すべきノードのタイプをプログラマが選択できるようにする。好ましい実施例において、ファシリティは、使用できるノードタイプに対してペーストコマンドのリストを表示し、ユーザがその1つを選択できるようにする。ファシリティは、各ノードタイプの子として挿入できるノードのタイプを指定するのが好ましい。ユーザがノードタイプをペーストするように選択した後、ファシリティは、選択されたタイプのノードを挿入点に適切に挿入できることを照合する。これは、このタイプのノードが挿入点より上のノードの適切な子であることをチェックすることを含む。選択されたタイプのノードを挿入点に適切に挿入できない場合、ファシリティは、プログラマが別のノードタイプ又は挿入点を選択できるようにするのが好ましい。

【0056】

ステップ904において、ファシリティは、選択されたタイプのノードを挿入点においてプログラムツリーに挿入する。挿入点がプログラムツリーのノードである場合には、ファシリティは、挿入点にあるノードを挿入されたノードと置き換える。一方、挿入点がプロ

グラムツリーのノード間の分離部にある場合には、ファシリティは、挿入されたノードを挿入点より上のノードの子としそして挿入点より下のノードを挿入されたノードの子とする。

【0057】

ファシリティは、各タイプのノードに対して最小数の子を指定するのが好ましい。ステップ905において、ファシリティは、選択されたタイプのノードに対していかに多くの子が必要とされるかを調べるようチェックする。選択されたタイプのノードが、挿入されたノードが現在有する以上の子を必要とする場合は、ファシリティは、ステップ906に続き、位置保持(placeholder)ノードと称する特殊なタイプのノードを、挿入されたノードが所要数の子をもつまでその挿入されたノードの子として挿入し、さもなくば、ファシリティは、ステップ907へ続く。

10

【0058】

ステップ907において、ファシリティは、新たな挿入点を選択する。ファシリティは、プログラマが次に選択するであろう挿入点を選択する。このため、ファシリティがステップ906において位置保持ノードを挿入した場合には、ファシリティは、挿入された位置保持ノードの1つを選択するのが好ましい。ファシリティが正しい挿入点を選択した場合には、プログラマは、次の挿入点を選択する段階をスキップし、挿入すべきノードの次のタイプの選択へと直接進むことができる。

【0059】

コマンドウエルエディタ

20

コマンドウエルエディタは、プログラマツリーエディタのユーザインターフェイスを構成を制御する。ファシリティは、ファシリティによって与えられる各コマンドを含むコマンドウエルを備えているのが好ましい。好ましい実施例において、各コマンドには、そのコマンドを実行するための独特の識別子及びファンクションが組み合わされる。好ましいコマンドウエルは、「コンピュータシステムにおいてユーザインターフェイスをカスタマイズする方法及び装置(METHOD AND SYSTEM FOR CUSTOMIZING A USER INTERFACE IN A COMPUTER SYSTEM)」と題する米国特許出願第07/819,194号に開示されている。プログラマツリーエディタは、メニューハイアラキー、種々のツールバー及び種々のキーマップを含むユーザインターフェイスを与える。コマンドウエルエディタは、プログラマがユーザインターフェイスをカスタマイズできるようにする。コマンドウエルエディタは、プログラマがメニューハイアラキーを指定できるようにする。プログラマは、メニュー項目のネームと、そのメニュー項目が選択されたときに実行するための対応するコマンドとを指定することができる。ユーザインターフェイスは水平及び垂直のツールバーを含むのが好ましい。コマンドウエルエディタは、プログラマがどのコマンドをツールバーに含ませるかを指定できるようにする。各コマンドは、典型的に、ツールバーのためのボタンを指定するビットマップを有している。ユーザインターフェイスは、左のマージンと、プログラムテキストキーマップとを含んでいるのが好ましい。コマンドウエルエディタは、プログラマがコマンドに対してキー(又はマウスイベント)のマッピング又はキーのシーケンスを指定できるようにする。例えば、プログラムテキストエリアにおけるダブルクリックを、ノードを選択するコマンドに対してマップしてもよく、一方、左マージンにおけるダブルクリックを、サブツリーを表示するコマンドに対してマップしてもよい。又、プログラマが文字「i f」(「i f」トークン)を、プログラムテキストキーマップにおける「P a s t e i f」コマンドに対してマップする場合には、プログラマがプログラムテキストエリア内に「i f」をタイプしたときに、「P a s t e i f」コマンドが実行され、現在挿入点にノードを挿入させる。

30

40

【0060】

図7は、コマンドウエルエディタのユーザインターフェイスの要素を示している。コマンドウエルエディタは、プログラマツリーエディタによって実行される全てのコマンドのリストを備えている。コマンドウエルエディタは、プログラマが各コマンドに関連したビットマップを指定できるようにする。コマンドウエルエディタは、プログラマがユーザイン

50

ターフェイスのどのカテゴリをカスタマイズすべきか選択できるようにするカテゴリリストボックスを備えている。メニューカテゴリは、プログラマがメニューハイアラーキをカスタマイズできるようにする。ツールバーカテゴリは、プログラマがツールバーをカスタマイズできるようにする。キーカテゴリは、プログラマがキーマップをカスタマイズできるようにする。

【0061】

ディスプレイ表示ジェネレータ

ディスプレイ表示ジェネレータは、プログラムツリーエディタから要求を受け取り、プログラムツリーの表示をディスプレイ装置に表示する。ディスプレイ表示ジェネレータは、現在選択されているディスプレイ表示フォーマットに基づいて表示を行うようにプログラムツリーの各ノードに要求する。ディスプレイ表示ジェネレータは、プログラムツリーを深さ優先で左から右ヘトラバースする。

10

【0062】

図10は、ステップ806を詳細に示すディスプレイ表示発生サブルーチンのフローチャートである。ツリーを表示するための公知技術が存在することが当業者に明らかであろう。ステップ1001ないし1006において、ファシリティは、深さ優先で左から右ヘプログラムツリーをトラバースしそして各ノードに対してディスプレイ表示を発生することによりプログラムツリーのディスプレイ表示を発生する。ステップ1001において、ファシリティは、プログラムツリーのサブツリーを選択することによりディスプレイ表示を発生すべきところのプログラムツリーの部分を決定する。サブツリーは、クラウンコードによって定められ、クラウンノードと、クラウンノードの全ての子孫とを含んでいる。ファシリティは、ディスプレイ表示に対して割り当てられたディスプレイ装置のスペースをディスプレイ表示が埋めようとするところのサブツリーを現在ノードが定義するまで、挿入点からツリーをアップ方向に移動することによりクラウンノードを選択するのが好ましい。

20

【0063】

ステップ1002において、ファシリティは、ディスプレイリストと称するデータ構造体を形成する。ファシリティは、ディスプレイリストを完全に拡張した後に、それを用いて表示を行う。ディスプレイリストは、各々拡張できるか又は非拡張であるかのいずれかであるディスプレイリスト項目を含むリンクされたリストである。非拡張のディスプレイリスト項目は、サブツリーのノードに対応する。一方、拡張ディスプレイリスト項目は、サブツリーの特定ノードのタイプに関連したディスプレイ表示の1つ以上の部片のうちの1つに対応する。ステップ1003において、ファシリティは、クラウンノードに対応する非拡張ディスプレイリスト項目をディスプレイリストに挿入する。ステップ1004ないし1006において、ファシリティは、ディスプレイリストが完全に拡張されるまで、ディスプレイリスト項目を徐々に拡張する。

30

【0064】

ステップ1004において、ディスプレイリストが非拡張ディスプレイリスト項目を含む場合には、ファシリティはステップ1005へ続き、さもなければ、サブルーチンは復帰となる。ステップ1005において、ファシリティは、ディスプレイリストの開始部に最も近い非拡張ディスプレイリスト項目を、選択された表示フォーマットで指定されたディスプレイリスト項目の交換セットと置き換える。ディスプレイリスト項目のあるものは、拡張できそして実際のディスプレイ表示に対応し、一方、他のものは、拡張できずそしてその置き換えられたディスプレイリストエントリに対応するノードの子であるサブツリーのノードに対応する。ステップ1006において、ファシリティは、交換セットの非拡張のディスプレイリスト項目の内容を変更し、その置き換えられるディスプレイリストエントリに対応するノードの特定の子ノードを各々指示するようにする。

40

【0065】

図11ないし14は、2つの新たなノードをサブツリーに挿入する間の種々の時間においてサブツリーの内容及びディスプレイ表示を例示した図である。図11は、挿入が実行さ

50

れる前のサブツリーの内容及びディスプレイ表示を例示している。サブツリー 1 1 1 0 は、ディスプレイ動作を指定するそのクラウン、即ちプリントノード 1 1 1 1 によって定義される。このプリントノードは、グループ分けノード（図示せず）のような別のノードの子である。「A」変数の参照ノード 1 1 1 2 及び「I S T H E A N S W E R」ストリングノード 1 1 1 3 は、プリントノードの子である。対角バーとして示された挿入点 1 1 1 4 は、プリントノードの上に配置される。ディスプレイ表示 1 1 2 0 は、サブツリーに対応するテキスト 1 1 2 1 の行を含む。垂直バーカーソルとして示された挿入点 1 1 2 2 は、この行の始めに配置される。

【0 0 6 6】

次いで、プログラマは、この挿入点をディスプレイ表示において文字「A」と「,」との間へ移動する。プログラマは、ポインタ装置で新たな位置を指すか、カーソルを新たな位置へ移動するためのカーソル位置設定キーを用いるか、又は子の位置への移動コマンドを用いることによって挿入点を移動する。図 1 2 は、ファシリティが「A」変数参照ノードの上に挿入点を移動した後のプログラムツリー及びディスプレイ表示を例示している。挿入点 1 1 4 2 は、現在、「A」文字と「,」文字との間に配置されている。挿入点 1 1 3 1 は、プリントノードと「A」変数参照ノードとの間へ移動されている。

【0 0 6 7】

次いで、プログラマは、現在挿入点に挿入（ペースト）すべき乗算ノードタイプを選択する。図 1 3 は、ファシリティが「A」変数参照ノードの親として乗算ノードを挿入した後のプログラムツリー及びディスプレイ表示を例示している。サブツリーにおいて、ファシリティは、乗算ノード 1 1 5 1 を「A」変数参照ノードの親として挿入している。乗算ノードは、少なくとも 2 つの子を必要とするので（乗算が 2 進演算であるとき）、ファシリティは、新たな乗算ノードの第 2 の子として位置保持ノード 1 1 5 2 も挿入している。ユーザは、この位置保持ノードに取って代わるノードを次に挿入し勝ちであるので、ファシリティは、挿入点 1 1 5 2 を位置保持ノードへ移動している。ディスプレイ表示においては、ファシリティは、新たな乗算ノードに対するディスプレイ表示としてアスタリスクと、新たな位置保持ノードに対するクエスチョンマークとを発生し、そして挿入点をクエスチョンマークへ移動している。

【0 0 6 8】

次いで、プログラムは、挿入点に挿入するために「B」変数参照ノードタイプを選択する。図 1 4 は、ファシリティが「B」変数参照ノードを乗算ノードの子として挿入した後のプログラムツリー及びディスプレイ表示を例示している。サブツリーにおいて、ファシリティは、位置保持ノードに置き換えるために「B」変数参照ノード 1 1 7 1 を挿入している。ユーザは、「B」変数参照ノードの後にノードを次に挿入し勝ちであるから、ファシリティは、「B」変数参照ノードの下に挿入点 1 1 7 2 を移動している。ディスプレイ表示において、ファシリティは、クエスチョンマークに代わり新たに挿入されたノードのディスプレイ表示として「B」を発生しそして挿入点を「B」の後に移動している。

【0 0 6 9】

図 1 1 ないし 1 4 において各ディスプレイ表示を発生するために、ファシリティは、ディスプレイ表示発生サブルーチンを実行する。図 1 5 ないし 2 0 は、図 1 4 に示したディスプレイ表示を発生するためのディスプレイ表示発生サブルーチンの実行を示している。図 1 5 は、ディスプレイ表示発生の開始におけるディスプレイリストを示している。サブツリー 1 2 1 0 は、ノード 1 2 1 1 ないし 1 2 1 5 を有している。ディスプレイリスト 1 2 2 0 は、単一ディスプレイリストエントリ 1 2 2 1 を有している。ディスプレイリストエントリ 1 2 2 1 は、ディスプレイリストエントリの内側の文字「U」で示されたように非拡張であり、その内容インジケータは、プリントノード 1 2 1 1 を示している。

【0 0 7 0】

図 1 6 は、ディスプレイリストエントリ 1 2 2 1 の拡張を示すディスプレイリストの図である。ディスプレイリストは、ディスプレイリスト項目 1 2 2 2 ないし 1 2 2 7 を有している。これらのディスプレイリストエントリは、ベーシック言語のディスプレイ表示フォ

10

20

30

40

50

ーマットのプリントノードに対する交換セット内のものである。又、エントリ 1 2 4 1 ないし 1 2 4 4 を有するストリングテーブル 1 2 4 0 も示されている。ディスプレイリストエントリ 1 2 2 2 は拡張され、その内容インジケータは、「P R I N T」ストリング 1 2 4 1 を示す。ディスプレイリストエントリ 1 2 2 2 が表示されたときには、「P R I N T」ストリングが表示される。ディスプレイリストエントリ 1 2 2 3、1 2 2 5 及び 1 2 2 7 も拡張され、それらの内容インジケータの各々も、ディスプレイリストが表示されるときに表示されるストリングを指示する。ディスプレイリストエントリ 1 2 2 4 は、非拡張であり、その内容インジケータは、乗算ノード 1 2 1 2 を指示する。ディスプレイリストエントリ 1 2 2 6 も、同様に非拡張であり、その内容インジケータは、「I S T H E A N S W E R」ストリングノード 1 2 1 5 を指示する。非拡張ノードについて更に述べる。

10

【0 0 7 1】

図 1 7 は、ディスプレイリストエントリ 1 2 2 4 の拡張を示すディスプレイリストの図である。ファシリティは、拡張のためにディスプレイリストエントリ 1 2 2 4 を選択する。というのは、これは、ディスプレイリストにおける最初の非拡張ディスプレイリストエントリだからである。ディスプレイリストエントリ 1 2 2 4 は、ディスプレイリスト項目 1 2 2 8 ないし 1 2 3 0 へと拡張されている。ディスプレイリストエントリ 1 2 2 8 は、非拡張であり、その内容インジケータは、「A」変数参照ノード 1 2 1 3 を指示する。ディスプレイリストエントリ 1 2 2 9 は拡張され、その内容インジケータは、「＊」ストリング 1 2 4 1 を指示する。ディスプレイリストエントリ 1 2 2 4 は非拡張であり、その内容インジケータは、「B」変数参照ノード 1 2 1 4 を指示する。

20

【0 0 7 2】

図 1 8 ないし 2 0 は、ディスプレイリストエントリ 1 2 2 6、1 2 2 8 及び 1 2 3 0 の拡張を示すディスプレイリストの図である。図 2 0 において、ディスプレイリストは完全に拡張されている。ファシリティは、ディスプレイリストをその始めから終わりへトラバースし、各ディスプレイリストエントリの内容インジケータによって指示されたストリングのテキストを表示することにより、図 1 4 に示すように非ペーストテキストを表示するための完全に拡張されたディスプレイリストを表示する。

【0 0 7 3】

本発明をその好ましい実施例について説明したが、本発明はこれらの実施例に限定されるものではない。本発明の精神から逸脱せずに変更がなされ得ることが当業者に明らかであろう。本発明の範囲は、特許請求の範囲によって限定されるものとする。

30

【図面の簡単な説明】

【図 1】ファシリティが好ましく動作する汎用コンピュータの高レベルブロック図である。

【図 2】好ましい実施例におけるファシリティの要素のブロック図である。

【図 3】例示的なプログラムツリーを示す図である。

【図 4】例示的なプログラムツリーの C ディスプレイ表示を示す図である。

【図 5】例示的なプログラムツリーのリスポディスプレイ表示を示す図である。

【図 6】プログラムツリーエディタの選択メニューの選択を示す図である。

40

【図 7】コマンドウェルエディタのユーザインターフェイスの要素を示す図である。

【図 8】ファシリティのプログラムツリー変更サブルーチンを示すフローチャートである。

【図 9】新たなノードを挿入するサブルーチンを示すフローチャートである。

【図 1 0】ディスプレイ表示発生サブルーチンを示すフローチャートである。

【図 1 1】2 つの新たなノードをサブツリーに挿入する間の種々の時間中におけるサブツリーの内容及びディスプレイ表示を例示する図である。

【図 1 2】2 つの新たなノードをサブツリーに挿入する間の種々の時間中におけるサブツリーの内容及びディスプレイ表示を例示する図である。

【図 1 3】2 つの新たなノードをサブツリーに挿入する間の種々の時間中におけるサブツ

50

リーの内容及びディスプレイ表示を例示する図である。

【図14】2つの新たなノードをサブツリーに挿入する間の種々の時間中におけるサブツリーの内容及びディスプレイ表示を例示する図である。

【図15】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【図16】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【図17】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【図18】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【図19】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【図20】ディスプレイ表示発生ツリーサブルーチンの実行を示す図である。

【符号の説明】

10

100 コンピュータシステム

101 CPU

102 コンピュータメモリ

103 入力／出力装置

104 記憶装置

105 キーボード

106 ポインタ装置

107 ディスプレイ装置

201 ソフトウェア機構（ファシリティ）

202 プログラムツリーエディタ

20

203 コマンドウエルエディタ

204 ディスプレイ表示ジェネレータ

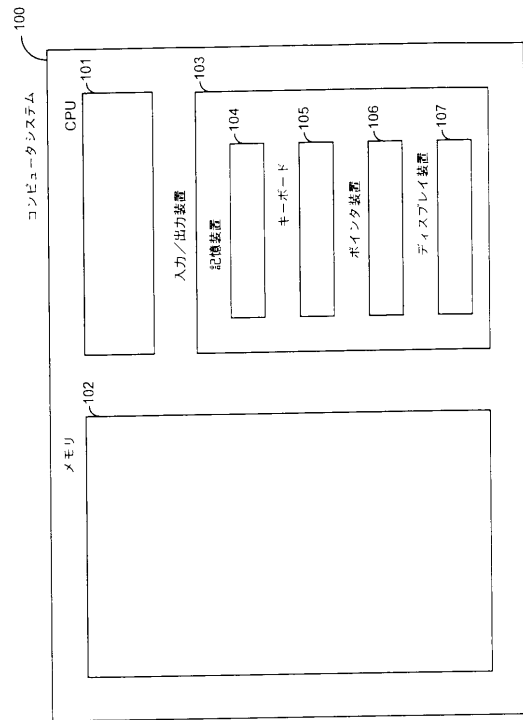
205 プログラムツリー変換器

206 コマンドウエル

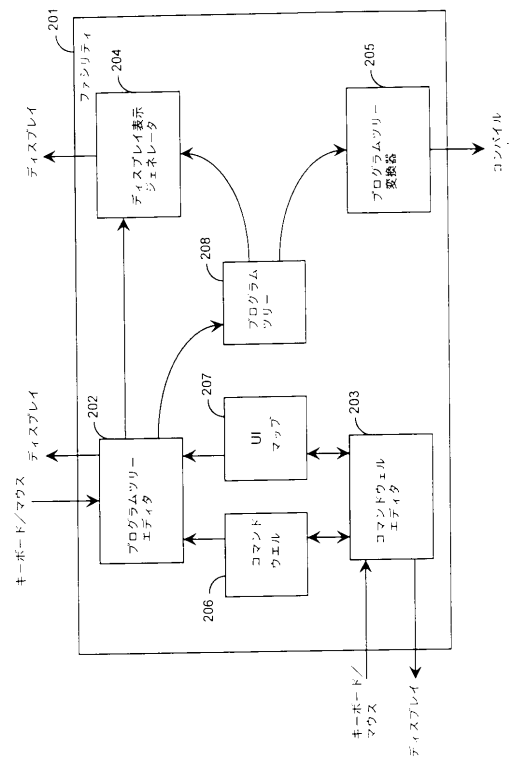
207 ユーザインターフェイスマップ

208 プログラムツリー

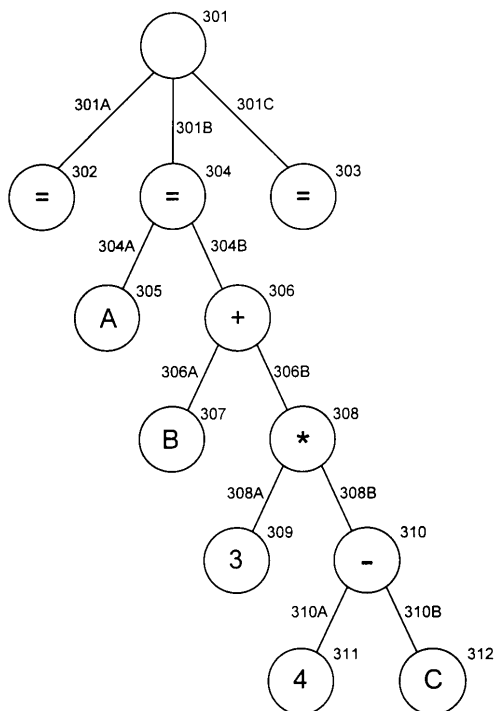
【図 1】



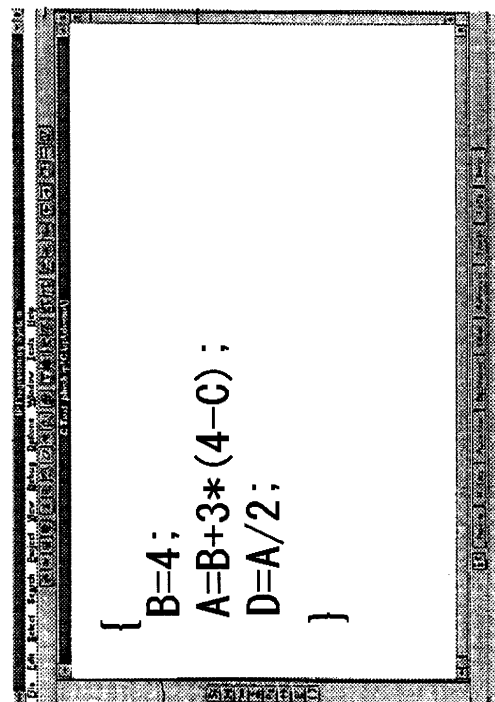
【図 2】



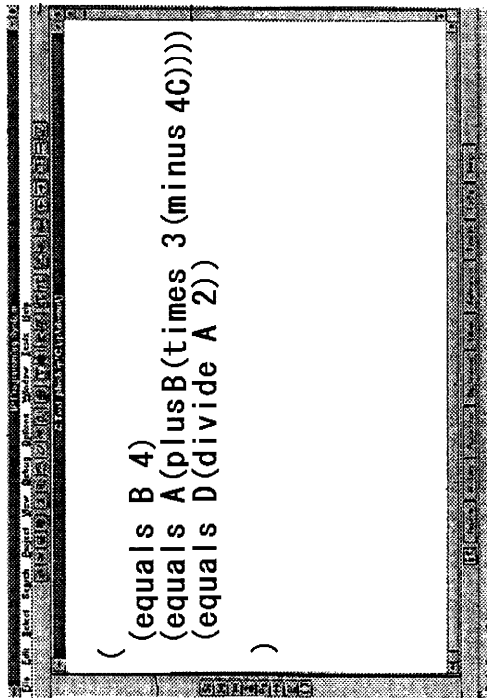
【図 3】



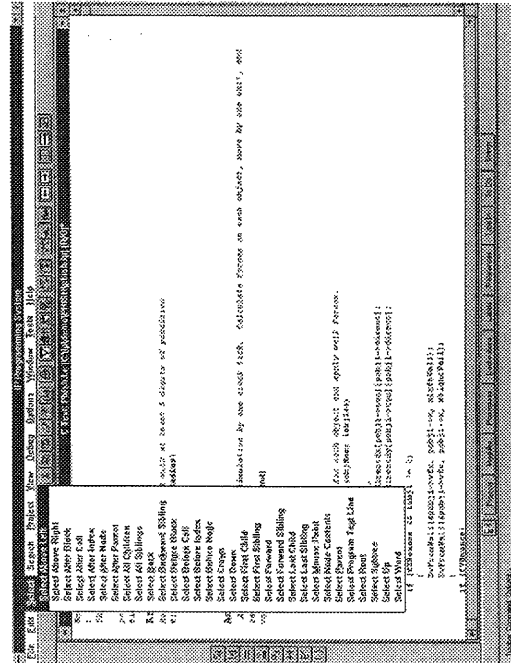
【図 4】



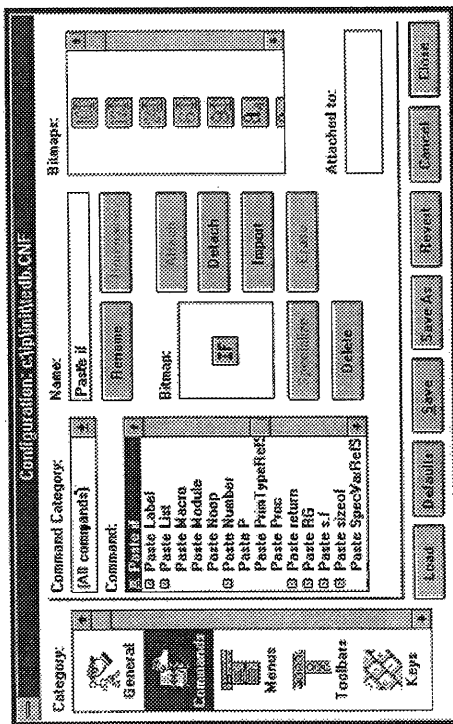
【図 5】



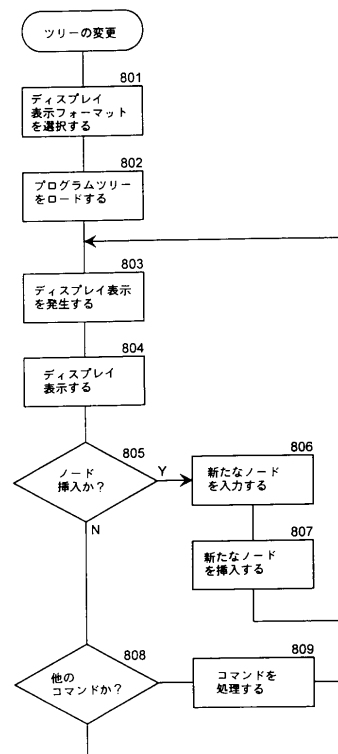
【図 6】



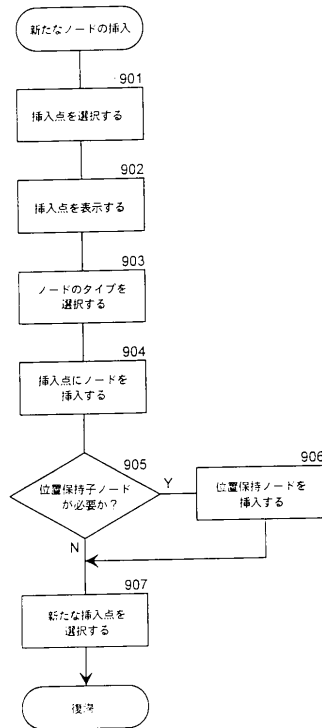
【图 7】



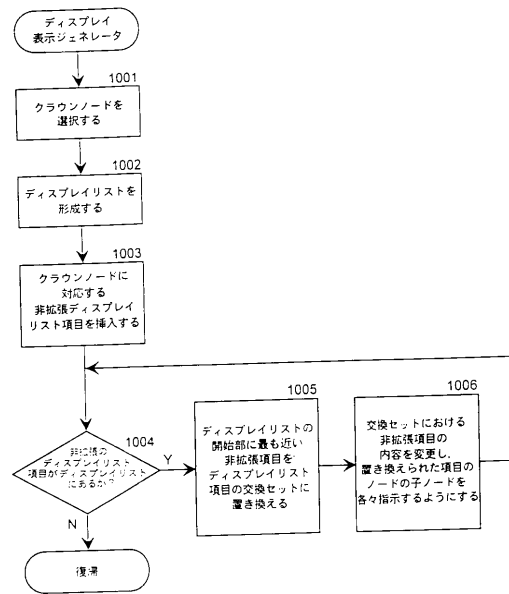
【图 8】



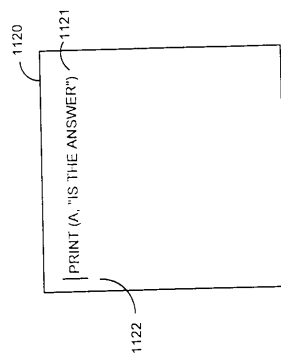
【図 9】



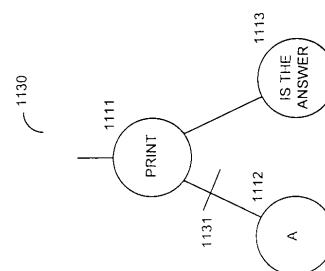
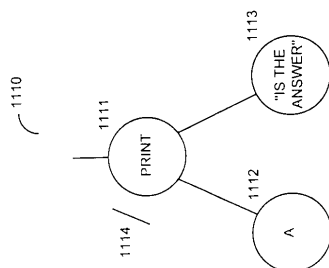
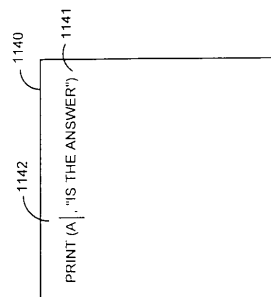
【図 10】



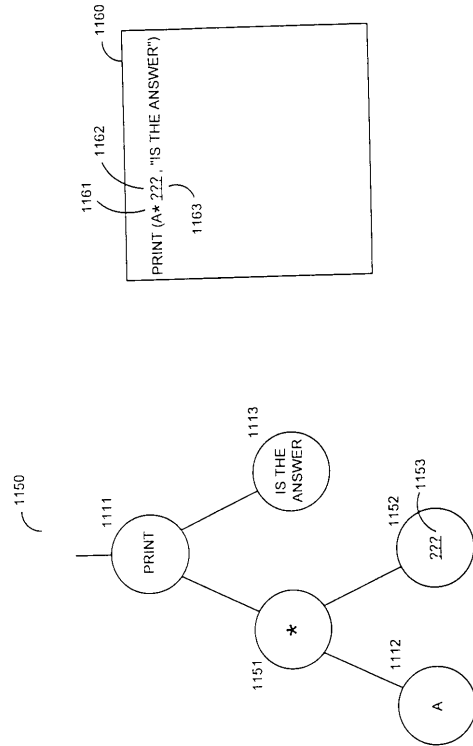
【図 11】



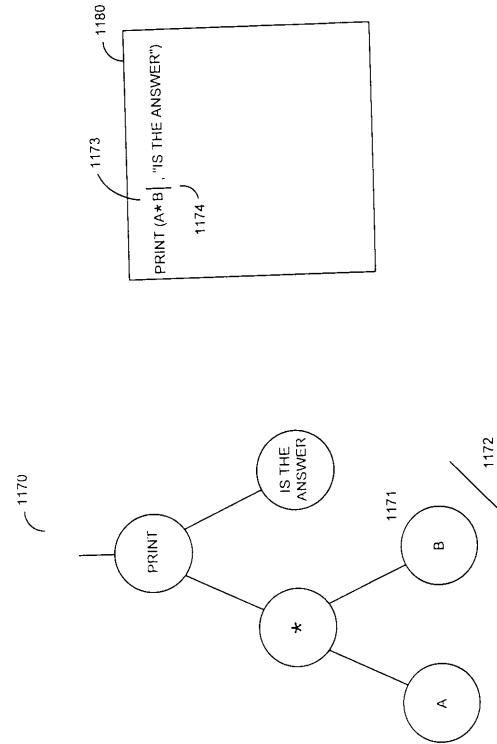
【図 12】



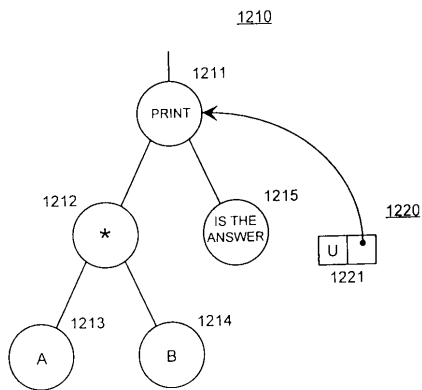
【図 13】



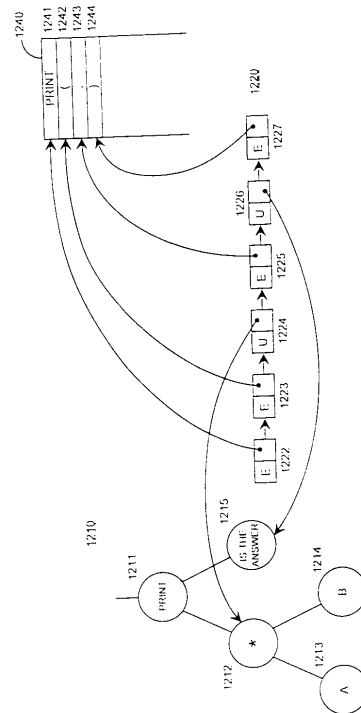
【図 14】



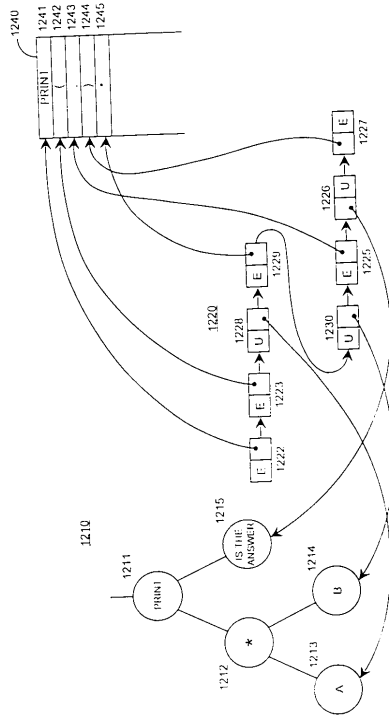
【図 15】



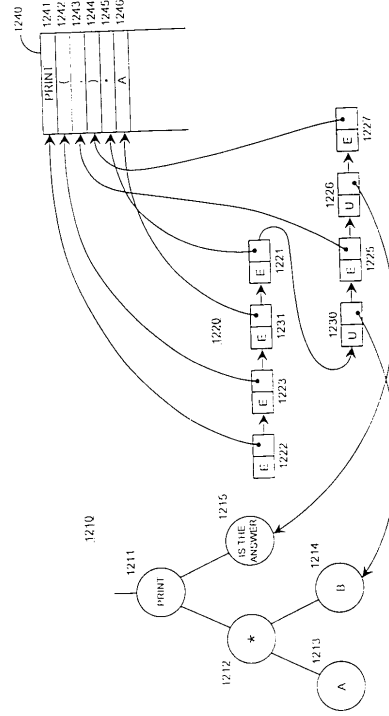
【図 16】



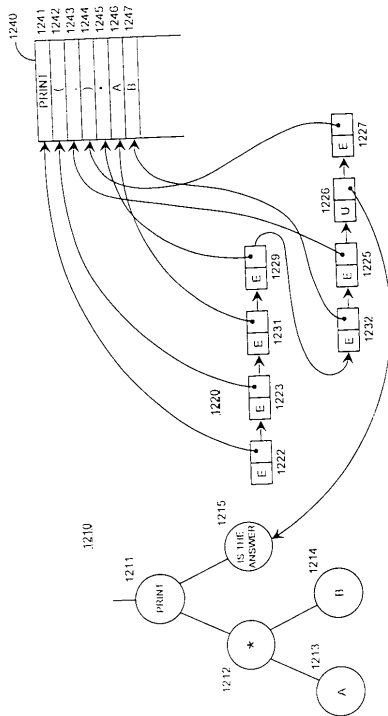
【図 17】



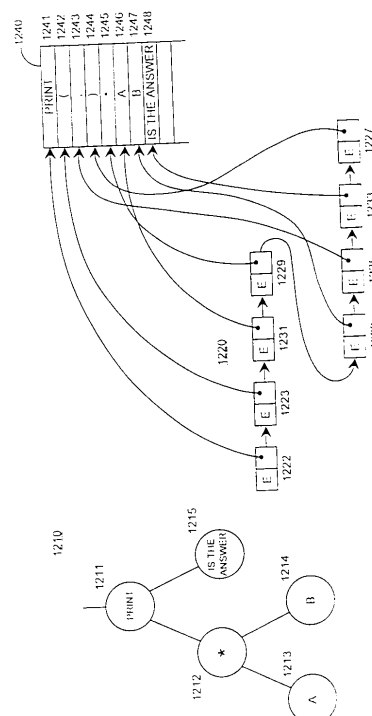
【図 18】



【図 19】



【図 20】



フロントページの続き

(74)代理人 100082821

弁理士 村社 厚夫

(74)代理人 100086771

弁理士 西島 孝喜

(72)発明者 チャールズ サイモニー

アメリカ合衆国 ワシントン州 98004 メディナ エイティフォース アベニュー ノース
イースト 109

合議体

審判長 赤川 誠一

審判官 鈴木 匡明

審判官 富吉 伸弥

(56)参考文献 特開昭61-282934号公報(JP, A)

特開平2-121030号公報(JP, A)

特開平4-140834号公報(JP, A)

渡辺喜道, 外2名, “構造エディタ生成系”, 情報処理 第32巻 第3号, 社団法人情報処理
学会, 1991年3月15日, 第32巻 第3号, P. 282-294

(58)調査した分野(Int.Cl., DB名)

G06F 9/06 530J