1. 付録

1.1. マニュアル

1.1.1. 設計図の開き方

［開くことの出来る設計図］

- 派生図
- オブジェクト図
- イベントトレース図
- ロジックテーブルエディタ

1.1.1.1. 派生図を開く

① ＳＯＭＥのアイコンをダブルクリックする。

② 起動すると無題の派生図が開かれる。そのとき、プロジェクトを表すクラスもあらかじめ表示される。

注）派生図のタイトルバーにそのプロジェクト名が表示され、プロジェクト名変更に伴って、タイトルバーのプロジェクト名も変更される。プロジェクト名の変更は、クラス名変更の場合と同じである。

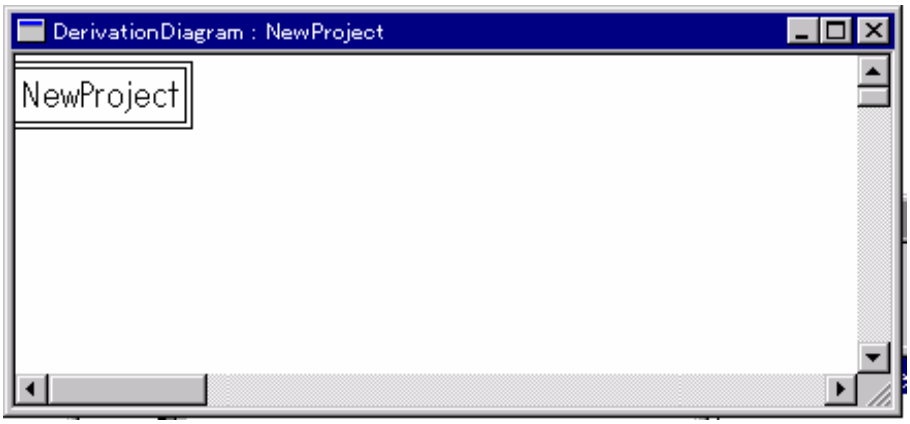


図 1-1 ＳＯＭＥ起動後の派生図

1.1.1.2. オブジェクト図を開く

① メニューバーの[ブラウズ]から[オブジェクト図]を選ぶ。

図 1-2 ブラウズメニュー

② オブジェクト図の選択ダイアログが開くので、どのクラスのオブジェクト図を開くのか選ぶ。

図 1-3 オブジェクト図を開く時のクラスの選択

③ [OK]をクリックすると、選んだクラスのオブジェクト図が開かれる。

注）もしくは派生図上で、クラス／テンプレートをダブルクリックするとそのクラスのオブジェクト図が開かれる。

注）オブジェクト図のタイトルバーにそのクラス名が表示され、クラス名の変更に伴い、タイトル名も変わる。

1.1.1.3. イベントトレース図を開く

① メニューバーの[ブラウズ]から［メソッド編集］を選ぶ。

② すると、メソッドの選択ダイアログが開くので、どのクラスのどのメソッドのイベントトレース図を開くのか選ぶ。

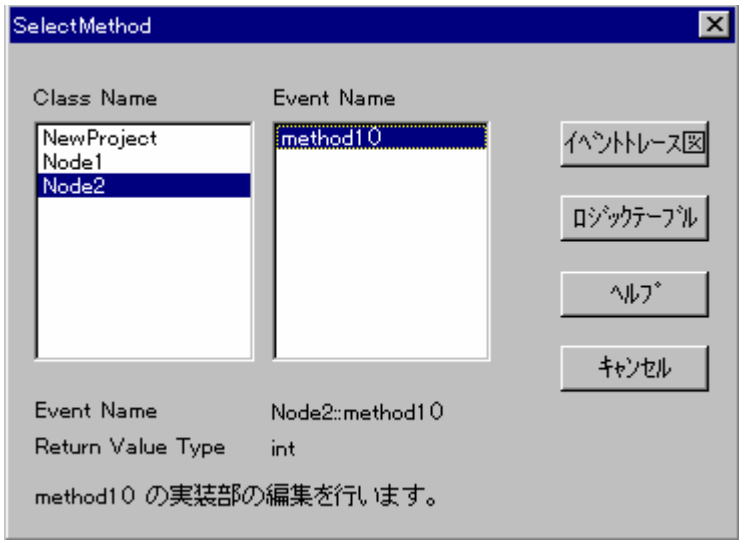


図 1-4 イベントトレース図を開く時のクラスとメソッドの選択

注）イベントがまだない状態のときには、イベントトレース図の選択ダイアログでは新規に作成できないので、オブジェクト図でイベントを定義する。その際、関数の引数は型だけでなく、仮引数名も書く。

③ ［イベントトレース図］をクリックすると、イベントトレース図が表示される。

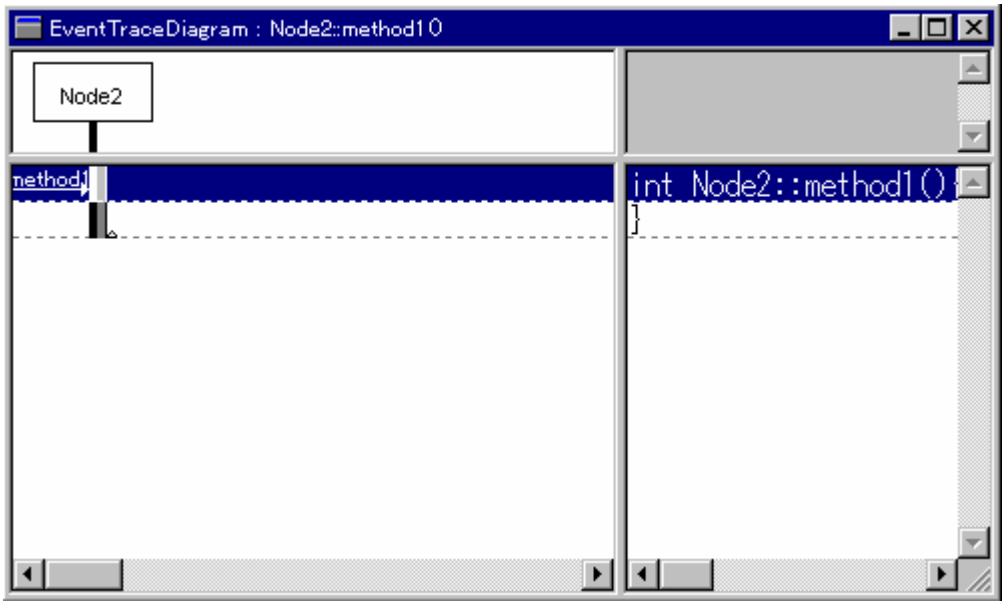


図 1-5 イベントトレース図

注）イベントトレース図のタイトルバーにはそのクラス名とメソッド名が表示される。

1.1.1.4. ロジックテーブルエディタを開く

① メニューバーの[ブラウズ]から［メソッド編集］を選ぶ。

② すると、メソッドの選択ダイアログが開くので、どのクラスのどのメソッドのロジックテーブルエディタを開くのか選ぶ。

③　［ロジックテーブル］をクリックするとロジックテーブルエディタが表示される。

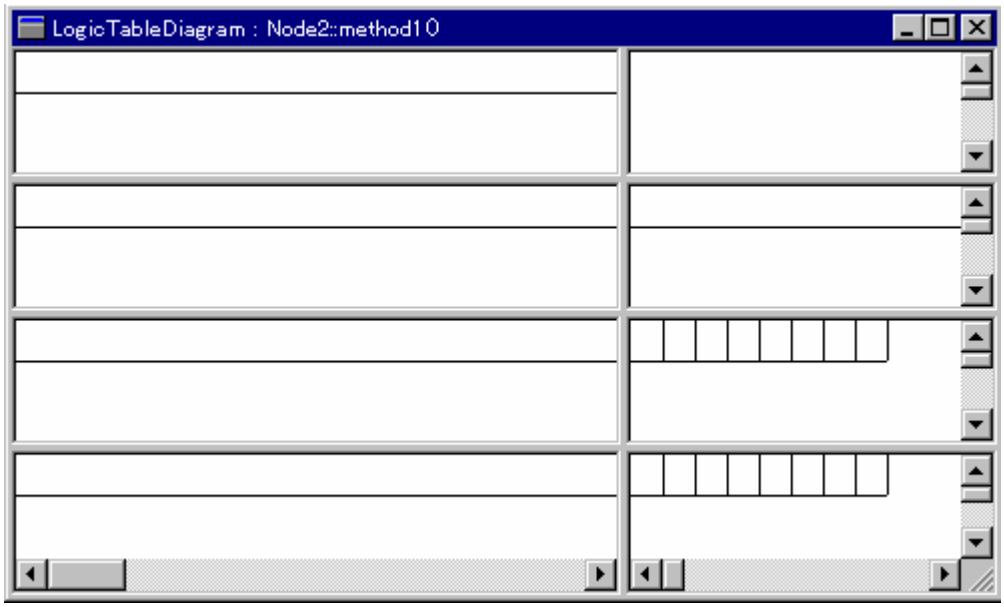


図 1-6 ロジックテーブルエディタ

1.1.2. 派生図

［派生図の目的］

- クラスの登録

  全てのクラスは派生図上で登録されなくてはならない

- 継承関係の指定

［派生図で出来ること］

- 編集対象の作成
- 編集対象の削除
- 編集対象のプロパティの設定
- オブジェクト図、イベントトレース図を開く（1.1.1設計図の開き方参照）

注）編集対象

- クラス
- テンプレート（型を引数として受け取るクラス）
- 派生関係

［派生図のツールバー］

ツールバーは、通常は編集対象選択ボタンになっている。

クラスノード生成ボタン、テンプレートノード生成ボタン、派生関係生成ボタンを押して、操作をした後は、自動的に編集対象選択ボタンが押された状態に戻る。

注）シフトキーを押しながら上記の操作を行うと、続けて操作を行うことが出来る。その場合、シフトキーを放すと自動的に編集対象選択ボタンに変わる。

1.1.2.1. 編集対象の作成

1. クラス作成

① クラス生成ボタンを押す。

② 派生図上で任意の位置にクリックすると、そこの位置を中央としてクラス（ノード）が作成される。

注）( )内はクラスのグラフ理論的な呼び方。

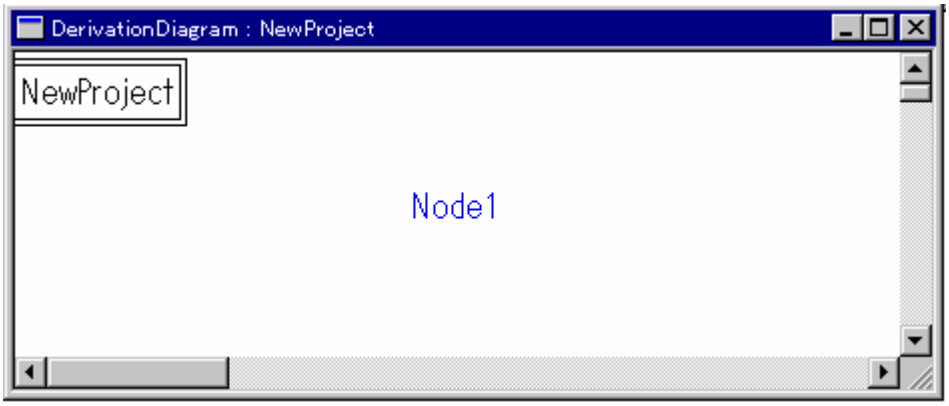


図 1-7 クラスの生成

2. テンプレートの作成

① テンプレート生成ボタンを押す。

② 派生図上でクリックすると、そこの位置を中央として、テンプレート（ノード）が作成される。

注）( )内はグラフ理論的な呼び方。

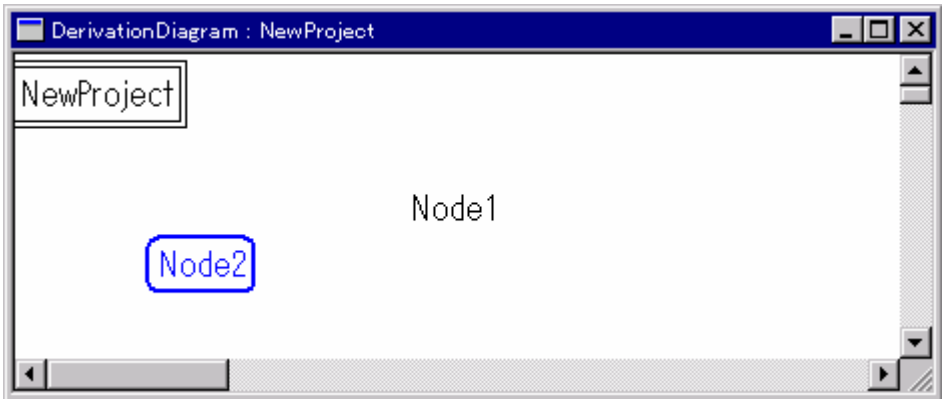


図 1-8 テンプレートの生成

3. 派生関係の作成

① 派生関係生成ボタンを押す。

② 派生図上の二つのクラス／テンプレート（ノード）を続けてクリックする。すると、一回目に選択されたクラス／テンプレート（ノード）から二回目に選択されたクラス／テンプレート（ノード）に向かって派生関係（アーク）が引かれる。

注）( )内はグラフ理論的な呼び方。

注）同じノードを続けてクリックした場合や、ノードをクリックできなかった場合は、それまでのクリックは無効となる。

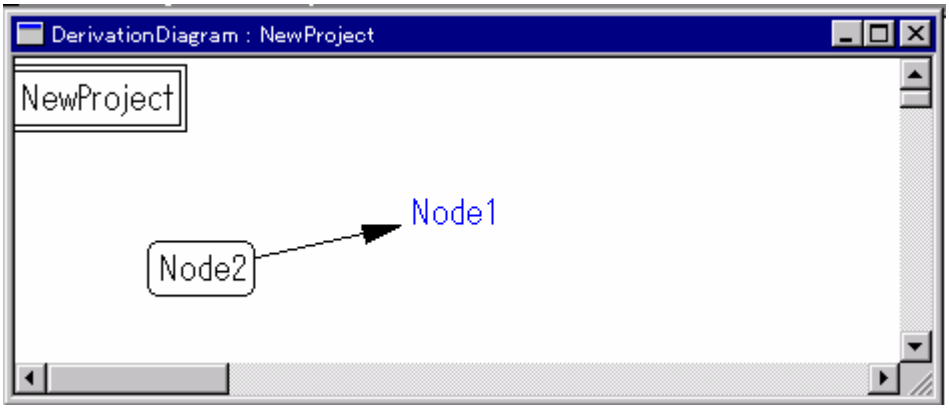


図 1-9 派生関係の作成

1.1.2.2. 編集対象の削除

① 編集対象選択ボタンを押した状態で、派生図上のある編集対象をクリックすると、色が青に変わり、現在その編集対象が選択状態にある事を示す。

注) この状態でノードをドラッグすると、ノードを任意の位置に移動することが出来る。

② 編集対象が選択された状態で、メニューバーの［編集］から［削除］を選ぶと、現在選択されている編集対象が削除される。

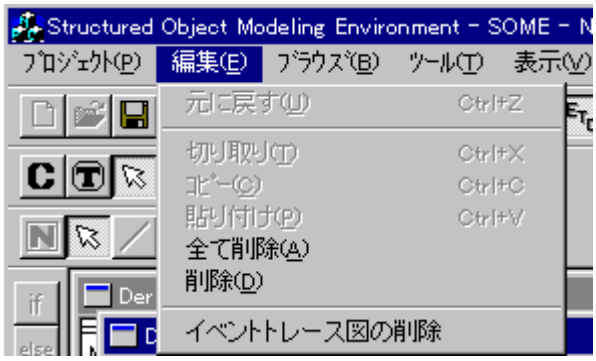


図 1-10 編集メニュー

注）クラス／テンプレート（ノード）を削除する場合、クラス／テンプレート（ノード）に繋がっている派生関係（アーク）も同時に削除される。

注）［全て削除］を選ぶと、現在の選択に関係なく、派生図上の全てが削除される。

1.1.2.3. 選択対象のプロパティの設定

ノード、アーク上で右クリックすると、プロパティダイアログが開かれる。

1. クラス／テンプレートのプロパティの設定

そのノードのクラス名、ノードタイプ、クラスタイプを決める。ここで、クラス名の変更が可能である。クラス名は生成時にはデフォルトの名前が付けられるので、ここで、クラス名を変更する。ノードタイプとは、そのノードが通常のクラスノードであれば”custom”、クラステンプレートを表すテンプレートノードであれば”template”を選択する。クラスタイプとは、そのノードが他のクラステンプレートのパラメタであれば”parameter”、ロジックテーブルで入力ファイルとして扱いたい場合は“inputfile”、出力ファイルとして扱いたい場合は”outputfile”、そうでなければ”normal”を選択する。クラスノードの場合、デフォルトは”custom”、”normal”で、テンプレートノードの場合、”template”、”normal”である。ファイルに関しては、さらに AutoFileOpen にチェックを入れておく。

図 1-11 Class Property

	normal	parameter	inputfile	Outputfile
custom	○	○	○	○
template	○	○	×	×

表 1-1 ノードタイプによるクラスタイプの制約

2. 派生関係の設定

その派生のアクセス権、継承種別を決める。継承、テンプレート化継承、テンプレート継承であれば”Inheritance”、カスタム継承であれば”Custom Inheritance”、カスタム化であれば”Customize”、展開であれば”Expand”を選択する。デフォルトは”Public”、”Inheritance”である。

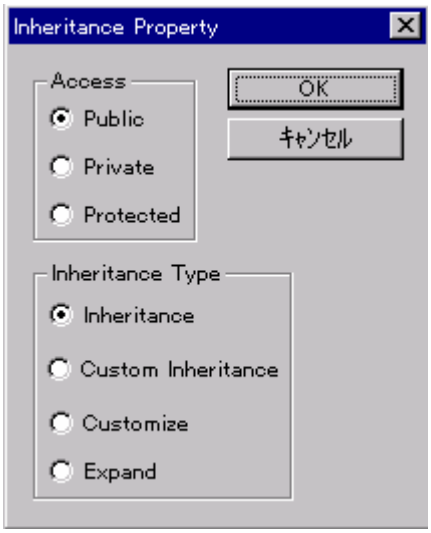


図 1-12 Inheritance Property

派生の種類	表記法	C＋＋での意味
継承	A ⟶ B	class B : A {.....};
テンプレート化継承	A ⟶ B	template < class T > class B : A {.....};
テンプレート継承	A ⟶ B	template < class T> class B : A<T> {.....};
カスタム化	A ⟹ B	T→CCの時 #define B A<CC>
カスタム継承	A ⟹ B	T→CCの時 class B : A<CC> {.....};
展開	A, A ⇢ B, B	コピー後に編集

表 1-2 派生関係の種類

1.1.3. オブジェクト図

［オブジェクト図の目的］

- 派生図に登録された全てのクラスの構造を表現する。すなわち、クラスの構成要素（利用者定義クラス型のオブジェクトと基本型の属性とメソッド）とオブジェクト間の関連を表現

［オブジェクト図で出来ること］

- 編集対象の作成
- 編集対象の削除
- 編集対象のプロパティの設定
- 属性プロパティによる登録
- ズームイン、ズームアウト操作
- 整合性チェック

注）編集対象

- オブジェクトノード
- グラフインタフェース
- ノードインタフェース
- 関連

［オブジェクト図のツールバー］

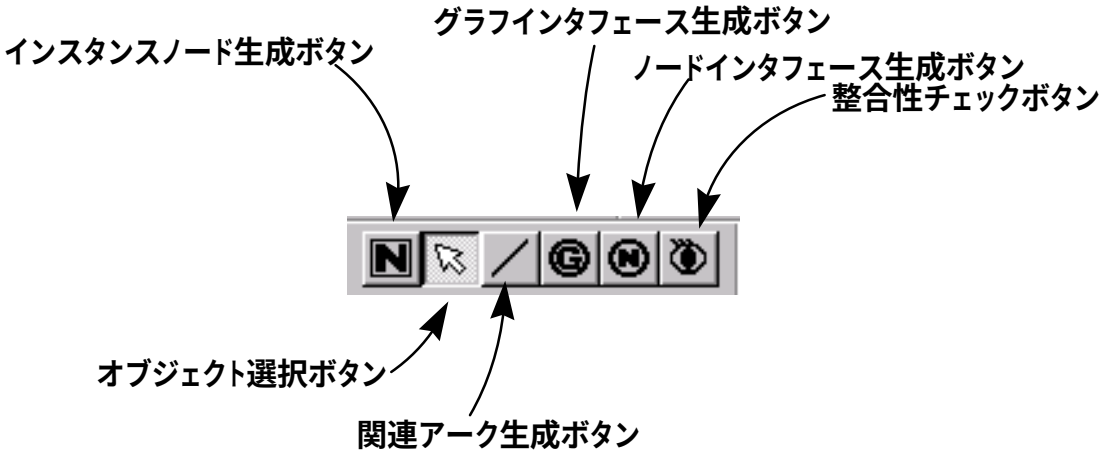
左から

- オブジェクトノード生成ボタン
- 編集対象選択ボタン
- 関連生成ボタン
- グラフインタフェース生成ボタン
- ノードインタフェース生成ボタン
- 整合性チェックボタン

ツールバーは、通常は編集対象選択ボタンになっている。

オブジェクトノード作成ボタン、グラフインタフェース作成ボタン、ノードインタフェース作成ボタン、関連作成ボタンを押して、操作をした後は、自動的に編集対象選択ボタンが押された状態に戻る。しかし、シフトキーを押しながら上記の操作を行うと、続けて操作を行うことが出来る。

その場合、シフトキーを放すと自動的に編集対象選択ボタンに変わる。



1.1.3.1. 選択対象の作成

1. オブジェクトノードの作成

① オブジェクトノード作成ボタンを押す。

② オブジェクト図上をクリックすると、そのオブジェクトの型を表すクラス名を選ぶダイアログが開かれる。

注）ダイアログ中のリストは、派生図に登録されているクラス名のリストとなっているので、そこからそのオブジェクトを実装させるクラス名を選択する。

③ ［ＯＫ］ボタンを押すと、最初にオブジェクト図上をクリックした位置を中央として、オブジェクトノードが作成される。

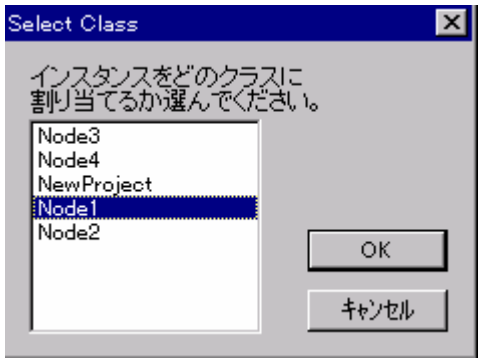


図 1-13 オブジェクトの実装時のクラスの選択

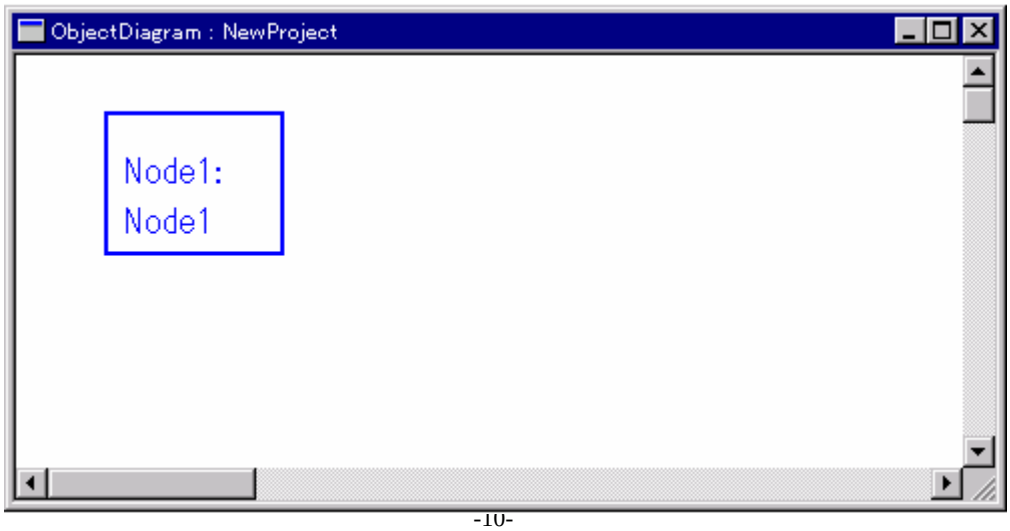


図 1-14 オブジェクトノードの生成

2. グラフインタフェースの作成

① グラフインタフェース作成ボタンを押す。

② オブジェクト図上でクリックすると、そこの位置を中央として、グラフインタフェースが作成される。

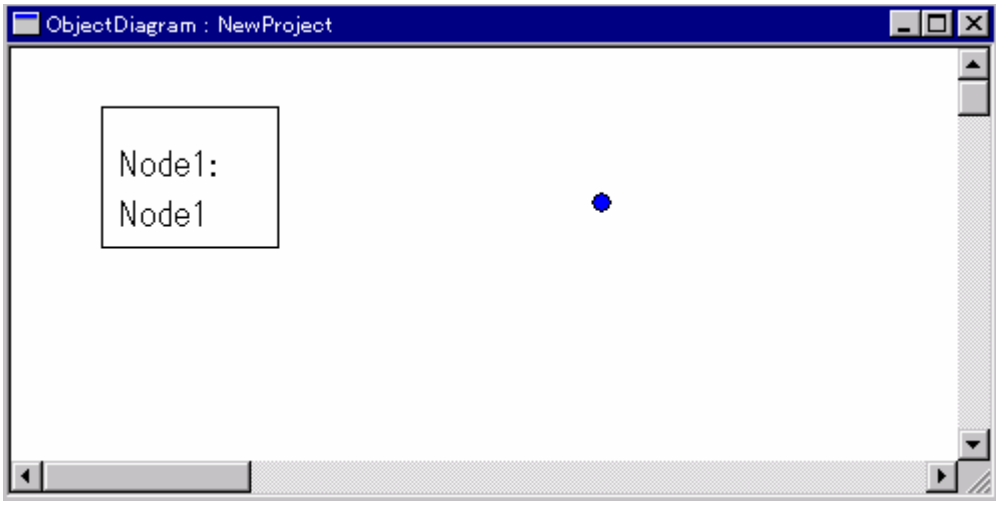


図 1-15 グラフインタフェースの生成

3. ノードインタフェースの作成

① ノードインタフェース作成ボタンを押す。

② オブジェクト図上のノードをクリックすると、そのノードの周りにノードインタフェースが作成される。

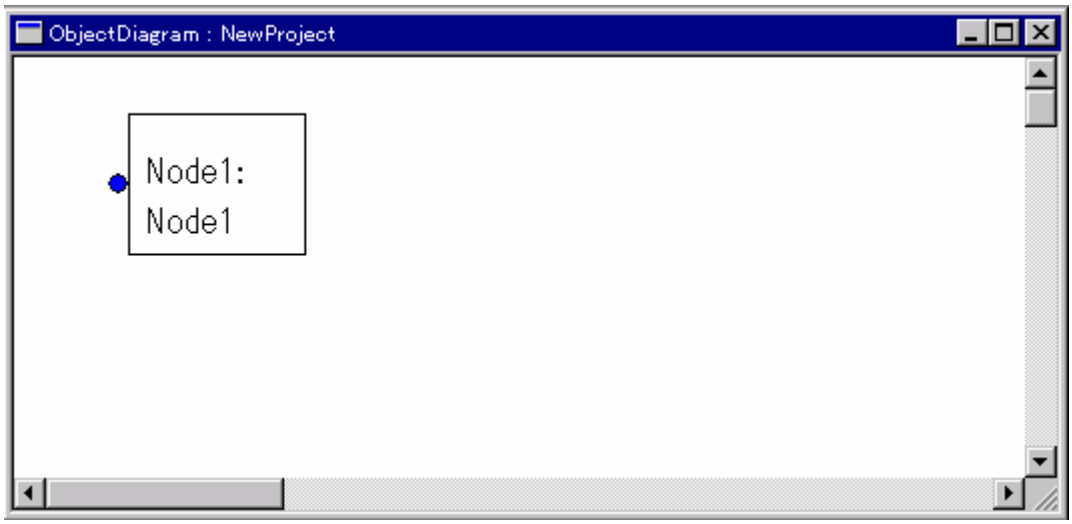


図 1-16 ノードインタフェースの生成

注）ノードインタフェースを移動させずに４つまで作ることができる。

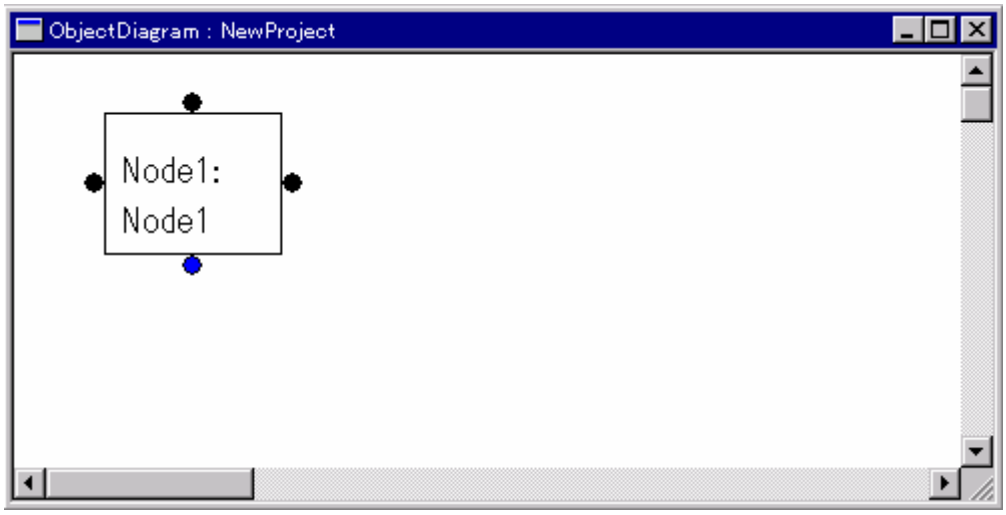


図 1-17 ノードインタフェースの生成（４つ）

注）ただし、任意の場所に移動すれば４つ以上作ることができる。

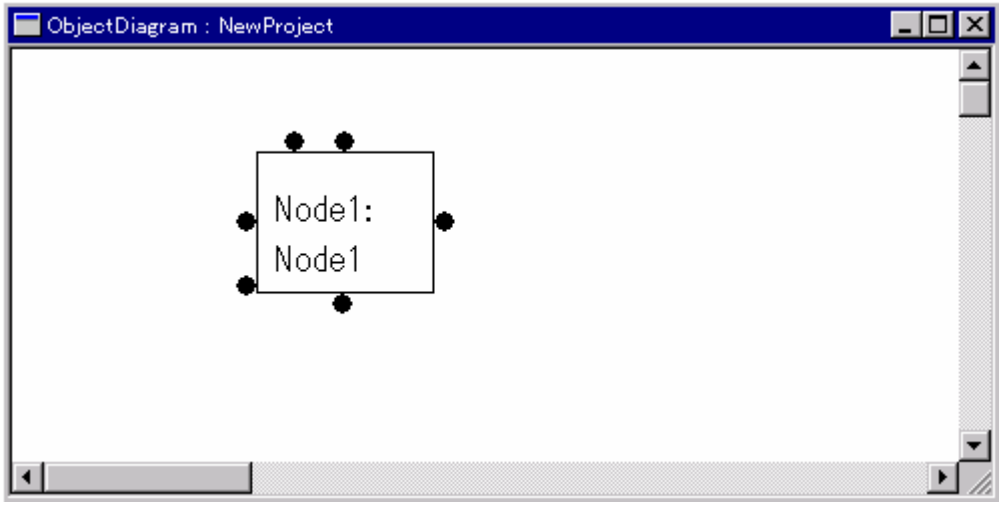


図 1-18 ノードインタフェースの生成（４つ以上）

- オブジェクトのプロパティ

ノード、インタフェース、アーク、グラフ上で右クリックすると、プロパティダイアログが開かれる。

［１］Instance Property

そのノードのインスタンス名、アクセス権、多重度を決める。インスタンスノードはそのオブジェクト図が表すクラスの集約要素なので、そのクラスに対してのインスタンス名をつける。アクセス権もそのクラス中のアクセス権である。多重度に関してはやや注意を要する。「１」はただ一つしか存在しないことを意味し、「＊」はリストとして存在することを意味する。任意に指定する場合は配列になるが、この時、[10]や[10][20]のように括弧まで書くようにする。

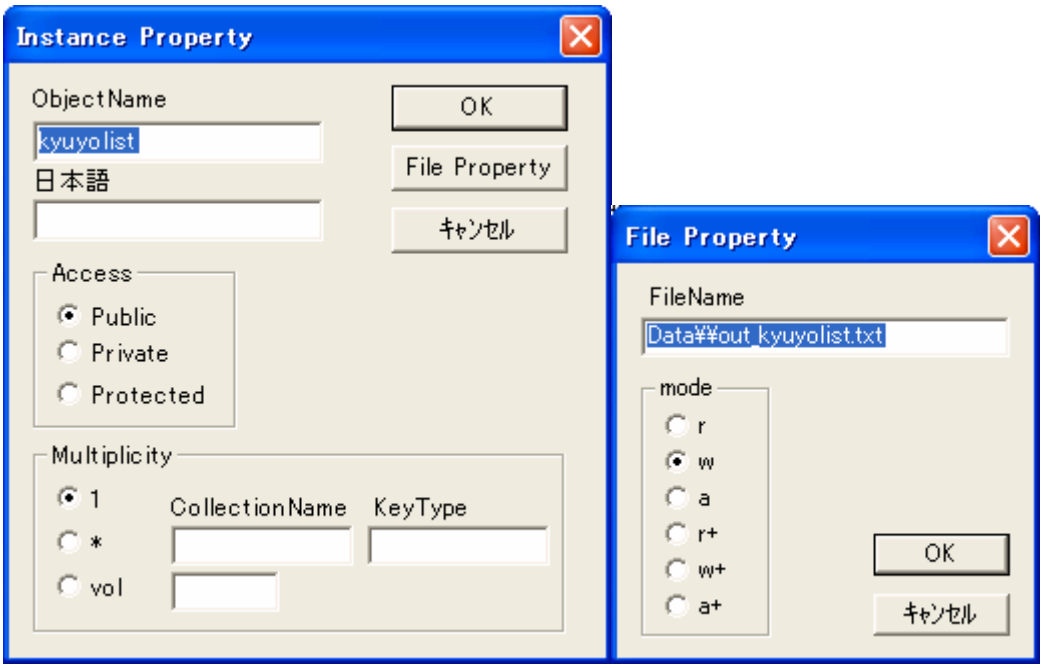


図 1-2　Instance Property

このように、ファイルに関してはオブジェクト図内において、ファイルを表すクラスを右クリックして、名前を入れるとともに、FileProperty をクリックして、そこで具体的なデータを格納するファイルのパス名をコード生成先のホルダを起点にして指定する。さらにファイルのオープンモード（ｒ：読み込み、ｗ：書き込み、など）を指定する。

1.1.3.2. 関連の作成

① 関連作成ボタンを押す。

② オブジェクト図上の二つのノード、インタフェースを続けてクリックすると、その二つのノードの間に関連アークが作成される。
注）同じノードを続けてクリックした場合や、ノードをクリックできなかった場合は、それまでのクリックは無効となる。

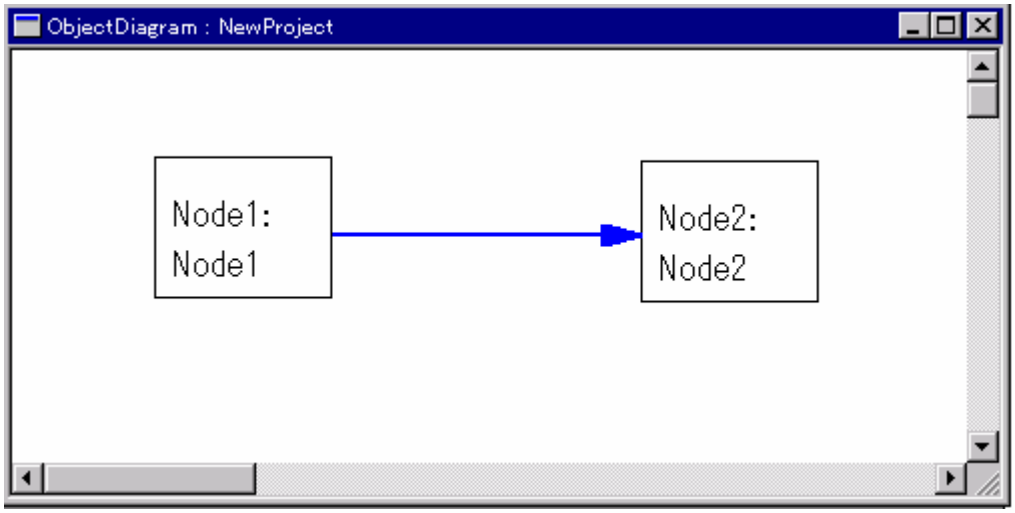


図 1-19 関連の生成

1. 編集対象の削除

① 編集対象選択ボタンを押した状態で、オブジェクト図上のある編集対象ををクリックすると、色が青に変わり、現在その編集対象が選択状態にある事を示す。
注）この状態でノードをドラッグすると、ノード、インタフェースを任意の位置に移動することが出来る。ただし、ノードインタフェースはその親ノードから離れることはない。

② 編集対象が選択された状態で、メニューバーの［編集］から［削除］を選ぶと、現在選択されている編集対象が削除される。

注）ノードやグラフインタフェースを削除する場合、そのノードやグラフインタフェースに繋がっているアークも同時に削除される。

注）［全て削除］を選ぶと、現在の選択に関係なく、オブジェクト図上の全てが削除される。

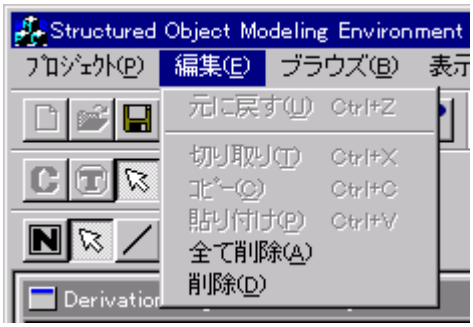


図 1-20 編集メニュー

2. 編集対象のプロパティの設定

ノード、インタフェース、アーク、グラフ上で右クリックすると、プロパティダイアログが開かれる。

(1) オブジェクトのプロパティの設定

ノードのオブジェクト名、アクセス権、多重度を決める。オブジェクトノードはそのノード自身を含むオブジェクト図が表すクラスの集約要素なので、そのクラスに対してのオブジェクト名をつける。アクセス権もそのクラス中のアクセス権である。多重度に関してはやや注意を要する。「１」はただ一つしか存在しないことを意味し、「＊」はリストとして存在することを意味する。任意に指定する場合は配列になるが、この時、[10]や[10][20]のように括アークまで書くようにする。

図 1-21 Instance Property

(2) インタフェースのプロパティの設定

インタフェースの型を指定する。インタフェースに入射アークがある場合、アークの始点ノードはアークがどのクラスに向かっているかを知っている必要がある。したがって、ここで行き先ノードの型を指定していないと、整合性チェックにひっかかる。これに対して、インタフェースが出射アークである場合は、アークの終点ノードはアークがどのクラスから来ているのか知る必要はない。

図 1-22 Interface Property

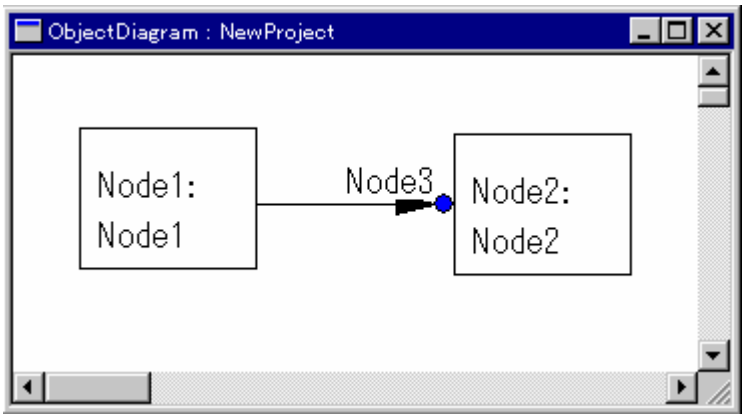


図 1-23 インスタンスプロパティ設定後の状態

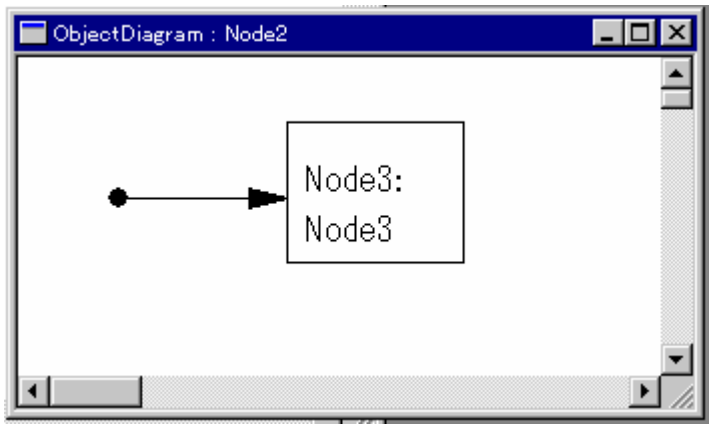


図 1-24 インスタンスプロパティ設定後の状態

1.1.3.3. 関連プロパティの設定

関連の終点ノードについて、関連名とロール名とアクセス権、多重度を入力できる。出発ノードからの関連名が表示されるが、ここで重要なのはロール名である。ロール名は、関連の始点クラスから見た終点クラスの呼び名で、実装時には始点クラスにおいて宣言される終点クラスへのポインタ変数となる。

図 1-25 Association Property

1.　属性プロパティによる登録

(1) 基本型の属性の登録

①　オブジェクト図上で右クリックすると属性プロパティが開く。

図 1-26 Attribute Property

②　“New　Attribute”をダブルクリックする。そうすると、入力ダイアログが開くので、そのクラスの（基本型を持つ）属性の型と属性名を入力する。

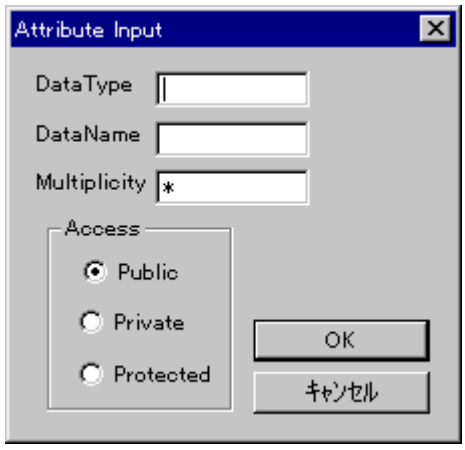
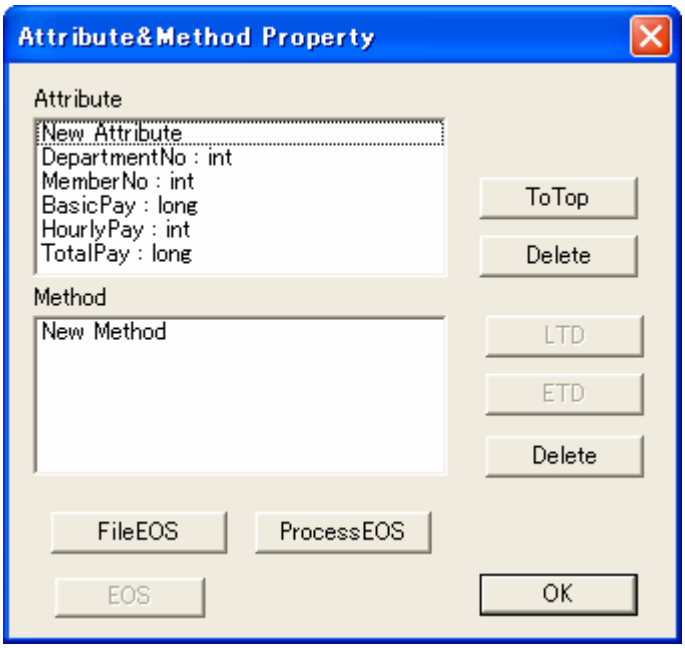


図 1-27 属性入力ダイアログ

③　“ＯＫ”ボタンを押すと、属性プロパティに追加される。

注）変更をするときは、変更する属性をダブルクリックして属性入力ダイアログで変更する。削除する場合は、削除する属性を選んで、“削除”ボタンを押す。



＊ファイルを表すクラスの属性定義においては上図の各属性の具体的なデータタイプを記入する。許されるのは,int、long、CString のみである。

(2) メソッドの登録

① オブジェクト図上で右クリックすると属性プロパティが開く。

図 1-28 Attribute Property

② “New Method”をダブルクリックする。そうすると、入力ダイアログが開くので、そのクラスのメソッドの返り値の型とメソッド名を入力する。そのメソッドがロジックテーブルでも使われるのであれば“EventTraceDiagram”をクリックする。

注）メソッド名はメソッド（）またはメソッド（仮引数型 仮引数名）で入力する。

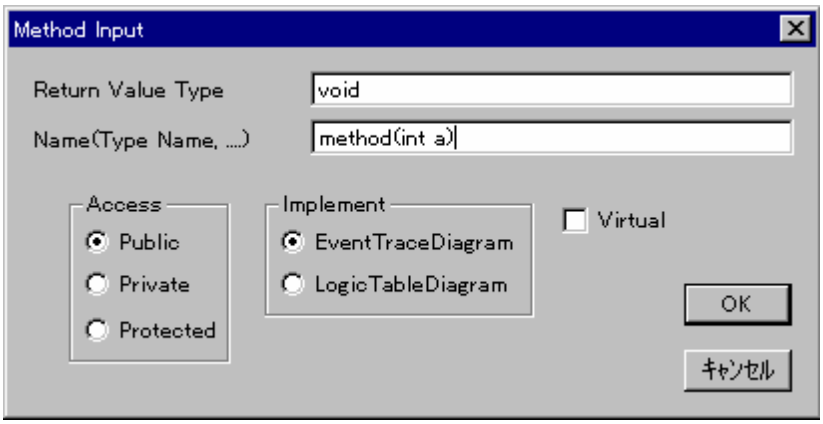


図 1-29 メソッド入力ダイアログ

③ “ＯＫ”ボタンを押すと、属性プロパティに追加される。

注）変更をするときは、変更するメソッドをダブルクリックしてメソッド入力ダイアログで変更する。削除する場合は、削除する属性を選んで、“削除”ボタンを押す。

1.1.4. イベントトレース図

［イベントトレース図の目的］

クラス自身とクラスの構成要素、および外部オブジェクト間でのイベントの送受信を表現。

［イベントトレース図の特徴］

- イベントトレース図は、各クラスのメソッド毎に高々１枚である。
- イベントトレース図の編集には、イベントトレースエディタを使用する。エディタの左側のフィールドがイベントトレース図の図フィールドに、右側がテキストフィールドに対応している。また、編集操作は、すべて行単位に行う。

［イベントトレース図で出来ること］

- 図フィールドの記述
- テキストフィールドの記述
- イベント行レベルでの切り取り、コピー、貼り付けなどの編集

1.1.4.1. イベントトレース図を開く

イベントトレースエディタは、イベントトレース図を編集することで起動される。

(①から②はメソッドを新規に作成する場合)

① オブジェクト図を開く。(方法は 2 ページ 10.1.1.2 参照)

② オブジェクト図上でメソッドを定義する。(方法は 17 ページ 10.1.3.3. の 1.－(2) 参照)

③ メニューの[ブラウズ]の[メソッド編集]を選ぶ。

④　イベントトレース図選択ダイアログが開く。

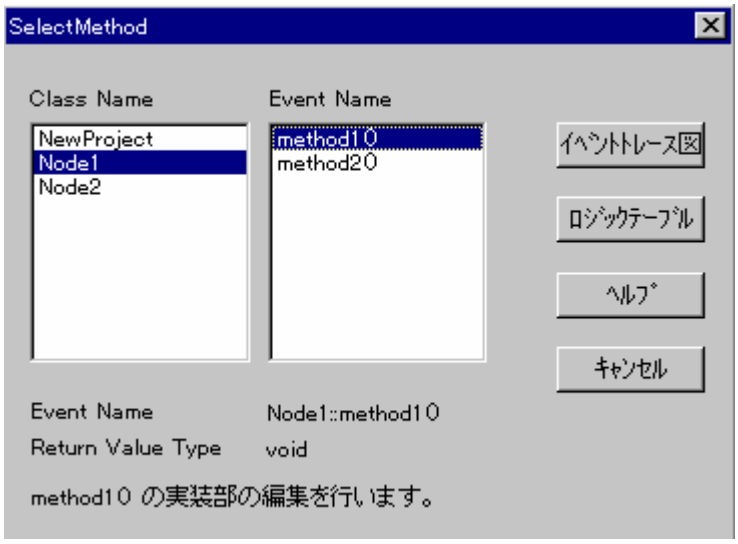


図 1-47 メソッド選択ダイアログ

⑤　編集するイベントトレース図のクラスを選択する（図 1-47 では Node1）

⑥　メソッド名リストからメソッド名を選択する。（図 1-47 では method1()）

⑦　[イベントトレース図]を押す。

⑧　選択したイベントトレース図を表示したウィンドウが開き、編集可能になる。

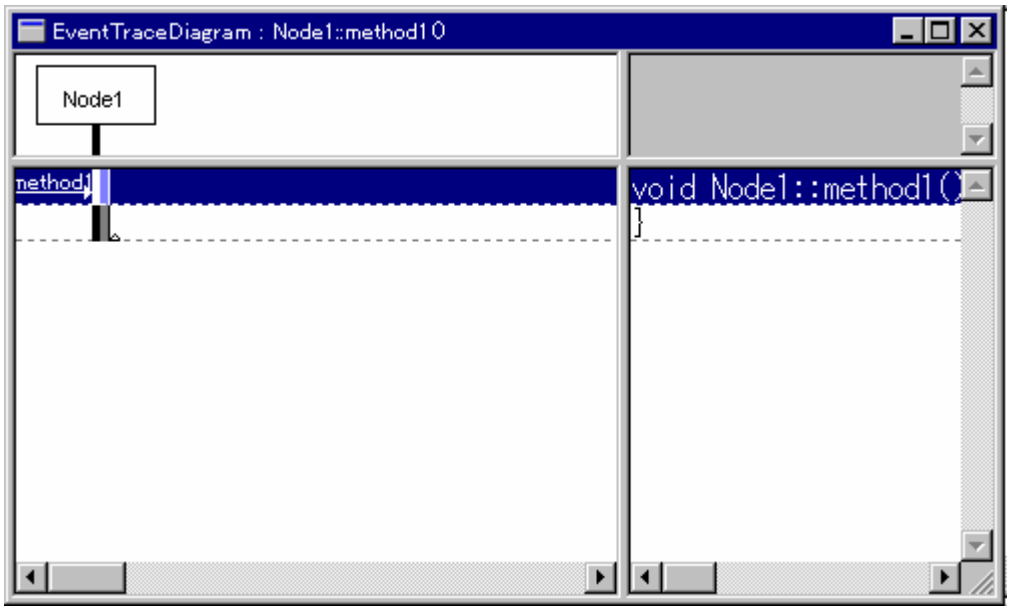


図 1-48 イベントトレースエディタ

注）メソッド名や返値の型を変更することはメソッド選択ダイアログではできない。変更したいときはオブジェクト図上で変更したいメソッドを選びそこをダブルクリックする。すると、メソッドの名前や返り値を入力できるダイアログが開かれるのでそこで変更する。

1.1.4.2. イベントトレース図の作図

1. 図フィールドを記述する

図フィールドを記述することは、図形シンボルの入力に相当する。図形シンボルの入力は、すべて新規行の挿入として扱われる。入力には、アイテムパレットを利用する。アイテムパレットとアイテムパレットの各ボタンが表す内容について、まとめたものが表 1-である。

ボタン	アイテムの意味	アイテムの説明
if else	if-else 文	C++における if-else 文を表します。
sw ca se de faut	switch-case-default 文	C++における switch 文を表します。
whil	while 文	C++における while 文を表します。
do	do-while 文	C++における do-while 文を表します。
for	for 文	C++における for 文を表します。
Decl	宣言文	変数の宣言を表します。
Com	代入文・計算式	主に代入文や四則演算などのオブジェクトの内部状態を変更する計算を表します。また、continue 文などの制御文、return 文なども、計算式として扱います。
Ev ent	イベント送信	イベントの送信を表します。内部イベントと、他のオブジェクトへのイベントの区別はありません。
	選択	行の選択モードに変わります。イベントトレース図を開いたときには、このボタンが押された状態になっています。

表 1-3 アイテムパレットとアイテムパレット対応表

図フィールド上では

- ブロックエンド △
- 宣言文　　□
- 代入文・計算式 ○
- イベント送信　→

で表されている。

① 図フィールド上の任意の位置をクリックする。

② アイテムパレットが利用可能な状態になる。

注）テキストフィールドにフォーカスがある場合には、アイテムパレットは利用できない状態となっている。

③ 入力する図形を表すアイテムパレット上のボタンをクリックして選択する。

④　アイテムパレットのボタンが押し込まれて、新規行入力状態になる。

⑤　図フィールドの挿入可能位置に行カーソル（黒の実線）が現れる。

注）新規行入力状態を解除するときには、マウスの右クリックもしくは Esc キーを押す。

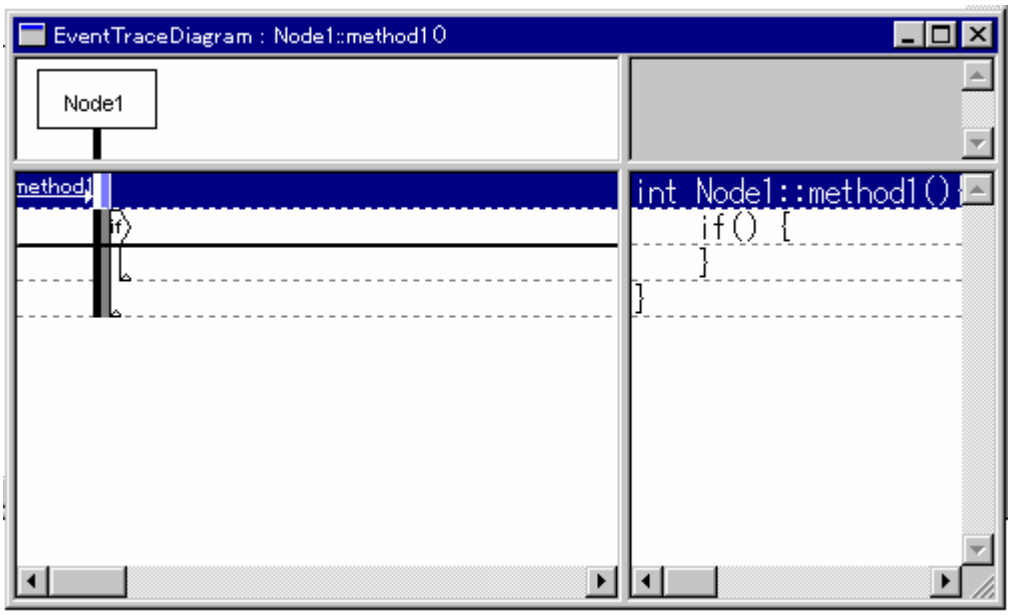


図 1-49 行カーソルの移動

⑥　マウスで行カーソルを動かしアイテムを挿入したい位置にあわせる。

⑦　行カーソルの位置が決定したらそこでクリックする。すると、選択した図形シンボルが描かれた新規行が挿入される。

注）このとき、入力された図形シンボルに対応したテキストがテキストフィールドに挿入される。

注）制御文の図形シンボルに関しては、制御文の終端を表す終端記号が挿入行の次行に自動的に挿入される。

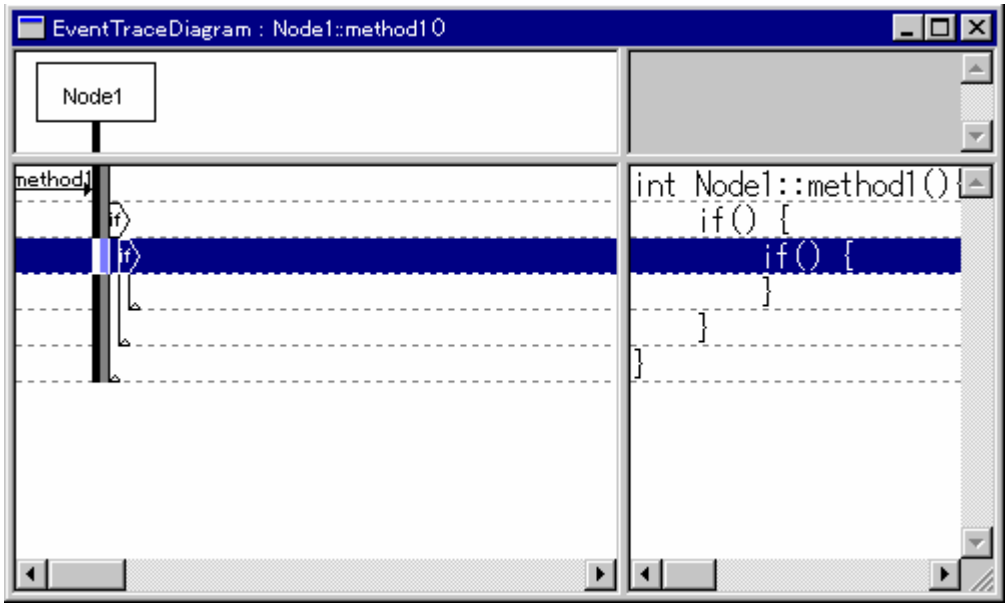


図 1-50 新規行の挿入

2. テキストフィールドを記述する

① 編集する行のテキストフィールドをクリックする。すると、テキスト編集が可能になる。

② テキスト編集を行う。

③ 改行、カーソルキーによる前後の行移動、もしくは編集行のテキストフィールド以外をクリックすれば、情報は更新される。

注）編集操作は、基本的なテキスト編集操作と同様である。また、上下カーソルキーによって、前後の行に編集カーソルを移動可能である。

3. イベント行の編集を行う

行の編集には、基本的な編集機能(切り取り・コピー・貼り付け)を利用することができる。切り取り・コピーした内容は、同一イベントトレース図内だけでなく、他のイベントトレース図間でも利用可能である。

① 行を選択する。

② 編集操作(切り取り・コピー・貼り付け)を行う。方法はメニューバーから、ツールバーから、ショートカットキーの３つの方法がある。

- メニューバーは「編集」から操作を選ぶ。

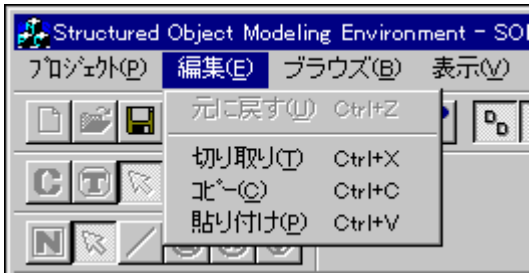


図 1-51 編集メニュー

- ツールバー 
  左から 切り取り、コピー、貼り付け である。
- ショートカットキー(図 1-51 参照)

注）ペーストに関しては、選択行の前にペーストされる。

注）行を編集するには、編集行を選択する必要がある。行の選択には、単一選択とブロック選択の２つがある。

- 単一選択

一行のみの選択である。図フィールド中で、編集行をクリックすることで選択をすることができる。

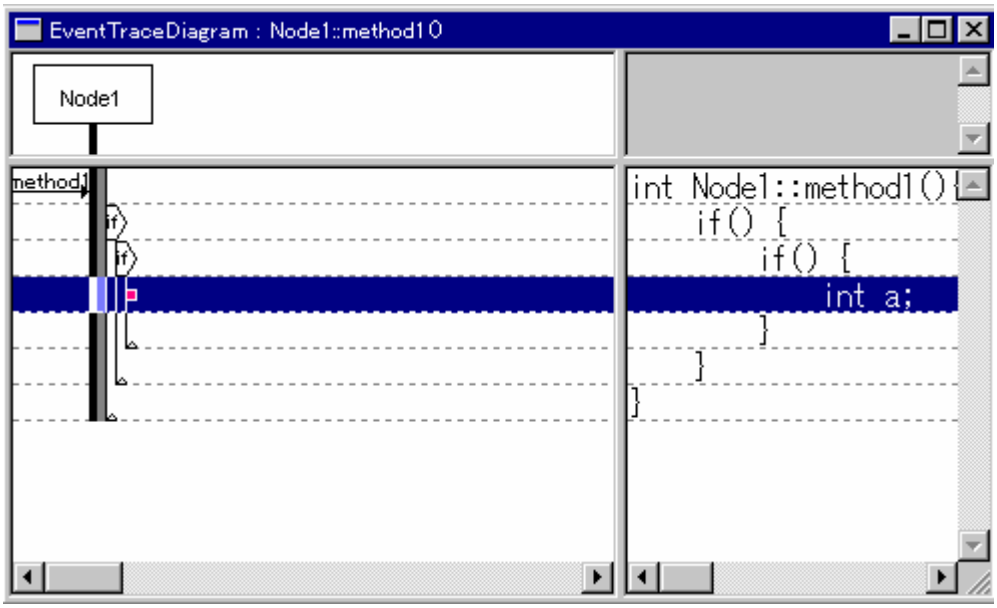


図 1-52 単一選択

- ブロック選択

複数行の選択で指定行を含む制御ブロック全体を選択する。図フィールド中で、指定行をダブルクリックすることで選択することができる。

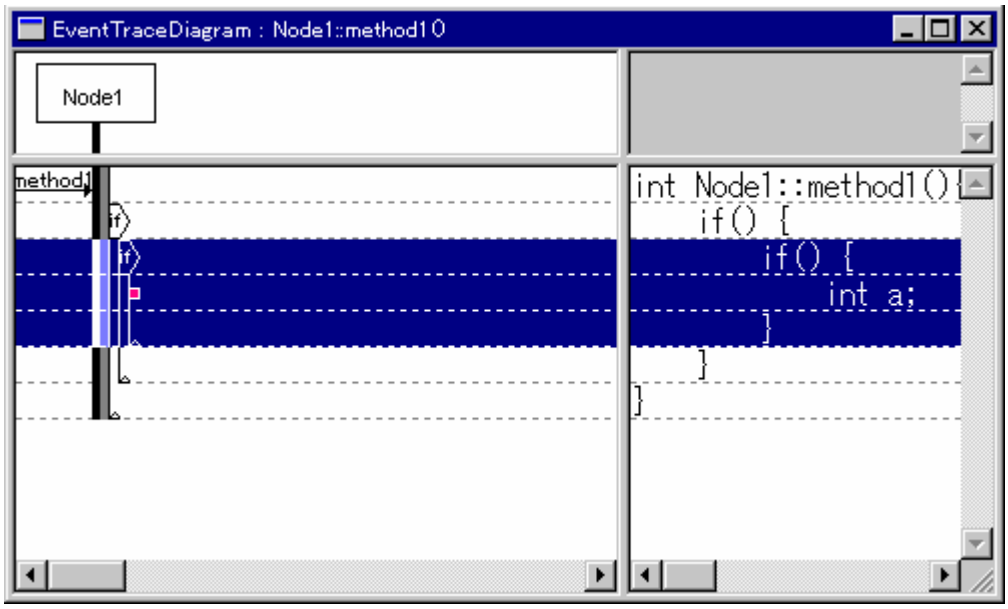
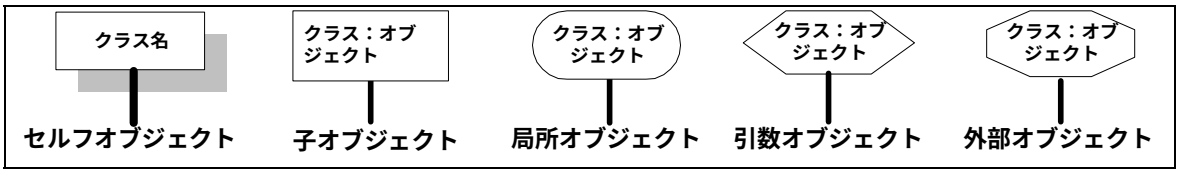


図 1-53 ブロック選択

1.1.4.3. イベントトレース図の表記法

1. イベントトレース図のノード



① セルフオブジェクト

イベントトレース図の当該メソッドが存在するオブジェクトのインスタンスを表す。

② 子オブジェクト

セルフオブジェクトの集約要素を表すオブジェクト。基本型の属性は表示されない。

③ 局所オブジェクト

当該メソッド内でローカルに宣言されるオブジェクトを表す。ライフタイムはメソッド内のみである。

④ 引数オブジェクト

当該メソッドが引数として受け取るオブジェクト。

⑤ 外部オブジェクト

セルフオブジェクトと関連を持っているオブジェクト、または複数の関連を経由して間接的に関連している間接参照オブジェクト

2.　ノードの表示

- 表示のタイミング

セルフオブジェクト、引数オブジェクト、子オブジェクトはイベントトレース図が作成された時点で、はじめからイベントトレース図中にノードとして表示する。

局所オブジェクトはその性質上、宣言された時点で存在確認後、表示する。

外部オブジェクトは最初から表示しない。これらは一般に、テキスト行で関連先のオブジェクトへのイベントが解析されたら自動表示される。ただし、直接関連でセルフオブジェクトとつながっているオブジェクトは必要ならばコンボボックスから選択して表示することができる。

最初から表示されるもの：	必要に応じて表示するもの：
①セルフオブジェクト	③局所オブジェクト
②子オブジェクト	⑤外部オブジェクト
④引数オブジェクト	

- 表示の画面上での順序

整列順序方針１：セルフオブジェクトは図の最左列に表示する。

整列順序方針２：ノードの種類ごとにまとめて表示する。

整列順序方針３：左から子、引数、局所、外部オブジェクトの順とする。

固定数のオブジェクトを左側に表示してゆく。

3.　イベントの記述方法

イベントの挿入には２通りある。

- 自動入力　イベント行を作成した後、テキストフィールドにC++の文を記述しこれをシステムが解析して、発見したイベントを自動表示する。
- マウス入力　Event ボタンを押して挿入すべき行をマウスで選び、行き先オブジェクト選ぶ。

(1) 自動入力の例

ClassA から ClassB のインスタンス b にイベントを送信する。

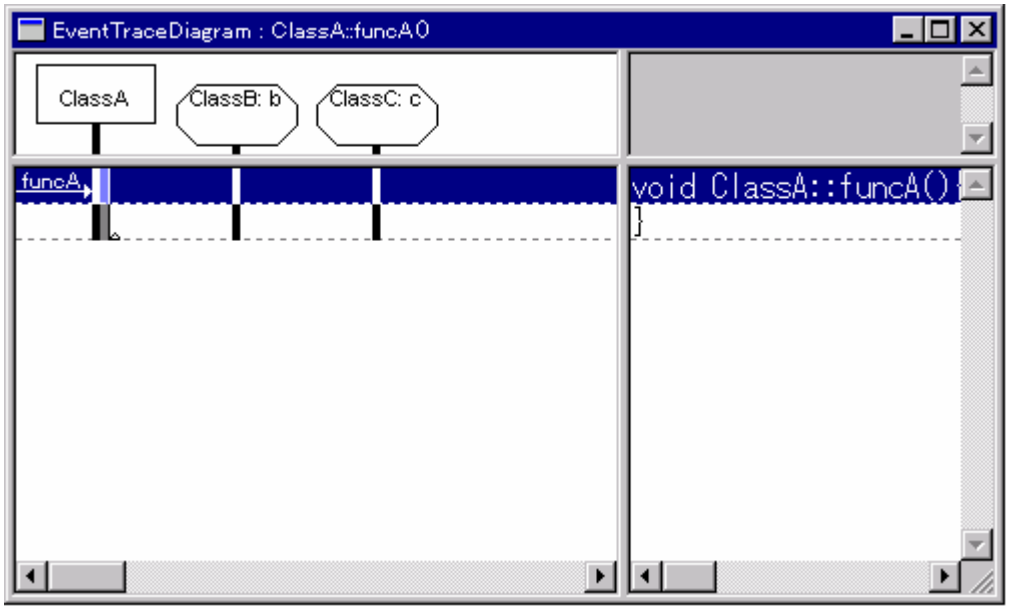


図 1-54 イベントトレースエディタ

① 図フィールド上の任意の位置をクリックする。

② アイテムパレットが利用可能な状態になる。

③ アイテムパレット上の「Com」ボタンをクリックして選択する。

④ アイテムパレットのボタンが押し込まれ新規行入力状態となる。

⑤ 図フィールドの挿入可能位置に行カーソル（黒の実線）が現れる。

⑥ マウスで行カーソルを動かしアイテムを挿入したい位置にあわせる。

⑦　行カーソルの位置が決定したらそこでクリックする。すると、選択した図形シンボルが描かれた新規行が挿入される。

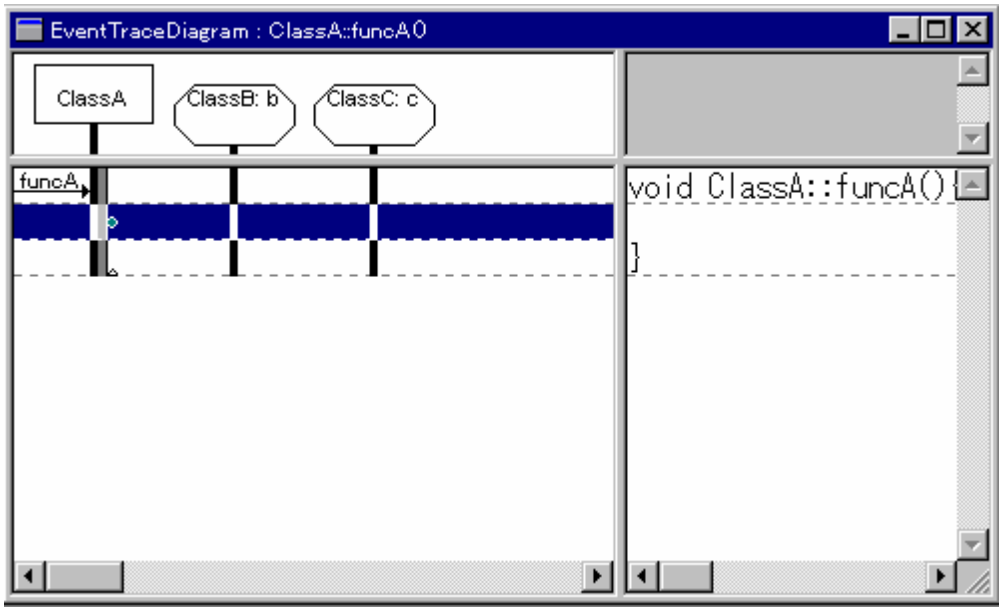


図 1-55 新規行の挿入

⑧　テキストフィールドに送りたいイベントを書き、Return キーを押せば図フィールドにもイベント名が記述される。

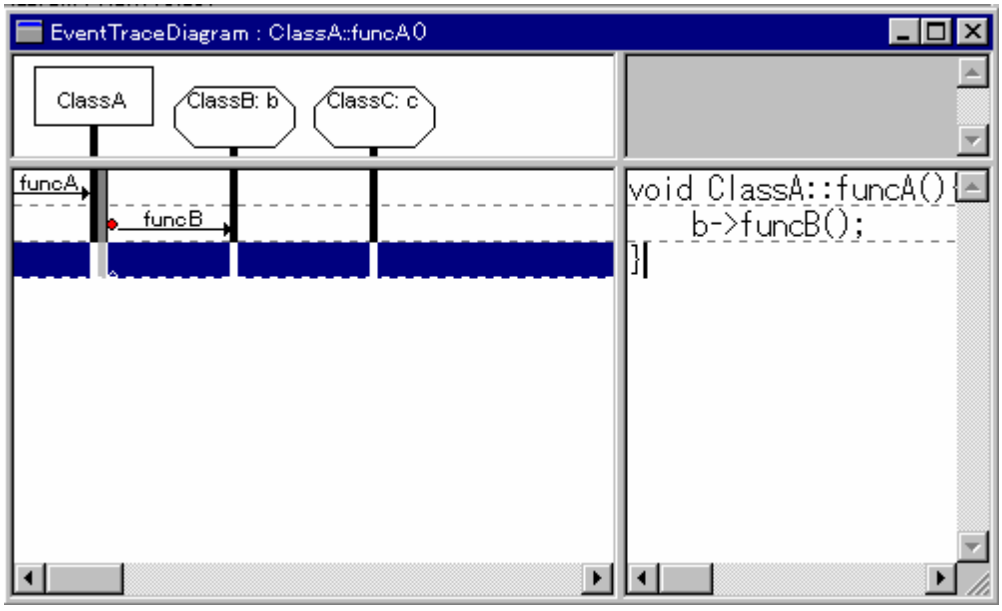


図 1-56 イベントの描画

(2) マウス入力

ClassA から ClassB のインスタンス b にイベントを送信する。

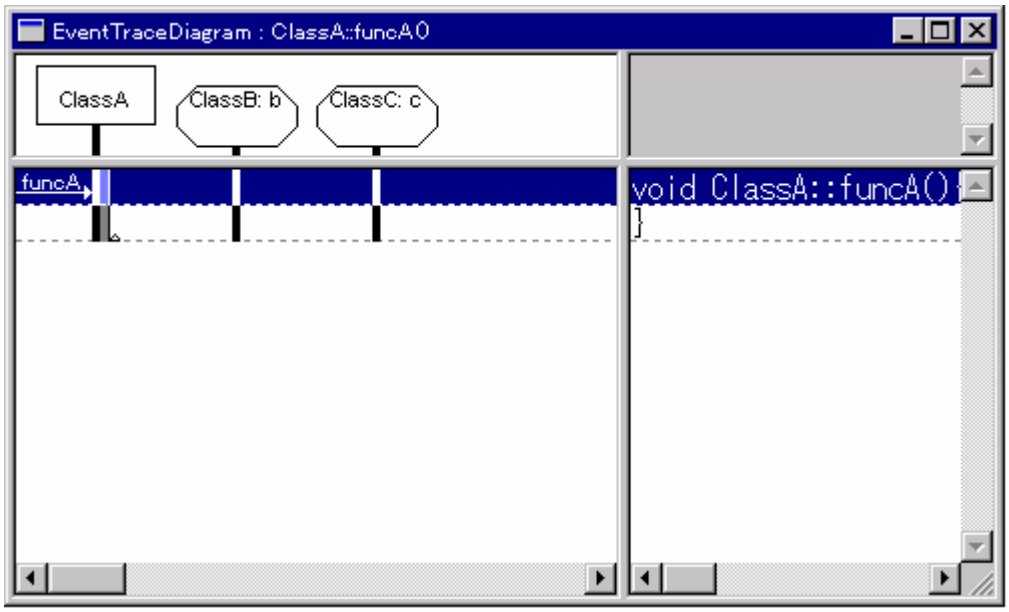


図 1-57 イベントトレースエディタ

① 図フィールド上の任意の位置をクリックする。

② アイテムパレットが利用可能な状態になる。

③ アイテムパレット上の「Event」ボタンをクリックして選択する。

④ アイテムパレットのボタンが押し込まれ新規行入力状態となる。

⑤ 図フィールドの挿入可能位置に行カーソル（黒の実線）が現れる。

⑥ マウスで行カーソルを動かしアイテムを挿入したい位置にあわせる。

⑦ 行カーソルの位置が決定したらそこでクリックする。すると、選択した図形シンボルが描かれた新規行が挿入される。

注）テキストフィールドの「b->」は自動的に記述される。

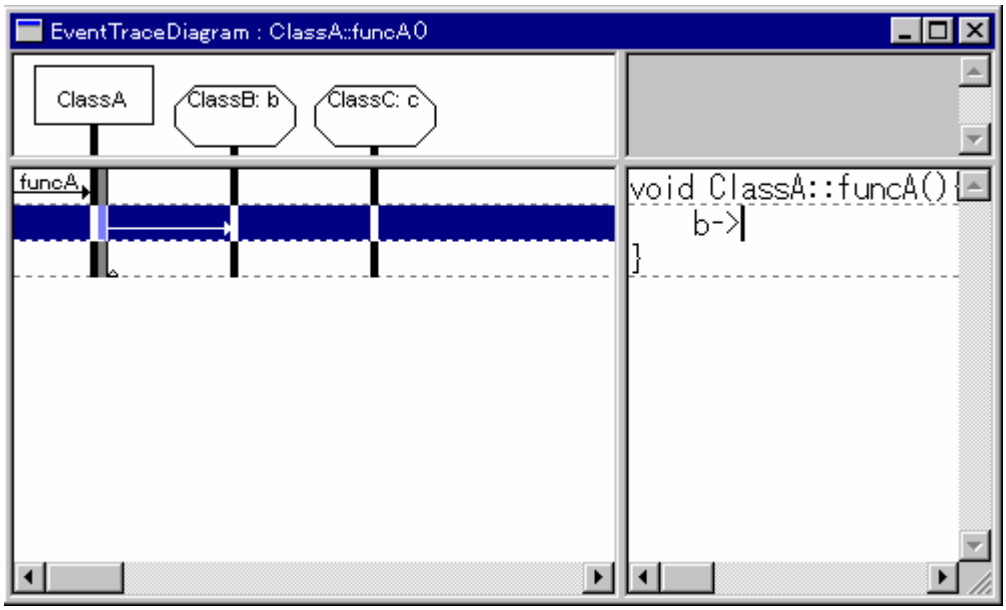


図 1-58 新規行の挿入

⑧ 続きを記述して Return キーを押せば図フィールドにもイベント名が記述される。

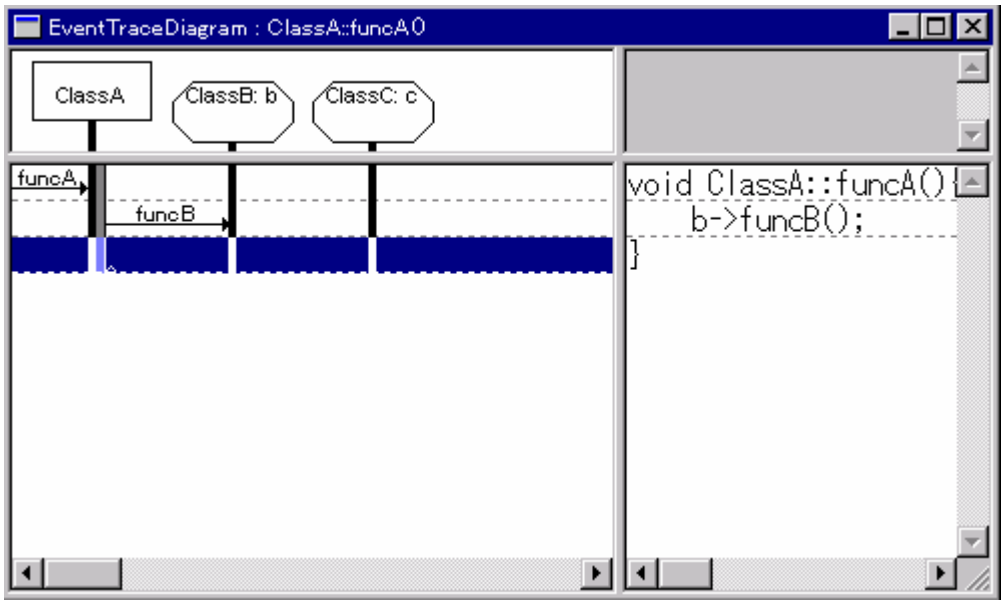


図 1-59 イベントの描画

注）もし、送信先にないイベント（ここでは funcD()）を記述してしまった場合は図フィールド上にイベント名が表示されないし、イベントトレースエディタでは自動的にイベントが作られない。

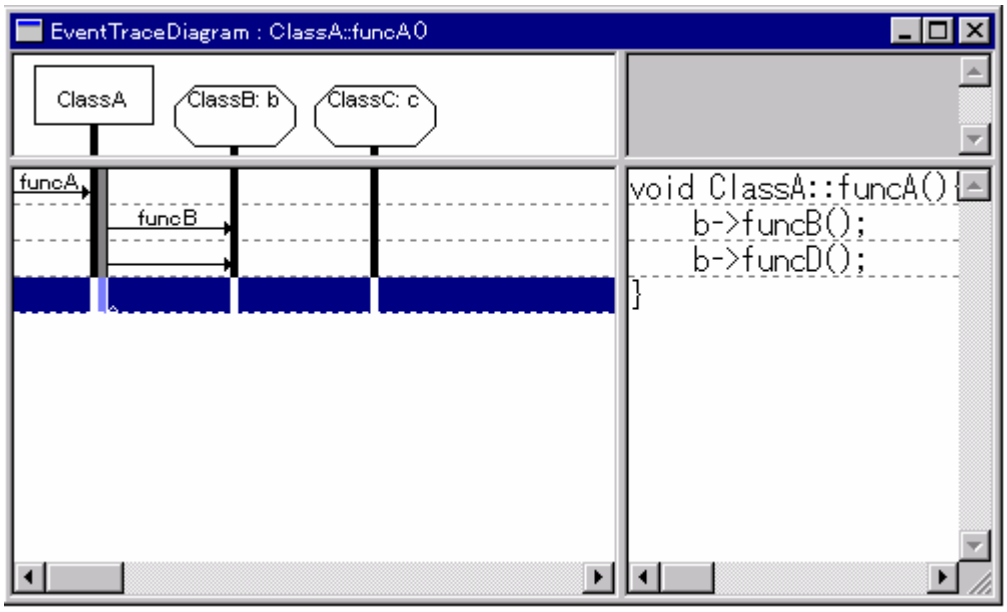


図 1-60 イベントが作られない例

4. コレクションクラス追加に伴うイベントトレース図の表現方法

子オブジェクト、外部オブジェクト（関連）で多重度＊で定義されたものは、オブジェクト図同様、型名（クラス名）、変数名、コレクション名の３行表示になる。

その際、コレクションの要素にメソッドを送る場合、その要素を表すために局所オブジェクトが必要になるが、その変数名は３行表示されたうちの第２項の名前とする。これは仕様として利用者に義務づけるか、最初からデフォルトでテキスト行に自動表示させるかは未定。義務づけた場合、自動的に局所オブジェクトを作るが、その際に検索をかけ、変数名と同じ場合は局所変数を作らないことにする。

従って、コレクションクラスとその要素が同じノードで表現されるため、コレクションクラスに対するメソッドか、要素に対するメソッドかは、矢印の先の形をかえて表現する。また、テキスト行解析部も手を入れる必要がある。

1.1.5. ロジックテーブル

［ロジックテーブルの目的］

メソッドの内容をデシジョンテーブル（決定表）を用いて表現する。

［ロジックテーブルの特徴］

- ロジックテーブルは、処理内容を簡単に記述するためのテーブルであり、イベントトレース図同様、各クラスのメソッド毎に高々１枚である。
- ロジックテーブルの編集には、ロジックテーブルエディタを使用する。編集操作は、行・列単位に行う。

● ロジックテーブルエディタは、初期処理部、前処理部、条件部、処理部の４つのエリアに分かれており、条件部及び処理部の右側に、対応する規則を入力する。

● ファイル照合処理・コレクション処理といった繰り返し・分岐処理を簡潔に表現できる。

［ロジックテーブルエディタで出来ること］

- テキストエリアの記述
- ルールエリアの記述
- 行、列単位での切り取り、コピー、貼り付けなどの編集
- エラーチェック

1.1.5.1. ロジックテーブルを開く

ロジックテーブルエディタは、ロジックテーブルを編集することで起動される。

(①から②はメソッドを新規に作成する場合)

① オブジェクト図を開く。(方法は 2 ページ 10.1.1.2 参照)

② オブジェクト図上でメソッドを定義する。(方法は 17 ページ 10.1.3.3 の 1.-（2）参照)

③ メニューの[ブラウズ][メソッド編集]を選ぶ。

④ メソッド選択ダイアログが開く。

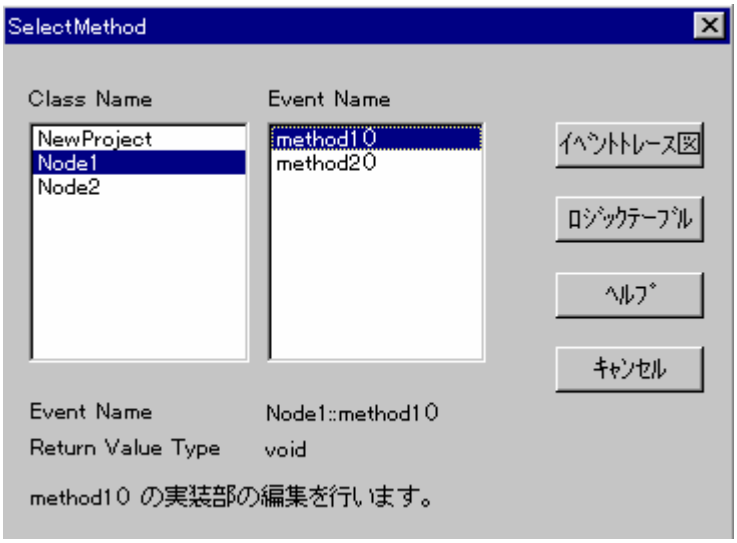


図 1-61 メソッド選択ダイアログ

⑤ 編集するロジックテーブルのクラスを選択する（図 1-61 では Node1）

⑥ メソッド名リストからメソッド名を選択する。(図 1-61 では method1())

⑦ ［ロジックテーブル］を押す。

⑧　選択したロジックテーブルを表示したウィンドウが開き、編集可能になる。

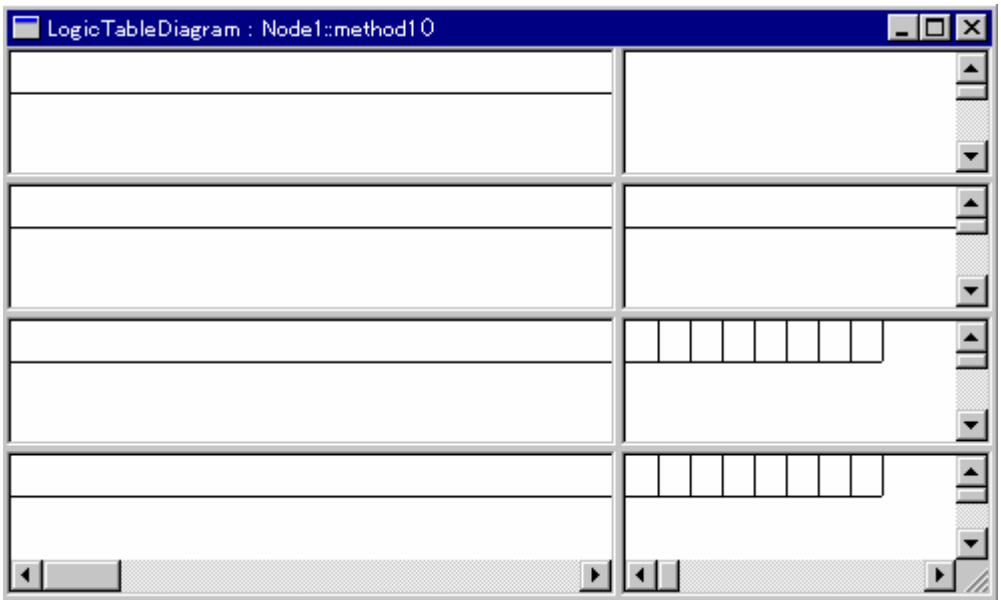


図 1-62 ロジックテーブルエディタ

注）メソッド名や返値の型を変更することはメソッド選択ダイアログではできない。変更したいときはオブジェクト図上で変更したいメソッドを選びそこをダブルクリックする。すると、メソッドの名前や返り値を入力できるダイアログが開かれるのでそこで変更する。

1.1.5.2. ロジックテーブルの作成

1.　ロジックテーブルの編集

①　編集する行のテキストエリアをクリックする。すると、テキスト編集が可能になる。

②　テキスト編集を行う。なおテキスト入力は、パレット入力または手作業で入力する。

③　改行で論理行（デシジョンテーブルで処理を指定する際の１単位。複数の命令を１論理行と定義可能）1 行が増え、Shift＋Enter キーで物理行（１論理行内にある画面表示する際の１単位）が 1 行増える。

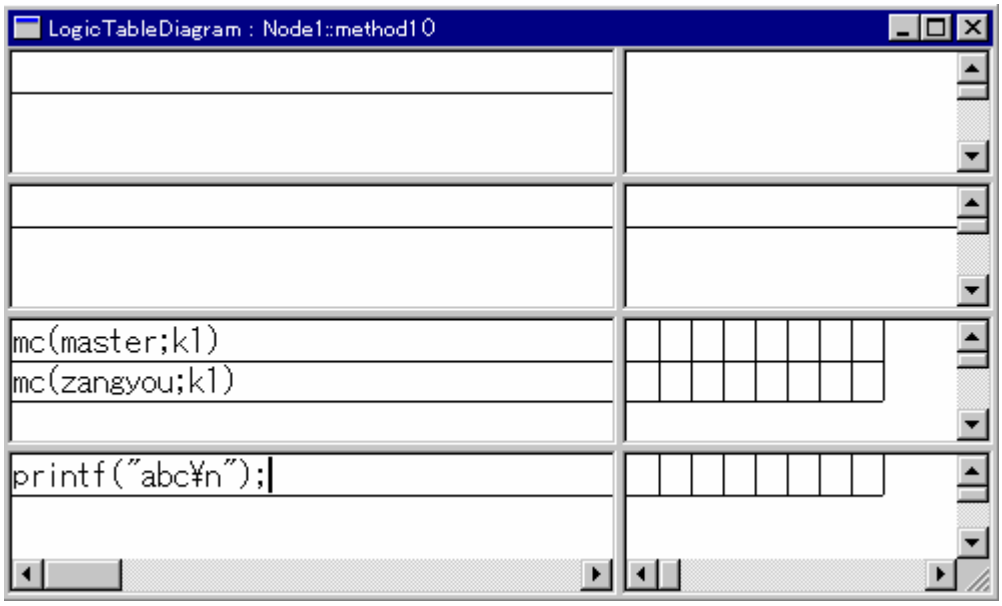


図 1-63 Shift＋Enter キーを押す前

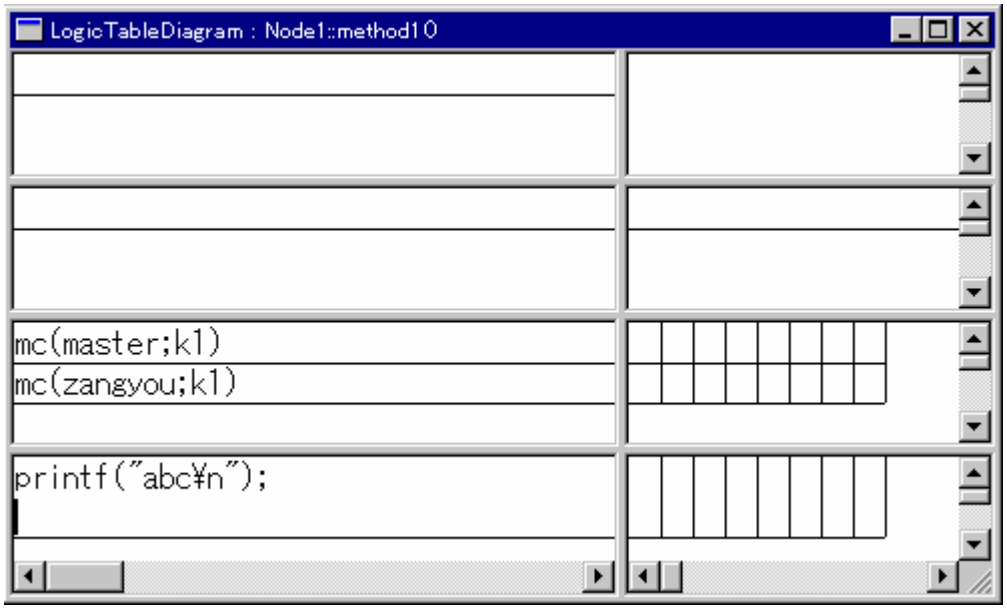


図 1-64 Shift＋Enter キーを押した後

④　改行、カーソルキーによる前後の行移動、もしくは編集行のテキストエリア以外をクリックすれば、情報は更新される。編集操作は、基本的なテキスト編集操作と同様である。また、上下カーソルキーによって、前後の行に編集カーソルを移動可能である。

2.　編集コマンド

テキストエリアでマウスを右クリックすると、編集コマンドがポップアップメニューで表示される。

＜条件部、処理部の編集＞

文字列を選択表示させ、右クリックすると、ポップアップメニューが表示される。

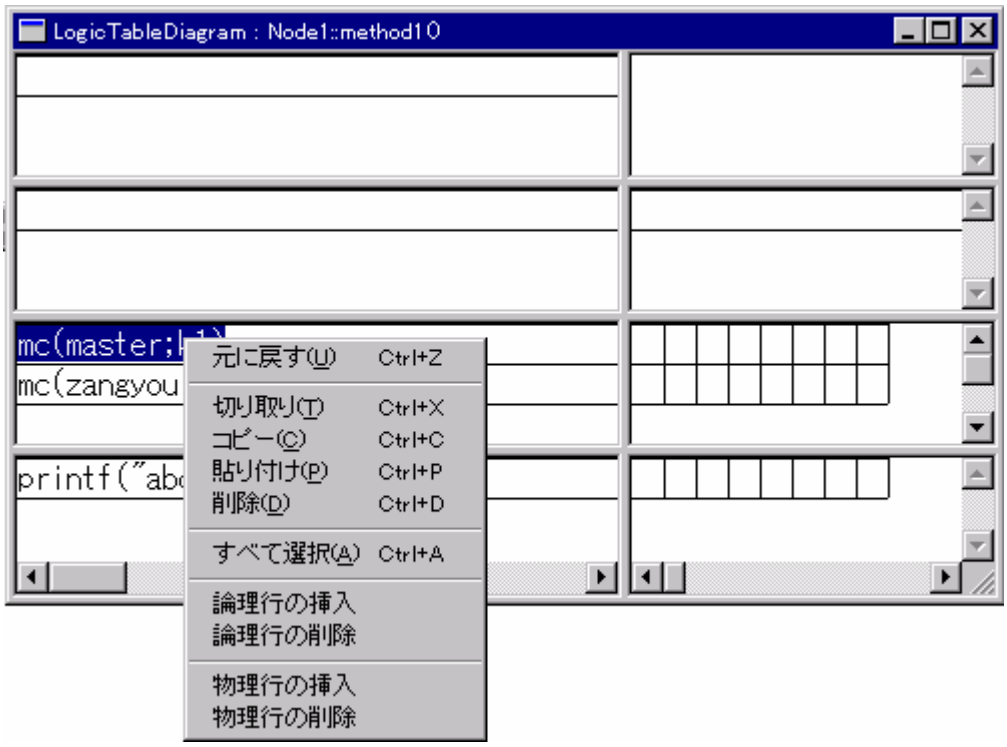


図 1-65 ポップアップメニュー

- 切り取り　　選択された文字列を画面から削除して、クリップボードにコピーする。
- コピー　　選択された文字列をコピーする。
- 貼り付け　　選択された部分にコピーされた内容を貼り付ける。
- 削除　　選択された文字列を削除する。
- すべて選択　　指定した行の全ての文字を選択し、反転表示する。
- 論理行の挿入　指定した行に新しく論理行を挿入する。
- 論理行の削除　指定した論理行を削除する。
- 物理行の挿入　指定した行に新しく物理行を挿入する。
- 物理行の削除　指定した物理行を削除する。

＜規則部の編集＞

規則部を右クリックすると、ポップアップメニューが表示される。

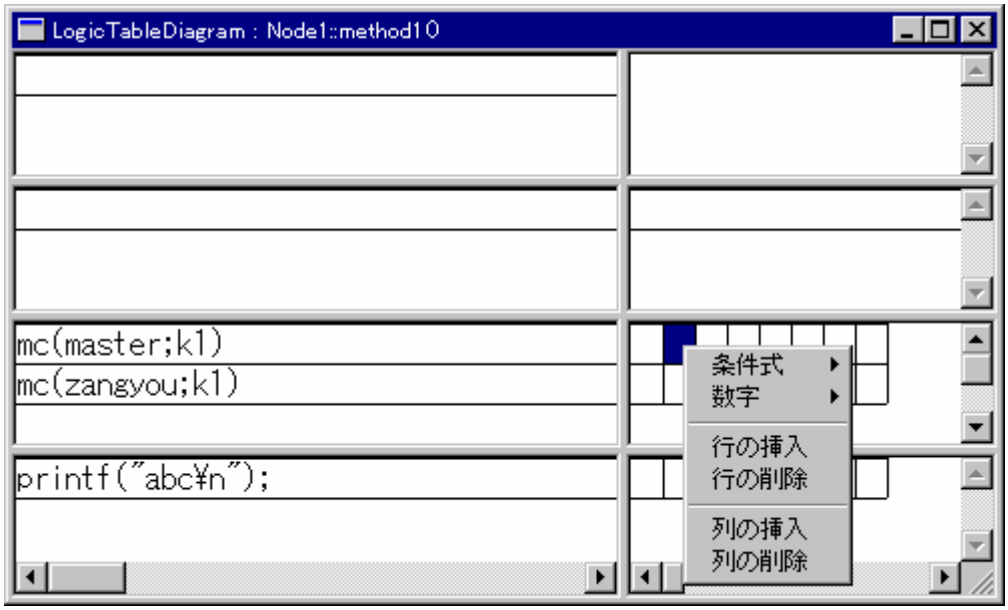


図 1-66 ポップアップメニュー

- 行の挿入　　選択された行に新たな論理行を挿入する。
- 行の削除　　選択された論理行を削除する。
- 列の挿入　　選択された列に新たな列を挿入する。
- 列の削除　　選択された列を削除する。なお条件部と処理部では対応する規則部の１列を同時に削除する。

3.　テキストエリアの記述

①　初期処理部

モジュールの状態を初期化するための命令を記述する。ただし、他のモジュールから do 呼び出しされた場合は初期処理部に書かれた命令は実行されず、前処理部に記述された命令から実行される。(do 呼び出しについては④を参照)

②　前処理部

条件部と処理部を実行する前に必要な命令を記述する。なお、局所変数の宣言はここで宣言する。

③　条件部

あらかじめ用意された条件式や論理式を記述する。次のページの表に条件式を示す。

＜SPACE 条件式一覧＞

種別	名前	書き方	条件値	読み方
データ構造条件式	照合式	MC（F；K）	Y，N，E，＊	ファイルFから入力したレコードのグループキーKと現在の処理対象を表わす照合キーKの値が等しい時Yes,等しくないときNo。＊はNまたはEの意味。
	制御切れ式	CB（F；K）	H，B，T，E，＊	当該モジュールが初期化された直後の状態の時あるいは、ファイルFから入力したレコードによって、グループキーKが同一値の新しいレコードグループに入った時Head,グループ内での各レコード処理状態の時Body,現在のグループから抜け出ようとする終了状態の時Tail。＊はTまたはEの意味。
	位置式	LC（F；K）	F，I，L，U，E，＊，＋	ファイルFから入力したレコードが、グループキーKが同一値のレコードグループを構成する唯一のレコードの時Unique、グループの先頭レコードの時First、最終レコードの時Last、中間レコードの時Intermediate。＊はLまたはEの意味。＋はLまたはUの意味。
	コレクション処理式	ForEach（List or Array or Map；C）	H，B，T	コレクションの中で条件式Cの成り立つ各要素に対して行う処理の実行状態を表す。H：当該モジュールが初期化された直後の状態。　B：各要素処理状態。T：処理対象となる全ての要素への処理が終わった直後の状態。
	読込み式	GT（F）	H，B，E	当該モジュールが初期化された直後の状態の時Head、各レコードの処理状態の時Body。
計算条件式	選択式	S＝｛S1；…..；Sn｝	0，1……，n	式S＝式Siとなる最小のi≧1が存在する時はそのi、そのようなiがない時0。
	論理式	C	Y，N	論理式Cの評価結果が真の時Yes、そうでない時No。

④　処理部

あらかじめ用意された処理文や、実際の C++命令を記述する。

以下に処理文を示す。

＜SPACE 処理文一覧＞

名前	記述	意味
実行制御文	SetNext	条件部に記入された条件式固有の暗黙の後処理を実行する。
	Repeat	当該ロジックテーブルを再実行する。
	DoAgain	後処理の後、当該ロジックテーブルを再実行する。
	Do 関数名(引数)	初期処理を行わないロジックテーブル型関数の呼び出し。
	Init 関数名	ロジックテーブル型関数の初期処理のみを行う。
	C++命令	そのまま展開される。

4.　ルールエリアの記述

ルールエリアは、どのような条件の時にどのような処理を行うかを、条件値を記入することで表現する。

以下に条件式と条件値の対応関係を示す。

条件式	条件値
MC(F;K)	Y, N, E, ＊
CB(F;K)	H, B, T, E, ＊
LC(F;K)	F, I, L, U, E, ＊, ＋
ForEach(List or Array or Map;C)	H, B, T
GT(F)	H, B, E
S＝{S1;……;Sn}	0, 1……, n
C	Y, N

(1) 条件値の入力

条件値の入力には 2 種類ある。

①　ポップアップメニューを使う。

条件値を入力したいセルを右クリックしてポップアップメニューから、条件値を選ぶ。

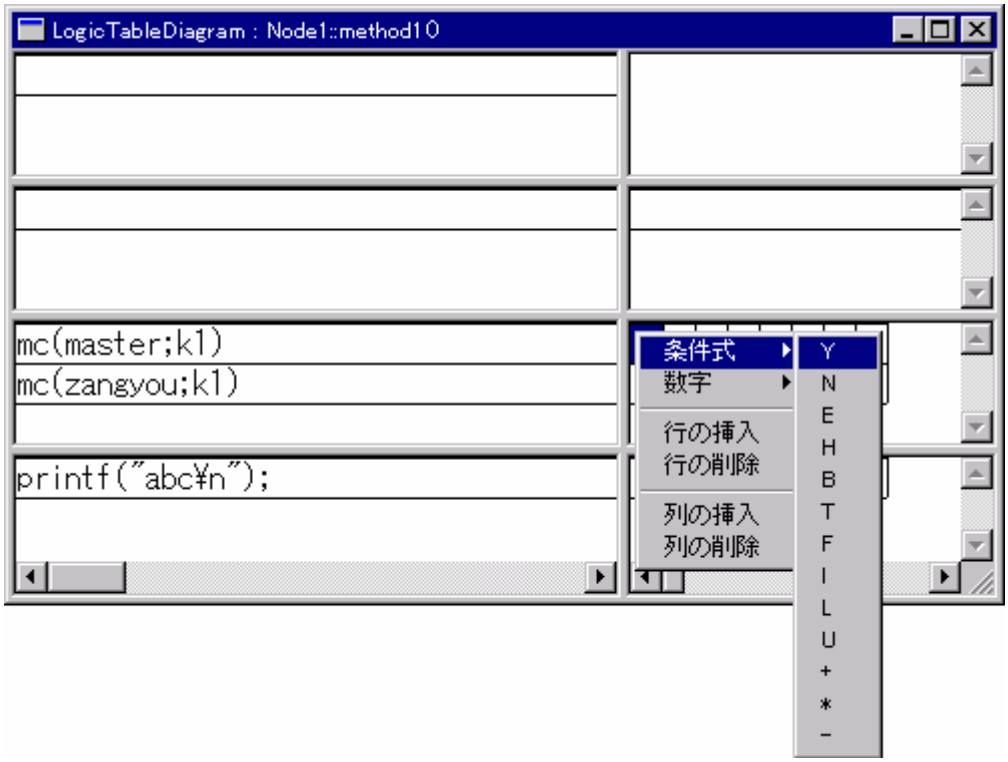


図 1-67 ポップアップメニュー

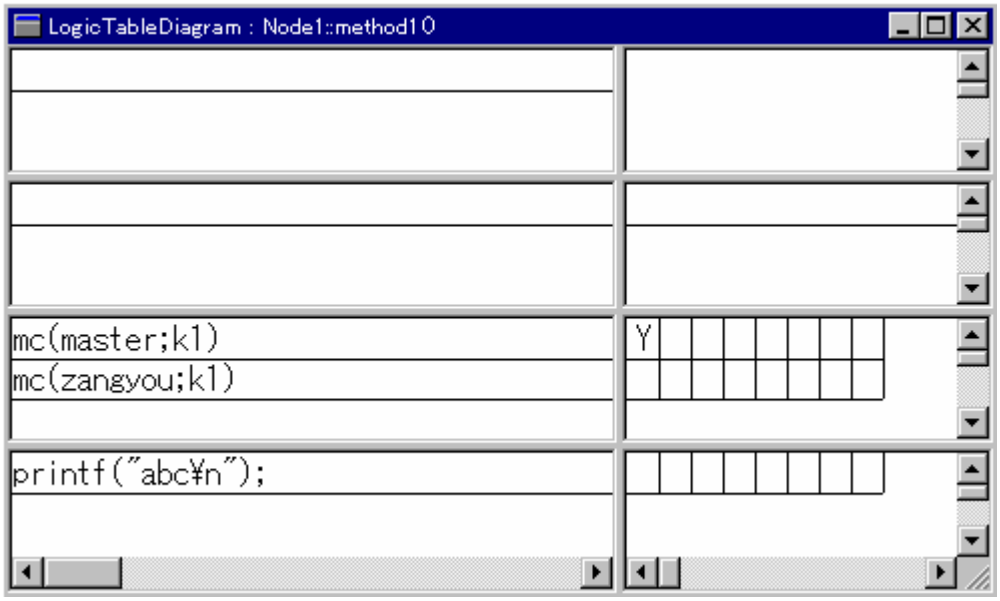


図 1-68 条件値の入力

② マウス入力

条件値を入力したいセルを左クリックしてキーボードから入力する。2 回目以降は 1 つ前の条件値を左クリックのみで入力できる。

(2) 条件値と処理の対応づけ

図のように、処理と条件の対応したセルを左クリックし、‘X’を入力する。既にX’が入力されているときは空白に戻る。

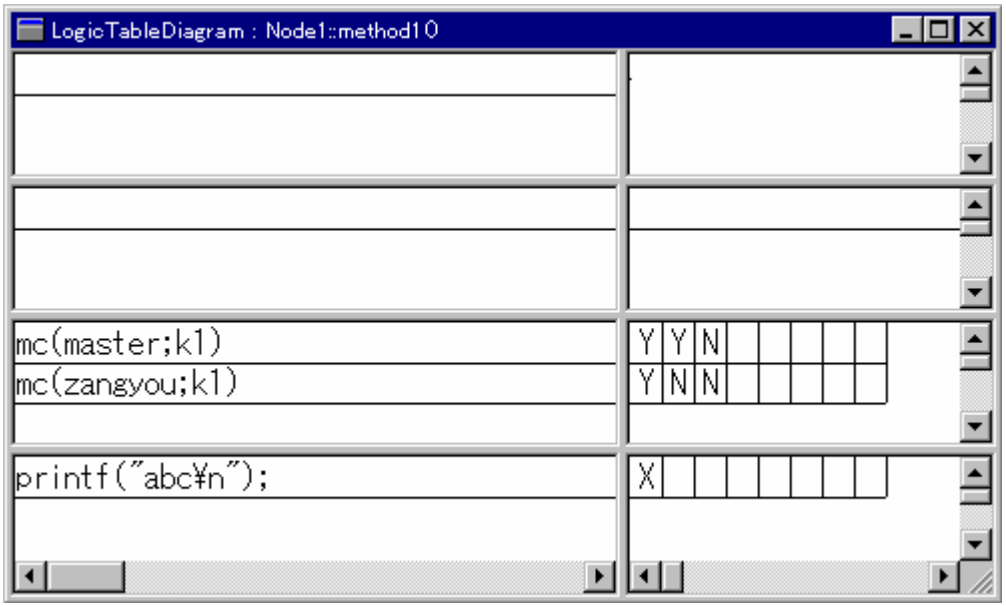


図 1-69 処理の選択

5. チェック機能

ロジックテーブルには、記述された内容が完全でなくてはならないという制約がある。例えば、条件値の記述に重複や漏れがあると、その条件と対応づけられた処理にも重複や漏れが生じてしまうからである。よって、ユーザはプログラムを生成する前にロジックテーブルの完全性をチェックする必要があるのだがこれは大変な労力を要する。そこで今回ロジックテーブルの完全性を自動的にチェックする機能を持たせた。ユーザはツールバーのチェックボタンを押すだけで完全性をチェックでき、エラーがあるときは図 1-72 のようなエラーダイアログが表示される。以下にエラーの種類を記す。

- 条件値の組み合わせに重複や漏れがある。
- 条件部に空白の論理行が存在する。
- 条件値が入力されていないのにそれに対応する実行指示子が入力されている場合など、不正な列が存在している。
- その条件式では取り得ない条件値が入力されている。
- 条件部において、CB 式、LC 式、GT 式を 1 つのロジックテーブル内で同時に使用している。
- 条件部において ForEach 式を 1 つのロジックテーブル内で複数使用している。

図 1-70 条件値欄が完全な場合

図 1-71 条件値欄が不完全な場合

図 1-72 条件値欄に洩れがある場合のエラーメッセージ

６．パレットによる入力

ロジックテーブルエディタにはプログラムの記述をマウスによって簡単に行えるパレットを用意した。パレットはメソッドごとに用意され、メソッドの所属するクラスや、集約クラス、関連クラスのオブジェクト名、メンバ変数、メソッドが登録されている。

(1) ロジックテーブルを起動し、メニューの[ツール][パレットの表示]を選ぶか、もしくはツールバーのパレットボタン  を指定すると図 1-73 のようなクラス選択パレットが表示される。

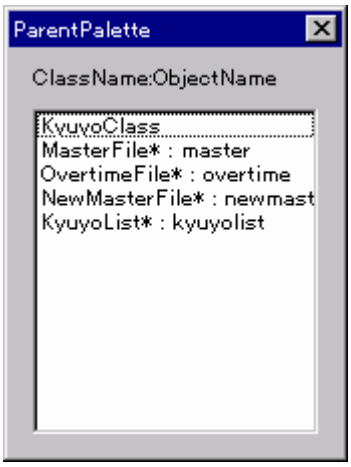


図 1-73 クラス選択パレット

(2) クラス選択パレットには当該メソッドが所属するクラス、集約クラス、関連クラスが表示されている。これらをダブルクリックすることにより、選択されたクラスのオブジェクト名、メンバ変数名、メソッド名が登録されたクラスパレットが表示される。図 1-74 にクラスパレットを示す。

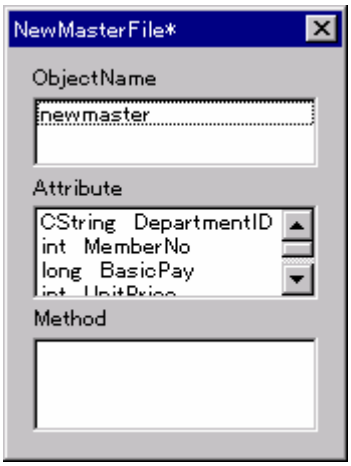


図 1-74 クラスパレット

(3) クラスパレットに表示されたメンバ変数名などをダブルクリックすることにより、ロジックテーブルの指定したカーソル位置にその文字列が挿入される。図 1-75 と図 1-76 に例を示す。

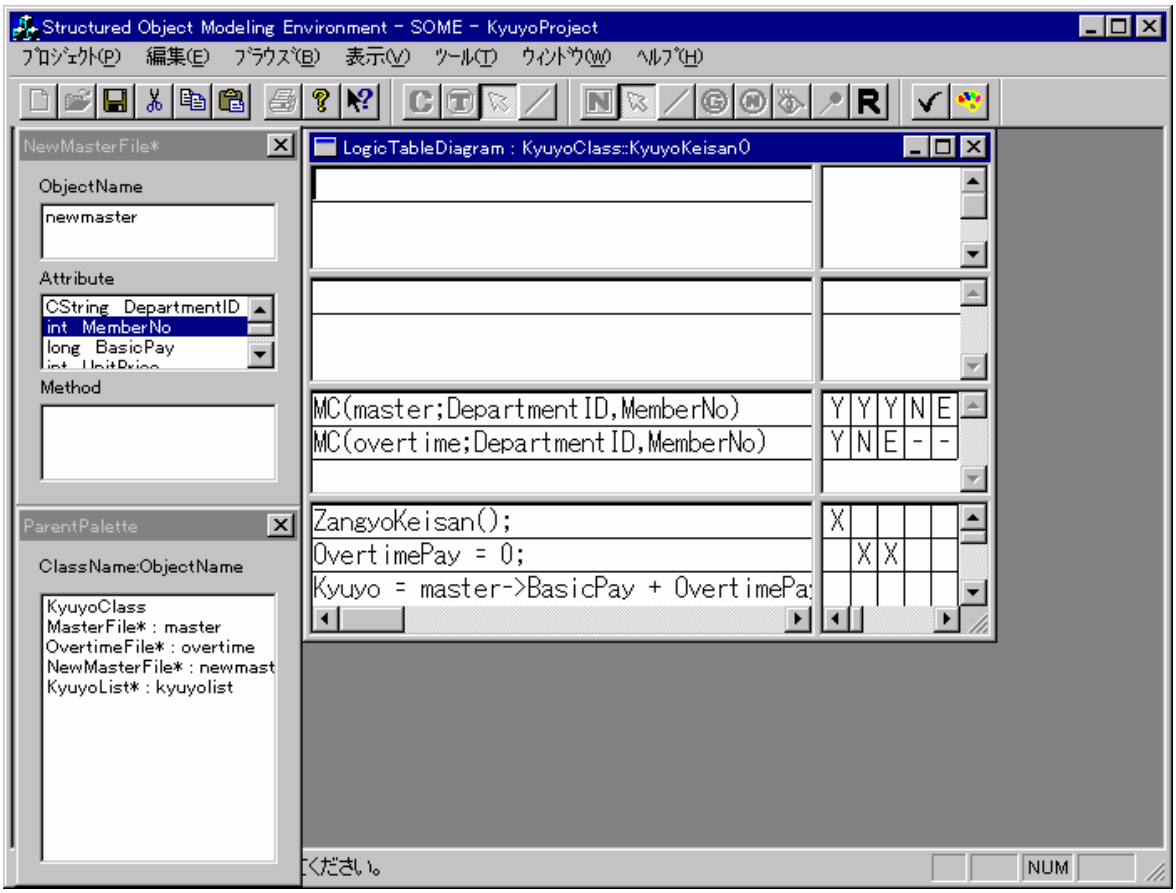


図 1-75 クラスパレットをダブルクリックする前

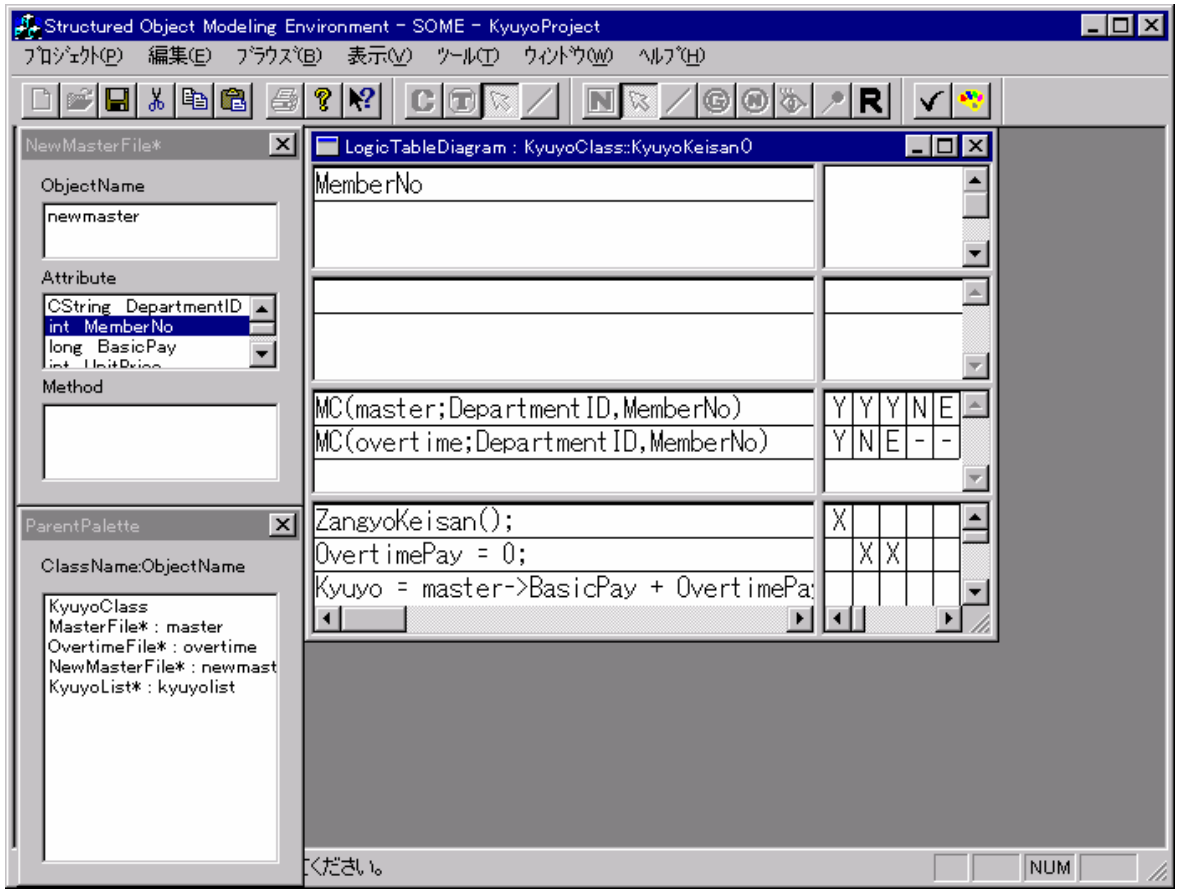


図 1-76 クラスパレットをダブルクリックした後

1.1.6. DSL 形式への import/export

●DSL の import

メニューバーの「ファイル」から「DSL のインポート」を選択する。ファイル名を選択するダイアログが開くので、これから復元する DSL ファイルの名前を指定する。

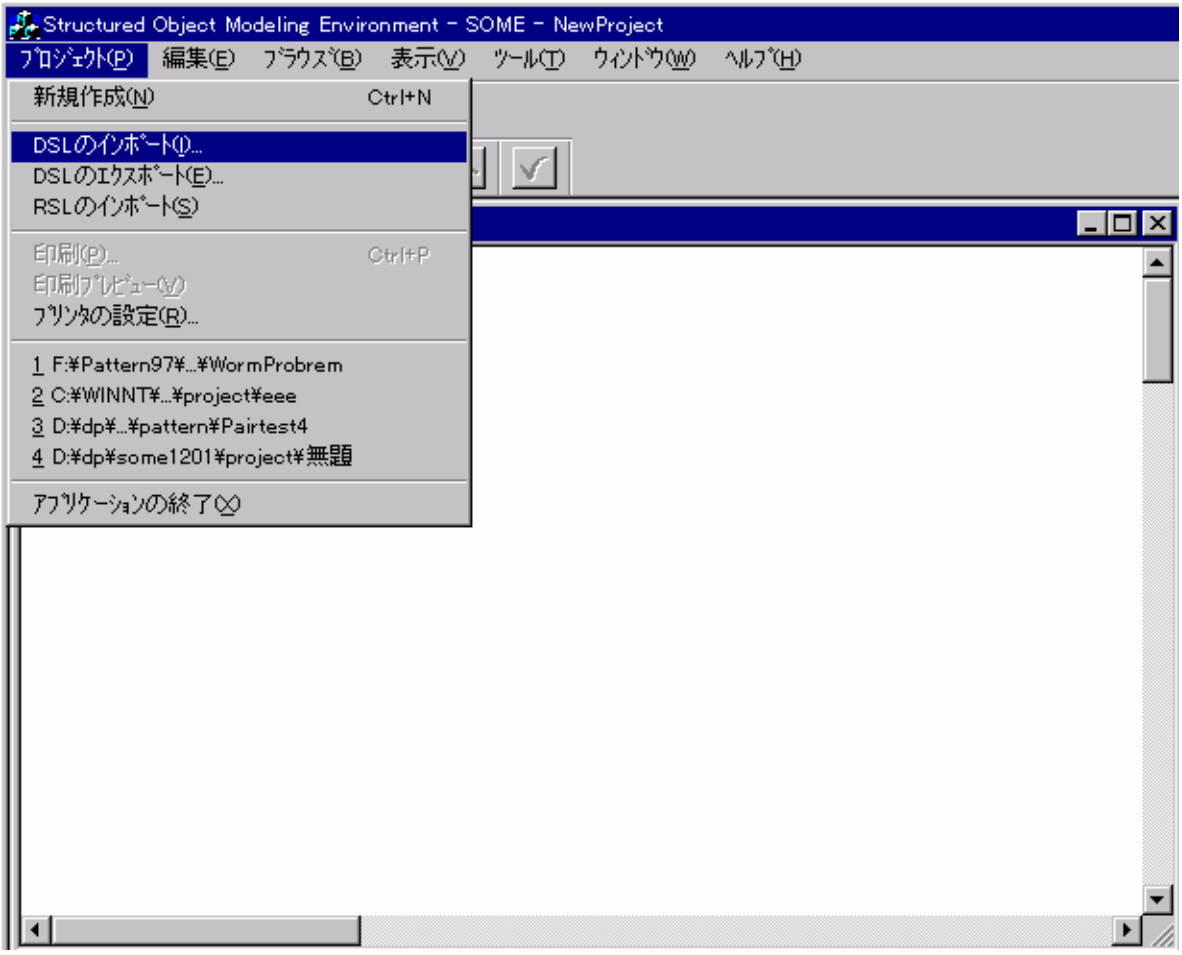


図 1-77 メニューバー

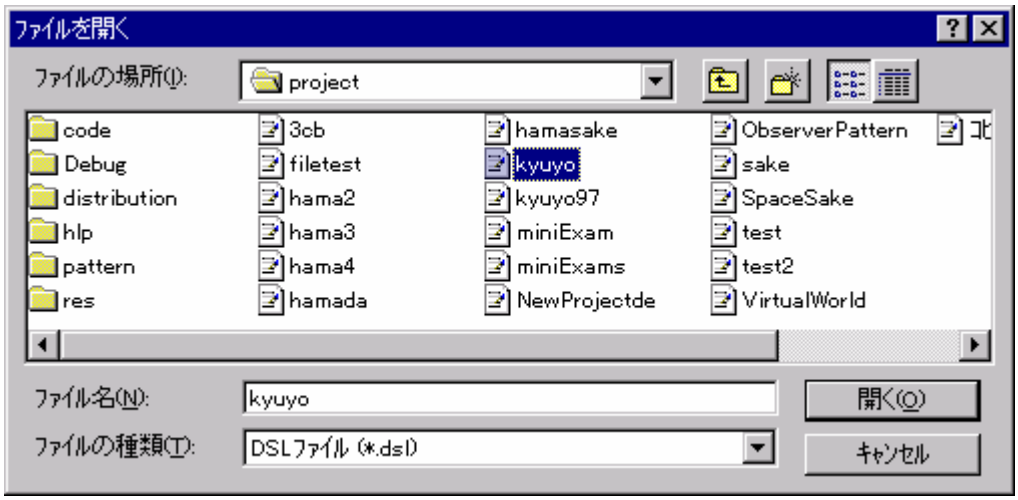


図 1-78　DSL のインポート

●DSL の export

メニューバーの「ファイル」から「DSL のエクスポート」を選択する。ファイル名を入力するダイアログが開くので、作成する DSL ファイルの名前を指定する。デフォルトはプロジェクト名.dsl である。

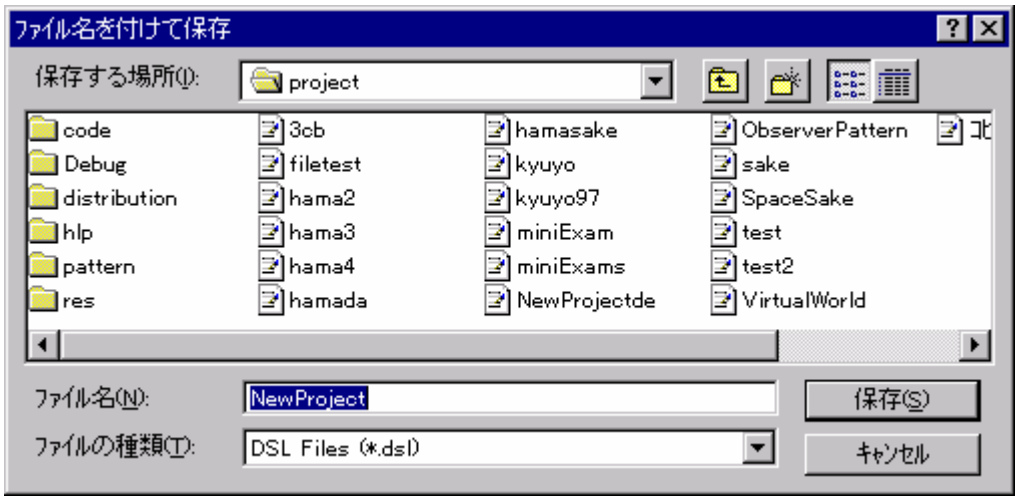


図 1-79　DSL のエクスポート

1.1.7. プログラム生成

1.1.7.1. プリプロセッサリスト

#include 文や#define 文などのプリプロセッサ命令を生成プログラムに組み込みたい場合に利用する。

①　メニューバーの「ツール」から「コード生成」「プリプロセッサリスト」を選択する。

図 1-80 ツールバー

② ダイアログが開くので、そこにプリプロセッサ文を入力する。

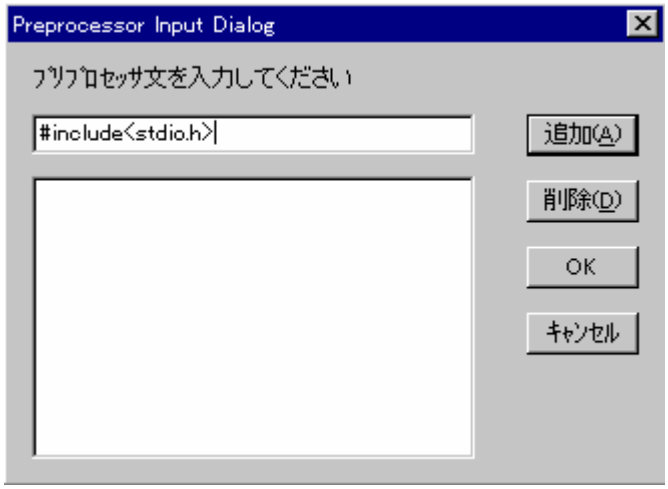


図 1-81 プリプロセッサ入力ダイアログ

③ ［追加］ボタンを押す。プリプロセッサリストに追加される。

図 1-82 プリプロセッサ入力ダイアログ

④ 逆にリストから削除する場合は、削除するプリプロセッサ文を選択して、［削除］ボタンを押す。

1.1.7.2. C++プログラムの生成

① C++プログラムの生成はメニューバーの「ツール」から「コード生成」「C++コード生成」を選択する。

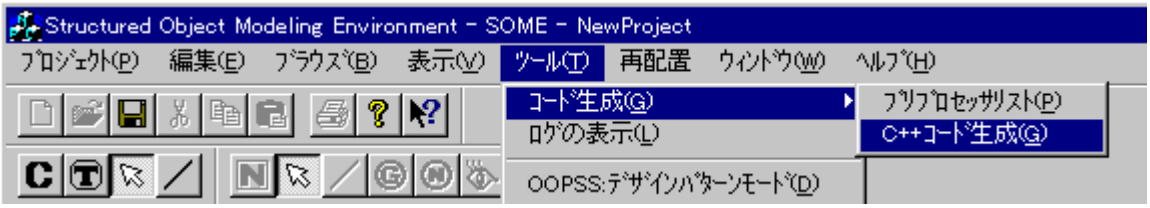


図 1-83 ツールバー

② ファイル名を入力するダイアログが開くので、ヘッダファイル名と、cpp ファイル名を指定する。デフォルトはそれぞれプロジェクト名.h、プロジェクト名．cpp である。

図 1-84 ダイアログ

1.2. 記述例

1.2.1. 給与計算問題

1.2.1.1. 事例の説明

このファイル処理は、マスタファイルＭ、残業ファイルＯを入力とし、新マスタファイルＭ2、社員給与リストＬを出力する。

[処理要件]

(1) マスタファイルＭのレコードは部 No、社員 No、基本給、単価、合計からなる。

(2) 社員給与リストＬのレコードは部 No、社員 No、給料、部合計からなる。

(3) マスタファイルＭの社員ごとに行を新マスタファイルＭ2 に出力する。

(4) マスタファイルＭの社員ごとに行を社員給与リストＬに出力する。

(5) マスタファイルＭの部ごとに部合計を社員給与リストに出力する。

(6) 給与は基本給に残業手当を加えたものである。

(7) 残業手当は残業代を社員ごとに集計したものである。

(8) 残業代は、残業ファイル０の残業レコードごとに時間、掛率およびその残業を行った社員の単価を用いて下記の式で求められる。

(9) 残業代＝単価×時間×掛率

(10) 部合計は部ごとに社員の給与を集計したものである。

1.2.1.2. 画面

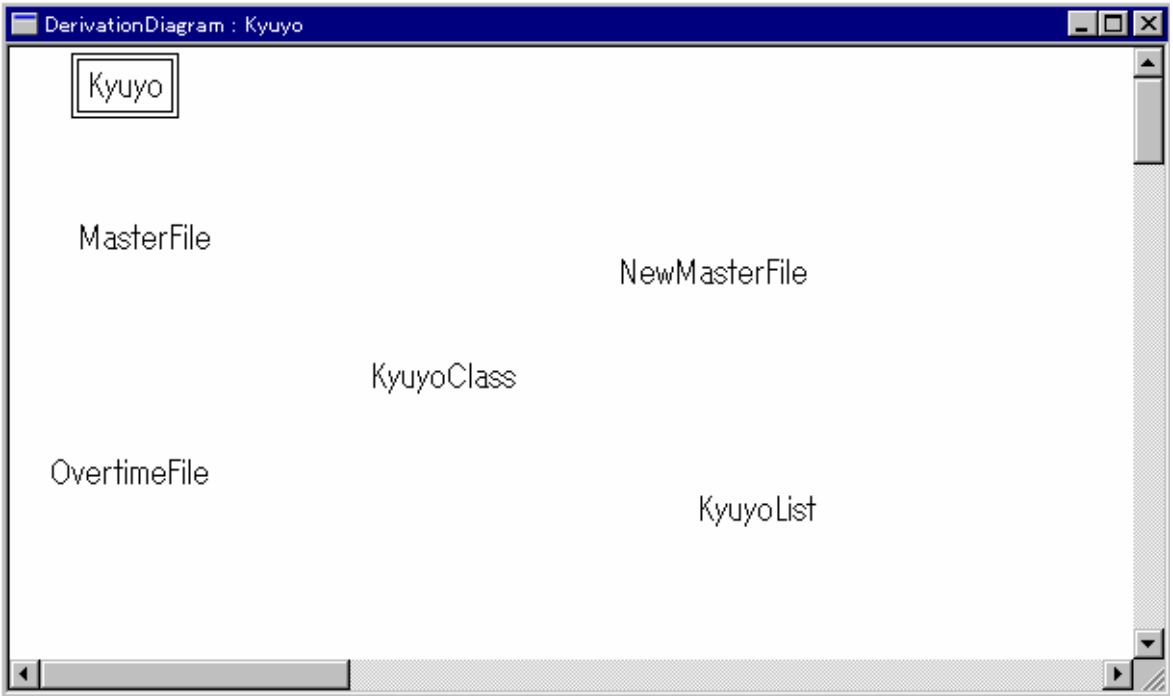


図 1-85 給与計算問題－派生図

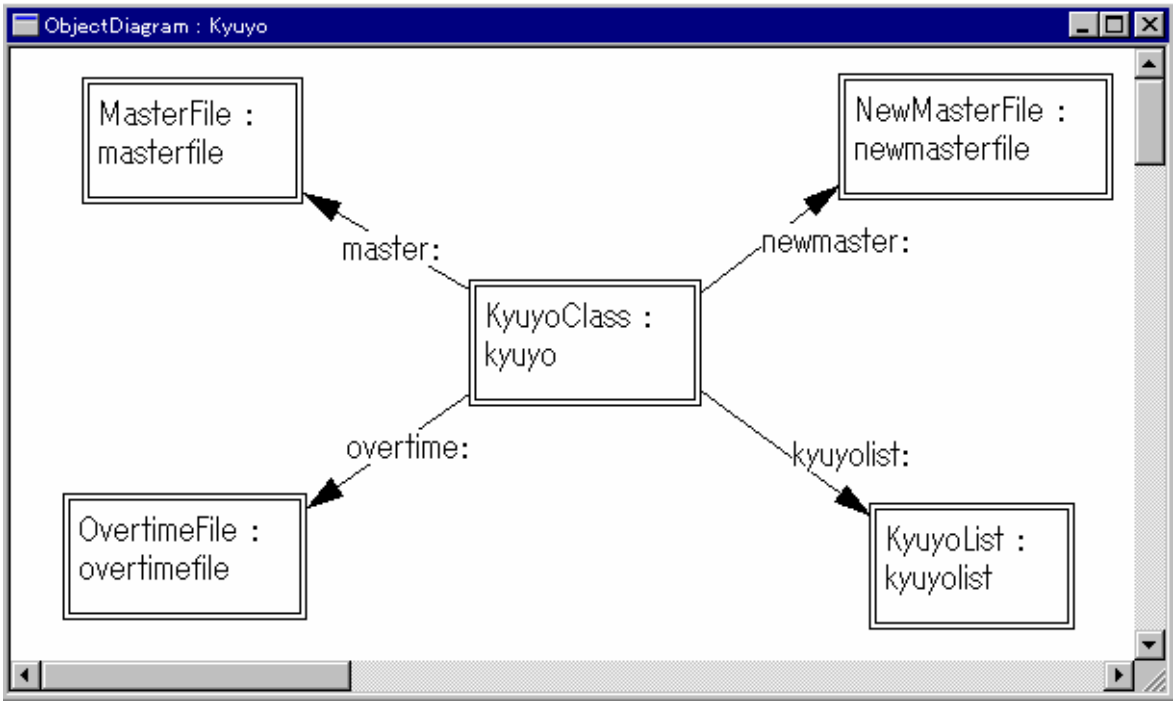


図 1-86 Kyuyo クラスのオブジェクト図

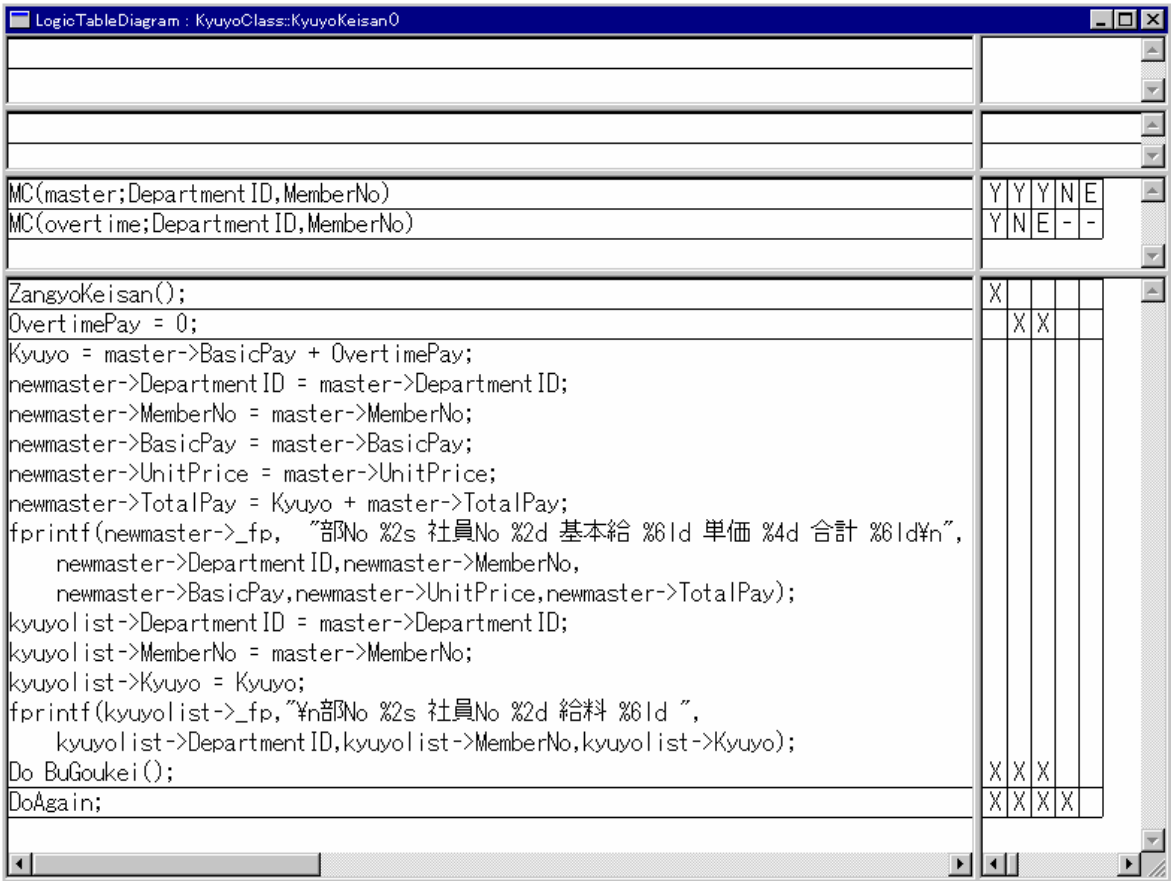


図 1-87 KyuyoClass::KyuyoKeisan()のロジックテーブル

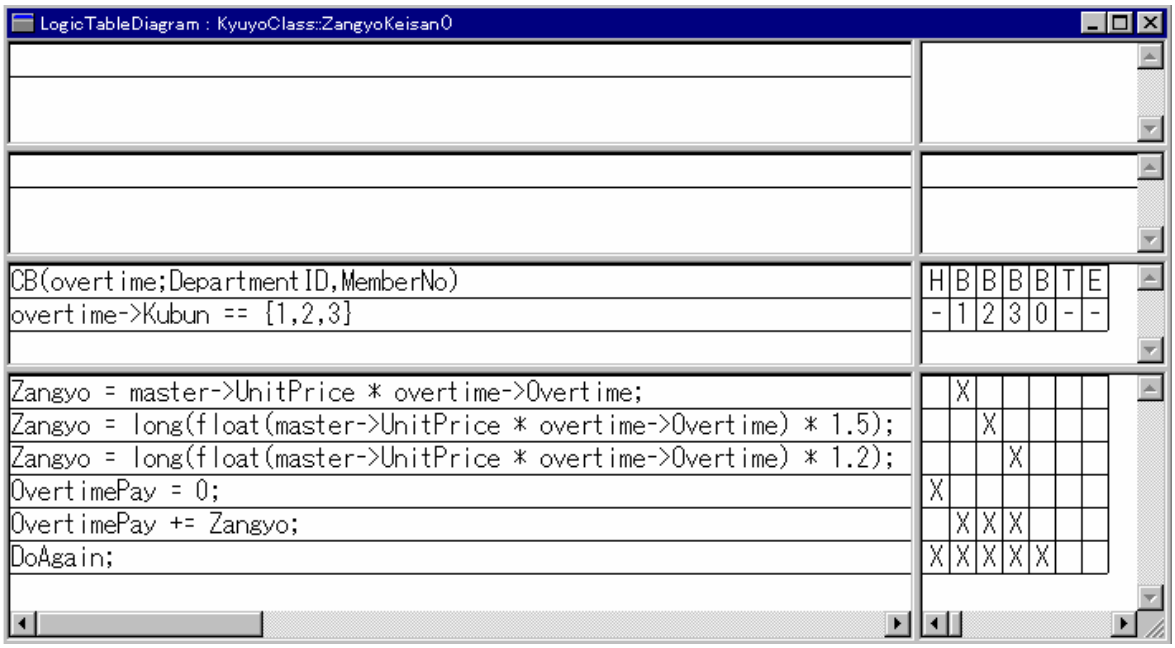


図 1-88 KyuyoClass::Zangyokeisan()のロジックテーブル

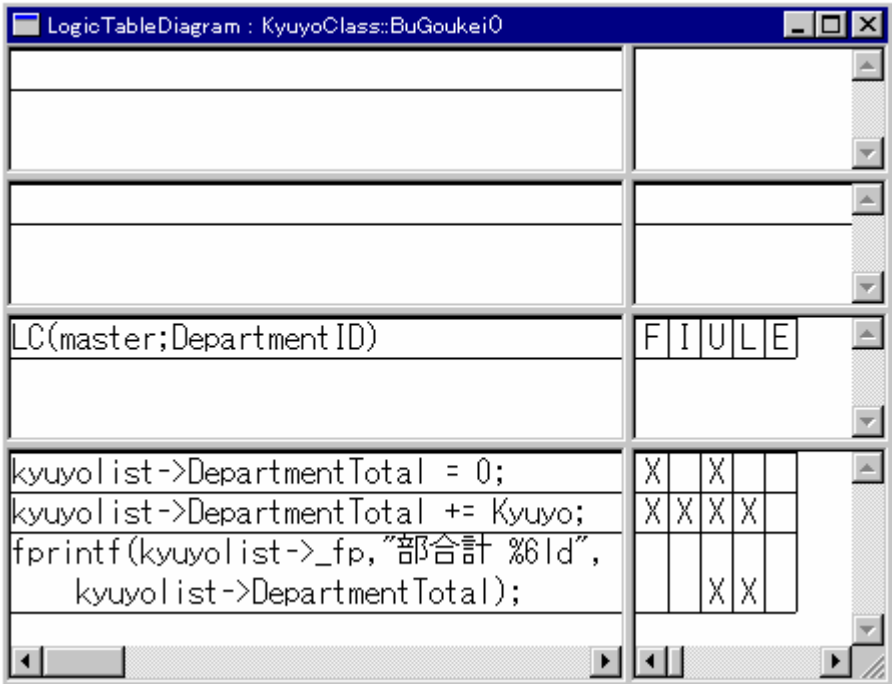


図 1-89 KyuyoClass::BuGoukei()のロジックテーブル

1.2.1.3. 生成プログラム

```
#include "space.h"

//CKeyCode クラスのメンバ関数定義

void CKeyCode::operator = (CKeyCode key)
{
```

```
     key_1 = key.key_1;
     key_2 = key.key_2;
}

void CKeyCode::SetKeyMax()
{
     key_1 = CHAR_MAX;
     key_2 = INT_MAX;
}

//_WhichIsSmallKey 関数定義
int CLogicTable::_WhichIsSmallKey(CKeyCode keyA,CKeyCode keyB,CString* Level)
{
     int k[_ALLKEY];
     int count = 0;

     if (Level[count] == "DepartmentID") k[count++] =
_SmallSingleKey(keyA.key_1,keyB.key_1);
     if (Level[count] == "MemberNo") k[count++] =
_SmallSingleKey(keyA.key_2,keyB.key_2);

     if (count==0) printf("指定されたキーが存在しません！！");
     if (count>0) if (k[0] != _EQUAL) return k[0];
     if (count>1) if (k[1] != _EQUAL) return k[1];
     return _EQUAL;
}
```

図 1-90 Key.cpp ファイル

```
#ifndef ___KEY_H
#define ___KEY_H

#include "stdafx.h"

#define _ALLKEY 2

//CKeyCode クラスの定義
class CKeyCode {
public:
     CString key_1;
     int key_2;

     void operator = (CKeyCode);
```

```
     void SetKeyMax();
};

#endif
```

図 1-91 key.h ファイル

```
#include "KyuyoProject.h"
//KyuyoKeisan 関数定義
void KyuyoClass::KyuyoKeisan()
{
     _KyuyoKeisan_SetTable();
     _KyuyoKeisan_Init();
     _KyuyoKeisan_Body();
}

//SetTable 関数定義
void KyuyoClass::_KyuyoKeisan_SetTable()
{
     _KyuyoKeisan_Map[0] = "YY:X XX";
     _KyuyoKeisan_Map[1] = "YN: XXX";
     _KyuyoKeisan_Map[2] = "YE: XXX";
     _KyuyoKeisan_Map[3] = "N-:   X";
     _KyuyoKeisan_Map[4] = "E-:    ";
     _KyuyoKeisan_Map[5] = '¥0';
}

//Init 関数定義
void KyuyoClass::_KyuyoKeisan_Init()
{
     //初期処理部

}

//Body 関数定義
void KyuyoClass::_KyuyoKeisan_Body()
{
     //宣言部

     char _currCondVal[3] = "";


     CString _Level[_ALLKEY+1];
```

```
     _Level[0] = "DepartmentID";
     _Level[1] = "MemberNo";
_DOAGAIN:

     //前処理部



     //MC 関数の前処理
     _KyuyoKeisan_MC_Key.SetKeyMax();
     _PreMC(master,_Level,&_KyuyoKeisan_MC_Key);
     _PreMC(overtime,_Level,&_KyuyoKeisan_MC_Key);

     //条件値の設定
     _currCondVal[0] = _MC(master,_Level,&_KyuyoKeisan_MC_Key);
     _currCondVal[1] = _MC(overtime,_Level,&_KyuyoKeisan_MC_Key);
     _currCondVal[2] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_KyuyoKeisan_Map,_currCondVal);

     //処理部の実行
     if (_KyuyoKeisan_Map[_rule][3] == 'X') {
          ZangyoKeisan();
     }
     if (_KyuyoKeisan_Map[_rule][4] == 'X') {
          OvertimePay = 0;
     }
     if (_KyuyoKeisan_Map[_rule][5] == 'X') {
          Kyuyo = master->BasicPay + OvertimePay;
          newmaster->DepartmentID = master->DepartmentID;
          newmaster->MemberNo = master->MemberNo;
          newmaster->BasicPay = master->BasicPay;
          newmaster->UnitPrice = master->UnitPrice;
          newmaster->TotalPay = Kyuyo + master->TotalPay;
          fprintf(newmaster->_fp,  "部 No %2s 社員 No %2d 基本給 %6ld 単価 %4d 合
計 %6ld¥n",newmaster->DepartmentID,newmaster->MemberNo,newmaster->BasicPay,newmaste
r->UnitPrice,newmaster->TotalPay);
          kyuyolist->DepartmentID = master->DepartmentID;
          kyuyolist->MemberNo = master->MemberNo;
          kyuyolist->Kyuyo = Kyuyo;
          fprintf(kyuyolist->_fp,"¥n 部 No %2s 社員 No %2d 給料 %6ld
",kyuyolist->DepartmentID,kyuyolist->MemberNo,kyuyolist->Kyuyo);
          _BuGoukei_SetTable();
```

```
          _BuGoukei_Body();
     }
     if (_KyuyoKeisan_Map[_rule][6] == 'X') {
          if (_currCondVal[0] == 'Y') master->_Read();
          if (_currCondVal[1] == 'Y') overtime->_Read();
          goto _DOAGAIN;
     }
}

//ZangyoKeisan 関数定義
void KyuyoClass::ZangyoKeisan()
{
     _ZangyoKeisan_SetTable();
     _ZangyoKeisan_Init();
     _ZangyoKeisan_Body();
}

//SetTable 関数定義
void KyuyoClass::_ZangyoKeisan_SetTable()
{
     _ZangyoKeisan_Map[0] = "H-:   X X";
     _ZangyoKeisan_Map[1] = "B1:X   XX";
     _ZangyoKeisan_Map[2] = "B2: X  XX";
     _ZangyoKeisan_Map[3] = "B3:  X XX";
     _ZangyoKeisan_Map[4] = "B0:     X";
     _ZangyoKeisan_Map[5] = "T-:      ";
     _ZangyoKeisan_Map[6] = "E-:      ";
     _ZangyoKeisan_Map[7] = '¥0';
}

//Init 関数定義
void KyuyoClass::_ZangyoKeisan_Init()
{
     //初期処理部

     _ZangyoKeisan_CB_NextSt = 1;
     _ZangyoKeisan_CB();
}

//Body 関数定義
void KyuyoClass::_ZangyoKeisan_Body()
{
     //宣言部
```

```
     char _currCondVal[3] = "";

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = _ZangyoKeisan_CB_Condition[0];
     _currCondVal[1] = _ZangyoKeisan_SelectValue(1);
     _currCondVal[2] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_ZangyoKeisan_Map,_currCondVal);

     //処理部の実行
     if (_ZangyoKeisan_Map[_rule][3] == 'X') {
          Zangyo = master->UnitPrice * overtime->Overtime;
     }
     if (_ZangyoKeisan_Map[_rule][4] == 'X') {
          Zangyo = long(float(master->UnitPrice * overtime->Overtime) * 1.5);
     }
     if (_ZangyoKeisan_Map[_rule][5] == 'X') {
          Zangyo = long(float(master->UnitPrice * overtime->Overtime) * 1.2);
     }
     if (_ZangyoKeisan_Map[_rule][6] == 'X') {
          OvertimePay = 0;
     }
     if (_ZangyoKeisan_Map[_rule][7] == 'X') {
          OvertimePay += Zangyo;
     }
     if (_ZangyoKeisan_Map[_rule][8] == 'X') {
          _ZangyoKeisan_CB();
          goto _DOAGAIN;
     }
}

//CB 関数定義
void KyuyoClass::_ZangyoKeisan_CB()
{
     static CKeyCode CB_Gid;
     CString Level1[_ALLKEY];
     Level1[0] = "DepartmentID";
```

```
     Level1[1] = "MemberNo";
     if (overtime->_EOF_flag == TRUE) {
          strcpy(_ZangyoKeisan_CB_Condition,"E");
          return;
     }

_LOOP:
     switch(_ZangyoKeisan_CB_NextSt) {
     case 1 :
          CB_Gid = overtime->_C_Key;
          _ZangyoKeisan_CB_NextSt = 3;
          strcpy(_ZangyoKeisan_CB_Condition,"H");
          break;

     case 2 :
          if (_WhichIsSmallKey(overtime->_N_Key,CB_Gid,Level1) == _EQUAL) {
               overtime->_Read();
               _ZangyoKeisan_CB_NextSt = 3;
               goto _LOOP;
          }
          else {
               strcpy(_ZangyoKeisan_CB_Condition,"T");
               _ZangyoKeisan_CB_NextSt = 4;
          }
          break;

     case 3 :
          _ZangyoKeisan_CB_NextSt = 2;
          strcpy(_ZangyoKeisan_CB_Condition,"B");
          break;

     case 4 :
          overtime->_Read();
          if (overtime->_EOF_flag == FALSE) {
               _ZangyoKeisan_CB_NextSt = 1;
               goto _LOOP;
          }
          else {
               strcpy(_ZangyoKeisan_CB_Condition,"E");
          }
          break;

     default :
          break;
```

```
     }
}

//SelectValue 関数定義
char KyuyoClass::_ZangyoKeisan_SelectValue(int k)
{
     switch(k) {
     case 1 :
          if (overtime->Kubun == 1) return '1';
          if (overtime->Kubun == 2) return '2';
          if (overtime->Kubun == 3) return '3';
          break;

     default :
          break;
     }
     return '0';
}

//BuGoukei 関数定義
void KyuyoClass::BuGoukei()
{
     _BuGoukei_SetTable();
     _BuGoukei_Init();
     _BuGoukei_Body();
}

//SetTable 関数定義
void KyuyoClass::_BuGoukei_SetTable()
{
     _BuGoukei_Map[0] = "F:XX ";
     _BuGoukei_Map[1] = "I: X ";
     _BuGoukei_Map[2] = "U:XXX";
     _BuGoukei_Map[3] = "L: XX";
     _BuGoukei_Map[4] = "E:   ";
     _BuGoukei_Map[5] = '¥0';
}

//Init 関数定義
void KyuyoClass::_BuGoukei_Init()
{
     //初期処理部

}
```

```
//Body 関数定義
void KyuyoClass::_BuGoukei_Body()
{
     //宣言部

     char _currCondVal[2] = "";

     CString _Level1[_ALLKEY];
     _Level1[0] = "DepartmentID";
_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = _LC(master,_Level1);
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_BuGoukei_Map,_currCondVal);

     //処理部の実行
     if (_BuGoukei_Map[_rule][2] == 'X') {
          kyuyolist->DepartmentTotal = 0;
     }
     if (_BuGoukei_Map[_rule][3] == 'X') {
          kyuyolist->DepartmentTotal += Kyuyo;
     }
     if (_BuGoukei_Map[_rule][4] == 'X') {
          fprintf(kyuyolist->_fp,"部合計 %6ld",kyuyolist->DepartmentTotal);
     }
}

//start 関数定義
void KyuyoProject::start()
{
     _start_SetTable();
     _start_Init();
     _start_Body();
}

//SetTable 関数定義
```

```
void KyuyoProject::_start_SetTable()
{
     _start_Map[0] = "-:X";
     _start_Map[1] = '¥0';
}

//Init 関数定義
void KyuyoProject::_start_Init()
{
     //初期処理部

}

//Body 関数定義
void KyuyoProject::_start_Body()
{
     //宣言部

     char _currCondVal[2] = "";

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = (1) ? 'Y' : 'N';
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_start_Map,_currCondVal);

     //処理部の実行
     if (_start_Map[_rule][2] == 'X') {
          kyuyo.KyuyoKeisan();
     }
}

//KyuyoProject 関数定義
 KyuyoProject::KyuyoProject()
{
     _KyuyoProject_SetTable();
     _KyuyoProject_Init();
```

```
     _KyuyoProject_Body();
}

//SetTable 関数定義
void KyuyoProject::_KyuyoProject_SetTable()
{
     _KyuyoProject_Map[0] = "-:X";
     _KyuyoProject_Map[1] = '¥0';
}

//Init 関数定義
void KyuyoProject::_KyuyoProject_Init()
{
     //初期処理部

}

//Body 関数定義
void KyuyoProject::_KyuyoProject_Body()
{
     //宣言部

     char _currCondVal[2] = "";

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = (1) ? 'Y' : 'N';
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_KyuyoProject_Map,_currCondVal);

     //処理部の実行
     if (_KyuyoProject_Map[_rule][2] == 'X') {
          kyuyo.master = &masterfile;
          kyuyo.overtime = &overtimefile;
          kyuyo.newmaster = &newmasterfile;
          kyuyo.kyuyolist = &kyuyolist;
```

```
            kyuyo.master->_Open("Data¥¥Sample1¥¥In_Master.txt","r");
            kyuyo.overtime->_Open("Data¥¥Sample1¥¥In_Overtime.txt","r");
            kyuyo.newmaster->_Open("Data¥¥Sample1¥¥Out_NewMaster.txt","w");
            kyuyo.kyuyolist->_Open("Data¥¥Sample1¥¥Out_KyuyoList.txt","w");
      }
}

//_KyuyoProject 関数定義
 KyuyoProject::~KyuyoProject()
{
      __KyuyoProject_SetTable();
      __KyuyoProject_Init();
      __KyuyoProject_Body();
}

//SetTable 関数定義
void KyuyoProject::__KyuyoProject_SetTable()
{
      __KyuyoProject_Map[0] = "-:X";
      __KyuyoProject_Map[1] = '¥0';
}

//Init 関数定義
void KyuyoProject::__KyuyoProject_Init()
{
      //初期処理部

}

//Body 関数定義
void KyuyoProject::__KyuyoProject_Body()
{
      //宣言部

      char _currCondVal[2] = "";

_DOAGAIN:

      //前処理部



      //条件値の設定
      _currCondVal[0] = (1) ? 'Y' : 'N';
```

```
	_currCondVal[1] = '¥0';

	//条件に合う行の検索
	int _rule = _Search(__KyuyoProject_Map,_currCondVal);

	//処理部の実行
	if (__KyuyoProject_Map[_rule][2] == 'X') {
		kyuyo.master->_Close();
		kyuyo.overtime->_Close();
		kyuyo.newmaster->_Close();
		kyuyo.kyuyolist->_Close();
	}
}

//CInputFile クラスからの派生クラスのメンバ関数定義

void MasterFile::_Read()
{
	_P_Key.key_1 = _C_Key.key_1;
	_C_Key.key_1 = _N_Key.key_1;
	_N_Key.key_1 = '¥0';
	DepartmentID = _C_Key.key_1;
	_P_Key.key_2 = _C_Key.key_2;
	_C_Key.key_2 = _N_Key.key_2;
	_N_Key.key_2 = 0;
	MemberNo = _C_Key.key_2;
	char temp1[256];
	str1 = _N_str1;
	char temp2[256];
	char temp3[256];
	str2 = _N_str2;
	char temp5[256];
	str3 = _N_str3;
	BasicPay = _N_BasicPay;
	char temp7[256];
	str4 = _N_str4;
	UnitPrice = _N_UnitPrice;
	char temp9[256];
	str5 = _N_str5;
	TotalPay = _N_TotalPay;
	if (_N_EOF_flag == TRUE) _EOF_flag = TRUE;
	if (fscanf(_fp,"%s %s %s %d %s %ld %s %d %s %ld
",temp1,temp2,temp3,&_N_Key.key_2,temp5,&_N_BasicPay,temp7,&_N_UnitPrice,temp9,&_N_
TotalPay) == EOF) {
```

```
            _N_EOF_flag = TRUE;
            temp1[0] = '¥0';
            temp2[0] = '¥0';
            temp3[0] = '¥0';
            _N_Key.key_2 = 0;
            temp5[0] = '¥0';
            _N_BasicPay = 0;
            temp7[0] = '¥0';
            _N_UnitPrice = 0;
            temp9[0] = '¥0';
            _N_TotalPay = 0;
        }
        _N_str1 = temp1;
        _N_Key.key_1 = temp2;
        _N_str2 = temp3;
        _N_str3 = temp5;
        _N_str4 = temp7;
        _N_str5 = temp9;
}

//CInputFile クラスからの派生クラスのメンバ関数定義

void OvertimeFile::_Read()
{
        _P_Key.key_1 = _C_Key.key_1;
        _C_Key.key_1 = _N_Key.key_1;
        _N_Key.key_1 = '¥0';
        DepartmentID = _C_Key.key_1;
        _P_Key.key_2 = _C_Key.key_2;
        _C_Key.key_2 = _N_Key.key_2;
        _N_Key.key_2 = 0;
        MemberNo = _C_Key.key_2;
        char temp1[256];
        str1 = _N_str1;
        char temp2[256];
        char temp3[256];
        str2 = _N_str2;
        char temp5[256];
        str3 = _N_str3;
        Kubun = _N_Kubun;
        char temp7[256];
        str4 = _N_str4;
        Overtime = _N_Overtime;
        if (_N_EOF_flag == TRUE) _EOF_flag = TRUE;
```

```
     if (fscanf(_fp,"%s %s %s %d %s %d %s %d
",temp1,temp2,temp3,&_N_Key.key_2,temp5,&_N_Kubun,temp7,&_N_Overtime) == EOF) {
          _N_EOF_flag = TRUE;
          temp1[0] = '¥0';
          temp2[0] = '¥0';
          temp3[0] = '¥0';
          _N_Key.key_2 = 0;
          temp5[0] = '¥0';
          _N_Kubun = 0;
          temp7[0] = '¥0';
          _N_Overtime = 0;
     }
     _N_str1 = temp1;
     _N_Key.key_1 = temp2;
     _N_str2 = temp3;
     _N_str3 = temp5;
     _N_str4 = temp7;
}

//関数定義：ETD より
void main(void){
     KyuyoProject*     __project = new KyuyoProject;
     __project->start();
     delete __project;
}
```

図 1-92 KyuyoProject.cpp ファイル

```
#ifndef KYUYOPROJECT_H
#define KYUYOPROJECT_H 1

#include "space.h"

/////プロトタイプ宣言
class KyuyoClass;
class NewMasterFile;
class KyuyoList;
class KyuyoProject;
class MasterFile;
class OvertimeFile;

class KyuyoClass : public CLogicTable {
/////declare attribute
```

```
private:
protected:
public:
     long OvertimePay;
     long Zangyo;
     long Kyuyo;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:
     MasterFile*     master;
     OvertimeFile*     overtime;
     NewMasterFile*     newmaster;
     KyuyoList*     kyuyolist;

/////declare methods
private:

          void     _KyuyoKeisan_Body();
          void     _KyuyoKeisan_Init();
          void     _KyuyoKeisan_SetTable();
          CString     _KyuyoKeisan_Map[7];
          CKeyCode     _KyuyoKeisan_MC_Key;
          void     _ZangyoKeisan_Body();
          void     _ZangyoKeisan_Init();
          void     _ZangyoKeisan_SetTable();
          CString     _ZangyoKeisan_Map[9];
          int     _ZangyoKeisan_CB_NextSt;
          char     _ZangyoKeisan_CB_Condition[_ALLKEY+1];
          void     _ZangyoKeisan_CB();
          char     _ZangyoKeisan_SelectValue(int);
          void     _BuGoukei_Body();
          void     _BuGoukei_Init();
          void     _BuGoukei_SetTable();
          CString     _BuGoukei_Map[7];
protected:

public:
```

```
          void     KyuyoKeisan();
          void     ZangyoKeisan();
          void     BuGoukei();

};

class NewMasterFile : public CLTFile {
/////declare attribute
private:
protected:
public:
     CString DepartmentID;
     int MemberNo;
     long BasicPay;
     int UnitPrice;
     long TotalPay;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class KyuyoList : public CLTFile {
/////declare attribute
private:
protected:
public:
     CString DepartmentID;
     int MemberNo;
     long Kyuyo;
```

```
     long DepartmentTotal;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class MasterFile : public CLTFile {
/////declare attribute
private:
protected:
public:
     CString str1;
     CString DepartmentID;
     CString str2;
     int MemberNo;
     CString str3;
     long BasicPay;
     CString str4;
     int UnitPrice;
     CString str5;
     long TotalPay;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
```

```
protected:
public:

/////declare methods
private:

protected:

public:

          void     _Read();
          void     _SetMask();
          CString     _N_str1;
          CString     _N_str2;
          CString     _N_str3;
          long     _N_BasicPay;
          CString     _N_str4;
          int     _N_UnitPrice;
          CString     _N_str5;
          long     _N_TotalPay;
};

class OvertimeFile : public CLTFile {
/////declare attribute
private:
protected:
public:
     CString str1;
     CString DepartmentID;
     CString str2;
     int MemberNo;
     CString str3;
     int Kubun;
     CString str4;
     int Overtime;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
```

```
public:

/////declare methods
private:

protected:

public:

          void     _Read();
          void     _SetMask();
          CString     _N_str1;
          CString     _N_str2;
          CString     _N_str3;
          int     _N_Kubun;
          CString     _N_str4;
          int     _N_Overtime;
};

class KyuyoProject : public CLogicTable {
/////declare attribute
private:
protected:
public:

/////declare aggregation
private:
protected:
public:
     MasterFile     masterfile;
     OvertimeFile     overtimefile;
     KyuyoClass     kyuyo;
     NewMasterFile     newmasterfile;
     KyuyoList     kyuyolist;

/////declare association
private:
protected:
public:

/////declare methods
private:

          void     _start_Body();
```

```
          void     _start_Init();
          void     _start_SetTable();
          CString     _start_Map[10];
          void     _KyuyoProject_Body();
          void     _KyuyoProject_Init();
          void     _KyuyoProject_SetTable();
          CString     _KyuyoProject_Map[10];
          void     __KyuyoProject_Body();
          void     __KyuyoProject_Init();
          void     __KyuyoProject_SetTable();
          CString     __KyuyoProject_Map[10];
protected:

public:
          void     start();
               KyuyoProject();
               ~KyuyoProject();

};

#endif
```

図 1-93 KyuyoProject.h ファイル

1.2.1.4. 生成プログラムの実行結果

入力ファイルは in_overtime ファイル、in_master ファイルの２つである。出力ファイルは Out_KyuyoList ファイル、Out_NewMaster ファイルの２つである。

```
部 No  1 社員 No  1 基本給  25000 単価 1200 合計      0
部 No  1 社員 No  2 基本給  30000 単価 1000 合計      0
部 No  1 社員 No  3 基本給  19000 単価 1300 合計  10000
部 No  2 社員 No  1 基本給  50000 単価 3000 合計      0
部 No  3 社員 No  1 基本給  45000 単価 2500 合計   5000
部 No  3 社員 No  2 基本給  23000 単価 2300 合計   3000
```

図 1-94 in_master ファイル

```
部 No  1 社員 No  1 区分 1 残業時間 3
部 No  1 社員 No  3 区分 3 残業時間 2
部 No  3 社員 No  2 区分 2 残業時間 4
```

図 1-95 in_master ファイル

```
部No  1 社員No  1 給料  28600
部No  1 社員No  2 給料  30000
部No  1 社員No  3 給料  22120 部合計  80720
部No  2 社員No  1 給料  50000 部合計  50000
部No  3 社員No  1 給料  45000
部No  3 社員No  2 給料  36800 部合計  81800
```

図 1-96 Out_KyuyoList ファイル

```
部No  1 社員No  1 基本給  25000 単価 1200 合計  28600
部No  1 社員No  2 基本給  30000 単価 1000 合計  30000
部No  1 社員No  3 基本給  19000 単価 1300 合計  32120
部No  2 社員No  1 基本給  50000 単価 3000 合計  50000
部No  3 社員No  1 基本給  45000 単価 2500 合計  50000
部No  3 社員No  2 基本給  23000 単価 2300 合計  39800
```

図 1-97 Out_NewMaster ファイル

1.2.2. 酒倉庫問題

1.2.2.1. 事例の説明

[処理概要]

(1) ある酒類販売会社の倉庫では、毎日数個のコンテナが搬入されてくる。

(2) その内容はビン詰めの酒で、１つのコンテナには１０銘柄まで混載できる。

(3) 扱い銘柄は約１０種類ある。

(4) 倉庫係は、コンテナを受け取りそのまま倉庫に保管し、コンテナに添付されている積荷票を受付係へ手渡す。

(5) また受付係からの出庫指示によって内蔵品を出庫することになっている。

(6) 内蔵品は別のコンテナに詰め替えたり、別の場所に保管することはない。

(7) 空になったコンテナはすぐに搬出される。

(8) 受付係は毎日数１０件の出庫依頼を受けそのつど倉庫係へ出庫指示書を出すことになっている。

(9) 出庫依頼は出庫依頼票または電話によるものとし、1件の依頼では1銘柄のみに限られている。

(10) 受付係は在庫が無いか数量が不足の場合、その旨依頼者に電話連絡し同時に在庫不足リストに記入。

(11) 倉庫係は空になる予定のコンテナを搬出する。

(12) 倉庫内のコンテナ数はできる限り最小にしたいと考えをしているからである。

(13)受付係の仕事（出庫依頼データを入力し、在庫なし連絡、出庫指示書作成および在庫不足リストを作成する）のための計算機プログラムを作成せよ。

［プロセスフロー］

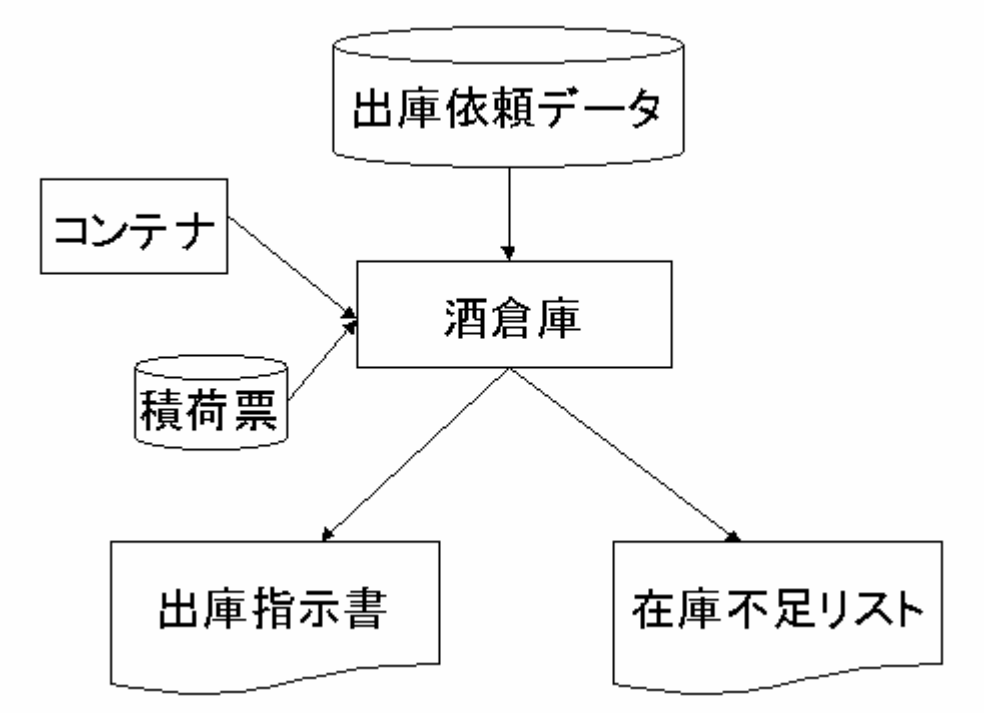


図 1-98 酒倉庫問題のプロセスフロー

［追加処理条件］

(1) まず、初期設定として乱数を用いてコンテナを生成し、コンテナに積荷票を付ける。コンテナを倉庫係に渡す。

(2) 随時、出庫依頼データを基に、依頼者を生成し、受付係に依頼する。

(3) 受付係は依頼に基づき、依頼データの数量に達するまで、積荷票のデータを基に、それが内蔵されている各コンテナから依頼データの数量に達するまで、出庫するための出庫指示書を出力する。依頼データの数量に達する前に、積荷票のデータを全て読み切った場合には、不足分の数量と共に在庫不足リストを出力する。出庫指示書は出庫依頼データ１件ごとに倉庫に送られる。

(4) 倉庫係りは出庫指示書に従って、出庫する。

［データ構造］

(1) 積荷票：
コンテナ番号と搬入年月、日時、{内臓品名、数量} の繰り返しからなる。

(2) 出庫依頼データ：
品名、数量 、送り先名からなる。

(3) 出庫指示書：
{注文番号、送り先名、コンテナ番号、品名、数量、空コンテナ搬出マーク} からなる。

(4) 在庫不足リスト：
送り先名、品名、数量

(5) ［関連する常識］

(6) 酒販売会社は倉庫係と受付係からなる。両者は相互に知っている。

(7) 倉庫係はコンテナを管理している。

(8) 受付係は依頼者を知っている。

(9) 受付係は倉庫係から受け渡された積荷表を管理している。

［シュミレーション問題としてプログラミング上の処理用件の変更］

1. 今回は会社全体の仕事のための、プログラムを作成せよ。
2. 積荷表はコンテナの内容を記したものなので、コンテナには、詳細なデータは持たせなくてよい。

1.2.2.2. 画面

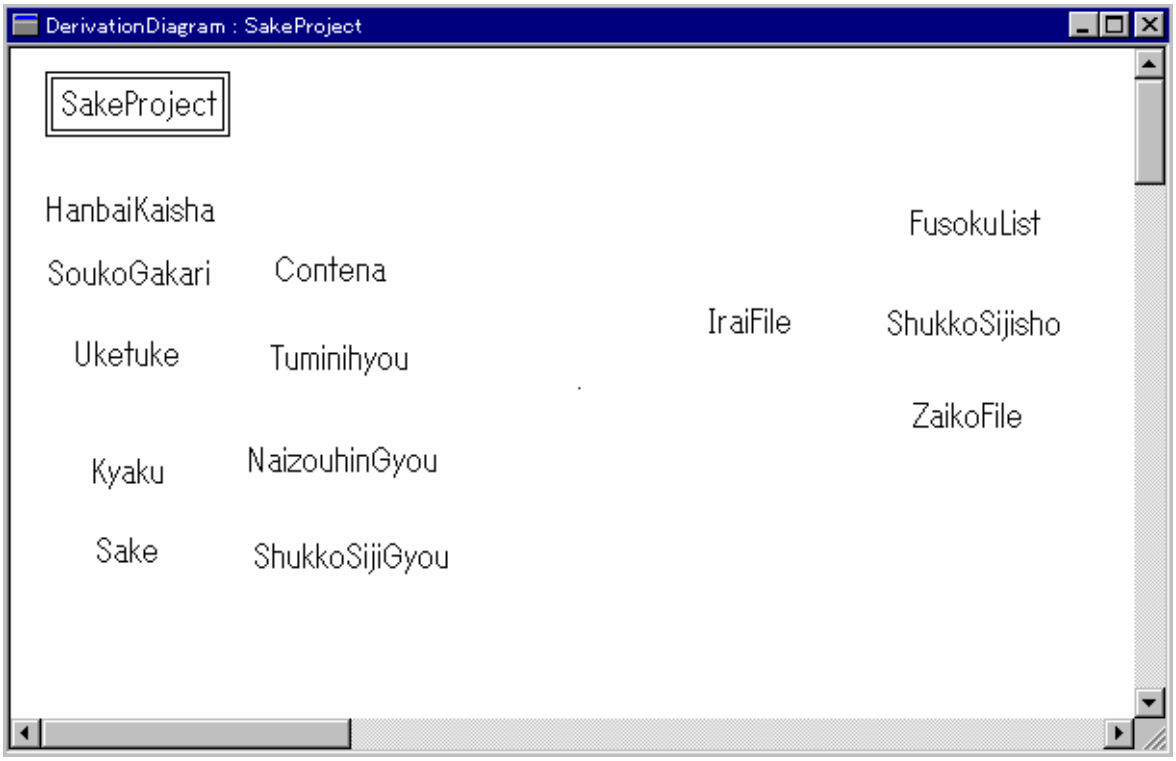


図 1-99 酒倉庫－派生図

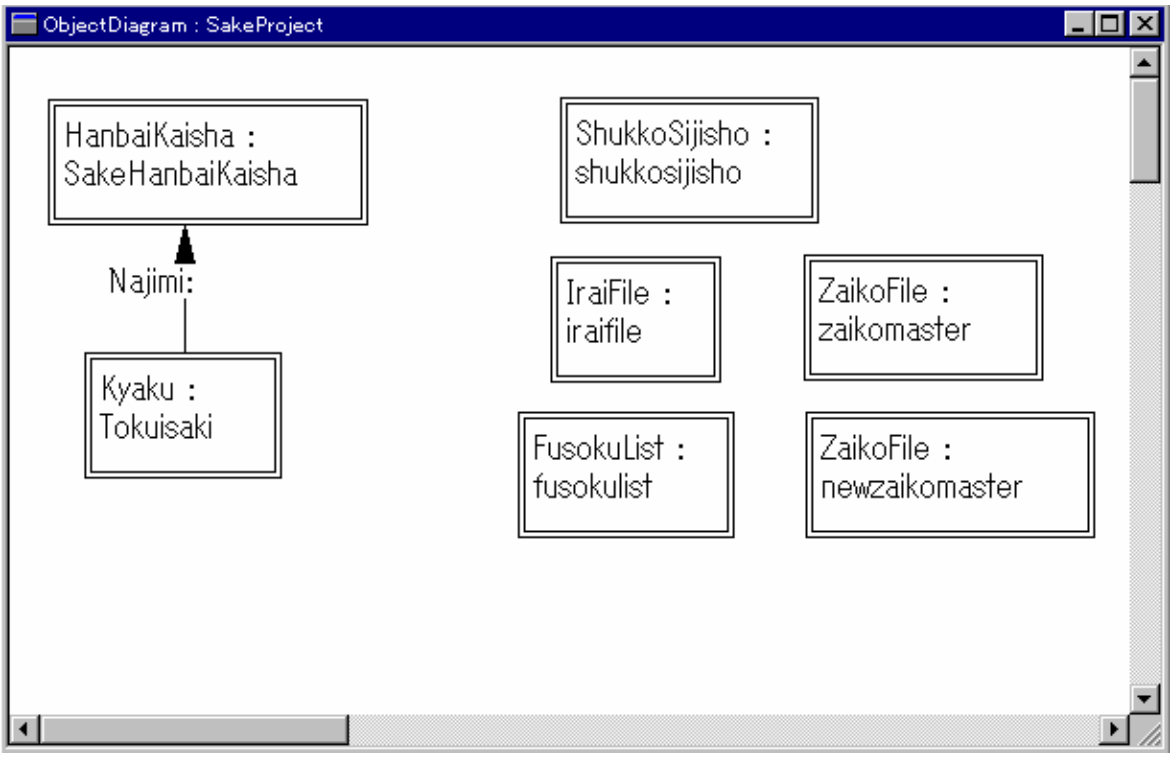


図 1-100 SakeProject クラスのオブジェクト図

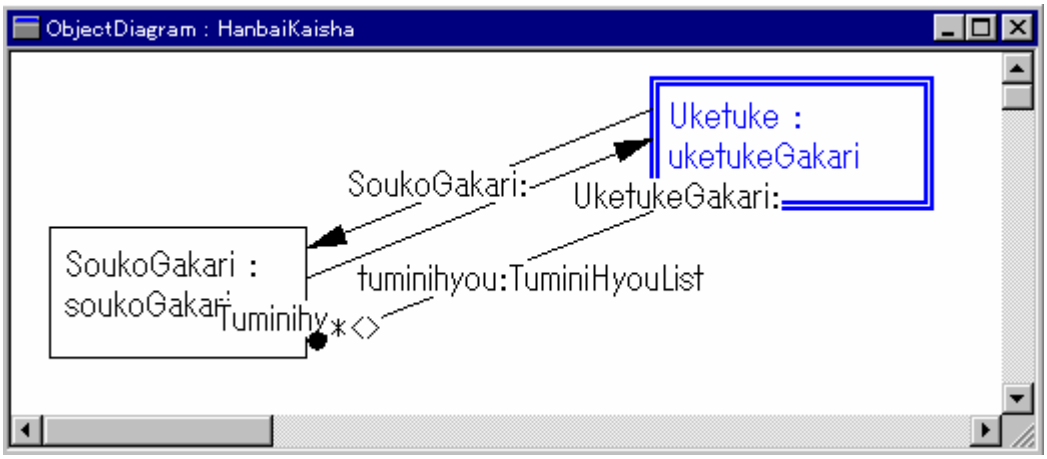


図 1-101 HanbaiKaisha クラスのオブジェクト図

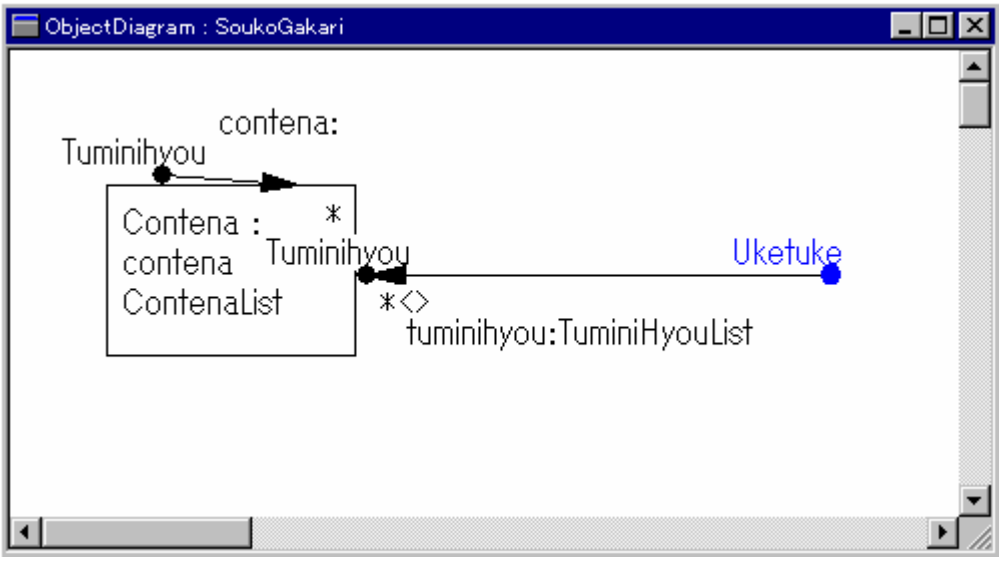


図 1-102 SoukoGakari クラスのオブジェクト図

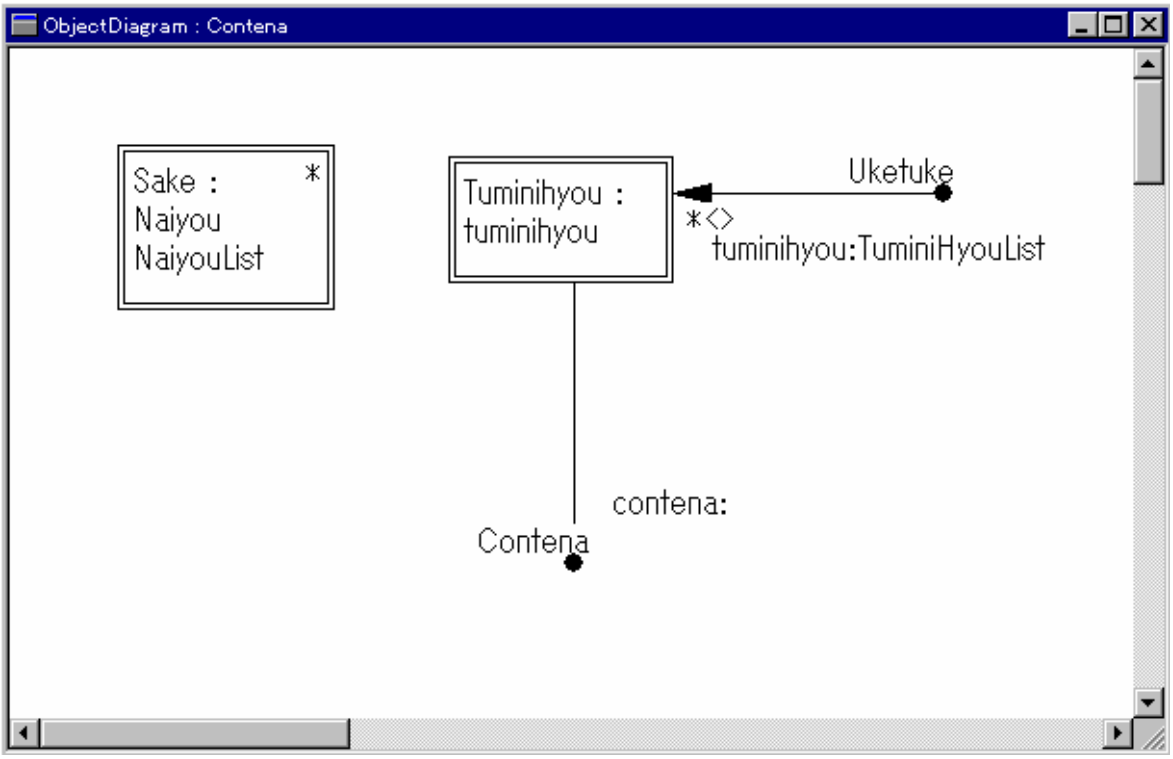


図 1-103 Contena クラスのオブジェクト図

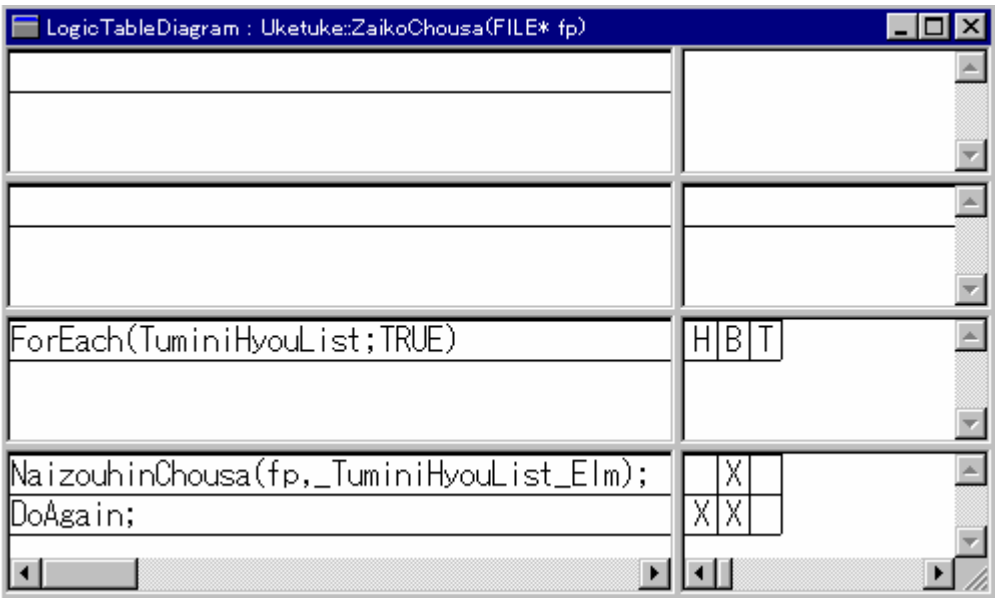


図 1-104 Uketuke::ZaikoChousa(FILE* fp)のロジックテーブル

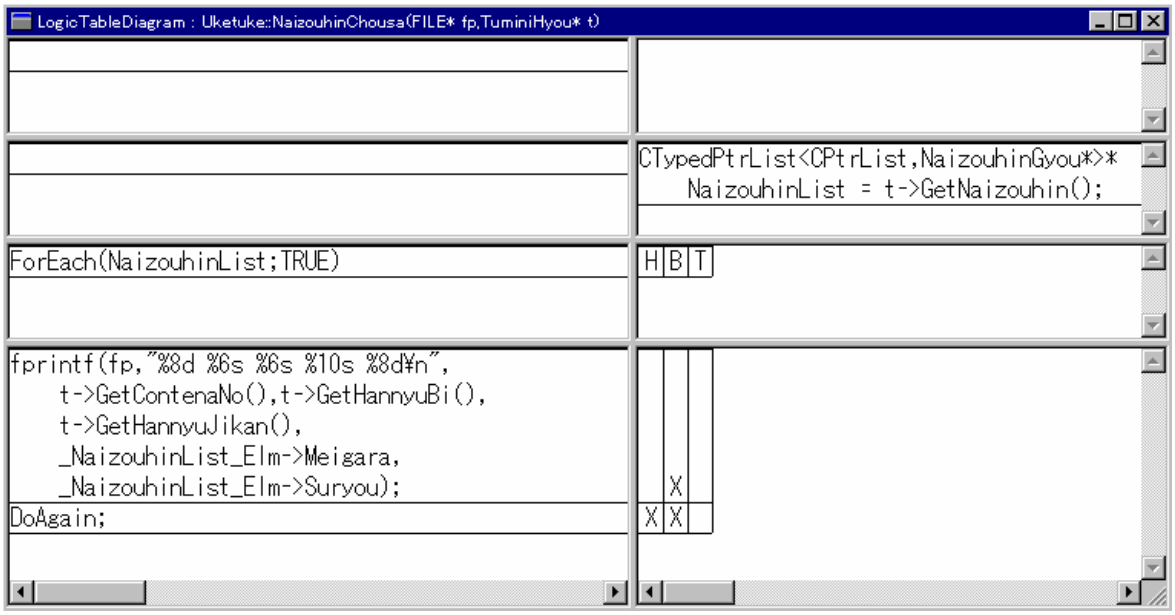


図 1-105 Uketuke::NaizouhinChousa(FILE* fp,TuminiHyou *t)のロジックテーブル

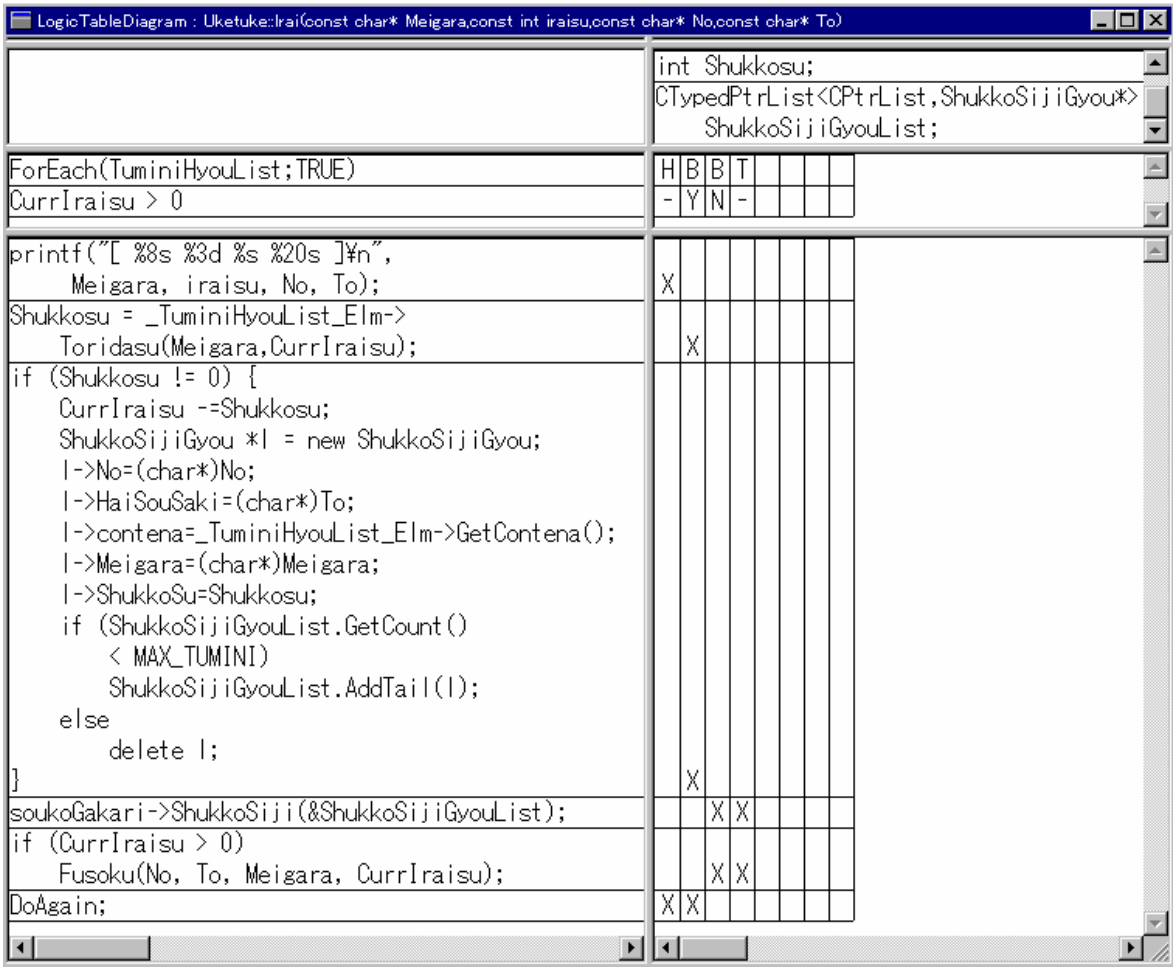


図 1-106 Uketuke::Irai(const char *Meigara, const int iraisu, const char *No, const char *To)のロジックテーブル

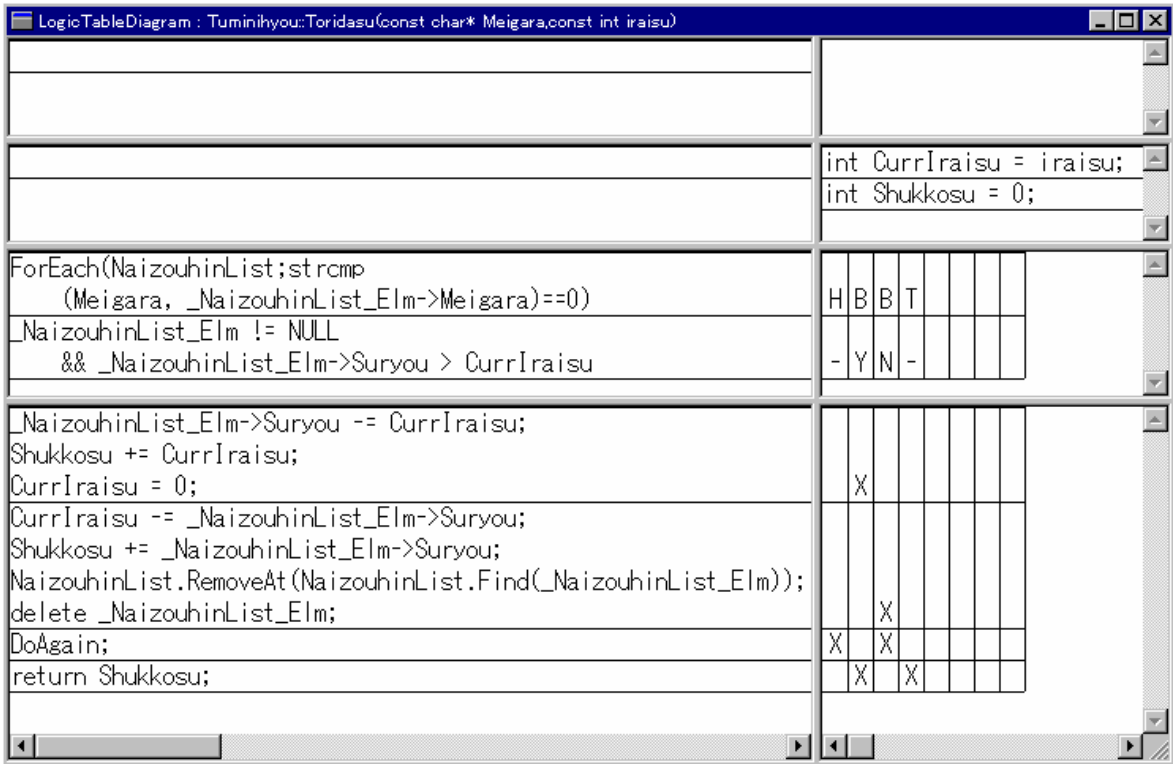


図 1-107 TuminiHyou::Toridasu(const char *Meigara, const int iraisu)の
ロジックテーブル

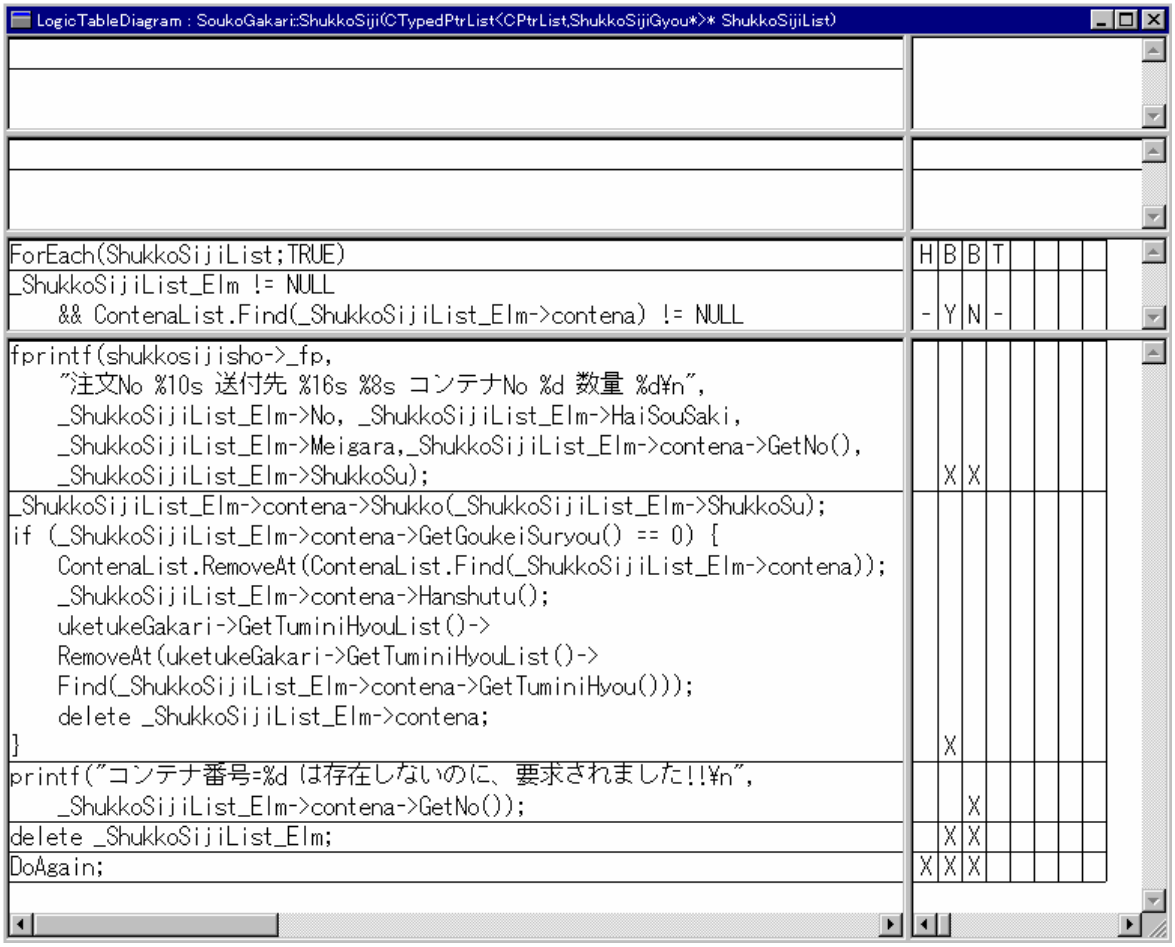


図 1-108 SoukoGakari::ShukkoSiji(CTypedPtrList<CPtrList,ShukkoSijiGyou*>*ShukkoSijiList)のロジックテーブル

1.2.2.3. 生成プログラム

```
#include "space.h"

//CKeyCode クラスのメンバ関数定義

void CKeyCode::operator = (CKeyCode key)
{
}

void CKeyCode::SetKeyMax()
{
}

//_WhichIsSmallKey 関数定義
int CLogicTable::_WhichIsSmallKey(CKeyCode keyA,CKeyCode keyB,CString* Level)
{
     int count = 0;
```

```
     if (count==0) printf("指定されたキーが存在しません！！");
     return _EQUAL;
}
```

図 1-109 Key.cpp ファイル

```
#ifndef ___KEY_H
#define ___KEY_H

#include "stdafx.h"

#define _ALLKEY 0

//CKeyCode クラスの定義
class CKeyCode {
public:

     void operator = (CKeyCode);
     void SetKeyMax();
};

#endif
```

図 1-110 Key.h ファイル

```
#include "SakeProject.h"
//_Contena 関数定義
 Contena::~Contena()
{
     __Contena_SetTable();
     __Contena_Init();
     __Contena_Body();
}

//SetTable 関数定義
void Contena::__Contena_SetTable()
{
     __Contena_Map[0] = "H: X";
     __Contena_Map[1] = "B:XX";
     __Contena_Map[2] = "T:  ";
     __Contena_Map[3] = '¥0';
}
```

```
//Init 関数定義
void Contena::__Contena_Init()
{
     //初期処理部

     __Contena_ForEach_List = 'H';
}

//Body 関数定義
void Contena::__Contena_Body()
{
     //宣言部

     char _currCondVal[2] = "";

     POSITION _pos = NaiyouList.GetHeadPosition();
     Sake* _NaiyouList_Elm = NULL;

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = __Contena_ForEach_List;
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(__Contena_Map,_currCondVal);

     //処理部の実行
     if (__Contena_Map[_rule][2] == 'X') {
          delete _NaiyouList_Elm;
     }
     if (__Contena_Map[_rule][3] == 'X') {
          do {
               if (_pos == NULL) {
                    __Contena_ForEach_List = 'T';
                    break;
               }
               else {
                    __Contena_ForEach_List = 'B';
```

```
                    _NaiyouList_Elm = NaiyouList.GetNext(_pos);
                }
            } while(!(1));
            goto _DOAGAIN;
        }
}

//_TuminiHyou 関数定義
 TuminiHyou::~TuminiHyou()
{
     __TuminiHyou_SetTable();
     __TuminiHyou_Init();
     __TuminiHyou_Body();
}

//SetTable 関数定義
void TuminiHyou::__TuminiHyou_SetTable()
{
     __TuminiHyou_Map[0] = "H: X";
     __TuminiHyou_Map[1] = "B:XX";
     __TuminiHyou_Map[2] = "T:  ";
     __TuminiHyou_Map[3] = '¥0';
}

//Init 関数定義
void TuminiHyou::__TuminiHyou_Init()
{
     //初期処理部

     __TuminiHyou_ForEach_List = 'H';
}

//Body 関数定義
void TuminiHyou::__TuminiHyou_Body()
{
     //宣言部

     char _currCondVal[2] = "";

     POSITION _pos = NaizouhinList.GetHeadPosition();
     NaizouhinGyou* _NaizouhinList_Elm = NULL;

_DOAGAIN:
```

```
     //前処理部



     //条件値の設定
     _currCondVal[0] = __TuminiHyou_ForEach_List;
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(__TuminiHyou_Map,_currCondVal);

     //処理部の実行
     if (__TuminiHyou_Map[_rule][2] == 'X') {
          delete _NaizouhinList_Elm;
     }
     if (__TuminiHyou_Map[_rule][3] == 'X') {
          do {
               if (_pos == NULL) {
                    __TuminiHyou_ForEach_List = 'T';
                    break;
               }
               else {
                    __TuminiHyou_ForEach_List = 'B';
                    _NaizouhinList_Elm = NaizouhinList.GetNext(_pos);
               }
          } while(!(1));
          goto _DOAGAIN;
     }
}

//Toridasu 関数定義
int TuminiHyou::Toridasu(const char* Meigara,const int iraisu)
{
     _Toridasu_SetTable();
     _Toridasu_Init();
     return _Toridasu_Body(Meigara,iraisu);
}

//SetTable 関数定義
void TuminiHyou::_Toridasu_SetTable()
{
     _Toridasu_Map[0] = "H-:  X ";
     _Toridasu_Map[1] = "BY:X  X";
     _Toridasu_Map[2] = "BN: XX ";
```

```
	_Toridasu_Map[3] = "T-:   X";
	_Toridasu_Map[4] = '¥0';
}

//Init 関数定義
void TuminiHyou::_Toridasu_Init()
{
	//初期処理部

	_Toridasu_ForEach_List = 'H';
}

//Body 関数定義
int TuminiHyou::_Toridasu_Body(const char* Meigara,const int iraisu)
{
	//宣言部
	int CurrIraisu = iraisu;
	int Shukkosu = 0;
	char _currCondVal[3] = "";

	POSITION _pos = NaizouhinList.GetHeadPosition();
	NaizouhinGyou* _NaizouhinList_Elm = NULL;

_DOAGAIN:

	//前処理部



	//条件値の設定
	_currCondVal[0] = _Toridasu_ForEach_List;
	_currCondVal[1] = (_NaizouhinList_Elm != NULL && _NaizouhinList_Elm->Suryou >
CurrIraisu) ? 'Y' : 'N';
	_currCondVal[2] = '¥0';

	//条件に合う行の検索
	int _rule = _Search(_Toridasu_Map,_currCondVal);

	//処理部の実行
	if (_Toridasu_Map[_rule][3] == 'X') {
		_NaizouhinList_Elm->Suryou -= CurrIraisu;
		Shukkosu += CurrIraisu;
		CurrIraisu = 0;
	}
```

```
     if (_Toridasu_Map[_rule][4] == 'X') {
          CurrIraisu -= _NaizouhinList_Elm->Suryou;
          Shukkosu += _NaizouhinList_Elm->Suryou;
          NaizouhinList.RemoveAt(NaizouhinList.Find(_NaizouhinList_Elm));
          delete _NaizouhinList_Elm;
     }
     if (_Toridasu_Map[_rule][5] == 'X') {
          do {
               if (_pos == NULL) {
                    _Toridasu_ForEach_List = 'T';
                    break;
               }
               else {
                    _Toridasu_ForEach_List = 'B';
                    _NaizouhinList_Elm = NaizouhinList.GetNext(_pos);
               }
          } while(!(strcmp(Meigara, _NaizouhinList_Elm->Meigara)==0));
          goto _DOAGAIN;
     }
     if (_Toridasu_Map[_rule][6] == 'X') {
          return Shukkosu;
     }
     int _dummy;
     return _dummy;
}

//ZaikoChousa 関数定義
void Uketuke::ZaikoChousa(FILE* fp)
{
     _ZaikoChousa_SetTable();
     _ZaikoChousa_Init();
     _ZaikoChousa_Body(fp);
}

//SetTable 関数定義
void Uketuke::_ZaikoChousa_SetTable()
{
     _ZaikoChousa_Map[0] = "H: X";
     _ZaikoChousa_Map[1] = "B:XX";
     _ZaikoChousa_Map[2] = "T:  ";
     _ZaikoChousa_Map[3] = '¥0';
}

//Init 関数定義
```

```
void Uketuke::_ZaikoChousa_Init()
{
     //初期処理部

     _ZaikoChousa_ForEach_List = 'H';
}

//Body 関数定義
void Uketuke::_ZaikoChousa_Body(FILE* fp)
{
     //宣言部

     char _currCondVal[2] = "";

     POSITION _pos = TuminiHyouList.GetHeadPosition();
     TuminiHyou* _TuminiHyouList_Elm = NULL;

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = _ZaikoChousa_ForEach_List;
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_ZaikoChousa_Map,_currCondVal);

     //処理部の実行
     if (_ZaikoChousa_Map[_rule][2] == 'X') {
          NaizouhinChousa(fp,_TuminiHyouList_Elm);
     }
     if (_ZaikoChousa_Map[_rule][3] == 'X') {
          do {
               if (_pos == NULL) {
                    _ZaikoChousa_ForEach_List = 'T';
                    break;
               }
               else {
                    _ZaikoChousa_ForEach_List = 'B';
                    _TuminiHyouList_Elm = TuminiHyouList.GetNext(_pos);
               }
```

```
          } while(!(TRUE));
          goto _DOAGAIN;
     }
}

//NaizouhinChousa 関数定義
void Uketuke::NaizouhinChousa(FILE* fp,TuminiHyou* t)
{
     _NaizouhinChousa_SetTable();
     _NaizouhinChousa_Init();
     _NaizouhinChousa_Body(fp,t);
}

//SetTable 関数定義
void Uketuke::_NaizouhinChousa_SetTable()
{
     _NaizouhinChousa_Map[0] = "H: X";
     _NaizouhinChousa_Map[1] = "B:XX";
     _NaizouhinChousa_Map[2] = "T:  ";
     _NaizouhinChousa_Map[3] = '¥0';
}

//Init 関数定義
void Uketuke::_NaizouhinChousa_Init()
{
     //初期処理部

     _NaizouhinChousa_ForEach_List = 'H';
}

//Body 関数定義
void Uketuke::_NaizouhinChousa_Body(FILE* fp,TuminiHyou* t)
{
     //宣言部
     CTypedPtrList<CPtrList,NaizouhinGyou*>* NaizouhinList = t->GetNaizouhin();
     char _currCondVal[2] = "";

     POSITION _pos = NaizouhinList->GetHeadPosition();
     NaizouhinGyou* _NaizouhinList_Elm = NULL;

_DOAGAIN:

     //前処理部
```

```
     //条件値の設定
     _currCondVal[0] = _NaizouhinChousa_ForEach_List;
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_NaizouhinChousa_Map,_currCondVal);

     //処理部の実行
     if (_NaizouhinChousa_Map[_rule][2] == 'X') {

fprintf(fp,"%8d %12s %6s %10s %8d¥n",t->GetContenaNo(),t->GetHannyuBi(),t->GetHanny
uJikan(),_NaizouhinList_Elm->Meigara,_NaizouhinList_Elm->Suryou);
     }
     if (_NaizouhinChousa_Map[_rule][3] == 'X') {
          do {
               if (_pos == NULL) {
                    _NaizouhinChousa_ForEach_List = 'T';
                    break;
               }
               else {
                    _NaizouhinChousa_ForEach_List = 'B';
                    _NaizouhinList_Elm = NaizouhinList->GetNext(_pos);
               }
          } while(!(TRUE));
          goto _DOAGAIN;
     }
}

//Irai 関数定義
void Uketuke::Irai(const char* Meigara,const int iraisu,const char* No,const char* To)
{
     _Irai_SetTable();
     _Irai_Init();
     _Irai_Body(Meigara,iraisu,No,To);
}

//SetTable 関数定義
void Uketuke::_Irai_SetTable()
{
     _Irai_Map[0] = "H-:X    X";
     _Irai_Map[1] = "BY: XX  X";
     _Irai_Map[2] = "BN:   XX ";
```

```
	_Irai_Map[3] = "T-:   XX ";
	_Irai_Map[4] = '¥0';
}

//Init 関数定義
void Uketuke::_Irai_Init()
{
	//初期処理部

	_Irai_ForEach_List = 'H';
}

//Body 関数定義
void Uketuke::_Irai_Body(const char* Meigara,const int iraisu,const char* No,const
char* To)
{
	//宣言部
	int CurrIraisu = iraisu;
	int Shukkosu;
	CTypedPtrList<CPtrList,ShukkoSijiGyou*> ShukkoSijiGyouList;
	char _currCondVal[3] = "";

	POSITION _pos = TuminiHyouList.GetHeadPosition();
	TuminiHyou* _TuminiHyouList_Elm = NULL;

_DOAGAIN:

	//前処理部



	//条件値の設定
	_currCondVal[0] = _Irai_ForEach_List;
	_currCondVal[1] = (CurrIraisu > 0) ? 'Y' : 'N';
	_currCondVal[2] = '¥0';

	//条件に合う行の検索
	int _rule = _Search(_Irai_Map,_currCondVal);

	//処理部の実行
	if (_Irai_Map[_rule][3] == 'X') {
		printf("[ %8s %3d %s %20s ]¥n", Meigara, iraisu, No, To);
	}
	if (_Irai_Map[_rule][4] == 'X') {
```

```
          Shukkosu = _TuminiHyouList_Elm->Toridasu(Meigara,CurrIraisu);
     }
     if (_Irai_Map[_rule][5] == 'X') {
          if (Shukkosu != 0) {
               CurrIraisu -=Shukkosu;
               ShukkoSijiGyou *l = new ShukkoSijiGyou;
               l->No=(char*)No;
               l->HaiSouSaki=(char*)To;
               l->contena=_TuminiHyouList_Elm->GetContena();
               l->Meigara=(char*)Meigara;
               l->ShukkoSu=Shukkosu;
               if (ShukkoSijiGyouList.GetCount() < MAX_TUMINI)
                    ShukkoSijiGyouList.AddTail(l);
               else
                    delete l;
          }
     }
     if (_Irai_Map[_rule][6] == 'X') {
          soukoGakari->ShukkoSiji(&ShukkoSijiGyouList);
     }
     if (_Irai_Map[_rule][7] == 'X') {
          if (CurrIraisu > 0) Fusoku(No, To, Meigara, CurrIraisu);
     }
     if (_Irai_Map[_rule][8] == 'X') {
          do {
               if (_pos == NULL) {
                    _Irai_ForEach_List = 'T';
                    break;
               }
               else {
                    _Irai_ForEach_List = 'B';
                    _TuminiHyouList_Elm = TuminiHyouList.GetNext(_pos);
               }
          } while(!(TRUE));
          goto _DOAGAIN;
     }
}

//CInputFile クラスからの派生クラスのメンバ関数定義

void IraiFile::_Read()
{
     strcpy(Meigara,_N_Meigara);
     Suryo = _N_Suryo;
```

```
     strcpy(ChumonNo,_N_ChumonNo);
     strcpy(Okurisaki,_N_Okurisaki);
     if (_N_EOF_flag == TRUE) _EOF_flag = TRUE;
     if (fscanf(_fp,"%s %d %s %s ",&_N_Meigara,&_N_Suryo,&_N_ChumonNo,&_N_Okurisaki)
== EOF) {
          _N_EOF_flag = TRUE;
          _N_Meigara[0] = '¥0';
          _N_Suryo = 0;
          _N_ChumonNo[0] = '¥0';
          _N_Okurisaki[0] = '¥0';
     }
}

//_SoukoGakari 関数定義
 SoukoGakari::~SoukoGakari()
{
     __SoukoGakari_SetTable();
     __SoukoGakari_Init();
     __SoukoGakari_Body();
}

//SetTable 関数定義
void SoukoGakari::__SoukoGakari_SetTable()
{
     __SoukoGakari_Map[0] = "H: X";
     __SoukoGakari_Map[1] = "B:XX";
     __SoukoGakari_Map[2] = "T:  ";
     __SoukoGakari_Map[3] = '¥0';
}

//Init 関数定義
void SoukoGakari::__SoukoGakari_Init()
{
     //初期処理部

     __SoukoGakari_ForEach_List = 'H';
}

//Body 関数定義
void SoukoGakari::__SoukoGakari_Body()
{
     //宣言部

     char _currCondVal[2] = "";
```

```
     POSITION _pos = ContenaList.GetHeadPosition();
     Contena* _ContenaList_Elm = NULL;

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = __SoukoGakari_ForEach_List;
     _currCondVal[1] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(__SoukoGakari_Map,_currCondVal);

     //処理部の実行
     if (__SoukoGakari_Map[_rule][2] == 'X') {
          delete _ContenaList_Elm;
     }
     if (__SoukoGakari_Map[_rule][3] == 'X') {
          do {
               if (_pos == NULL) {
                    __SoukoGakari_ForEach_List = 'T';
                    break;
               }
               else {
                    __SoukoGakari_ForEach_List = 'B';
                    _ContenaList_Elm = ContenaList.GetNext(_pos);
               }
          } while(!(1));
          goto _DOAGAIN;
     }
}

//ShukkoSiji 関数定義
void SoukoGakari::ShukkoSiji(CTypedPtrList<CPtrList,ShukkoSijiGyou*>*
ShukkoSijiList)
{
     _ShukkoSiji_SetTable();
     _ShukkoSiji_Init();
     _ShukkoSiji_Body(ShukkoSijiList);
}
```

```
//SetTable 関数定義
void SoukoGakari::_ShukkoSiji_SetTable()
{
     _ShukkoSiji_Map[0] = "H-:    X";
     _ShukkoSiji_Map[1] = "BY:XX XX";
     _ShukkoSiji_Map[2] = "BN:X XXX";
     _ShukkoSiji_Map[3] = "T-:     ";
     _ShukkoSiji_Map[4] = '¥0';
}

//Init 関数定義
void SoukoGakari::_ShukkoSiji_Init()
{
     //初期処理部

     _ShukkoSiji_ForEach_List = 'H';
}

//Body 関数定義
void SoukoGakari::_ShukkoSiji_Body(CTypedPtrList<CPtrList,ShukkoSijiGyou*>*
ShukkoSijiList)
{
     //宣言部

     char _currCondVal[3] = "";

     POSITION _pos = ShukkoSijiList->GetHeadPosition();
     ShukkoSijiGyou* _ShukkoSijiList_Elm = NULL;

_DOAGAIN:

     //前処理部



     //条件値の設定
     _currCondVal[0] = _ShukkoSiji_ForEach_List;
     _currCondVal[1] = (_ShukkoSijiList_Elm != NULL &&
ContenaList.Find(_ShukkoSijiList_Elm->contena) != NULL) ? 'Y' : 'N';
     _currCondVal[2] = '¥0';

     //条件に合う行の検索
     int _rule = _Search(_ShukkoSiji_Map,_currCondVal);
```

```
     //処理部の実行
     if (_ShukkoSiji_Map[_rule][3] == 'X') {
          fprintf(shukkosijisho->_fp,"注文No %10s 送付先 %16s %8s コンテナNo %d 数
量 %d¥n",
               _ShukkoSijiList_Elm->No,
_ShukkoSijiList_Elm->HaiSouSaki,_ShukkoSijiList_Elm->Meigara,

_ShukkoSijiList_Elm->contena->GetNo(),_ShukkoSijiList_Elm->ShukkoSu);
     }
     if (_ShukkoSiji_Map[_rule][4] == 'X') {
          _ShukkoSijiList_Elm->contena->Shukko(_ShukkoSijiList_Elm->ShukkoSu);
          if (_ShukkoSijiList_Elm->contena->GetGoukeiSuryou() == 0) {

ContenaList.RemoveAt(ContenaList.Find(_ShukkoSijiList_Elm->contena));
               _ShukkoSijiList_Elm->contena->Hanshutu();

uketukeGakari->GetTuminiHyouList()->RemoveAt(uketukeGakari->GetTuminiHyouList()->Fi
nd(_ShukkoSijiList_Elm->contena->GetTuminiHyou()));
               delete _ShukkoSijiList_Elm->contena;
          }
     }
     if (_ShukkoSiji_Map[_rule][5] == 'X') {
          printf("コンテナ番号=%d は存在しないのに、要求されまし
た!!¥n",_ShukkoSijiList_Elm->contena->GetNo());
     }
     if (_ShukkoSiji_Map[_rule][6] == 'X') {
          delete _ShukkoSijiList_Elm;
     }
     if (_ShukkoSiji_Map[_rule][7] == 'X') {
          do {
               if (_pos == NULL) {
                    _ShukkoSiji_ForEach_List = 'T';
                    break;
               }
               else {
                    _ShukkoSiji_ForEach_List = 'B';
                    _ShukkoSijiList_Elm = ShukkoSijiList->GetNext(_pos);
               }
          } while(!(TRUE));
          goto _DOAGAIN;
     }
}
```

```
//関数定義：ETD より
void Contena::Tumikomi(Sake* sake,const int x){
     if(NaiyouList.GetCount() < MAX_TUMINI) {
          NaiyouList.AddTail(sake);
     }
     GoukeiSuryou += x;
     if(NaiyouList.GetCount() < MAX_TUMINI) {
          NaizouhinGyou* newLine = new NaizouhinGyou;
          strcpy(newLine->Meigara, sake->GetName());
          newLine->Suryou = x;
          if(tuminihyou.GetNaizouhin()->GetCount() < MAX_TUMINI) {
               tuminihyou.GetNaizouhin()->AddTail(newLine);
          }
          else {
               delete newLine;
          }
     }
}

 Contena::Contena(const int n) : tuminihyou(this,n) {
     ContenaNo = n;
     GoukeiSuryou = 0;
     int     TsuminiSu     = rnd(MAX_TUMINI)+1;
     for(int i=0;i<TsuminiSu;i++) {
          Sake *s     = new Sake();
          Tumikomi(s,rnd(MAX_SURYOU)+1);
     }
}

int Contena::GetNo(){
     return ContenaNo;
}

int Contena::GetGoukeiSuryou(){
     return GoukeiSuryou;
}

TuminiHyou* Contena::GetTuminiHyou(){
     return &tuminihyou;
}

void Contena::Shukko(int x){
     GoukeiSuryou -= x;
}
```

```
void Contena::Hanshutu(){
     printf("コンテナ番号=%d は空になったので搬出されました。¥n",ContenaNo);
}

 Kyaku::Kyaku(HanbaiKaisha* kaisha){
     Najimi = kaisha;
}

BOOL Kyaku::Iraidasi(IraiFile* irai){
     if (irai->_EOF_flag) return FALSE;

Najimi->GetUketuke()->Irai(irai->Meigara,irai->Suryo,irai->ChumonNo,irai->Okurisaki
);
     irai->_Read();
     return TRUE;
}

 TuminiHyou::TuminiHyou(Contena* c,const int n){
     char     Day[12], Time[6];
     contena = c;
     ContenaNo = n;
     sprintf(Day, "%d 月%d 日", rnd(12)+1, rnd(30)+1);
     sprintf(Time, "%d:%d", rnd(24), rnd(60));
     strcpy(HannyuBi, Day);
     strcpy(HannyuJikan, Time);
}

char* TuminiHyou::GetHannyuBi(){
     return     HannyuBi;
}

char* TuminiHyou::GetHannyuJikan(){
     return     HannyuJikan;
}

Contena* TuminiHyou::GetContena(){
     return     contena;
}

int TuminiHyou::GetContenaNo(){
     return     ContenaNo;
}
```

```
CTypedPtrList<CPtrList,NaizouhinGyou*>* TuminiHyou::GetNaizouhin(){
     return &NaizouhinList;
}

 Uketuke::Uketuke(SoukoGakari* s){
     soukoGakari     = s;
     s->Set(this);
}

 Uketuke::~Uketuke(){
}

CTypedPtrList<CPtrList,TuminiHyou*>* Uketuke::GetTuminiHyouList(){
     return &TuminiHyouList;
}

void Uketuke::Ukewatasi(TuminiHyou* t){
     if (TuminiHyouList.GetCount() < MAX_CONTENA) TuminiHyouList.AddTail(t);
}

void Uketuke::Fusoku(const char* No,const char* To,const char* Meigara,const int Kazu){
     fprintf(fusokulist->_fp,"%10s %20s %10s %4d¥n",No,To,Meigara,Kazu);
}

 Sake::Sake(){
     char *Meigara[] =
{"BEER","BOURBON","BRANDY","JIN","KEEL","RAM","SCOTCH","TEQUILA","WHISKEY","WINE",}
;
     strcpy(Name, Meigara[rnd(MAX_TUMINI)]);
}

char* Sake::GetName(){
     return Name;
}

void SakeProject::start(){

     SakeHanbaiKaisha.GetUketuke()->ZaikoChousa(zaikomaster._fp);



     int kennsuu=0;
     int flag = TRUE;
     while(flag) {
```

```
          Kyaku     *TokuiSaki     = new     Kyaku(&SakeHanbaiKaisha);
          flag = TokuiSaki->Iraidasi(&iraifile);
          delete TokuiSaki;
          if (flag == TRUE) kennsuu++;
     }

     SakeHanbaiKaisha.GetUketuke()->ZaikoChousa(newzaikomaster._fp);

     printf("処理終了 件数 = %d¥n", kennsuu);
}

 SakeProject::SakeProject(){

     iraifile._Open("data¥¥shuko-ir.dat","r");
     zaikomaster._Open("data¥¥Zaiko.dat","w");
     newzaikomaster._Open("data¥¥New-Zaiko.dat","w");
     shukkosijisho._Open("data¥¥Shukko.dat","w");
     fusokulist._Open("data¥¥Fusoku-List.dat","w");


     SakeHanbaiKaisha.GetsoukoGakari()->shukkosijisho = &shukkosijisho;
     SakeHanbaiKaisha.GetUketuke()->fusokulist = &fusokulist;
}

 SakeProject::~SakeProject(){

     iraifile._Close();
     zaikomaster._Close();
     newzaikomaster._Close();
     shukkosijisho._Close();
     fusokulist._Close();
}

 SoukoGakari::SoukoGakari(){
     for(int i=1;i<=MAX_CONTENA;i++) {
          Contena *c = new Contena(i);
          Hannyu(c);
     }
}

void SoukoGakari::Set(Uketuke* u){
     uketukeGakari = u;
}
```

```
void SoukoGakari::Hannyu(Contena* contena){
     if (ContenaList.GetCount() < MAX_CONTENA) ContenaList.AddTail(contena);
     uketukeGakari->Ukewatasi(contena->GetTuminiHyou());
}

 HanbaiKaisha::HanbaiKaisha() : uketukeGakari(&soukoGakari){
}

 HanbaiKaisha::~HanbaiKaisha(){
}

SoukoGakari* HanbaiKaisha::GetsoukoGakari(){
     return &soukoGakari;
}

Uketuke* HanbaiKaisha::GetUketuke(){
     return &uketukeGakari;
}

void main(void){
     SakeProject*     __project = new SakeProject;
     __project->start();
     delete __project;
}
```

図 1-111 SakeProject.cpp ファイル

```
#ifndef SAKEPROJECT_H
#define SAKEPROJECT_H 1

#include "space.h"
#define RANDOM_SEED 1
#define MAX_TUMINI 10
#define MAX_SURYOU 20
#define MAX_CONTENA 6
#define MEIGARA_NO_NAGASA 11
#define rnd(r) (rand() % r)

/////プロトタイプ宣言
class Contena;
class NaizouhinGyou;
class Kyaku;
class ShukkoSijiGyou;
```

```
class TuminiHyou;
class ShukkoSijisho;
class Uketuke;
class Sake;
class SakeProject;
class IraiFile;
class SoukoGakari;
class ZaikoMaster;
class HanbaiKaisha;
class FusokuList;

class TuminiHyou : public CLogicTable {
/////declare attribute
private:
     int ContenaNo;
     char HannyuBi[12];
     char HannyuJikan[6];
     CTypedPtrList<CPtrList,NaizouhinGyou*> NaizouhinList;
protected:
public:

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:
     Contena*     contena;

/////declare methods
private:

          void     __TuminiHyou_Body();
          void     __TuminiHyou_Init();
          void     __TuminiHyou_SetTable();
          CString     __TuminiHyou_Map[10];
          char     __TuminiHyou_ForEach_List;
          int     _Toridasu_Body(const char* Meigara,const int iraisu);
          void     _Toridasu_Init();
          void     _Toridasu_SetTable();
          CString     _Toridasu_Map[10];
```

```
          char     _Toridasu_ForEach_List;
protected:

public:
               TuminiHyou(Contena* c,const int n);
               ~TuminiHyou();
          char*     GetHannyuBi();
          char*     GetHannyuJikan();
          Contena*     GetContena();
          int     GetContenaNo();
          CTypedPtrList<CPtrList,NaizouhinGyou*>*     GetNaizouhin();
          int     Toridasu(const char* Meigara,const int iraisu);

};

class Contena : public CLogicTable {
/////declare attribute
private:
     int ContenaNo;
     int GoukeiSuryou;
protected:
public:

/////declare aggregation
private:
protected:
public:
     CTypedPtrList<CPtrList,Sake*> NaiyouList;
     TuminiHyou     tuminihyou;

/////declare association
private:
protected:
public:

/////declare methods
private:
          void     Tumikomi(Sake* sake,const int x);

          void     __Contena_Body();
          void     __Contena_Init();
          void     __Contena_SetTable();
          CString     __Contena_Map[10];
          char     __Contena_ForEach_List;
```

```
protected:

public:
               Contena(const int n);
               ~Contena();
          int     GetNo();
          int     GetGoukeiSuryou();
          TuminiHyou*     GetTuminiHyou();
          void     Shukko(int x);
          void     Hanshutu();

};

class NaizouhinGyou {
/////declare attribute
private:
protected:
public:
     char Meigara[11];
     int Suryou;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class Kyaku {
/////declare attribute
private:
protected:
```

```
public:
     char Name[32];
     HanbaiKaisha* Najimi;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:
               Kyaku(HanbaiKaisha* kaisha);
          BOOL     Iraidasi(IraiFile* irai);

};

class ShukkoSijiGyou {
/////declare attribute
private:
protected:
public:
     char* No;
     char* HaiSouSaki;
     Contena* contena;
     char* Meigara;
     int ShukkoSu;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
```

```
public:

/////declare methods
private:

protected:

public:

};

class ShukkoSijisho : public CLTFile {
/////declare attribute
private:
protected:
public:
     char ChumonNo[20];
     char Okurisaki[30];
     char Meigara[20];
     int ContenaNo;
     int Suryo;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class Uketuke : public CLogicTable {
/////declare attribute
private:
```

```
protected:
public:

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:
     SoukoGakari*     soukoGakari;
     CTypedPtrList<CPtrList,TuminiHyou*> TuminiHyouList;
     FusokuList*     fusokulist;

/////declare methods
private:

          void     _ZaikoChousa_Body(FILE* fp);
          void     _ZaikoChousa_Init();
          void     _ZaikoChousa_SetTable();
          CString  _ZaikoChousa_Map[5];
          char     _ZaikoChousa_ForEach_List;
          void     _NaizouhinChousa_Body(FILE* fp,TuminiHyou* t);
          void     _NaizouhinChousa_Init();
          void     _NaizouhinChousa_SetTable();
          CString  _NaizouhinChousa_Map[5];
          char     _NaizouhinChousa_ForEach_List;
          void     _Irai_Body(const char* Meigara,const int iraisu,const char*
No,const char* To);
          void     _Irai_Init();
          void     _Irai_SetTable();
          CString  _Irai_Map[10];
          char     _Irai_ForEach_List;
protected:

public:
               Uketuke(SoukoGakari* s);
               ~Uketuke();
          CTypedPtrList<CPtrList,TuminiHyou*>*     GetTuminiHyouList();
          void     Ukewatasi(TuminiHyou* t);
          void     Fusoku(const char* No,const char* To,const char* Meigara,const int
Kazu);
```

```
          void      ZaikoChousa(FILE* fp);
          void      NaizouhinChousa(FILE* fp,TuminiHyou* t);
          void      Irai(const char* Meigara,const int iraisu,const char* No,const char*
To);

};

class Sake {
/////declare attribute
private:
protected:
public:
     char Name[11];

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:
               Sake();
          char*     GetName();

};

class ZaikoMaster : public CLTFile {
/////declare attribute
private:
protected:
public:
     char ChumonNo[20];
     char HannyuBi[10];
     char HannyuJikan[10];
     char Meigara[20];
```

```
     int Suryo;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class SoukoGakari : public CLogicTable {
/////declare attribute
private:
protected:
public:

/////declare aggregation
private:
protected:
public:
     CTypedPtrList<CPtrList,Contena*> ContenaList;

/////declare association
private:
protected:
public:
     Uketuke*     uketukeGakari;
     ShukkoSijisho*     shukkosijisho;

/////declare methods
private:

          void     __SoukoGakari_Body();
```

```
          void     __SoukoGakari_Init();
          void     __SoukoGakari_SetTable();
          CString     __SoukoGakari_Map[10];
          char     __SoukoGakari_ForEach_List;
          void     _ShukkoSiji_Body(CTypedPtrList<CPtrList,ShukkoSijiGyou*>*
ShukkoSijiList);
          void     _ShukkoSiji_Init();
          void     _ShukkoSiji_SetTable();
          CString     _ShukkoSiji_Map[10];
          char     _ShukkoSiji_ForEach_List;
protected:

public:
               SoukoGakari();
               ~SoukoGakari();
          void     Set(Uketuke* u);
          void     Hannyu(Contena* contena);
          void     ShukkoSiji(CTypedPtrList<CPtrList,ShukkoSijiGyou*>*
ShukkoSijiList);

};

class HanbaiKaisha {
/////declare attribute
private:
protected:
public:

/////declare aggregation
private:
protected:
public:
     Uketuke     uketukeGakari;
     SoukoGakari     soukoGakari;

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:
```

```
public:
              HanbaiKaisha();
              ~HanbaiKaisha();
         SoukoGakari*     GetsoukoGakari();
         Uketuke*     GetUketuke();

};

class FusokuList : public CLTFile {
/////declare attribute
private:
protected:
public:
     char ChumonNo[20];
     char Okurisaki[30];
     char Meigara[20];
     int Suryo;

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

};

class IraiFile : public CLTFile {
/////declare attribute
private:
protected:
public:
     char Meigara[20];
```

```
     int Suryo;
     char ChumonNo[20];
     char Okurisaki[30];

/////declare aggregation
private:
protected:
public:

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:

          void     _Read();
          void     _SetMask();
          char     _N_Meigara[20];
          int     _N_Suryo;
          char     _N_ChumonNo[20];
          char     _N_Okurisaki[30];
};

class SakeProject {
/////declare attribute
private:
protected:
public:

/////declare aggregation
private:
protected:
public:
     ZaikoMaster     zaikomaster;
     ZaikoMaster     newzaikomaster;
     HanbaiKaisha     SakeHanbaiKaisha;
     FusokuList     fusokulist;
     ShukkoSijisho     shukkosijisho;
```

```
     IraiFile     iraifile;

/////declare association
private:
protected:
public:

/////declare methods
private:

protected:

public:
          void     start();
               SakeProject();
               ~SakeProject();

};

#endif
```

図 1-112 SakeProject.h ファイル

1.2.2.4. 生成プログラムの実行結果

入力ファイルはShuko-irファイルである。出力ファイルはFusoku-Listファイル、New-Zaikoファイル、Shukkoファイル、Zaikoファイルの４つである。

```
WHISKEY    0000000018 AA-0001-01 SakamotoSaketen
BEER       0000000019 AA-0001-02 HaradaSaketen
WINE       0000000028 AA-0001-03 SaitouSaketen
BOURBON    0000000013 AA-0001-04 MatumotoSaketen
TEQUILA    0000000023 AA-0001-05 SawadaSaketen
JIN        0000000030 AA-0002-01 NishimuraSaketen
RAM        0000000038 AA-0002-02 TakedaSaketen
BEER       0000000012 AA-0002-03 IzumidaSaketen
SCOTCH     0000000032 AA-0003-01 MiyakeSaketen
BRANDY     0000000116 AA-0003-02 NakajimaSaketen
KEEL       0000000058 AA-0003-03 NagaiSaketen
WINE       0000000123 AA-0003-04 HaradaSaketen
SAKE       0000000002 AA-0003-05 IzumidaSaketen
```

図 1-113 Shuko-ir ファイル

```
AA-0002-02        TakedaSaketen        RAM   21
AA-0002-03       IzumidaSaketen       BEER    8
AA-0003-02     NakajimaSaketen     BRANDY   98
AA-0003-03        NagaiSaketen       KEEL    4
AA-0003-04       HaradaSaketen       WINE  112
AA-0003-05       IzumidaSaketen       SAKE    2
```

図 1-114 Fusoku-List ファイル

```
    1  12/12   4:34    BOURBON        3
    2  10/23  14:56    BOURBON       19
    2  10/23  14:56    TEQUILA        6
    3   10/3   8:33    BOURBON       14
    4  11/15  21:57     SCOTCH        2
    5   10/7  20:33     SCOTCH       12
    5   10/7  20:33    WHISKEY        5
    5   10/7  20:33        JIN        7
    5   10/7  20:33    WHISKEY       19
    6    8/6   0:18    TEQUILA        7
```

図 1-115 New-Zaiko ファイル

```
注文No AA-0001-01 送付先  SakamotoSaketen  WHISKEY コンテナNo 1 数量 3
注文No AA-0001-01 送付先  SakamotoSaketen  WHISKEY コンテナNo 3 数量 8
注文No AA-0001-01 送付先  SakamotoSaketen  WHISKEY コンテナNo 4 数量 7
注文No AA-0001-02 送付先    HaradaSaketen     BEER コンテナNo 4 数量 16
注文No AA-0001-02 送付先    HaradaSaketen     BEER コンテナNo 5 数量 3
注文No AA-0001-03 送付先    SaitouSaketen     WINE コンテナNo 2 数量 13
注文No AA-0001-03 送付先    SaitouSaketen     WINE コンテナNo 4 数量 15
注文No AA-0001-04 送付先  MatumotoSaketen  BOURBON コンテナNo 1 数量 13
注文No AA-0001-05 送付先    SawadaSaketen  TEQUILA コンテナNo 1 数量 2
注文No AA-0001-05 送付先    SawadaSaketen  TEQUILA コンテナNo 2 数量 21
注文No AA-0002-01 送付先 NishimuraSaketen      JIN コンテナNo 2 数量 12
注文No AA-0002-01 送付先 NishimuraSaketen      JIN コンテナNo 4 数量 2
注文No AA-0002-01 送付先 NishimuraSaketen      JIN コンテナNo 5 数量 16
注文No AA-0002-02 送付先    TakedaSaketen      RAM コンテナNo 1 数量 2
注文No AA-0002-02 送付先    TakedaSaketen      RAM コンテナNo 2 数量 15
注文No AA-0002-03 送付先   IzumidaSaketen     BEER コンテナNo 5 数量 4
注文No AA-0003-01 送付先    MiyakeSaketen   SCOTCH コンテナNo 1 数量 12
注文No AA-0003-01 送付先    MiyakeSaketen   SCOTCH コンテナNo 4 数量 20
注文No AA-0003-02 送付先  NakajimaSaketen   BRANDY コンテナNo 1 数量 8
注文No AA-0003-02 送付先  NakajimaSaketen   BRANDY コンテナNo 5 数量 10
注文No AA-0003-03 送付先     NagaiSaketen     KEEL コンテナNo 1 数量 28
```

```
注文No AA-0003-03 送付先     NagaiSaketen     KEEL コンテナNo 4 数量 9
注文No AA-0003-03 送付先     NagaiSaketen     KEEL コンテナNo 5 数量 17
注文No AA-0003-04 送付先    HaradaSaketen     WINE コンテナNo 4 数量 4
注文No AA-0003-04 送付先    HaradaSaketen     WINE コンテナNo 5 数量 7
```

図 1-116 Shukko ファイル

```
1  12/12   4:34      KEEL      19
1  12/12   4:34   WHISKEY       3
1  12/12   4:34      KEEL       6
1  12/12   4:34       RAM       2
1  12/12   4:34   TEQUILA       2
1  12/12   4:34   BOURBON      16
1  12/12   4:34    BRANDY       8
1  12/12   4:34    SCOTCH      12
1  12/12   4:34      KEEL       3
2  10/23  14:56   TEQUILA       7
2  10/23  14:56   BOURBON      19
2  10/23  14:56      WINE      13
2  10/23  14:56   TEQUILA      20
2  10/23  14:56       RAM      15
2  10/23  14:56       JIN      12
3   10/3   8:33   BOURBON      14
3   10/3   8:33   WHISKEY       8
4  11/15  21:57       JIN       2
4  11/15  21:57      WINE      19
4  11/15  21:57    SCOTCH      16
4  11/15  21:57      BEER       3
4  11/15  21:57   WHISKEY       7
4  11/15  21:57      BEER       3
4  11/15  21:57      KEEL       9
4  11/15  21:57    SCOTCH       6
4  11/15  21:57      BEER      10
5   10/7  20:33       JIN       5
5   10/7  20:33      KEEL      17
5   10/7  20:33      BEER       7
5   10/7  20:33    SCOTCH      12
5   10/7  20:33   WHISKEY       5
5   10/7  20:33      WINE       7
5   10/7  20:33       JIN      18
5   10/7  20:33   WHISKEY      19
5   10/7  20:33    BRANDY      10
6    8/6   0:18   TEQUILA       7
```

図 1-117 Zaiko ファイル