情報処理学会第37回(昭和63年後期)全国大会

印刷レイアウト機能

5X-6

＊(株)東芝　山谷祐二，池田丈男，島岡司郎
＊＊東芝ｿﾌﾄｳｪｱｴﾝｼﾞﾆｱﾘﾝｸﾞ(株) 田中強

1.はじめに

プリンタの高性能化や高機能化に伴い、事務処理用の帳票処理のアプリケーションにおいても、そのような機能の使用の要求が増えてきた。従来は、その手段がないか、またはあってもサブルーチンなどの形で提供されているため、パラメタの指定が複雑であり使われることは、特殊な場合を除きほとんどなかった。

また、帳票処理などにおける行制御や桁送りは、ほとんどをプログラム内で行い、帳票の形式が少し変更になっただけでプログラムの変更を伴い、作業量も大きいものであった。

そこで、プログラムの開発・保守の生産性を向上させ、かつ高度な帳票処理印刷を可能とする「印刷レイアウト機能」を提案する。

2.印刷レイアウトとは

一つの印刷頁（帳票）内に、図１のような階層構造を持ったレイアウト情報の定義を行う。

レイアウトには、共通レイアウトとサブレイアウトがある。出力する帳票が一種類しかないときは共通レイアウトのみで十分である。サブレイアウトは、複数の帳票がある場合に、各帳票ごとに定義するレイアウトである。サブレイアウトは共通レイアウトと重ねて印刷することができ、より柔軟な帳票処理を行うことも可能である。一つのレイアウトに対しては、レイアウト見出し（一レコード）と領域（複数）の定義を行う。

領域には固定領域と相対領域がある。更に、領域の変更（つまり印刷する場所の変更を行うのであるが、相対領域のときは領域の縮小も可能）を行うための代替領域というものも定義できる。

固定領域には、頁内の特定の場所に印刷を行うために複数のレコードが配置される。

相対領域は、印刷領域の開始位置と終了位置を指定し、その中に定義されたレコードの連続的な印刷を行うための領域である。開始行位置から始まり、レコードの持っている改行情報により行位置が自動的に更新される。つまり、印刷ごとに行位置が移動することになる。これは、明細行の印刷などに用いられる。また相対領域には、二つ以上に分割した領域を一つのつながった領域として扱える分割領域も定義できる（これにより袋とじ印刷などが簡単にできる）。つまり、一つの領域に余白がなくなったら、次の領域に印刷位置が自動的に移動する。更に、一頁の相対分割領域に余白がなくなったときに自動改頁を行うことも可能である。もちろん、改頁制御を自分で行うこともできる。

相対領域、固定領域には、それぞれ複数個の相対レコード、固定レコードの定義を行う。

レコードには、見出しレコード、固定レコード、相対レコードがある。またレコードは、フィールドの集まりであり、レイアウトへの出力の論理的な出力の単位である。固定印刷レコードの場合、「継続印刷」の指定があると、一度そのレコードに対して出力の命令が出されると、それ以降の頁では出力の命令を実行しなくても同じ内容のレコードが自動的に印刷される機能がある。また、相対レコードの場合、残りの行数を知ることもできる。それぞれのレコードに対して、複数のフィールドの定義を行う。

フィールドには、見出しフィールドとデータフィールドがある。フィールドは、プログラムで扱うことのできる最小単位であり、各種の属性が定義できる。この属性は、プログラムとは関係がなく定義の変更が可能であり、プログラムでは、型と長さのみが関係する。

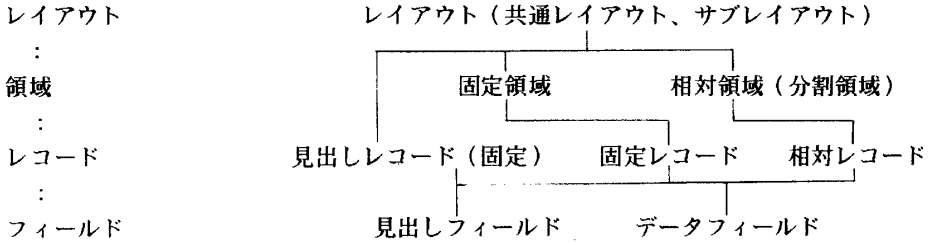

図１．階層化されたレイアウト情報

Print Layout Feature For COBOL

* YUJI YAMAYA, TAKEO IKEDA, SIROU SIMAOKA -- TOSHIBA Corp. OME WORKS
** TSUYOSI TANAKA -- TOSHIBA SOFTWEAR ENGINEERING Co.

3.機能概要

印刷レイアウト機能の実現のために、レイアウト定義部（FORMALユーティリティ）、プログラム部（COBOLコンパイラ）、制御部（ＯＳ）を一体としてサポートを考慮しなければならない。

まず、レイアウト定義を行う FORMAL ユーティリティでは、図１に示したようなレイアウト情報の定義や印刷書式（印刷属性、フォームオーバレイ）の定義を行い、テスト印刷などを繰り返してイメージに合う帳票を作り出す。ただし、フィールド属性や印刷領域の位置の変更などは、プログラムから切り離して考えることができるため、型や長さなどプログラムのロジックに影響を及ぼす部分の変更を除けば、設計終了後プログラムと同時に開発することも可能となる。もっとも、たとえロジックに影響がある変更があったとしても、従来の変更量に比べれば、行制御やカラム位置に捕らわれずに済むため、変更は格段に少なくなる。

プログラムは、図２に示す基本的な流れに沿ってコーディングすればよい。命令は図２の各処理を示した６個の基本命令と、行制御情報を取り出すACCEPT 命令、継続印刷の指定を取り消す CANCEL 命令、印刷領域を代替領域に変更する CHANGE 命令、出力されたレコードの内容を無効にする CLEAR 命令、相対印刷領域内の行送り制御を行う OPERATE 命令がある。簡単な帳票印刷処理を行う限りでは図２に示す基本命令を使うだけで済み、その他の命令を使うことによって更に複雑な帳票印刷が可能となる。

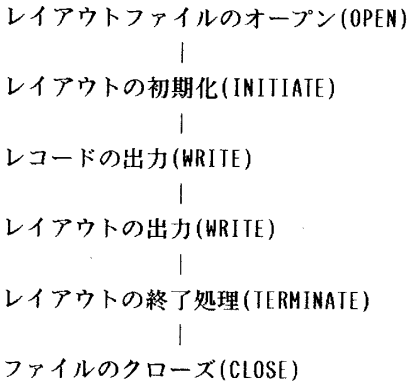

図２　基本処理の流れ

実行時、制御部はローカルなエリアをレイアウト制御領域として使用し、レイアウト情報の定義、印刷書式、印刷データをそこにプールする。

制御部は、OPEN 命令によってレイアウトファイルに対してオープンが要求されると、印刷書式の読み込みを行う。INITIATE 命令によってこれから使用する共通レイアウトが指定されるとレイアウト定義の読み込み、レイアウト制御領域の初期化を行う。ここで、更に別のサブレイアウトの INITIATE を行うことも可能である。予めMOVE 命令などによってフィールドへデータを格納しておき、WRITE（レコードの出力）命令により、実際に印刷されるデータをプールする。WRITE（レイアウト出力）命令によって、レイアウトに従った物理的な印刷を行う。以上の WRITE 命令の繰り返しによって、指定されたレイアウトの印刷が行われるが、レイアウトの変更を行うには、TERMINATE 命令を実行した後にINITIATE命令を実行すればよい。

4.従来の問題と本機能サポートによる効果

従来の帳票設計で問題となっていたのは、プリントレイアウトシートという実際のイメージとはかなり異なったイメージをもとに設計を進めるため、出来上がった帳票が実際のものとかなり異なったものとなることであった。

また別の問題は、印刷を行うデータ、レコードの配置（レイアウト）のために、本来はプログラムの流れとは無関係であるはずの制御を強いられることである。つまり、印刷帳票の上から順番に出力処理をしていかなくてはならないということである。

更に、プリンタの性能がアップしているにもかかわらず、それをうまくプログラム上で使いこなせないことである。

出来上がる帳票のイメージとのギャップについては、FORMALユーティリティにあるテスト印刷機能により、実際のレイアウトイメージを確かめながら、プログラムとは関係なく帳票の設計ができることで問題とならない。

また、レイアウト定義を行うことでレイアウトの出力制御に関して、プログラムと切り離すことが可能である。つまり、プログラムの流れをデータに依存した論理的なものとすることができ、帳票形式の変更は印刷レイアウトや印刷書式を変更するだけでよく、データレコードなどの定義の変更も従来のプログラムの変更に比べて大幅に少なくなる。

更に、レイアウト定義を、出力するプリンタを意識して行うことで、ページプリンタやシリアルプリンタにも出力することができ、そのプリンタの持つ機能を十分に生かすことができる。

5.おわりに

印刷レイアウト機能を使うことにより、ほぼ従来の帳票印刷における不満や問題が解決できるようになり、プログラムの開発・保守の生産性が向上するだけでなく、従来プログラムでは困難であった印刷制御、帳票の印刷が可能となった。

最後に、実際に帳票印刷を行う立場に立って考慮し使い易いものとしたり、更なる機能の向上を計ってゆかなければならない。