システム・ソフトウェア開発

地図画像のためのソフトウェア:

(1) ベクトルデータを用いた地図画像の作成

中野 司*

Softwares for Map Images:

(1) Map Image Construction Using Vector Data

Tsukasa NAKANO*

Abstract : Procedure to construct map images using vector data is explained. The DLG-format data stored in the CD-ROM of the Geological Map of Japan 1 : 1000000, 3rd edition (GSJ, 1995) is used as an example of the vector data. To construct the digital geological map images, line data for all geological unit is firstly extracted from the DLG data. Applying the rasterization technique to the line data, the map image is drawn as a description in the PostScript language or a TIFF image data file. For the above process, the software executable on the UNIX operating system is introduced.

Key Words : Map Image, Vector Data, Geological Map of Japan 1 : 1000000, CD-ROM, GIS, DLG, Rasterization.

1. はじめに

地図画像とは地図データがプロットされた画像のことである。地図データとして様々な形式のものが考えられるが、画像そのものが与えられた場合には、それに対する処理や表示の手法は通常の画像に対するもの（たとえば、中野, 1995a）とほとんど同じである。地図画像に固有な処理として画像の変形（座標変換もしくは地図投影）がやや特殊であると言えるが、実際にはそれは時間を要するだけでさほど複雑な処理内容ではない。地図画像の取り扱いがやや複雑な理由は、それを画像以外の様々な形式の地図データから作成する処理が必要なことであろう。

ここでは、地図データとしてグリッド（格子）データとベクトルデータの2種類のものを考える。グリッドデータは、たとえば数値地形モデル（Digital Elevation Model）のように、データ値（たとえば、標高値）が面的に密に分布する点上で与えられたものである。ただし、データ点は通常は規則正しく並ぶ格子点であるが、そうでない場合には適当なデータ補間の手法（たとえば、中野・藤井, 1989b ; Briggs, 1974）を用いて格子点上での値を求める必要がある。地図データとしてグリッドデータが与えられた場合、それから地図画像を作成することは容易である。

ベクトルデータは、GIS（Geographical Informa-

tion System ; 地理情報システム）などで取り扱われる、地図上に示された領域・線分・点などの要素を、それらの位置関係や接続関係を含めたグラフ（Wilson, 1985）として表したものである。これを利用する場合には、グラフの解読（線分要素の接続状態のチェックなど）を行ない、かつ、ラスター化（rasterization）と呼ばれる輪郭線内部の領域塗りつぶしを行なうなどして画像を作成しなければならない。

本文では、このようなベクトルデータを用いた地図画像の作成手法について述べる。以下では、数値地質図の場合を例に取り、実際のベクトルデータの構成、およびそれから地図画像を作成するための具体的な処理手順について詳しく説明する。また、その実例として、100万分の1日本地質図第3版CD-ROM版（地質調査所, 1995）に含まれるベクトルデータから、PostScriptで記述した数値地質図、もしくはTIFF画像データファイルとしておさめた数値地質図を作成するためのソフトウェアなどを紹介する。

2. DLGデータファイル

2.1. 100万分の1日本地質図第3版CD-ROM版

本文ではベクトルデータによって表現された地図データの例として、地質調査所から最近出版された100万分の1

日本地質図第3版CD-ROM版(地質調査所, 1995)におさめられているDLG形式(Digital Line Graph Format; 村田, 1995)の数値地質図データファイル(以下では, DLGデータファイル("DLG" file)と呼ぶ)のものだけを取り扱う。このCD-ROMは, 地質調査所がこれまで長年に渡って印刷物として発行してきた地質図類の一層の利用・活用を促進するために, それらをデジタル化した数値データの形でCD-ROMなどの電子媒体上に記録して出版した最初のものである(地質調査所, 1995)。これにはDLGデータファイル以外にも複数の形式の数値地質図データベース(データファイル群とそれらを取り扱うためのソフトウェア), およびそれらの解説書などがおさめられている。本文で紹介するソフトウェアは, これらの応用例として位置づけられる。そこで, DLGデータファイルの説明の前に, まずこのCD-ROMにおさめられているデータベースの概略を説明する。なお, このCD-ROM全体の概要は, CD-ROMに添付の小冊子, もしくは地質ニュースの特集記事(たとえば, 長谷川, 1995; 鹿野ほか, 1995; 松本・鹿野, 1995)などを参照していただきたい。

100万分の1日本地質図第3版CD-ROM版には以下の3種類の形式の数値地質図のデータベースおさめられている: (1)ベクトル形式地質図データベース, (2)メッシュ形式地質図データベース, (3)地質図画像データベース。これらは各々, 複数のデータファイルとそれらをパーソナルコンピュータやワークステーションで取り扱うためのソフトウェア, そしてデータおよびソフトウェアの詳細な解説書の入った文書ファイルなどからなる。

ベクトル形式地質図データベースでは, ベクトルデータで表された数値地質図などを作成・編集する際に広く用いられている市販のGISのソフトウェア(たとえば, ESRI社の"Arc/Info")で取り扱いが可能なDLG形式(正確にはDLG-3オプション形式), Arc export形式("Arc/Info"用), およびDXF形式(Autodesk社のCADソフトウェア"AutoCAD"用)の3種類のファイル形式のデータが用意されている(村田, 1995)。そして, これら各々のファイル形式で, 円筒図法と多円錐図法で地図投影した座標値表現を持つ数値地質図のデータセットが, 合計6個のデータファイルとしてCD-ROMにおさめられている。これらのうち本文で利用するDLGデータファイルは, 以下の2つである。

/cdrom/database/vectmap/japan_g.dlg

円筒図法で地図投影された座標値(X座標値が経度, Y座標値が緯度; 単位は秒)の入ったデータファイル。

/cdrom/database/vectmap/japan_p.dlg

多円錐図法の座標値(単位はm)のデータファイル。

ただし, 上記の記法はUNIXのもので, CD-ROMはディレクトリ"/cdrom"にマウントされていると仮定し

ている。また, ファイル名はすべて小文字で表した。これら2つのDLGデータファイルはまったく同じ構成である(後述)。それゆえ, 以下では必要なとき以外にはその区別をしない(より詳しく言うと, 円筒図法の座標データから多円錐図法の座標データを算出可能なので, 実際には前者のデータファイルだけを考えればよい)。

ベクトル形式地質図データベースに含まれるソフトウェアは, パーソナルコンピュータ用とワークステーション用のものが用意されており, 本文で紹介するもの(後述)と同様にDLGデータファイルからPostScriptで記述した数値地質図データを作成するために利用できる(村田, 1995)。ただし, 本文のものとは異なり, これらは指定した任意の広がり数値地質図ではなく, 1次メッシュ領域(国土地理院発行の20万分の1地図群の領域に相当する東西1度, 南北40分の広がりを持つ矩形領域)ごとの数値地質図データを作ることができるだけである。

メッシュ形式地質図データベースと地質図画像データベースにはどちらも, 日本全体をカバーする1次メッシュ領域ごとに画像化した数値地質図や地形陰影図などが含まれている。ただし, 前者は各領域の画像データがASCIIテキストの独自形式で書き込まれており, 添付のソフトウェアはNEC PC9801シリーズ(+フルカラー画像表示ボード)で用いられることを想定している(松本・野呂, 1995)。また, 後者は, 画像データがTIFF画像データファイル(中野, 1995a)としておさめられており, 添付のUNIX+Xウィンドウ用ソフトウェア, もしくは市販のソフトウェアなどを用いれば様々なコンピュータで利用可能である(中野, 1995b)。

メッシュ形式地質図データベースの画像データファイルは, GISソフトウェアを用いてベクトルデータファイル(DLGデータファイル)から作成された。また, 地質図画像データベースのデータファイルは, メッシュ形式データベースのデータから作成された。後述する本文のソフトウェアのうちの1つは, 地質図画像データベースのものと同じ数値地質図の画像データを, 指定された任意の領域についてDLGデータファイルだけを用いて作成することができる。

2.2. DLG データファイルの構成

以下では, 100万分の1日本地質図第3版CD-ROM版のDLGデータファイルの構成を, 地質図画像データを作成する際に必要な部分を中心に説明する。このファイルのより詳細な説明(特にファイル上でのデータの位置や記述形式)は, 村田(1995)の解説書を参照していただきたい。また, ここで説明する構成が一般的なものかどうかを確認していないので, ここの説明を参考に100万分の1日本地質図第3版以外のDLGデータファイルを取り扱う場合には注意が必要である。

DLGデータファイルはFORTRANで取り扱うことを想定した一行80文字(行末は改行コード)のASCIIテキ

第 1 表 DLG データファイル（"DLG" file）の概要

ヘッダ部		
項目名		説 明
原資料の情報	資料名, 原図の縮尺・地図投影法, デジタイズの解像度など.	
変換パラメータ	内部座標系の値で記されたデータ点の位置を実座標系のものに変換するための4つの係数データ (注1).	
ノード数	数値地図を構成するノード要素の個数. 後述するデータ部のノード要素の記述は, ここに記された個数分だけ繰り返される.	
領域個数	数値地図を構成する領域要素の個数. 後述するデータ部の領域要素の記述は, ここに記された個数分だけ繰り返される.	
線分個数	数値地図を構成する線分要素の個数. 後述するデータ部の線分要素の記述は, ここに記された個数分だけ繰り返される.	
データ部		
大項目	項目名	説 明
ノード 要素 記述部	内部 ID 番号	各ノード要素を識別するためにつけられた自然数 (注2).
	位置	ノードに相当する点の座標値. 内部座標系の値で記されている.
	線分個数	そのノードを始点もしくは終点とする線分要素の個数.
	線分番号	そのノードを始点もしくは終点とする線分要素 (複数) の内部 ID 番号. ただし, そのノードが終点となる線分では, 内部 ID 番号に負符号をつけた値が記述されている.
領域 要素 記述部	内部 ID 番号	各領域要素を識別するためにつけられた自然数 (注2).
	線分個数	領域の輪郭線を構成する線分を記述する線分番号 (後述) の個数.
	"島" の個数	領域内部のドーナツ状に穴があいている部分 ("島") の個数.
	線分番号	領域の輪郭線を構成する線分 (複数) の内部 ID 番号. ただし, 領域内部の"島"の輪郭線は, ダミーの番号0に続いて記述される (それゆえ, 上記の線分個数から"島"の個数を引いたものがここで有効な線分の個数となる).
	領域の属性	その領域を占める地質単元に相当する地質コード (1~200) と領域の属する1次メッシュのコード (注3).
線分 要素 記述部	内部 ID 番号	各線分要素を識別するためにつけられた自然数 (注2).
	始点ノード ID 番号	線分の始点に相当するノードの内部 ID 番号.
	終点ノード ID 番号	線分の終点に相当するノードの内部 ID 番号.
	左側領域 ID 番号	線分をベクトルと見なした時にその左側に位置する領域の内部 ID 番号
	右側領域 ID 番号	線分をベクトルと見なした時にその右側に位置する領域の内部 ID 番号.
	点の個数	線分を構成する点の個数.
	点の位置	線分を構成する点 (複数) の座標値. 内部座標系の値で上記の個数分だけ記述されている.
	線分の属性	線分の種類を表す断層コード (注4).

- 注 1 内部座標系から実座標系への変換式の詳細は村田 (1995) を参照のこと.
- 注 2 100 万分の 1 日本地質図第 3 版の数値データファイルの場合, 内部 ID 番号は各要素ごとのファイル上での順番 (1~) と一致する.
- 注 3 100 万分の 1 日本地質図第 3 版の数値データファイルの場合の領域の属性値. 1 次メッシュ (領域) とは国土地理院発行の 20 万分の 1 地図に相当する経度方向 1 度, 緯度方向 40 分の矩形領域で, その南西隅の緯度・経度を識別可能なコードがつけられている.
- 注 4 100 万分の 1 日本地質図第 3 版の数値データファイルの場合の線分の属性値. 断層コードは, 値 0 が 1 次メッシュ領域の境界線を, 1 が地質単元の境界線, 2 が地質境界を兼ねない断層線, そして 3 が地質境界を兼ねる断層線を表す.

ストファイルである. その内容は, 大別するとヘッダ部とデータ部に分類できる (第 1 表).

第 1 表に示されているように, ヘッダ部には変換パラメータ, そして後述する 3 種類の描画要素の各々の個数などが記されている. ここで, 変換パラメータは, DLG データファイル上に記録された点の座標値をファイル上の値から実際のものへと変換 (一次変換) する際に用いら

れる (その詳細は村田 (1995) の解説書, もしくは付録に示したソースリストを参照のこと).

データ部には, ノード, 領域, 線分要素の各記述部がこの順番でおさめられている. これら 3 つの要素記述部はヘッダ部に記されている個数分だけ各々繰り返されるが, それ以外にも, これらの識別のために各記述部の先頭に各々 "N" (ノード), "A" (領域), および "L" (線分) という文字が書き込まれている.

DLG データファイルにおさめられた地図データでは, 何らかの属性によって区別される地図上の領域の輪郭線や断層線のような線構造は, 多数の点からなる線分によって表される (第 1 図). これらの点のうち, 線分の端点や交点のような特別な点 (第 1 図の N1 ~ N6) は, 特にノード要素と呼ばれる. 線分を構成する点の座標値はノード要素の分も含めてすべて線分要素記述部 (後述) に書き込まれているため, 数値地図を描く際に DLG データファイル中のノード要素記述部の情報 (第 1 表) は特に必要ない. しかしながら, ノード要素それぞれに一意的につけられた内部 ID 番号は, 同一点のチェックのために重要である. すなわち, DLG データファイル中の線分要素をつなぎあわせる必要があるとき, それらの端点 (= ノード要素) の座標値の比較ではなく, ノード要素の内部 ID 番号の比較によって同一点を検索することができる.

領域要素記述部には, 地図上で同じ属性がつけられたひ

第 1 図 DLG データファイルの構成要素の概念図

とつながりの領域ごとにその広がり（情報）などが書き込まれている。各領域の属性は属性コードによって表され、100万分の1日本地質図第3版のDLGデータファイルの場合、それには地質コード（地質単元ごとに付けられた1～200の整数値）と1次メッシュコード（その領域が属する1次メッシュ領域につけられた4桁の整数値）の2つがある。たとえば、第1図のA1とA5はそれぞれ独立な領域要素で、また、A2～A4は地質コードは同じだが異なる1次メッシュ領域に属するためにそれぞれ2つに分断された領域要素である。領域要素は、複数の線分（線分要素）からなる輪郭線によって取り囲まれる。領域要素記述部では、その領域の広がり（輪郭線）をこれらの線分要素それぞれにつけられた一意的な内部ID番号列で表す。ただし、第1図のA1のように領域内部に“島”（第1図のA5）を持つ領域要素では、“島”の輪郭線をダミーの内部ID番号0の後に記述する。

100万分の1日本地質図第3版のDLGデータファイルの線分要素には、その属性として以下の4種類の断層コード（0～3）がつけられている：(0) 1次メッシュ領域の境界線（第1図L8～L10）、(1) 地質単元の境界線（L1～L5, L11）、(2) 地質境界を兼ねない断層線（L12）、(3) 地質境界を兼ねる断層線（L6, L7）。これに加えて、それぞれの線分要素記述部には先に述べた線分要素の内部ID番号、線分を構成する点の情報、その端点のノード要素の内部ID番号が書き込まれている。さらに、数値地図を描画する際には必要ないが、線分要素をベクトルと見なした場合にその左右に位置する領域要素の内部ID番号もそこに記録されている。

以上に加えて、100万分の1日本地質図第3版のDLGデータファイル固有の注意点を2つコメントしておく。まず第1の点は、データ部の各要素記述部にある内部ID番号についてである。これは、100万分の1日本地質図第3版のDLGデータファイルの場合、各要素ごとにファイル上で出現する順番（1～）に一致している。後述するソフトウェアではDLGデータファイルのアクセスの際にそれを大いに利用しているが、一般に成り立っているとは限らないことに注意すべきである。第2の点は、領域要素の属性についてである。100万分の1日本地質図第3版のDLGデータファイルの場合、先に述べたメッシュ形式データベースを作成することを考慮して、各領域要素には地質コードに加えて1次メッシュコードを属性値として与えている。このため、第1図のA2～A4のように、地図上の領域は地質単元だけで分類されないことになる。言い換えると、1次メッシュ領域の境界線で分割された領域（たとえば第1図A2）は、数値地質図描画のためにその境界線（第1図L9）を取り除く必要がある。後述するソフトウェアでも、各地質領域に対してこのような1次メッシュ領域の境界線を取り除いた輪郭線の再構成（線分のつなぎあわせ）を行なっている。

先に説明したノード要素記述部のように、DLGデータ

ファイルにはそれを用いて数値地図を作成する際に不必要な情報が多数含まれている。これは、DLGデータファイルが数値地図描画の際の領域→線分→ノードのようなトップダウンのアクセスだけでなく、データファイルの作成時に行なわれるノード→線分→領域のようなボトムアップのアクセスにも対応しているためである。

3. 地図画像の作成

100万分の1日本地質図第3版のDLGデータファイルから地質図画像を作成するための処理の概要を第2図に示す。ただし、第2図の長楕円中に示されているものは、以下で説明する処理の各ステップで用いられるプログラムの名称である。これらは、UNIX上で起動可能なC言語で書かれた実行プログラムもしくはC-Shellとawk言語用のスクリプトで、第2、5および6表に使用法を、また付録にソースリストを示してある。

ここで説明する手法では、DLGデータファイルから画像を直接作成するのではなく、まずそれから2種類の形式の中間ファイルを作成する。これらはどちらも、フリーフォーマットの数値が書き込まれたASCIIテキスト形式のファイルで、その各々を領域データファイル（“area” file；第3表参照）と線データファイル（第4表参照）と呼んでいる。ただし、線データファイルは、第2図に示した断層線データファイル（“fault” file）、海岸線データファイル（“coast” file）、海域データファイル（“sea” file）、そして地質データファイル（複数；“geology” files）の総称である。また、領域データファイルはそれらを作成するためだけに用いるファイルなので、画像作成の際には必要ない。

線データファイルを用いて、PostScriptで記述したものとTIFF画像データファイル（中野, 1995a）の2種類の形式の地質図画像を作成する。本文で紹介するソフト

第2図 100万分の1日本地質図第3版の数値地質図を作成する処理のフローチャート。楕円および四角で示したものは処理に用いられるプログラムおよびデータファイルの名称である（詳細は本文を参照のこと）。

第2表 DLGデータファイルから線データファイルを取り出すためのソフトウェアの説明

dlg2fault.g および dlg2fault.p	
機能	DLG形式で書かれている100万分の1日本地質図第3版の数値データファイル（DLGデータファイル；"DLG" file）から断層線のデータだけを取り出し、標準出力に書く（それを格納したファイルを断層線データファイル（"fault" file）と呼ぶ）。プログラム名の拡張子".g"と".p"は、プログラムで取り扱うDLGデータファイルで用いられている地図投影法の違いを示している（詳細は本文を参照のこと）。
起動法	dlg2fault.g {DLGFile} もしくは dlg2fault.p {DLGFile}
パラメータ	DLGFile DLGデータファイルの名前。省略時は標準入力からデータの読み込みが行なわれる。
dlg2area.g および dlg2area.p	
機能	DLGデータファイルに含まれているすべての地質領域のデータを取り出し、標準出力に書く（それを格納したファイルを領域データファイル（"area" file）と呼ぶ）。プログラム名の拡張子の意味はdlg2fault.gとdlg2fault.pの場合と同じ。
起動法	dlg2area.g {DLGFile} もしくは dlg2area.p {DLGFile}
パラメータ	DLGFile プログラムdlg2fault.gの場合と同じ。
area2coast	
機能	領域データファイル中から海岸線のデータだけを取り出し、標準出力に書く（それを格納したファイルを海岸線データファイル（"coast" file）と呼ぶ）。
起動法	area2coast {AreaFile}
パラメータ	AreaFile 領域データファイルの名前。省略時は標準入力からデータの読み込みが行なわれる。
area2geology	
機能	領域データファイル中に含まれているすべての地質領域の境界線データを取り出し、それらを地質単元ごとにまとめて、その各々の属性コード（地質コード）を名前とするファイルに格納する（これによって得られた地質単元ごとのファイルを総称して、地質データファイル（"geology" files）と呼ぶ）。
起動法	area2geology {AreaFile} Filter Extension
パラメータ	AreaFile プログラムarea2coastの場合と同じ。 Filter 地質データファイルを格納する際に用いるフィルタの記述（詳細は本文を参照のこと）。 Extension 地質データファイルを格納する際にその各々の名前につける拡張子（詳細は本文を参照のこと）。

ウェア（第5、6表）は、指定した地域・広がり領域について、地質領域の輪郭線などの線画を作成すること、および指定した色で塗り分けた地質領域などをカラー画像として描画することができる。さらに、TIFF画像データを作成するプログラムline2map（第6表）は、指定した地質単元の領域だけを抜き出した画像を作成することも可能である（後述）。

以下では、上記の2つの処理内容（DLGデータファイルからの線データファイルの作成とそれを用いた地質図画像の作成）をそれぞれ説明する。

3.1. 線データの取り出し

先にも述べたようにDLGデータファイルには、ノード要素記述部のような数値地質図を描画するために不必要な情報が多数含まれている。そこで、ここでは、数値地質図の描画に必要な情報だけをDLGデータファイルから抜き出して、取り扱いの容易なASCIIテキストのフリーフォーマットのデータファイル（線データファイル）に書き込む。

DLGデータファイルから線データファイルを作成する際の具体的な処理内容は以下のとおりである。まず、プログラムdlg2fault（第2表）を用いて、DLGデータファイル中の線分要素のうちで断層コードから断層線の一部であると判断できる線分の情報だけを取り出し、断層線データファイルに書き込む。ただし、第2表にも示されているように、このプログラムと後述するdlg2areaにはプログラム名の拡張子として".g"と".p"のついた2つのバージョン

がある。これらは各々、先に説明した100万分の1日本地質図第3版CD-ROM版の円筒図法と多円錐図法の2つのDLGデータファイルに対応しており、DLGデータファイルからのデータの抜き出しと同時に座標値をそれぞれ度単位とkm単位に変換する（付録に示した"Makefile"のリストを見ればわかるように、拡張子".g"と".p"のプログラムは同じソースリストから座標値変換のための定数"UNIT"に異なる値を与えられて作成（コンパイル）される）。

次に、プログラムdlg2area（第2表；dlg2faultと同様にこれにも2つのバージョンがある）を用いて、地質単元の占める領域の輪郭線の情報だけをDLGデータファイルから抜き出して領域データファイル（第3表）に書き込む。この際に、数値地質図を描画する場合に必要なDLGデータファイル中のノード要素の情報と1次メッシュ領域の境界線をなす線分要素の情報は、断層コードなどを利用して取り除かれる。

領域データファイルに書き込まれた地質領域の輪郭線をすべて描くと、隣り合う地質領域の境界線をなす線分はその両側の領域の輪郭線の一部として2回使用される。ところで、"海"という属性を持つ領域は100万分の1日本地質図第3版のDLGデータファイルでは定義されていないので、その際に地質領域の輪郭線のうち海岸線をなす線分だけは1度しか使用されないことになる（ついでながら、"湖"という属性は定義されている；村田・鹿野、1995）。これを利用すれば、領域データファイルに含まれる線分の情報から海岸線のデータを取り出すことができ

第3表 領域データファイル("area" file)のフォーマット

線分データの個数 (M)	
領域の輪郭線データの個数 (N)	
線分データ (M 回の記述が繰り返される; $i=1 \dots M$)	
i 番目の線分を構成する点の個数 (P_i)	注1
i 番目の線分の始点のノード番号 (自然数; $1 \sim$)	注2
i 番目の線分の終点のノード番号 (自然数; $1 \sim$)	注2
i 番目の線分を構成する点の座標データ ($ P_i $ 回の記述が繰り返される; $j=1 \dots P_i $)	
j 番目の点の X 座標値 (浮動少数点数)	注3
j 番目の点の Y 座標値 (浮動少数点数)	注3
領域の輪郭線データ (N 回の記述が繰り返される; $k=1 \dots N$)	
k 番目の輪郭線を構成する線分の個数 (Q_k)	
k 番目の輪郭線がとり囲む領域の属性コード (地質コード; $\pm 1 \sim \pm 200$)	注4
k 番目の輪郭線を構成する線分の名前 (Q_k 回の記述が繰り返される; $l=1 \dots Q_k$)	
l 番目の線分の名前 (番号; このファイル上での線分データの格納の順番 i に対応)	注5

- 注1 P_i が負の値の場合、その線分は1次メッシュの境界線 (詳細は本文を参照のこと) を構成するものである (線分を構成する点の個数は $-P_i$ である)。
- 注2 ノード番号は、同一の点を識別するために使われる。
- 注3 座標値の単位はもともとなった DLG データファイルなどに依存する。
- 注4 属性コードが負の値の場合には、輪郭線データは地質領域の内部にある別の地質領域 ("島") の輪郭線を記述している (詳細は本文を参照のこと)。
- 注5 線分の番号が正か負かによって輪郭線上での線分のつながり (輪郭線の周回方向) が決まる。番号が正ならばそれで示される線分を始点から終点につないだ向きが、また番号が負ならばその絶対値で示される線分を終点から始点につないだ向きが輪郭線の周回方向となる。

る。プログラム area2coast (第2表) は、このようにして海岸線データファイルを作成する。ただし、領域データファイル中の線分から上に定義される海岸線を抜き出ただけでは単に線分の集まりとなり、たとえば本州全体で1本のつながった (そして閉じている) 輪郭線とはならない。そこで、area2coast は、DLG データファイルの線分要素記述部からそのまま領域データファイルにコピーした、各々の線分の端点に相当するノード要素の内部 ID 番号を利用してこれらの線分をつなぎ合わせ、海岸線がすべて閉じた線分になるようにする。

前述したように、100万分の1日本地質図第3版の DLG データファイルでは "海" は独立な領域要素として定義されていない。しかしながら、上で求めた海岸線の内部の "陸" の領域を除く部分を便宜上 "海" の領域と定義できる。さらに、日本列島全体を含む十分に大きな領域の輪郭線をそれにつけ加えれば、数値地質図を描画する際に海域を他の地質領域と同様にして取り扱うことができる (DLG データファイルの説明 (第1表) で使った用語に従えば、その領域に対して "陸" の部分は文字どおり "島" となる)。プログラム append.box (第2図) はこのような目的に使用する。すなわち、このプログラムは、海岸線データファイルの中の線分を構成する点の座標をすべて調べて、それらを内部に含む十分に大きな長方形の領域を決め、その輪郭線のデータを海岸線データファイルの最後につけ加えた海域データファイルを作成する。

線データファイルの取り出しの処理の最後として、プロ

第4表 線データファイルのフォーマット (注1)

以下のような線分データの記述がファイルの終了、もしくは線分を構成する点の個数が0であるデータセットが見つかるまで繰り返される ($i=1 \sim$)	
線分を構成する点の個数 (P_i)	注2
線分を構成する点の座標データ ($ P_i $ 回の記述が繰り返される; $j=1 \dots P_i $)	
j 番目の点の X 座標値 (浮動少数点数)	注3
j 番目の点の Y 座標値 (浮動少数点数)	注3

- 注1 断層線 ("fault" file)、地質 (地質領域の輪郭線; "geology" files)、海岸線 ("coast" file) および海域 ("sea" file) の各データファイルはすべてここで示したフォーマットで書き込まれる。
- 注2 地質データファイルでは P_i が負の値の線分データが存在する。その場合の線分データは地質領域の内部にある別の地質領域 ("島"; 詳細は本文を参照のこと) の輪郭線の一部である (線分を構成する点の個数は $-P_i$ となる)。
- 注3 座標値の単位はもともとなった DLG データファイルなどに依存する。

グラム area2geology (第2表) は、領域データファイルに含まれるすべての地質領域の輪郭線情報を取り出し、それらを地質単元ごとにまとめて、地質単元に対応する地質コード ($1 \sim 200$ の整数値) を名前とする地質データファイルに書き込む (地質単元ごとの地質データファイルを自動的に作成する)。ただし、DLG データファイルから作成する際に1次メッシュ領域の境界線の情報を取り除いたため、領域データファイル上では地質領域の輪郭線は必ずしもひとつつながりの閉じた線分として記録されているわけではない。それゆえ、海岸線の場合と同様に、area2geology は、すべての地質領域の輪郭線が閉じたものとなるように、領域データファイル上の線分の端点のノードの内部 ID 番号を用いて、そのつなぎ合わせを行なっている (海岸線の場合は、もともとが地質領域の輪郭線をなす線分のうち1度だけ用いられるものを寄せ集めたブツ切れの状態の線分をつなぎ合わせる必要があったが、地質領域の輪郭線の場合は、1次メッシュ領域の境界線を取り除いたことによる部分的なつなぎ合わせを行なえばよいだけである)。

以上で、数値地質図を描画するために必要なすべての線データファイルが作成できた。これらのうち、断層線データファイルには閉じていない線分が、そして、それ以外の線データファイルはすべて閉じた線分 (領域の輪郭線) のデータがおさめられている。しかしながら、線データファイル上にはそれを区別する情報を埋め込まずに、どちらも同じ形式のデータを書き込むことにした (第4表参照; 後述する数値地質図描画のためのプログラム line2map では、その区別は利用者が指定するようになっている)。また、後述する数値地質図を描くためのソフトウェアはすべて、これらの線データファイルが同じディレクトリにおさめられていることを仮定している。area2geology の場合、通常はそれが自動的に作成する地質データファイルはすべて同じディレクトリに書き込まれるが、それ以外のものの場合は、各自がデータファイルをそれと同じディレクトリに書き込むように指定する必要がある。

なお、ここで作成した線データファイル (および領域データファイル) は ASCII テキスト形式のものである、

ファイル圧縮によってそのサイズを大幅に削減できる（後述）。第2表に示したプログラムのうち area2geology を除くものはすべてデータを標準出力に書き出すので、それを適当なファイル圧縮用のフィルタプログラム（たとえば UNIX の compress コマンドなど）にリダイレクトした結果をファイルに書き込めばよい。また、area2geology は地質单元ごとの地質データファイルを自動的に作成するが、それら各々にデータを書き込む際に起動時のパラメータとして指定されたプログラムをフィルタとして用いる。そして、たとえば compress コマンドによって圧縮されたデータの入ったファイルには拡張子 ".Z" をつけ加える必要がある。area2geology では、自動作成される地質データファイルの名前につけ加えるべき拡張子の指定を起動時に行なうことにしてある（第2表）。以上で述べたファイル圧縮の手法は、線分データファイル作成のプロセスとともに後でその実例を紹介する。

3.2. 地図画像データの作成

3.2.1. PostScript データファイルの作成

上で作成した線データファイルを用いて、PostScript で記述した数値地質図を作成するプログラムには、第5表に示した make.ps.line と make.ps.rgb の2つがある。これらはどちらも UNIX の C-Shell スクリプトで、パターン処理言語 awk を用いて線データファイルのアクセスなどを行なっている。

プログラム make.ps.line は、起動時に指定されたサイズの用紙上に、指定された地域（長方形を仮定）に含まれるすべての地質領域の輪郭線と断層線を、各々指定された線の太さで描いた数値地質図のための PostScript データを作成する（第5表）。このプログラムは描画処理のために2つのサブプログラムを呼び出す（これらも UNIX の C-Shell スクリプトで、各々の使用法の詳細は付録に示した make.ps.line のリストを参照のこと）。これらのうちサブプログラム eps.window は、数値地質図を描く際

第5表 線データファイルを用いて PostScript で記述した数値地質図を作成するためのソフトウェアの説明

make.ps.line	
機能	地質および断層線データファイルから指定された地域のすべての地質領域の境界線および断層線データを取り出し、それらをポストスクリプトで記述した線画のデータに変換する（領域内部の塗りつぶしは行なわない）。
起動法	make.ps.line Directory x1 y1 x2 y2 h0 v0 dh dv Geology Fault {"lay"}
パラメータ	Directory 地質データファイルがおさめられたディレクトリの名前。 x1, y1 地質境界線などを描きたい領域の西南隅の座標値（座標値の単位については本文を参照のこと）。 x2, y2 地質境界線などを描きたい領域の東北隅の座標値（座標値の単位については本文を参照のこと）。 h0, v0 出力用紙上で図の左および下におく余白の幅（横マージンと縦マージン）。単位はインチ。 dh, dv 出力用紙上で図の範囲の横および縦幅。単位はインチ。なお、通常は出力用紙上で図の東西方向が横、南北方向が縦となるが、図を横倒しに出力する場合（後述）にはその対応関係が逆になることに注意が必要である。 Geology 地質領域の境界線の太さ。単位はポイント（= 1/72 インチ）。 Fault 断層線の太さ（単位はポイント）。 "lay" 出力用紙上で図を90度横倒しに描きたい場合に指定する（引用符号はタイプする必要なし）。
make.ps.rgb	
機能	地質、断層線および海岸線データファイルから指定された地域のすべての地質領域、断層線そして海岸線データを取り出し、それらをポストスクリプトで記述したカラー地質図のデータに変換する。
起動法	make.ps.rgb Directory x1 y1 x2 y2 h0 v0 dh dv Palette Fault Coast {"lay"}
パラメータ	Directory プログラム make.ps.line のパラメータの意味と同じ。 x1, y1 同上。 x2, y2 同上。 h0, v0 同上。 dh, dv 同上。 Palette 地質図の色づけに用いられる情報をおさめたファイル（カラーパレットデータファイル；"palette" file）の名前。 Fault 断層線の太さ（単位はポイント）。なお、断層線はパレットデータファイルのパレット番号 250 の色で描かれる。 Coast 海岸線の太さ（単位はポイント）。なお、海岸線はパレットデータファイルのパレット番号 254 の色で描かれる。 "lay" プログラム make.ps.line のパラメータの意味と同じ。

第6表. 線データファイルを用いて TIFF 画像データファイルとして数値地質図を作成するためのソフトウェアの説明

line2map	
機能	地質、断層線、海岸線データファイルおよび海域データファイル（"sea" file；本文参照）を用いて指定された地域の地質図画像を作成し、それを TIFF 画像データファイルとして格納する。地質図上に描きたい地質領域や断層線などの種類、描画の方法（境界線だけを描くことや領域内部を塗りつぶしを行なうことなど）、そして領域や線を描く際に使用する色や線の太さは起動時の指定によって自由に選ぶことができる。
起動法	line2map Filter Script x1 y1 x2 y2 Palette dx dy TIFF
パラメータ	Filter 地質図上に描きたい線データファイルを読み込む際に用いるフィルタの指定を行なう（詳細は本文を参照のこと）。 Script 地質図上に描きたい地質領域や断層線などの種類（データファイルの名前）、描画の方法、そして領域や線を描く際に使用する色や線の太さを記述したファイル（描画記述ファイル；"script" file）の名前。 x1, y1 プログラム make.ps.rgb のパラメータの意味と同じ。 x2, y2 同上。 Palette 同上。 dx, dy 画像の横および縦サイズ。単位はピクセル。 TIFF TIFF 画像データファイルの名前。

のウィンドウ処理（スケーリングとクリッピング）のための PostScript による宣言などを行なう。また、サブプログラム `eps.draw` は、与えられた線データを、`awk` を用いてクリッピングを行ないながら PostScript による線描画コマンドに変換する。

100万分の1日本地質図第3版のすべての地質領域を、それらの地質単位ごとに指定された色で塗りつぶしたカラーの数値地質図の PostScript データを作成するためには、プログラム `make.ps.rgb`（第5表）を用いる。このプログラムは、起動時に指定されたカラーパレットデータファイル（"palette" file；第2図）中に記述されたデータを用いて、各地質単元の領域の色を決める（このファイルの記述法の詳細については後述する）。プログラム `make.ps.line` と同様に、`make.ps.rgb` は4つのサブプログラム（`awk` を用いた C-Shell スクリプト）を呼び出すことによって描画処理を行なう（それらの使用法の詳細は付録に示した `make.ps.rgb` のリストを参照のこと）。これらのうち `eps.window` と `eps.draw` は、先に説明した `make.ps.line` で使用するものと同じである。これ以外の2つのうち、サブプログラム `eps.palette` は、カラーパレットデータファイルに記述された色データを PostScript で描画時に用いる形式に変換する。また、サブプログラム `eps.fill` は、`eps.draw` と同様に `awk` を用いて数値地質図のすべての領域の輪郭線データをスキャンしてクリッピングを行ないながら、それらを PostScript の領域塗りつぶしコマンドに変換する。ただしその際に、輪郭線内部の領域の塗りつぶしは、領域の内部性を奇・偶交差規則（even-odd rule）によって判定する PostScript コマンド `"eofill"` を用いて行なっている。このような PostScript による領域塗りつぶし処理の詳細は、そのリファレンスマニュアル（Adobe System, 1988）などを参照していただきたい。

3.2.2. TIFF 画像データファイルの作成

線データファイルから TIFF 画像データファイルの形式（中野, 1995a）でファイルに書き込んだ数値地質図を作成するためのプログラムが、`line2map`（第6表）である。このプログラムは、起動時に `make.ps.line` や `make.ps.rgb` と同様に指定したサイズ・領域の数値地質図を `make.ps.line` と同様な輪郭線だけの線画、もしくは `make.ps.rgb` と同様な領域内部を指定した色で塗りつぶしたカラー画像として描くことができる。さらにこれらに加えて、`line2map` では線データファイルのうちの指定したものだけを用了数値地質図を作成することが可能である（`make.ps.rgb` などでもこれは可能だが、そのためにはプログラムの書き換えを要する）。

`line2map` は、その起動時に与えた描画記述ファイル（"script" file）の指定によって、描画すべき地質領域や断層線などの種類、およびそれらの表示色（正確にはカラーパレット番号；後述）や線の太さを決める。また、こ

れと同時に指定するカラーパレットデータファイル（これは `make.ps.rgb` で用いられるものと同じ形式のものである）のデータによって、画像上での表示色の詳細を決める。`line2map` が読み込むことのできる描画記述ファイルとカラーパレットデータファイルの例（ただし一部を省略した）を第7表に示した。

第7表に示したように、描画記述ファイルは1行が1つの線データファイルに対応する ASCII テキストファイルである。各行には空白やタブコードで区切られた以下の3もしくは4個の項目のデータを書く：(1) 線データファイルの名前、(2) 線（輪郭線）を描く際に用いるカラーパレット番号、(3) 線（輪郭線）の太さ（単位はピクセル）、(4) 領域を塗りつぶす際に用いるカラーパレット番号（省略可能）。これらの指定に関してはそれぞれ以下のような注意点がある。まず線データファイルの名前の指定に関して、先にも述べたように、`line2map`（および `make.ps.line` と `make.ps.rgb`）ではこれらがすべて同じディレクトリに置かれていることを仮定している。また、これらはファイル圧縮されていてもよいわけだが、このようなディレクトリ指定およびファイル圧縮の解凍処理は `line2map` の起動時のフィルタの指定（第6表；後述）によって決定されるため、ここでの線データファイルの名前にはそれに関する情報をつけ加えなくてもよい。次に、線のカラーパレット番号と太さに関する指定についてであるが、これらの一方を0と指定すると画像上にその線は描かれない。たとえば第7表に示した描画記述ファイルの例では、断層線（線データファイル名 "fault"）と海岸線（"coast"）以外のパレット番号は0なので、これら以外の線（領域の輪郭線）は作成される画像上には描かれない。描画記述ファイルの各行の最後に位置する領域塗りつぶし用のパレット番号の指定は、上にも記したように省略が可能である。もし、この指定を省略すると（行の項目数が3であれば）、そこで指定された線データファイルは（輪郭線ではない）単なる閉じていない線であると見なされ、領域内部の塗りつぶしは行なわれない（たとえば、第7表の断層線）。また、ここに0を指定すると、その線を描くことを指定した場合には閉じた輪郭線であると見なされるが、その内部の塗りつぶしは行なわれない（たとえば、第7表の海岸線）。

描画記述ファイル上でカラーパレット番号として示されている線や領域の色データの詳細は、カラーパレットデータファイル上に記述する（第7表；前述したように、このファイルはプログラム `make.ps.rgb` でも用いられる）。第7表に示されているように、このファイルは各行に空白やタブコードで区切った、以下の4項目のデータを書き込んだ ASCII テキストファイルである：(1) カラーパレット番号（0～）、(2) 色の赤（R）成分の強度、(3) 色の緑（G）成分の強度、(4) 色の青（B）成分の強度。ここで、色の各成分の強度データは0～255の整数値で表す（すべて0なら黒、すべて255なら白となる）。また、先に述べ

たようにカラーパレット番号0は描画記述ファイル中で特別扱いされるが、ここでもそれに対する記述は画像の背景色を指定するためだけに用いられる。ファイル上で、色の記述はカラーパレット番号の順になっている必要はないが、負の値などの不適当な値が書き込まれているとそれ以降の記述は無視される。プログラムline2mapは、色の成分の記述がないカラーパレット番号に対してその初期値として与えられた黒を描画に用いるようになっている。

第6表にも記されているように、プログラムline2mapでは線データのおさめられたディレクトリの指定やそれらがファイル圧縮されている場合の解凍処理を起動時に与えられたフィルタの記述を用いて行なう（プログラムmake.ps.lineとmake.ps.rgbでは、その処理はプログラム中に埋め込まれている；付録のソースリストを参照のこと）。さらにそればかりでなく、フィルタをうまく用いれば、線データファイルを書き換えることなく、line2

mapのデータの読み込みと同時にその座標変換を行なうこともできる。このようなフィルタの指定の実例、およびより高度な応用例については後述する。

プログラムline2mapの説明の最後として、それが用いている作画処理の手法についてコメントしておく。このプログラムで行なっているような画像データ作成のために輪郭線の内部を塗りつぶす処理は、ラスタ化と呼ばれている。ラスタ化には様々な手法があるが（たとえば、中野・藤井、1989a）、line2mapの関数Raster（付録に示したソースリスト"raster.c"を参照のこと）では、それをmake.ps.rgbで用いたPostScriptコマンド"eofill"と同じ奇・偶交差規則で領域の内部性を判定する、高速な手法によって行なっている。この手法では、まず、読み込んだ領域の輪郭線を作成する画像にあわせてデジタル化した後、それとX方向に無限に延びる線の交点（エッジ）を求める。そして、同じY座標値のものをX座標の順に

第7表 描画記述ファイル（"script" file）とカラーパレットデータファイル（"palette" file）の例

描画記述ファイル				カラーパレットデータファイル				備 考
線データ ファイル名	(境界) 線		領 域	パレット 番号	色成分の強度			
	パレット 番号	線の 太さ	パレット 番号		R (赤)	G (緑)	B (青)	
なし	なし	なし	なし	0	255	255	255	注 1
1	0	1	1	1	236	255	255	
...	...	...	...	...	...	...	...	
17	0	1	17	17	188	140	92	← 注 2
19	0	1	19	19	255	255	156	
20	0	1	20	20	255	220	92	← 注 2
22	0	1	22	22	188	255	156	
...	...	...	...	...	...	...	...	
44	0	1	44	44	188	255	92	← 注 2
46	0	1	46	46	0	217	217	
...	...	...	...	...	...	...	...	
53	0	1	53	53	0	250	0	← 注 2
56	0	1	56	56	188	255	188	
...	...	...	...	...	...	...	...	
61	0	1	61	61	0	248	0	← 注 2
63	0	1	63	63	248	133	85	
...	...	...	...	...	...	...	...	
111	0	1	111	111	157	254	0	← 注 2
115	0	1	115	115	235	205	170	
116	0	1	116	116	235	205	130	← 注 2
117	0	1	117	117	235	205	0	
119	0	1	119	119	180	30	0	
...	...	...	...	...	...	...	...	
124	0	1	124	124	230	214	0	← 注 2
126	0	1	126	126	255	0	0	
...	...	...	...	...	...	...	...	
165	0	1	165	165	0	0	255	← 注 2
200	0	1	200	200	150	150	236	
fault	250	2	なし	250	0	0	0	注 3
coast	254	1	0	254	128	128	128	注 3, 4
sea	0	0	255	255	180	180	234	注 4

注 1 画像の背景色に用いるカラーパレットデータ。

注 2 地質領域が存在しないために地質データファイル18などはない。

注 3 領域の塗りつぶしを行わない（領域のパレット番号を指定しない）。

注 4 海岸線は海域データファイル（"sea" file；本文参照）だけを使って描くこともできる。

第 8 表 ソフトウェアの使用例。(a) 線データファイルの作成。(b) 地質図画像データファイルの作成。これらはいずれも UNIX の適当な Shell のもとでのコマンドラインからの入力行を示している。

(a) 線データファイルの作成

1	set cdrom=/cdrom	# CD-ROM のマウントディレクトリ名
2	set dlgl=\${cdrom} /database/vectmap/japan_g.dlg	# DLG データファイル名
3	set area=../area	# 領域データファイル名
4	set g=../g ; mkdir \$g	# 線データファイルを格納するディレクトリ名
5	dlg2fault.g \$dlg compress > \${g} /fault.Z	# 断層線データファイル
6	dlg2area.g \$dlg compress > \${area} .Z	# 領域データファイル
7	zcat \$area area2coast compress > \${g} /coast.Z	# 海岸線データファイル
8	zcat \${g} /coast append.box compress > \${g} /sea.Z	# 海域データファイル
9	zcat \$area area2geology "compress > \${g} /" .Z	# 地質データファイル

(b) 地質図画像データファイルの作成

1	set g=../g	# 線データファイルを格納してあるディレクトリ名
2	set extent=(134.75 34 136.75 36)	# 地質図の描画範囲
3	set paper=(1.5 2.5 5.2 6.4)	# 紙上の左下隅の位置と描画領域の横・縦幅
4	set gl=0.1 ; set fl=0.5 ; set cl=0.1	# 地質境界線、断層線と海岸線の太さ
5	set palette=legend.pal2	# カラーパレットデータファイル名
6	set image=(1365 1680)	# 画像の横・縦幅
7	make.ps.line \$g \$extent \$paper \$gl \$fl > line.ps	# 線画 PostScript
8	make.ps.rgb \$g \$extent \$paper \$palette \$fl \$cl > rgb.ps	# 塗りつぶし PostScript
9	line2map "zcat \${g} /" geology.line \$extent \$palette \$image line.tif	# 線画 TIFF
10	line2map "zcat \${g} /" geology.rgb \$extent \$palette \$image rgb.tif	# 塗りつぶし TIFF

ソートしたエッジのうち、引き続く奇数番目と偶数番目のものを端点とする X 方向の線分によって領域を塗りつぶしを行なう。この手法の詳細については、付録のソースリスト、もしくは適当なコンピュータグラフィックスの教科書（たとえば、Harrington, 1987）を参照していただきたい。なお、このような手法を実現した関数 Raster は、入出力（線データの読み込みとラスター化した線分の画像への書き込み）を引数として与えられた関数（コールバック関数）を用いて行なっている。それゆえ、関数 Raster は、使用する線データや画像データの形式にほとんど依存しない広く利用可能なものとなっている。

4. ソフトウェアの使用例

ここまでで説明した 100 万分の 1 日本地質図第 3 版 CD-ROM 版の DLG データファイルからの線データファイルの作成、およびそれから数値地質図の描画の処理（第 2 図）はやや複雑なので、以下ではそのためのプログラム群（第 2, 5, 6 表）の使用例を示す（第 8 表）。ただし、ここでは、UNIX の適当な Shell のもとで必要なすべてのプログラムがコマンドラインから起動可能であるとする。また、以下では円筒図法で座標値が記された DLG データファイルだけを取り扱うが、多円錐図法の場合もほぼ同じ要領で処理を行なうことができる。

4.1. 線データの取り出し

DLG データファイルから線データファイルを作成する処理の実例を第 8(a) 表に示す。今、100 万分の 1 日本地質図第 3 版 CD-ROM 版が第 8(a) 表の 1 行目の Shell 変数 cdrom に設定されたディレクトリ（"/cdrom"）にマウントされており、また、それに含まれる円筒図法の DLG データファイルの名前が 2 行目で設定した Shell 変

数 dlgl で表されるとする（UNIX によっては、CD-ROM 上のファイル名を大文字で指定する必要があるかもしれない；中野・田中, 1995）。変数 area にその名前を設定した領域データファイル（3 行目）は、線データファイルを作成する間だけ用いるテンポラリなものなので、ハードディスクの容量を考慮して適当なディレクトリに置き、その使用後は消去するとよい。そして、前準備の最後として、線データファイルのために適当なディレクトリを作成しておく（4 行目）。

第 8(a) 表の 5～9 行に示したものが DLG データファイルから線データファイルを作成する実際の処理である。ただし、ここでは、作成するすべてのデータファイルには UNIX の compress コマンドによってファイル圧縮を行

第 9 表 100 万分の 1 日本地質図第 3 版の数値データファイルのサイズ

円筒図法のデータファイル		
ファイル	サイズ (バイト)	圧縮後の サイズ(注)
DLG データファイル (ファイル "japan_g.dlg")	15,539,988	—
領域データファイル ("area" file)	8,246,277	3,120,119
断層線データファイル ("fault" file)	519,612	137,312
海岸線データファイル ("coast" file)	930,041	250,976
海域データファイル ("sea" file)	930,125	368,517
地質データファイル ("geology" files ; 155 個)	11,665,913	4,603,660
多円錐図法のデータファイル		
ファイル	サイズ (バイト)	圧縮後の サイズ(注)
DLG データファイル (ファイル "japan_p.dlg")	15,539,994	—
領域データファイル ("area" file)	15,539,994	3,892,221
断層線データファイル ("fault" file)	555,274	262,412
海岸線データファイル ("coast" file)	1,032,574	475,593
海域データファイル ("sea" file)	1,032,676	475,647
地質データファイル ("geology" files ; 155 個)	2,466,498	5,933,176

注 UNIX の "compress" コマンドによる圧縮後のファイルサイズ。

第3図 プログラムline2mapを用いて作成した数値地質図の画像の例。(a) 地質領域の輪郭線と断層線だけを描いた画像。(b) 地質領域内部を塗りつぶしたカラー地質図画像。

なった結果を書き込んでいる（それを表すためにファイル名の拡張子“.Z”を付け加えた）。また、領域データファイルもファイル圧縮したため、それを用いる7～9行目の処理ではzcatコマンドを用いてその読み出しを行なっている。なお、前述したように、area2geologyの場合（9行目）はファイル圧縮をそれ以外とはやや異なった記法で指定する（第2表）ことに注意していただきたい。

上記の処理に要した時間は合計すると、SUN SPARC Classic（24メガバイトメモリ）+SunOS 4.1.3上でおよそ40分程度であった。また、作成されたデータファイルのサイズは第9表に示したようになった（ただし、上記の計算機以外では、C言語の数値出力用ライブラリ関数の違いに起因する若干の増減があるかもしれない）。この表からわかるように、ここまでの処理には、100万分の1日本地質図第3版の円筒図法と多円錐図法の数値データのどちらの場合も最低限15メガバイト程度のハードディスクの容量が必要である（このうちおよそ3メガバイトはテンポラリな領域データファイルの分なので、後で消去可能である）。ここでは、ファイルの圧縮にcompressコマンドを用いたが、GNUのgzipコマンドを用いればこれらよりもやや圧縮率の高い（サイズの小さい）データファイルを得ることができる。

4.2. 地図画像の作成

数値地質図の描画処理の実例を第8(b)表に示す。そのために、まず、線データファイルのおさめられたディレク

トリ名（第8(b)表の1行目）、描画を行なう地域の対角点（西南と東北の隅）の座標値（2行目）、PostScriptで記述する地質図の紙上での位置・サイズ（3行目）と線の太さ（4行目；単位はポイント）、カラー地質図の色指定のためのカラーパレットファイル名（5行目）、そしてTIFF画像データファイルを作成する際の画像サイズ（6行目）をそれぞれShell変数に設定している。ここでは、画像サイズ（横・縦幅がそれぞれ1365と1680ピクセル）および紙上での描画領域のサイズ（横・縦幅がそれぞれ5.2と6.4インチ）は、地質図の地域の広がり（東経134.75～136.75度、北緯34～36度）と日本周辺での経・緯度1度当たりの実距離の比（およそ13:16）を考慮して決めた。また、PostScriptで記述した数値地質図をA4サイズ用の紙（横・縦幅がそれぞれ8.3と11.7インチ）上にセンタリングして出力することを想定して、紙上での描画領域の左下隅の位置を指定している（紙の左端から1.5、下端から2.5インチとした）。

PostScriptで記述した数値地質図は、第8(b)表7行目（地質領域の輪郭線と断層線の線画）もしくは8行目（カラー地質図）のようにして作成する。ただし、これらではプログラムが出力するPostScriptの記述をそのままファイルに書き込んでいるが、実際にはそれを直接PostScriptプリンタに出力したり、また、ファイル圧縮を行ないながらファイルに書き込んだ方がよいだろう。

プログラムline2mapを用いて、PostScriptで記述したものと同様な地質図のTIFF画像データファイルを作

成する例が、第 8(b) 表 9 行目（線画）と 10 行目（カラー地質図）である。これらの例で、プログラム名（"line2map"）の次に記した "zcat \${g} /" が線データファイルを読み込むためのフィルタの記述である。これによって Shell 変数 *g* の示すディレクトリ（"\${g} /"）におさまられているファイル圧縮されたデータファイルを、読み込み時に *zcat* コマンドを用いて解凍するように指定している。その次に記された "geology.line" および "geology.rgb" が、描画記述ファイルの名前である。これらのうち前者は、地質と断層線データファイルだけをカラーパレット番号 250 の色（黒；第 7 表参照）で線画として描画するように記述したものである（地質データファイルに対しては、領域塗りつぶしに使うカラーパレット番号を 0 として輪郭線だけを描くように指定した）。また、後者は、第 7 表に示した内容とほぼ同じ記述を含むファイルである。このようにして得られた TIFF 画像データファイル "line.tif" と "rgb.tif" の画像を第 3 図に示した。

以上で説明した描画処理に要する時間はいずれも、前記の SUN の計算機で 20 分程度であった。これらの処理時間のほとんどは、データファイルのスキャンに費やされているようである。また、作成されたファイルのサイズは、PostScript のもので 1 メガバイト弱に、そして、TIFF 画像データファイルの場合は、"line.tif"、"rgb.tif" がそれぞれ 120 および 200 キロバイト程度になった。ただし、PostScript のファイルは、圧縮するとおよそ 1/3 のサイズになる。そして、TIFF 画像データファイルは、プログラム line2map が画像データを LZW 法（中野, 1995a）で圧縮して書き込むようにしてあるので、比較的小さなサイズとなっている（LZW 法によるデータ圧縮の指定の詳細は付録のソースリストを参照のこと）。

5. おわりに

本文では、ベクトルデータから地図画像を作成するために、一般の GIS ソフトウェアが取り扱う複雑な構造のデータファイル（DLG データファイル）からダイレクトに描画を行なうのではなく、それから必要な部分だけを抜き出した線データファイルを用いて描画処理を行なうというやや冗長な手続きを踏んでいる。このようにした理由は、ソフトウェア作成の負担を軽減するためだけではなく、利用者によるベクトルデータの作成（もしくは修正）を容易にすることを意図したためである（ただし、100 万分の 1 日本地質図第 3 版の数値データの著作権は地質調査所が保持しているため、それを利用するには地質調査所の承諾が必要である；100 万分の 1 日本地質図第 3 版 CD-ROM 版（地質調査所, 1995）に添付の小冊子中の著作権等に関する記述を参照のこと）。すなわち、個々の地質单元ごとに線データファイルを単純な形式にしてあるために新たな線データファイルの作成が容易で、また、プログラム line2map を使えば（描画記述ファイルの若干の改変で）それを追加した数値地質図画像の作成が簡単に行なえ

る（たとえば、調査地域の位置を数値地質図上に示したインデックスマップなどは容易に作成できる）。

しかしその反面、前述したように数値地質図の描画時の線データファイルのスキャンにはかなりの時間がかかる。これを高速化するためには、線データファイルの構造を以下のように変える必要があるかもしれない：(1) 線データファイル中の各線分のデータセット（第 4 表）にそれが含まれる領域の広がり的情報を持たせる（これを利用すれば、スキャンの際に描画領域外のデータの読み飛ばしが高速に行なえる）、(2) 線分データセットに対するインデックスを作成して、それを用いてスキャンをする、(3) 線データファイルをバイナリファイルにする。また、これとは別に、PostScript で記述した数値地質図を作成するプログラムの高速化のために、それを C-Shell + awk ではなく、もっとデータ読み込みの速い言語（たとえば C 言語）でコーディングし直すべきかもしれない。

プログラム line2map 起動時のフィルタとして適当なものを指定すれば、本文で紹介したような線データファイルの解凍処理だけでなく、線を構成する点の座標変換をデータの読み込み時に同時に行なうことができる。これは、衛星画像を地質図と重ね合わせて表示する場合などに利用できる（中野, 1996b）。すなわち、これら 2 枚の地図画像の座標系を一致させるために、画像の位置決めデータ（画像上の座標と実空間での座標（経度と緯度）の関係式）を利用して衛星画像を変形（座標変換）するのではなく、それを逆に用いて座標変換したベクトルデータから地質図を作成の方が効率的である。また、線データファイルのフォーマットを変更した場合にも、フィルタでその違いを吸収するようにすれば line2map のソースリストを書き換える必要はない。

本文では地図画像を色だけで表現したが、プログラム line2map に少し手を加えれば、その領域塗りつぶしを適当なパターン（ハッチパターン）を用いて行なうこともできる（なお、100 万分の 1 日本地質図第 3 版 CD-ROM 版に含まれるソフトウェア DLG2PS（村田, 1995）は、地質領域をパターンで塗りつぶした数値地質図を作成できる）。さらに、パターンとして適当な画像データを与えることにすれば、それは画像中の任意の輪郭線内部だけを表示するプログラムとなる。これは、衛星画像のうち陸に相当する部分だけを表示したい場合などに利用できる。

以上で作成した数値地質図は、地形データと組み合わせで陰影つき地質図や鳥瞰図として表示することができる（中野, 1996a）。また、前述したように、衛星画像と重ね合わせた表示も可能である（中野, 1996b）。このような手法は、別稿において詳しく述べる予定である。

なお、本文で紹介したソフトウェアはすべて配付が可能である。ご希望のかたは著者まで連絡していただきたい。また、100 万分の 1 日本地質図第 3 版 CD-ROM 版に関しては、地質調査所地質情報センター情報管理普及室（電話：0298 - 54 - 3606、電子メール：postmaster@gsj.

go.jp)に問い合わせていただきたい。

文 献

- Adobe System (石田晴久監修, 松村邦仁訳) (1988) ページ記述言語 PostScript リファレンス・マニュアル. アスキー出版局, 338 p.
- Briggs, I. C. (1974) Machine contouring using minimum curvature, *Geophys.*, vol. 39, pp. 39-48.
- 地質調査所 (編) (1995) 100 万分の 1 日本地質図第 3 版 CD-ROM 版, 数値地質図 G-1. 地質調査所.
- Harrington, S. (郡山 彬訳) (1988) アルゴリズムとプログラムによるコンピュータグラフィックス [I]. マグロウヒル, 324 p.
- 長谷川功 (1995) 特集: 100 万分の 1 日本地質図第 3 版 CD-ROM 版. 地質ニュース, no. 493, p. 8.
- 鹿野和彦・長谷川功・松本則夫・村田泰章・中野 司・野呂春文 (1995) 数値地質図「100 万分の 1 日本地質図第 3 版 CD-ROM 版」の出版. 地質ニュース, no. 493, pp. 9-11.
- 松本則夫・鹿野和彦 (1995) 「100 万分の 1 日本地質図第 3 版 CD-ROM 版」CD-ROM 解説. 地質ニュース, no. 493, pp. 12-16.
- 松本則夫・野呂春文 (1995) メッシュ形式地質図データベース数値ファイル解説, および, メッシュ形式地質図データベースソフトウェア解説. 地質調査所 (編) 100 万分の 1 日本地質図第 3 版 CD-ROM 版, 数値地質図 G-1. 地質調査所.
- 村田泰章 (1995) ベクトル形式地質図データベース数値ファイル解説, および, ベクトル形式地質図データベースソフトウェア DLG2PS 解説. 地質調査所 (編) 100 万分の 1 日本地質図第 3 版 CD-ROM 版, 数値地質図 G-1. 地質調査所.
- 村田泰章・鹿野和彦 (1995) 「100 万分の 1 日本地質図第 3 版 CD-ROM 版」から求めた日本列島を構成する岩石の分布面積. 地質ニュース, no. 493, pp. 26-29.
- 中野 司 (1995a) 画像処理のためのソフトウェア: (7) 色数削減のアルゴリズム. 情報地質, vol. 5, no. 4, pp. 187-210.
- 中野 司 (1995b) 地質図画像データベース数値ファイル及びソフトウェア解説. 地質調査所 (編) 100 万分の 1 日本地質図第 3 版 CD-ROM 版, 数値地質図 G-1. 地質調査所.
- 中野 司 (1996a) 地図画像のためのソフトウェア: (2) 地図画像の陰影表現. (準備中)
- 中野 司 (1996b) 地図画像のためのソフトウェア: (3) 衛星画像の利用法. (準備中)
- 中野 司・藤井直之 (1989a) 画像処理のためのソフトウェア: (1) 領域塗りつぶしと仮想スクリーン処理. 情報地質, no. 14A, pp. 93-107.
- 中野 司・藤井直之 (1989b) 二次元分布データの処理システム: (2) 二次元分布データの内挿補間. 情報地質, no. 14B, pp. 113-132.
- 中野 司・田中明子 (1995) Clementine 探査機のデータ: その概要と利用法. 情報地質, vol. 6, no. 3, pp. 147-164.
- Wilson, R. J. (斉藤伸自・西関隆夫訳) (1985) グラフ理論入門. 近代科学社, 182 p.

要 旨

地図画像のためのソフトウェア: (1) ベクトルデータを用いた地図画像の作成

: 中野 司

ベクトルデータとして与えられた地図データから画像 (地図画像) を作成するための手続きを説明する. ここでは, 100 万分の 1 日本地質図第 3 版 CD-ROM 版 (地質調査所, 1995) におさめられた DLG 形式のベクトルデータファイルを例にとり, そこから取り出した地質単元の線データをラスター化して数値地質図を作成する手法について説明する. その実例として, DLG 形式のベクトルデータから PostScript で記述した数値地質図, もしくは TIFF 画像形式の数値地質図を作成するための UNIX で使用可能なソフトウェアを紹介する.

キーワード: 地図画像, ベクトルデータ, 100 万分の 1 日本地質図, CD-ROM, GIS, DLG, ラスター化.

付録：プログラムリスト

以下の名前のファイルの内容（ソースリスト）を示した。

```
Makefile:
dig2fault.c, dlg2area.c, area2coast.c, area2geology.c:
line2map.c, raster.c, clip.c, line.c:
(cmake, h, cmake.c, writetiff.h, writetiff.c):
stderr.c:
append_box:
make.ps, line, make.ps, rgb:
eps, window, eps, draw, eps, palette, eps, fill:
```

Makefile:

```
1: LIBTIFF          # change this as suitable for your system
2:
3: dig2fault.g:      dig2fault.c
4: # cc -DUNIT=(1.0/60.0) -o dig2fault.g dig2fault.c
5: # cc -DUNIT=(1.0/60.0) dig2fault.c ; mv a.out dig2fault.g
6:
7: dlg2fault.p:      dlg2fault.c
8: # cc -DUNIT=0.001 -o dlg2fault.p dlg2fault.c
9: # cc -DUNIT=0.001 dig2fault.c ; mv a.out dlg2fault.p
10:
11: dlg2area.g:       dlg2area.c
12: # cc -DUNIT=(1.0/60.0) -o dlg2area.g dlg2area.c
13: # cc -DUNIT=(1.0/60.0) dlg2area.c ; mv a.out dlg2area.g
14:
15: dlg2area.p:       dlg2area.c
16: # cc -DUNIT=0.001 -o dlg2area.p dlg2area.c
17: # cc -DUNIT=0.001 dlg2area.c ; mv a.out dlg2area.p
18:
19: area2coast:       area2coast.c
20: cc -o area2coast area2coast.c
21:
22: area2geology:     area2geology.c
23: cc -o area2geology area2geology.c
24:
25: line2map:         line2map.c raster.c clip.c line.c y
26: cmake, h, cmake.c, writetiff.h, writetiff.c
27: cc -DLZW -I$(LIBTIFF) -I$(LIBTIFF) -o line2map line2map.c y
28: raster.c clip.c line.c cmake.c writetiff.c -ltiff
29:
30: stderr:           stderr.c
31: cc -o stderr stderr.c
```

dig2fault.c:

```
1: #include <stdio.h>
2:
3: #define UNIT 1.0
4: #define UNIT 1.0
5: #define FILE
6: #endif
7: #define FAULT(attr) (attr==2 || attr==3)
8:
9: static void Error(msg)
10: {
11:     char *msg;
12:     fputs(msg, stderr); fputs("\n", stderr); exit(1);
13: }
14:
15: #define BUFILE 256
```

```
16: static void FGetS(buf, file)
17: {
18:     char *buf;
19:     FILE *file;
20:     if (fgets(buf, BUFILE, file)==NULL)
21:         *buf='\0';
22:     else
23:         if (buf[strlen(buf)-1]!='\n') Error("Buffer overflow.");
24: }
25:
26: typedef struct {
27:     int num;
28:     double *x, *y;
29: }
```

```
31: ) Line;
32: static void ReadLine(dlg, buf, line, attr)
33: FILE *dlg;
34: char *buf;
35: int *line;
36: int *attr;
37: {
38:     int idx, rem;
39:     sscanf(&buf[42], "%d", &(line->num));
40:     if ((line->x==double *) || (line->y==double *) || (line->num==0)) {
41:         Error("Memory for line allocation error.");
42:         return;
43:     }
44:     for (idx=0; idx<line->num; idx++) {
45:         if (rem-idx%4==0) FGetS(buf, dlg);
46:         sscanf(&buf[rem%4], "%21f%21f", &(line->x[idx]), &(line->y[idx]));
47:         FGetS(buf, dlg);
48:     }
49:     main(argc, argv)
50:     int argc;
51:     char **argv;
52: {
53:     FILE *dlg;
54:     int idx, attr;
55:     char buf[BUFILE];
56:     double a, b, c, d;
57:     Line line;
58:     if (argc!=1 && argc!=2) Error("usage: dig2fault [DLG]");
59:     if (argc==1)
60:         dlg=stdin;
61:     else if ((dlg=fopen(argv[1], "r"))==NULL) Error("DLG file not found.");
62:     FGetS(buf, dlg); fputs(buf, stderr);
63:     for (idx=2; idx<=10; idx++) FGetS(buf, dlg);
64:     for (idx=0; buf[idx]!='\0'; idx++) if (buf[idx]!='D') buf[idx]='E';
65:     sscanf(buf, "%18f%18f%18f%18f", &a, &b, &c, &d);
66:     for (idx=11; idx<=15; idx++) FGetS(buf, dlg);
67:     do FGetS(buf, dlg); while (*buf!='L');
68:     do {
69:         ReadLine(dlg, buf, line, attr);
70:         if (FAULT(attr)) {
71:             printf("%d\n", line->num);
72:             for (idx=0; idx<line->num; idx++)
73:                 printf("%18f%18f", (a*line.x[idx]+b*line.y[idx]+c)*UNIT,
74:                     (a*line.y[idx]-b*line.x[idx]+d)*UNIT);
75:             free(line); free(line.x); FGetS(buf, dlg);
76:             while (*buf!='L');
77:         }
78:         if (dlg!=stdin) fclose(dlg);
79:     }
80: }
81:
82: #include <stdio.h>
83:
84: #define UNIT 1.0
85: #define FILE
86: #endif
87:
88: #define GEOLOGY(attr) (attr!=0)
89:
90: static void Error(msg)
91: {
92:     char *msg;
93:     fputs(msg, stderr); fputs("\n", stderr); exit(1);
94: }
```

dlg2area.c:

```
1: #include <stdio.h>
2:
3: #define UNIT 1.0
4: #define FILE
5: #endif
6:
7: #define GEOLOGY(attr) (attr!=0)
8:
9: static void Error(msg)
10: {
11:     char *msg;
12:     fputs(msg, stderr); fputs("\n", stderr); exit(1);
13: }
```

```

16: #define BUFLen 256
17:
18: static void FGetS(buf, file)
19: char *buf;
20: FILE *file;
21: {
22:     if (fgetc(buf, BUFLen, file) == NULL)
23:         *buf = '\0';
24:     else
25:         if (buf[strlen(buf)-1] != '\n') Error("Buffer overflow.");
26: }
27:
28: typedef struct {
29:     double a, b, c, d;
30: } TP;
31:
32: static void ReadTransformParameter(buf, tp)
33: char *buf;
34: TP *tp;
35: {
36:     char *ptr;
37:     for (ptr=buf; *ptr != '\0'; ptr++) if (*ptr == 'D') *ptr = 'E';
38:     sscanf(buf, "%18lf%18lf%18lf%18lf", &(tp->a), &(tp->b), &(tp->c), &(tp->d));
39: }
40:
41:
42:
43: static void SkipNode(dlg, buf)
44: FILE *dlg;
45: char *buf;
46: {
47:     int num;
48:     for (sscanf(&buf[36], "%d", &num); num>0; num-=12) FGetS(buf, dlg);
49: }
50:
51:
52: #define Area struct AREA
53: Area {
54:     int num, *names, attr;
55:     Area *next;
56: };
57:
58: static Area *AllocArea(names, next)
59: int *names;
60: Area *next;
61: {
62:     Area *area = (Area *) malloc(sizeof(*area));
63:     if (area == NULL) Error("memory for Area allocation error.");
64:     area->num = 0; area->names = names; area->next = next; return(area);
65: }
66:
67:
68: static Area *ReadArea(dlg, buf)
69: FILE *dlg;
70: char *buf;
71: {
72:     int num, island, *names, cnt, idx, rem, name, attr;
73:     Area *ptr, *top=NULL;
74:     sscanf(&buf[36], "%d", &num);
75:     sscanf(&buf[60], "%d", &island);
76:     if (names != NULL) Error("memory for area line name allocation error.");
77:     top = AllocArea(names, NULL); cnt = 0;
78:     for (idx = 0; idx < num; idx++) {
79:         if ((rem = idx%12) == 0) FGetS(buf, dlg);
80:         sscanf(&buf[rem*6], "%d", &name);
81:         if (names == NULL)
82:             top = AllocArea(&(names[cnt]), top);
83:         else
84:             ++(top->num); names[cnt++] = name;
85:     }
86: }
87:
88:
89: FGetS(buf, dlg); sscanf(buf, "%d", &attr);
90: for (ptr=top; ptr->next != NULL; ptr=ptr->next) ptr->attr = (-attr);
91: ptr->attr = attr; return(top);
92:
93:
94:
95:
96: typedef struct {
97:     int bgn, end, num;
98:     double *x, *y;
99: } Line;
100:
101: static void ReadLine(dlg, buf, line)
102: FILE *dlg;
103: char *buf;
104: Line *line;
105: {
106:     int idx, rem;
107:     sscanf(&buf[6], "%d", &(line->bgn));
108:     sscanf(&buf[12], "%d", &(line->end));
109:     sscanf(&buf[42], "%d", &(line->num));
110:     if ((line->x = (double *) calloc(line->num, sizeof(*(line->x)))) == NULL ||
111:         (line->y = (double *) calloc(line->num, sizeof(*(line->y)))) == NULL)
112:         Error("memory for points of line allocation error.");
113:     for (idx = 0; idx < line->num; idx++) {
114:         if ((rem = idx%3) == 0) FGetS(buf, dlg);
115:         sscanf(&buf[rem*24], "%12lf%12lf", &(line->x[idx]), &(line->y[idx]));
116:     }
117: }
118:
119:
120:
121: static void ReadDLG(dlg, tp, aNum, lNum, area, line)
122: FILE *dlg;
123: TP *tp;
124: int *aNum, *lNum;
125: Area *area;
126: Line *line;
127: {
128:     int idx, nNum, iid, num, name, attr;
129:     char buf[BUFLen];
130:     Area *ptr;
131:     FGetS(buf, dlg); fputs(buf, stderr);
132:     for (idx = 2; idx <= 10; idx++) FGetS(buf, dlg);
133:     ReadTransformParameter(buf, tp);
134:     for (idx = 11; idx <= 15; idx++) FGetS(buf, dlg);
135:     sscanf(&buf[30], "%d", &nNum);
136:     sscanf(&buf[46], "%d", &aNum); if (*aNum == 0) Error("no Area exists.");
137:     sscanf(&buf[62], "%d", &lNum);
138:     for (idx = 0; idx < nNum; idx++) {
139:         FGetS(buf, dlg); if (*buf != 'N') Error("Node read error.");
140:         SkipNode(dlg, buf);
141:     }
142:     num = 0;
143:     for (idx = 0; idx < *aNum; idx++) {
144:         FGetS(buf, dlg); if (*buf != 'A') Error("Area read error.");
145:         if (idx == 0)
146:             *area = ptr = ReadArea(dlg, buf);
147:         else
148:             ptr = (ptr->next = ReadArea(dlg, buf));
149:         for (iNum = 0; iNum < tp->num; iNum++) {
150:             if ((ptr->names[iNum] > 0) && (ptr->names[iNum] > *lNum) ||
151:                 (ptr->names[iNum] < 0) && (-ptr->names[iNum] > *lNum))
152:                 Error("bad line name of area.");
153:             ptr = ptr->next;
154:         }
155:         *aNum = num;
156:         if (*lNum == (Line *) calloc(*lNum, sizeof(**line))) == NULL)
157:             Error("memory for Line allocation error.");
158:         for (idx = 0; idx < *lNum; idx++) {
159:             FGetS(buf, dlg); if (*buf != 'L') Error("Line read error.");
160:

```



```

176: sscanf(&(buf[1]), "%s", &iid);
177: if (iid!=idx+1) Error("bad internal ID of Line.");
178:
179: ReadLine(dlg, buf, &((*(line)[idx]));
180: Fgetc(buf, dlg); sscanf(buf, "%d", &attr);
181: if ((BLOG(attr)) (*(line)[idx].num==(-(*(line)[idx].num);
182:
183:
184:
185: static void PrintXY(tp.x.y)
186: TP *tp;
187: double x.y;
188: {
189: printf("%lf%lf%lf\n", (tp->x*tp->b*tp->c)*UNIT,
190: (tp->y*tp->b*tp->c)*UNIT);
191: }
192:
193: void main(argc, argv)
194: argc;
195: **argv;
196: {
197: FILE *dlg;
198: TP *tp;
199: int aNum, lNum, name, num, idx;
200: Area *area, *ptr;
201: Line *line;
202:
203: if (argc!=1 && argc!=2) Error("usage : dlg2area (DLG) ");
204: if (argc==1)
205:     dlg=stdin;
206: else
207:     if ((dlg=fopen(argv[1], "r"))==NULL) Error("DLG file not found.");
208: ReadDLG(dlg, &tp, &aNum, &lNum, &area, &line); if (dlg!=stdin) fclose(dlg);
209:
210: printf("%d%lf%lf\n", lNum, aNum);
211: for (name=0; name<lNum; name++) {
212:     printf("%d%lf%lf%lf\n", line[name].num, line[name].bgn, line[name].end);
213:     num=(line[name].num>0)?line[name].num:-line[name].num;
214:     for (idx=0; idx<num; idx++)
215:         PrintXY(&tp, line[name].x[idx], line[name].y[idx]);
216: }
217: for (ptr=area, ptr!=NULL; ptr->ptr->next) {
218:     printf("%d%lf%lf\n", ptr->num, ptr->attr);
219:     for (idx=0; idx<ptr->num; idx++) printf("%d%lf", ptr->names[idx]);
220: }
221:
222: }

```

area2coast.c:

```

1: #include <stdio.h>
2:
3: static void Error(msg)
4:     *msg;
5: char {
6:     fputs(msg, stderr); fputs("\n", stderr); exit(1);
7: }
8:
9: typedef struct {
10:     int num, bgn, end;
11:     double *x, *y;
12: } Line;
13:
14: static Line *ReadLine(file, lNum)
15:     FILE *file;
16:     int lNum;
17: {
18:     Line *line=(Line *)calloc(lNum, sizeof(*line)); *ptr;
19:     int i, j;
20:     if (line==NULL) Error("memory for Line allocation error.");
21:     for (ptr=line, i=0; i<lNum; i++, ptr++) {
22:         fscanf(file, "%d %d", &(ptr->num), &(ptr->bgn), &(ptr->end));
23:         if (ptr->num<0) ptr->num=(-ptr->num);
24:         if (ptr->x=calloc(ptr->num, sizeof(*ptr->x)))==NULL,
25:             if (ptr->y=calloc(ptr->num, sizeof(*ptr->y)))==NULL,
26:                 Error("memory for point in Line allocation error.");
27:         for (ptr->y=(double *)calloc(ptr->num, sizeof(*ptr->y)); ptr->y; ptr++)
28:             Error("memory for point in Line allocation error.");
29:     }
30: }

```

```

31: for (j=0; j<ptr->num; j++)
32:     fscanf(file, "%lf", &(ptr->x[j]), &(ptr->y[j]));
33: return(line);
34: }
35:
36: typedef struct {
37:     int num, *names;
38: } Area;
39:
40: static Area *ReadArea(file, aNum)
41:     FILE *file;
42:     int aNum;
43: {
44:     Area *area=(Area *)calloc(aNum, sizeof(*area)); *ptr;
45:     int i, attr, j;
46:     if (area==NULL) Error("memory for Area allocation error.");
47:     for (ptr=area, i=0; i<aNum; i++, ptr++) {
48:         fscanf(file, "%d", &(ptr->num), &attr);
49:         if ((ptr->names=
50:             (int *)calloc(ptr->num, sizeof(*ptr->names)))==NULL)
51:             Error("memory for line name in Area allocation error.");
52:         for (j=0; j<ptr->num; j++) fscanf(file, "%d", &(ptr->names[j]));
53:     }
54:     return(area);
55: }
56:
57: static void PrintNormal(line)
58:     *line;
59: {
60:     int idx;
61:     for (idx=0; idx<(line->num-1; idx++)
62:         printf("%lf%lf%lf\n", line->x[idx], line->y[idx]);
63: }
64:
65: static void PrintReverse(line)
66:     *line;
67: {
68:     int idx;
69:     for (idx=line->num-1; idx>0; idx--)
70:         printf("%lf%lf%lf\n", line->x[idx], line->y[idx]);
71: }
72:
73: static void PrintNormal(line)
74:     *line;
75: {
76:     int idx;
77:     for (idx=line->num-1; idx>0; idx--)
78:         printf("%lf%lf%lf\n", line->x[idx], line->y[idx]);
79: }
80:
81: static void PrintConnect(lNum, line, num, bgn, end)
82:     lNum, num, bgn, end;
83: {
84:     int idx;
85:     *ptr;
86:     for (ptr=line, idx=0; idx<lNum; idx++, ptr++)
87:         if (ptr->num>0) {
88:             if (end==ptr->bgn) {
89:                 printf("%d%lf", num+ptr->num-1);
90:                 ptr->num=(-ptr->num);
91:                 PrintConnect(lNum, line, num+ptr->num-1, bgn, ptr->end);
92:                 ptr->num=(-ptr->num);
93:             }
94:             PrintReverse(ptr); ptr->num=(-ptr->num); return;
95:         }
96:         if (end==ptr->end) {
97:             if (bgn==ptr->bgn)
98:                 printf("%d%lf", num+ptr->num-1);
99:             else
100:                 ptr->num=(-ptr->num);
101:             PrintConnect(lNum, line, num+ptr->num-1, bgn, ptr->bgn);
102:             ptr->num=(-ptr->num);
103:         }
104:         PrintNormal(ptr); ptr->num=(-ptr->num); return;
105:     }
106:     Error("open line found.");
107: }

```

```

111: static void PrintCoast(aNum, lNum, area, line)
112: aNum, lNum;
113: Area
114: *area;
115: *line;
116: {
117: int i, j, idx;
118: Line
119: *ptr;
120: for (i=0; i<aNum; i++)
121: for (j=0; j<area[i].num; j++)
122: if (area[i].names[j]>0)
123: line[area[i].names[j]-1].num=
124: (-line[area[i].names[j]-1].num);
125: else
126: line[area[i].names[j]-1].num=
127: (-line[area[i].names[j]-1].num);
128: for (ptr=line, idx=0; idx<lNum; idx++, ptr++) ptr->num=(-ptr->num);
129:
130: for (ptr=line, idx=0; idx<lNum; idx++, ptr++)
131: if (ptr->num>0) {
132: if (ptr->bgn==ptr->end || idx==lNum-1)
133: printf("%d\n", ptr->num);
134: else
135: PrintConnect(lNum-idx-1, &(ptr[l]),
136: ptr->num, ptr->bgn, ptr->end);
137:
138: PrintReverse(ptr); printf("%d\n", *(ptr->x), *(ptr->y));
139: }
140:
141: }
142:
143: void main(argc, argv)
144: int argc;
145: char **argv;
146: {
147: FILE *file;
148: int lNum, aNum;
149: Line *line;
150: Area *area;
151:
152: if (argc!=1 && argc!=2) Error("usage : area2coast {area}");
153: if (argc==1)
154: file=stdin;
155: else
156: if (((file=fopen(argv[1], "r"))==NULL) Error("area file not found.");
157:
158: fscanf(file, "%d %d", &lNum, &aNum); line=ReadLine(file, lNum);
159: area=ReadArea(file, aNum);
160: PrintCoast(aNum, lNum, area, line);
161: }

```

area2geology. c:

```

1: #include <stdio.h>
2:
3: static void Error(msg)
4: msg;
5: char
6: {
7: fputs(msg, stderr); fputc("\n", stderr); exit(1);
8: }
9:
10: typedef struct {
11: int num, bgn, end;
12: double *x, *y;
13: } Line;
14:
15: static Line *ReadLine(file, lNum)
16: FILE *file;
17: int lNum;
18: {
19: Line *line=(Line *)calloc(lNum, sizeof(*line)); *ptr;
20: int i, num, j;
21: if (line==NULL) Error("memory for Line allocation error.");
22:
23: for (ptr=line, i=0; i<lNum; i++, ptr++) {
24: fscanf(file, "%d %d", &(ptr->num), &(ptr->bgn), &(ptr->end));
25: num=(ptr->num>0)?ptr->num:-ptr->num;
26:

```

```

27: if ((ptr->x)=(double *)calloc(num, sizeof(*(ptr->x)))==NULL ||
28: (ptr->y)=(double *)calloc(num, sizeof(*(ptr->y)))==NULL)
29: Error("memory for point in Line allocation error.");
30:
31: for (j=0; j<num; j++)
32: fscanf(file, "%lf", &(ptr->x[j]), &(ptr->y[j]));
33:
34: return(line);
35: }
36:
37: #define struct AREA
38: Area {
39: int num, attr, *names, tot, bgn, end;
40: Area *next;
41: };
42:
43: static Area *AllocArea(attr, names)
44: attr, *names;
45: {
46: Area *area=(Area *)malloc(sizeof(*area));
47:
48: if (area==NULL) Error("memory for Area allocation error.");
49:
50: area->num=i; area->attr=attr; area->names=names; area->tot=0;
51: area->next=NULL; return(area);
52: }
53:
54: static Area *ReadArea(file, aNum, lNum, line)
55: *file;
56: int aNum, lNum;
57: Line
58: *line;
59: {
60: *ptr.*top=NULL;
61: int i, num, attr, *names, stat, j, name, bgn, end=0;
62: char *badLineName="bad line name of Area.";
63: *badLineName="bad line connection of Area.";
64:
65: for (i=0; i<aNum; i++) {
66: fscanf(file, "%d %d", &num, &attr);
67: if ((names=(int *)calloc(num, sizeof(*names)))==NULL)
68: Error("memory for names in Area allocation error.");
69:
70: stat=0;
71: for (j=0; j<num; j++) {
72: fscanf(file, "%d", &(names[j]));
73: if (names[j]==0) Error(badLineName);
74: if (names[j]>0) {
75: if ((name=names[j]-1)>=lNum) Error(badLineName);
76: if (j>0 && end!=line[name].bgn) Error(badLineConnection);
77:
78: bgn=line[name].bgn; end=line[name].end;
79: }
80: else {
81: if ((name=-(names[j]-1))>=lNum) Error(badLineName);
82: if (j>0 && end!=line[name].end) Error(badLineConnection);
83:
84: bgn=line[name].end; end=line[name].bgn;
85: }
86: if (line[name].num>0) {
87: if (stat==0) {
88: top=ptr=AllocArea(attr, &(names[j]));
89: else
90: ptr=(ptr->next=AllocArea(attr, &(names[j])));
91:
92: ptr->bgn=bgn; stat=1;
93: }
94: else
95: ++(ptr->num);
96:
97: ptr->tot+=(line[name].num-1); ptr->end=end;
98:
99: stat=0;
100: }
101: }
102:
103: return(top);
104: }
105:
106: static void PrintLineNormal(file, line)

```

```

107: FILE *file;
108: Line *line;
109: {
110:     int idx;
111:     for (idx=0; idx<line->num-1; idx++)
112:         fprintf(file, "%ift%MfWn", line->x[idx], line->y[idx]);
113:     }
114: }
115: static void
116: PrintLineReverse(file, line)
117: FILE *file;
118: Line *line;
119: {
120:     int idx;
121:     for (idx=line->num-1; idx>0; idx--)
122:         fprintf(file, "%ift%MfWn", line->x[idx], line->y[idx]);
123:     }
124: }
125: static void
126: PrintAreaNormal(file, area, line)
127: FILE *file;
128: Area *area;
129: Line *line;
130: {
131:     int idx;
132:     for (idx=0; idx<area->num; idx++)
133:         if (area->names[idx]>0)
134:             PrintLineNormal(file, &(line[area->names[idx]-1]));
135:     else
136:         PrintLineReverse(file, &(line[area->names[idx]-1]));
137:     }
138: }
139: static void
140: PrintAreaReverse(file, area, line)
141: FILE *file;
142: Area *area;
143: Line *line;
144: {
145:     int idx;
146:     for (idx=area->num-1; idx>0; idx--)
147:         if (area->names[idx]>0)
148:             PrintLineReverse(file, &(line[area->names[idx]-1]));
149:     else
150:         PrintLineNormal(file, &(line[area->names[idx]-1]));
151:     }
152: }
153: static void
154: PrintConnect(file, attr, area, line, tot, bg, end)
155: FILE *file;
156: int attr, tot, bg, end;
157: Area *area;
158: Line *line;
159: {
160:     Area *ptr;
161:     for (ptr=area, ptr!=NULL; ptr->ptr->next;
162:         if (ptr->attr==attr && ptr->num>0)
163:         {
164:             if (end==ptr->bgn) {
165:                 if (bgn==ptr->end)
166:                     if (attr>0)
167:                         fprintf(file, "%dWn", tot+ptr->tot);
168:                     else
169:                         fprintf(file, "%dWn", -(tot+ptr->tot));
170:             }
171:             else {
172:                 ptr->num=(-ptr->num);
173:                 PrintConnect(file, attr, area, line, tot+ptr->tot, bg, ptr->end);
174:                 ptr->num=(-ptr->num);
175:             }
176:             PrintAreaReverse(file, ptr, line); ptr->num=(-ptr->num); return;
177:         }
178:         if (end==ptr->end) {
179:             if (bgn==ptr->bgn)
180:                 if (attr>0)
181:                     fprintf(file, "%dWn", tot+ptr->tot);
182:                 else
183:                     fprintf(file, "%dWn", -(tot+ptr->tot));
184:             else {
185:                 ptr->num=(-ptr->num);
186:                 PrintConnect(file, attr, area, line, tot+ptr->tot, bg, ptr->bgn);
187:             }
188:             ptr->num=(-ptr->num);
189:             PrintAreaNormal(file, ptr, line); ptr->num=(-ptr->num); return;
190:         }
191:         Error("open line found.");
192:     }
193: }
194: #define STRLEN 256
195: static void
196: PrintGeology(filter, extension, area, line)
197: FILE *file;
198: char *area;
199: Line *line;
200: {
201:     int min, max, attr;
202:     Area *ptr;
203:     char name[STRLEN];
204:     FILE *pipe;
205:     for (ptr=area; ptr!=NULL; ptr=ptr->next) {
206:         attr=ptr->attr; if (attr<0) attr=(-attr);
207:         if (ptr==area) min=area;
208:         else if (attr<min) min=ptr; else if (attr>max) max=ptr;
209:         if (ptr==area) min=area;
210:         else if (attr<min) min=ptr; else if (attr>max) max=ptr;
211:         for (attr=min; attr<max; attr++) {
212:             pipe=NULL;
213:             for (ptr=area; ptr!=NULL; ptr=ptr->next)
214:                 if ((ptr->attr==attr || ptr->attr==(-attr)) && ptr->num>0)
215:                     if (pipe==NULL) {
216:                         sprintf(name, "%s%ks", filter, attr, extension);
217:                         printf(stderr, "%sWn", name);
218:                         if ((pipe=fopen(name, "w"))==NULL) Error("pipe not open.");
219:                     }
220:                     if (ptr->bgn==ptr->end)
221:                         if (ptr->attr>0)
222:                             fprintf(pipe, "%dWn", ptr->tot+1);
223:                         else
224:                             fprintf(pipe, "%dWn", -(ptr->tot+1));
225:                     else
226:                         PrintConnect(pipe, ptr->attr, ptr->next, line,
227:                             ptr->tot+1, ptr->bgn, ptr->end);
228:                     PrintAreaReverse(pipe, ptr, line);
229:                     if ((ptr->names)>0)
230:                         fprintf(pipe, "%ift%MfWn", line[(ptr->names)-1], x[0]);
231:                         fprintf(pipe, "%ift%MfWn", line[(ptr->names)-1], y[0]);
232:                     else
233:                         fprintf(pipe, "%ift%MfWn",
234:                             line[(ptr->names)-1], x[(ptr->names)-1], num-1);
235:                         fprintf(pipe, "%ift%MfWn",
236:                             line[(ptr->names)-1], y[(ptr->names)-1], num-1);
237:                     if (pipe!=NULL) fclose(pipe);
238:                 }
239:                 main(argc, argv)
240:                 argc;
241:                 **argv;
242:                 FILE *file;
243:                 int lNum, aNum;
244:                 Line *line;
245:                 Area *area;
246:                 if (argc!=3 && argc!=4)
247:                     Error("usage : area2geology {area} filter extension");
248:                 if (argc==3)
249:                     file=stdin;
250:                 else if ((file=fopen(argv[1], "r"))==NULL) Error("area file not found.");
251:                 fscanf(file, "%d %d", &lNum, &aNum); line=ReadLine(file, lNum);
252:                 area=ReadArea(file, aNum, lNum, line);
253:                 PrintGeology(argv[argc-2], argv[argc-1], area, line);
254:                 }
255:                 }
256:                 }

```



```

157:     }
158: } #endif
159: }
160:
161: main(argc, argv)
162: int argc;
163: char **argv;
164: {
165:     double x1, y1, x2, y2;
166:     char *cm;
167:     TIFF *tiff;
168:     int code;
169:     char msg[STRLEN];
170:
171:     if (argc!=1)
172:         Error("usage : line2map filter script x1 y1 x2 y2 palette xSize ySize TIFF");
173:
174:     sscanf(argv[3], "%lf", &x1);
175:     sscanf(argv[4], "%lf", &y1);
176:     sscanf(argv[5], "%lf", &x2);
177:     sscanf(argv[6], "%lf", &y2);
178:     if (x1>x2 || y1>y2) Error("bad map area.");
179:
180:     cm=ReadPalette(argv[7]);
181:     AllocateImageatoi(argv[8], atoi(argv[9]));
182:
183:     ParseScript(argv[1], argv[2], x1, y1, x2, y2);
184:     if ((tiff=TIFFOpen(argv[10], "r"))==NULL)
185:         Error("TIFF file not created.");
186:
187:     sprintf(msg, "%lg %lg %lg", x1, y1, x2, y2);
188:     TIFFSetField(tiff, TIFFTAG_IMAGEDESCRIPTION, msg);
189:     TIFFSetField(tiff, TIFFTAG_ROWSPERSTRIP,
190:                 (image->xSize>8192)?1L:(long)(8192/image->xSize));
191:
192:     #ifdef LZW
193:     TIFFSetField(tiff, TIFFTAG_COMPRESSION, COMPRESSION_LZW);
194:     #endif
195:     if ((code=WriteTIFF(tiff, image, cm))>0) {
196:         sprintf(msg, "TIFF %s", WriteTIFFError(code));
197:         TIFFClose(tiff);
198:     }
199: }
200:
201: raster.c:
202:
203: 1: #include <stdio.h>
204: 2:
205: 3: static void Error(msg)
206: 4: char *msg;
207: 5: {
208: 6:     fputs(msg, stderr); fputc('\n', stderr); exit(1);
209: 7: }
210:
211: 10: typedef struct {
212: 11:     int x, y;
213: 12: } Point;
214:
215: 13: #define Line struct LINE
216: 14: Line {
217: 15:     int num;
218: 16:     Point *point;
219: 17:     Line *next;
220: 18: };
221:
222: #define NUM (line->num)
223: #define P (line->point)
224:
225: 24: static Line *ReadLine(readPoint, xL, yL, xR, yL, nx, ny)
226: 25: int *readPoint;
227: 26: double xL, yL, xR, yL;
228: 27: {
229: 28:     int cnt, num;
230: 29:     double x, y, dx=(xR-xL)/(double)nx, dy=(yL-yR)/(double)ny;
231: 30:     Point *p, min, max;
232: 31:     Line *line, *ptr=NULL, *top=NULL;
233: 32:     while ((cnt=*readPoint)(&x, &y)>0) {
234: 33:         if ((p=(Point *)calloc(cnt+1, sizeof(*p)))==NULL)
235: 34:             Error("edge data memory allocation error (2).");

```

```

35:         Error("point data memory allocation error.");
36:
37:         p->x=(int)((x-xL)/dx);
38:         p->y=(int)((y-yR)/dy);
39:         min=max-p[0]; num=1;
40:         while (--cnt>0) {
41:             if (p[num].x!=p[num-1].x || p[num].y!=p[num-1].y) {
42:                 if (min.x>p[num].x) min.x=p[num].x; else
43:                 if (max.x<p[num].x) max.x=p[num].x;
44:                 if (min.y>p[num].y) min.y=p[num].y; else
45:                 if (max.y<p[num].y) max.y=p[num].y; ++num;
46:             }
47:
48:             if (min.x<nx && max.x>=0 && min.y<ny && max.y>=0) {
49:                 if ((line=(Line *)malloc(sizeof(*line)))==NULL)
50:                     Error("line data memory allocation error.");
51:
52:                 if (min.y==max.y) {
53:                     p[0]=min; p[1]=max; if (num>1) { p[2]=p[0]; num=2; }
54:                 }
55:                 else if (p->x!=p[num-1].x || p->y!=p[num-1].y)
56:                     p[num]=p[0];
57:                 else
58:                     --num;
59:
60:                 NUM=num; P=p; line->next=NULL;
61:                 if (ptr==NULL) top=line; else ptr->next=line; ptr=line;
62:             }
63:             else
64:                 free(p);
65:
66:             return(top);
67:         }
68:
69:         static int CompareX(x1, x2)
70:         int *x1, *x2;
71:         {
72:             return(*x1-(*x2));
73:         }
74:
75:         static void RasterInside(top, nx, ny, drawInside)
76:         int nx, ny;
77:         void (*drawInside)();
78:         {
79:             int *cnt, **x, y, idx, dy0, dyN, y1, y2, x1, x2, dx, nx1=nx-1, ny1=ny-1;
80:             Line *line;
81:
82:             if ((cnt=(int *)calloc(ny, sizeof(*cnt)))==NULL ||
83:                 (x=(int **)calloc(ny, sizeof(*x)))==NULL)
84:                 Error("edge data memory allocation error (1).");
85:
86:             for (y=0; y<ny; y++) cnt[y]=0;
87:
88:             for (line=top; line!=NULL; line=line->next)
89:                 if (NUM!=1 && (NUM!=2 || P->y!=P[1].y)) {
90:                     for (idx=0; idx<nx; idx++) if ((dy0=P[idx+1].y-P[idx].y)!=0) break;
91:
92:                     for (idx=0; idx<nx; idx++) if (dy0>0 && y1<ny && y2=0) {
93:                         if (y1<0) y1=0; else if (dy0<0) ++y1;
94:                         for (y=y2-ny/ny1; y2; y=y1; y--)+cnt[y];
95:                     }
96:
97:                     if ((dyN=P[idx+1].y-P[idx].y)!=0) {
98:                         if (y1>ny) y1=ny; else if (dy0<0) --y1;
99:                         for (y=y2<0?0:y2; y=y1; y--)+cnt[y];
100:                     }
101:                     if (dyN<0 && y1>=0 && y2<ny) {
102:                         if (y1>ny) y1=ny; else if (dy0<0) --y1;
103:                         for (y=y2<0?0:y2; y=y1; y--)+cnt[y];
104:                     }
105:                     dy0=dyN;
106:                 }
107:
108:             int cnt, num;
109:             double x, y, dx=(xR-xL)/(double)nx, dy=(yL-yR)/(double)ny;
110:             Point *p, min, max;
111:             Line *line, *ptr=NULL, *top=NULL;
112:             while ((cnt=*readPoint)(&x, &y)>0) {
113:                 if ((p=(Point *)calloc(cnt+1, sizeof(*p)))==NULL)
114:                     Error("edge data memory allocation error (2).");

```

```

115:         cnt[y]=0;
116:     }
117:     for (line=top; line=NULL; line=line->next)
118:     {
119:         if (NUM==1 && (NUM==2 || p->y==p[1].y)) {
120:             for (idx=NUM-1; idx>0; idx--)
121:                 if ((dy0-P[idx+1].y-P[idx].y)!=0) break;
122:             for (idx=0; idx<NUM; idx++)
123:                 if ((dyN-P[idx+1].y-P[idx].y)!=0) {
124:                     y1=P[idx].y; y2=P[idx+1].y; dx=(x2-P[idx+1].x)-P[idx].x;
125:                     if (dyN>0 && y1<ny && y2>0) {
126:                         if (y1<0) y1=0; else if (dy0>0) ++y1;
127:                         for (y=(y2-ny); y1<y2; y=y1, y--)
128:                             x[y][cnt[y]+1]=x2+(dx*(y-y2))/dyN;
129:                     } else
130:                         if (dyN<0 && y1>0 && y2<ny) {
131:                             if (y1>ny) y1=ny; else if (dy0<0) --y1;
132:                             for (y=(y2<0)?0:y2; y<y1; y++)
133:                                 x[y][cnt[y]+1]=x2+(dx*(y-y2))/dyN;
134:                         }
135:                     dy0=dyN;
136:                 }
137:             }
138:         }
139:     }
140:     for (y=0; y<ny; y++)
141:         if (cnt[y]>0) {
142:             if (cnt[y]&1) Error("odd edge count number.");
143:             qsort(x[y], cnt[y].sizeof(x[y])).CompareX;
144:             for (idx=0; idx<cnt[y]; idx++)
145:                 if ((x1=x[y][idx])<nx && (x2=x[y][idx+1])>0)
146:                     if (*drawInside)((x1<0)?0:x1.y, (x2>nx)?nx1:x2.y);
147:             free(x[y]);
148:         }
149:     free(x); free(cnt);
150: }
151:
152: static void
153: *line;
154: DrawLine(line, nx, ny, adjust, drawLine)
155: {
156:     nx, ny, adjust;
157:     (*drawLine)();
158:     int
159:     x1, y1, num, idx, x2, y2, x, y, nx1=nx-1, ny1=ny-1;
160:     Line
161:     *next;
162:     while (line=NULL) {
163:         if (drawLine=NULL) {
164:             x1=P[0].x; y1=P[0].y; num=NUM+adjust;
165:             for (idx=1; idx<num; idx++) {
166:                 x2=x+p[idx].x; y2=y+p[idx].y;
167:                 if (clipline(dx1, dy1, dx2, dy2, 0, 0, nx1, ny1))
168:                     (*drawLine)(x1, y1, x2, y2);
169:                 x1=x; y1=y;
170:             }
171:         }
172:         free(P); next=line->next; free(line); line=next;
173:     }
174: }
175:
176: Raster(readPoint, xL, yL, xR, yR, nx, ny, drawBoundary, drawInside)
177: {
178:     (*readPoint)(); nx, ny;
179:     double
180:     xL, yL, xR, yR;
181:     void
182:     {
183:         *line=ReadLine(readPoint, xL, yL, xR, yR, nx, ny); *next;
184:         int
185:         idx, y, x1, x2, nx1=nx-1;
186:         if (drawInside=NULL) RasterInside(line, nx, ny, drawInside);
187:         if (drawBoundary=NULL)
188:             DrawLine(line, nx, ny, 1, drawBoundary);
189:         else while (line=NULL) {
190:             if (drawInside=NULL)
191:                 for (idx=0; idx<NUM; idx++)
192:                     if ((y=P[idx].y)-P[idx+1].y && y>0 && y<ny)
193:                         if ((x1=P[idx].x)<(x2=P[idx+1].x)) {
194:
195:             if (x1<nx && x2>0)
196:                 (*drawInside)((x1<0)?0:x1.y, (x2>nx)?nx1:x2.y);
197:             }
198:             else
199:                 if (x2<nx && x1>0)
200:                     (*drawInside)((x2<0)?0:x2.y, (x1>nx)?nx1:x1.y);
201:             free(P); next=line->next; free(line); line=next;
202:         }
203:     }
204:     Draw(readPoint, xL, yL, xR, yR, nx, ny, drawLine)
205:     (*readPoint)(); nx, ny;
206:     int
207:     xL, yL, xR, yR;
208:     double
209:     (*drawLine)();
210:     DrawLine(readLine(readPoint, xL, yL, xR, yR, nx, ny), nx, ny, 0, drawLine);
211: }
212:
clip.c:
1:
2: #ifndef TYPE
3: #define TYPE int
4: #endif
5:
6: #define CLIP clipline
7:
8: static int
9: TYPE
10: {
11:     if (x1<xL) return(1); if (x1>xR) return(2); return(0);
12: }
13:
14: static void
15: TYPE
16: {
17:     if (*x1==x1)
18:         switch(side(*x2, xL, xR)) {
19:             case 1 : *y2=y1+((x2-y1)*(xL-x1))/(x2-x1); *x2=xL; break;
20:             case 2 : *y2=y1+((x2-y1)*(xR-x1))/(x2-x1); *x2=xR; break;
21:         }
22: }
23:
24: int
25: TYPE
26: *x1, *y1, *x2, *y2, xL, yL, xR, yR;
27: {
28:     if (side(*x1, xL, xR)&side(*x2, xL, xR) ||
29:         side(*y1, yL, yR)&side(*y2, yL, yR)) return(0);
30:     cut(*x1, *y1, x2, y2, xL, xR);
31:     cut(*x1, *y1, x2, y2, xL, xR); if (side(*x2, xL, xR) ||
32:         side(*y2, yL, yR)) return(0);
33:     cut(*x2, *y2, x1, y1, xL, xR);
34:     cut(*y2, *x2, y1, x1, yL, yR); if (side(*x1, xL, xR) ||
35:         side(*y1, yL, yR)) return(0); return(1);
36: }
213:
line.c:
1:
2: #define LINE drawline
3:
4: void
5: LINE(x1, y1, x2, y2, plot)
6: {
7:     int
8:     x1, y1, x2, y2;
9:     (*plot)();
10:     int
11:     dx1, dy1, dx2, dy2, m, p, e, x, y;
12:     if (x2>x1) { dx1=x2-x1; dx2=1; } else { dx1=x1-x2; dx2=(-1); }
13:     if (y2>y1) { dy1=y2-y1; dy2=1; } else { dy1=y1-y2; dy2=(-1); }
14:     x2=dx1; y2=dy1; m=2*dx1; e=m-dx1; p=e-dx1;
15:     for (y=y1, x=x1; x!=x2; x+=dx2)
16:         (*plot)(x, y); if (e>0) { y+=dy2; e+=p; } else e+=m;
17:     }
18:     else {
19:         y2=dy2; m=2*dy1; e=m-dy1; p=e-dy1;
20:         for (x=x1, y=y1; y!=y2; y+=dy2)
21:             (*plot)(x, y); if (e>0) { x+=dx2; e+=p; } else e+=m;
22:     }
23: }

```

```

22:     }
23: }
24: }

cmimage.h, cmimage.c, writetiff.h, writetiff.c:
これらのソースリストは、中野 (1995a) を参照のこと。

stderr.c:
1: #include <stdio.h>
2:
3:
4: void main(argc, argv)
5: int argc;
6: char **argv;
7: {
8:     int code;
9:
10:    if (argc==1)
11:        while ((code=getchar())!=EOF) fputc(code, stderr);
12:    else {
13:        while (--argc>0) { fputs(*(++argv), stderr); fputc(" ", stderr); }
14:        fputc("\n", stderr);
15:    }
16: }

append_box: execute after "chmod +x append_box"
1: #!/bin/csh
2: set awk='BEGIN { s=1 } { if ($1==0) print " " } { print; } NF=2; if (s==1) { { x1=$1; x2=$1; y1=$2; y2=$2; s=0 } else { y { if (x1>$2) x1=$1; if (x2<$1) x2=$1; if (y1>$2) y1=$2; if (y2<$2) y2=$2 } } } END { printf "4\n"; y { dx=x2-x1; dy=y2-y1; x1-dx, y1-dy; y { printf "%1f%1f\n", x1-dx, y1-dy; y { printf "%1f%1f\n", x2-dx, y1-dy; y { printf "%1f%1f\n", x2-dx, y2-dy; y { printf "%1f%1f\n", x1-dx, y2-dy; } } } } }'
14: # $awk
15:

make.ps, line: execute after "chmod +x make.ps, line"
1: #!/bin/csh
2: # usage : make.ps, line directory x1 y1 x2 y2 h0 v0 dh dv geology fault ("lay")
3: #
4: alias filter zcat # change this as suitable for your environment
5: #
6: if ($argv != 11 && $argv != 12) then
7:     stderr usage : make.ps, line directory x1 y1 x2 y2 y
8:     stderr usage : h0 v0 dh dv geology fault ("lay") ; exit 1
9: endif
10: #
11: set dir=$argv[1]
12: set area=$(argv[2] $argv[3] $argv[4] $argv[5])
13: set paper=$(argv[6] $argv[7] $argv[8] $argv[9]) ; set geology=$argv[10]
14: set fault=$argv[11]
15: if ($argv == 11) then
16:     eps.window $area $paper
17: else
18:     eps.window $area $paper $argv[12] ; if ($status != 0) exit 1
19: endif
20: echo $geology W | filter $ {dir} / [0-9]* | eps.draw $area
21: echo $fault W | filter $ {dir} / fault | eps.draw $area ; echo showpage

make.ps, rgb: execute after "chmod +x make.ps, rgb"
1: #!/bin/csh
2: # usage :
3: # make.ps, rgb directory x1 y1 x2 y2 h0 v0 dh dv palette fault coast ("lay")
4: #
5: alias filter zcat # change this as suitable for your environment
6: #
7: set faultcode=250
8: set coastcode=254
9: #

10: if ($argv != 12 && $argv != 13) then
11:     stderr usage :
12:     stderr make.ps, rgb directory x1 y1 x2 y2 y
13:     h0 v0 dh dv palette fault coast ("lay") ; exit 1
14: endif
15: #
16: set dir=$argv[1]
17: set area=$(argv[2] $argv[3] $argv[4] $argv[5])
18: set paper=$(argv[6] $argv[7] $argv[8] $argv[9]) ; set fault=$argv[11]
19: set coast=$argv[12]
20: if ($argv == 12) then
21:     eps.window $area $paper
22: else
23:     eps.window $area $paper $argv[13] ; if ($status != 0) exit 1
24: endif
25: #
26: eps.palette $argv[10] ; if ($status != 0) exit 1
27: #
28: foreach file ($ {dir} / [0-9]*)
29:     basename $file | awk -F. { printf "%s ", $1 }
30:     echo LS : filter $file | eps.fill $area ; echo LE
31: end
32: echo $fault W $faultcode LS : filter $ {dir} / fault | eps.draw $area ; echo LE
33: echo $coast W $coastcode LS : filter $ {dir} / coast | eps.draw $area ; echo LE
34: echo showpage

eps.window: execute after "chmod +x eps.window"
1: #!/bin/csh
2: # usage : eps.window x1 y1 x2 y2 h0 v0 dh dv ("lay")
3: #
4: set box=true # flag to draw box surrounding the map area
5: #
6: set usage='usage : eps.window x1 y1 x2 y2 h0 v0 dh dv ("lay")'
7: switch ($argv)
8:     case 8 :
9:         set lay=false ; breaksw
10:     case 9 :
11:         if ($argv[8] != lay) then
12:             stderr "$usage" ; exit 1
13:         endif
14:         set lay=true ; breaksw
15:     default :
16:         stderr "$usage" ; exit 1
17: endswh
18: #
19: set x1=$argv[1] ; set x2=$argv[3] ; set h0=$argv[5] ; set dh=$argv[7]
20: set y1=$argv[2] ; set y2=$argv[4] ; set v0=$argv[6] ; set dv=$argv[8]
21: #
22: set a1='echo $h0 | awk { printf "%1f\n", $1 *72 } ;'
23: set b1='echo $v0 | awk { printf "%1f\n", $1 *72 } ;'
24: set a2='echo $h0 $dh | awk { printf "%1f\n", ($1-$2)*72 } ;'
25: set b2='echo $v0 $dv | awk { printf "%1f\n", ($1-$2)*72 } ;'
26: #
27: cat <<fin
28: %!PS-Adobe-2.0 EPSF-2.0
29: %%BoundingBox: $a1 $b1 $a2 $b2
30: %%EndComments
31: %
32: /h0 $h0 72 mul def /dh $dh 72 mul def /v0 $v0 72 mul def /dv $dv 72 mul def
33: /x1 $x1 def /x2 $x2 def /y1 $y1 def /y2 $y2 def
34: /lay $lay def
35: /box $box def
36: %
37: /N { newpath } def /M { moveto } def /L { lineto } def /S { stroke } def
38: /W { x2 x1 sub h div y2 y1 sub v div add 2 div mul setlinewidth } def
39: %
40: lay { /h dv def /v dh def h0 dh add v0 translate 90 rotate } ifelse
41: { /h dh def /v dv def h0 v0 translate
42:   { N 0 0 M h 0 L h v L 0 0 L S } if
43: }
44: %
45: h x2 x1 sub dv v y2 y1 sub div scale
46: x1 neg y1 neg translate
47: N x1 y1 M x2 y1 L x2 y2 L x1 y2 L closepath clip
48: %
49: fin

eps.draw: execute after "chmod +x eps.draw"
1: #!/bin/csh

```



```

2: # usage : eps.fill x1 y1 x2 y2
3: #
4: if ($#argv != 4) then
5:   stderr usage : eps.fill x1 y1 x2 y2 ; exit 1
6: endif
7: #
8: set x1=$argv[1] ; set y1=$argv[2] ; set x2=$argv[3] ; set y2=$argv[4]
9: #
10: cat <<fin >eps.fill.awk
11: BEGIN {
12:   x1 = $x1 ; y1 = $y1 ; x2 = $x2 ; y2 = $y2
13:   fin <<'fin' >>eps.fill.awk
14:   cat <<'fin' >>eps.fill.awk
15:   start = 1 ; exist = 0 ; old = 0
16: }
17: NF == 1 {
18:   if (exist == 1) printf "S\n" ; start = 1 ; exist = 0 ; old = 0
19: }
20: NF == 2 {
21:   x = $1 ; y = $2
22:   if (x > x1 && x < x2 && y > y1 && y < y2) new = 1 ; else new = 0
23:   if (new == 1) {
24:     if (old == 1) {
25:       printf "%i %i L\n", x, y ; exist = 1
26:     } else {
27:       if (start == 1) {
28:         printf "N %i %i M\n", x, y ; start = 0
29:       } else {
30:         printf "N %i %i M %i %i L\n", x0, y0, x, y ; exist = 1
31:       }
32:     }
33:   } else {
34:     if (old == 1) {
35:       printf "%i %i L S\n", x, y ; exist = 0
36:     }
37:     x0 = x ; y0 = y ; old = new
38:   }
39: }
40: END {
41:   if (exist == 1) printf "S\n"
42: }
43: 'fin'
44: #
45: awk -f eps.draw.awk ; rm eps.draw.awk

```

eps.palette: execute after "chmod +x eps.palette"

```

1: #!/bin/csh
2: # usage : eps.palette
3: #
4: if ($#argv != 1) then
5:   stderr usage : eps.palette ; exit 1
6: endif
7: #
8: set palette=$argv[1] ; if (!(-e $palette)) then
9:   stderr palette file not found. ; exit 1
10: endif
11: cat <<'fin'
12: /paletteDict 3 dict def
13: paletteDict begin
14:   /F { closepath eofill } def
15:   /rgb 256 array def 0 1 255 { rgb exch [0 0 0 0.0] put } for
16:   /R68 { dup { 255 div } forall 4 -1 roll astore rgb 3 1 roll put } def
17:   %'fin'
18: 'fin'
19: #
20: set awk="(NF==4 && $1<256 && $2<256 && $3<256 && $4<256 { ' %
21:   printf \"%d %d %d RGB\n\", $1, $2, $3, $4 } )' ; awk \"$awk\" $palette
22: #
23: cat <<'fin'
24: end
25: %
26: /LS { gsave paletteDict begin rgb exch get aload pop setrgbcolor } def
27: /LE { end gstore } def
28: %
29: 'fin'

```

eps.fill: execute after "chmod +x eps.fill"

```

1: #!/bin/csh

```