パーソナルコンピュータ用 GEOLIS 検索利用プログラムの開発とその利用

菅原 義明*

Development and Application of Retrieval Program of GEOLIS for Personal Computer

Yoshiaki, SUGAWARA*

ABSTRACTS : GEOLIS FD is a floppy disk version of GEOLIS (Bibliographic data base for geology of Japan) and is distributed without cost for scientific use by Geological Survey of Japan. This article describes a retrieval program for searching data from the GEOLIS FD. The functions of the program are 1) to search data from GEOLIS FD using logical combination of keywords, 2) to format outputs, and that 3) output device file can be selected by user. The program is written by ANSI C language and does not use any nonstandard system call of MS-DOS. Consequently the program runs on almost all Japanese MS-DOS based personal computers.

KEY WORDS : Personal Computer, GEOLIS, Document Retrieval, MS-DOS

1 はじめに

地質調査所ではGEOLIS FD版を無償で試行配布している(本荘・菅原, 1990)。GEOLIS FD版は地質調査所で作成している日本地質文献データベース(GEOLIS)を簡略化してフロッピーディスクに収めたものである。1990年9月現在で約150件の申込に対して配布を行っている。GEOLIS FD版に採録されたデータ件数は約29,000件であり、フロッピーディスク8枚に収められている。その内容と簡単な利用方法は野呂・村田(1989)、採録範囲等の構築方法については本荘・菅原(1989)に詳しく述べてあるので参考にしたい。

GEOLIS FD版に採録されたデータをパーソナルコンピュータ上で手軽に利用できるように、市販されているリレーショナルデータベースを使用したGEOLIS FD版の検索システムについて報告した(菅原・本荘, 1990)。そのシステムは簡単に作成できる反面、高価なデータベースソフトを各自が購入することが前提である。またハードディスクが備わっていることが前提となってしまう、フロッピーディスクのみの機器構成システムでの検索はほとんど不可能である。そのためより汎用性の高い利用プログラムの必要性がある。また外部から要求されているのも、手軽に検索でき

る多くの機種に対応できるプログラムである。そこで、その要望に答えるための利用プログラムを作成した。本稿ではその利用プログラムの仕様と利用方法を中心に述べる。

2. プログラムの仕様

2. 1 開発のための前提条件

利用プログラムを作成するに当たって次の4点を可能にすることを前提にした。

(1) 日本語 MS-DOS が実行される機種で単独作動が可能なこと

プログラムの配布等を考えた場合、特別なシステムがなくても作動可能なことが必要である。また特定機種の機能に依存しないことも必要である。今回は、ANSI C で記述し Zortech 社の C コンパイラを利用した。実行ファイル(拡張子が exe や com のファイル)は特定ソフトやライブラリーを持たなくても実行が可能である。しかも MS-DOS の標準システムコールのみを使い、機種依存の機能を使っていないので日本語 MS-DOS の走るほとんどのパソコン上で作動する。

(2) 論理式がある程度自由に書けること

GEOLIS FD 版のデータは MS-DOS のテキスト形式で記述されている。MS-DOS の Find コマンドで検索すれば、1つの文字列だけの検索という単純な検索は可能だが、論理演算子など使用した多重検索はできない。バッチファイルを使用すれば多重検索は可能だが、一定の検索式以外を行おうとするとバッチファイルをそのつど更新する必要がある。

*地質調査所地質情報センター Geological Information Center,
Geological Survey of Japan, 1-1-3 Higashi, Tsukuba, Ibaraki
305, Japan.

ある。また検索する立場からは、検索データ、論理条件の変更などを行いながらの検索が多く、バッチファイルをそのつど更新することはわずらわしい。したがって論理式が自由に書ければ検索の幅が広がり、手軽な検索が可能である。

(3) 検索出力結果の出力機器を指定できること

検索結果は画面に出力するだけでなく、プリンターやファイルに出力することが多い。他の検索システムやデータベースシステムでは、それが可能である。画面出力のみであると情報はその場限りとなってしまう。またプリンターやファイルに出力することによって必要な情報だけ手元に残しておくことができ、効率のよい運用が可能である。

(4) 検索出力結果は整形して出力できること

GEOLIS FD 版のデータ内容は区切り記号を使用したデータの羅列で記述されているので検索結果がそのままでは見にくく、整形した出力が望ましい。

2. 2 論理演算子

本稿の検索プログラムでは検索文字列を MS-DOS のコマンドラインから与える方式、すなわちプログラム名のあとに検索文字列を与える方式をとっている。

論理演算子には次の 3 種類があり、検索文字列の中に記述できるようにしてある。なお論理演算子は、必ず半角文字とする。

- 1 論理積 &
- 2 論理和 /
- 3 否定 !

検索プログラムが実行されると与えられた検索文字列の中に論理演算子の文字パターンがあるかどうかを探す。もし論理演算子がなければ検索文字列全体と一致した文字列を探す。論理演算子があった場合は、論理演算子の前後で文字列をわけ、以下の手順で 1 行 (1 データレコード) づつ検索をおこなう。

(1) & がある場合

データレコード中に & の前の検索文字列が含まれる場合のみ後の検索文字列を探す。前の文字列がない場合は後の文字列は探さず検索結果は偽となる。後の文字列が含まれた場合のみ検索結果が真となる。

(2) / がある場合

データレコード中に / の前の文字列が含まれない場合のみ後の文字列を探す。前の文字列が含まれる場合は検索結果が自動的に真となるからである。後の文字列が含まれる場合は検索結果は真となる。後の文字列が含まれない場合のみ検索結果は偽となる。

(3) ! がある場合

! が検索文字列の一文字目にあった場合のみ論理演算子と

88C02272@野藤 孝裕, 升本 真二, 弘原海 清, 塩野 清治@NOTOU TAKAHIRO, → MASUMOTO SHINJI, WADATSUMI KIYOSHI, SHIONO KIYOJI@1988@断層不→ 連続を含む地層面の推定-最適化原理による地層面推定法の拡張用サブルー→ チンAD I S C O N T-@Numerical Estimation of the Faulted Bedding Plane → - BASIC Subroutine Program *DISCONT-@@情報地質@Geological Data→ Processing@13865-82@断層不連続, 地層面推定法
88C02273@小池 克明, 西山 孝, 大村 誠@KOIKE KATSUAKI, NISHIYAMA TAKASHI, → OOMURA MAKOTO@1988@ヒマラヤ山脈から天山山脈にかけての地形解析@ Topographic Analysis of the Himalaya-Tien Shan Area@情報地質@Geological→ Data Processing@13883-92@ヒマラヤ山脈, 天山山脈, 地形解析
88C02279@松江 千佐世, 坂巻 幸雄, 山田 直利@MATSUE CHISAYU, SAKAMAKI → YUKIO, YAMADANAOTOSHI@1988@改良型地質標本管理・検索システム (GE → MS-I) の開発@Revision of the Database System for the Geological → Museum Specimens - from "GEMS-I" to "GEMS-II"-@情報地質@Geological→ Data Processing@138187-203@地質標本管理・検索システム, → 88C02280@岩松 輝@IWAMATSU AKIRA@1988@小断層解析のトータルシステム → @Total System of Fracture Analysis - with Special Reference to Paleo→ stress Field-@情報地質@Geological Data Processing@138205-228@断層 → 解析, トータルシステム

第 1 図 GeoLis に収められたデータ形式

右向の矢印は行が続いていることを表わす。
実際のデータには含まれないことに注意。

禁則処理対象文字
行頭禁則
、 。 、 .] 」 】 』 }) " : ; ? あ い う え お や ゆ よ
行末禁則
【 「 『 ‹ { [(

第 2 図 禁則処理対象文字の一覧

なる。データレコード中に ! を除いた検索文字列を含まない場合に検索結果を真とする。検索文字列中にある場合はデータの一部分とする。例えば abc&delg は "abc" and "de!g" であって "abc" and "de" not "g" ではない。以上の原則をもとに検索文字列を探す。複数の論理演算子をもっている場合も同じ原則で行う。ただし優先順位はないので論理演算子の記載順に上記の手続きを実行する。

2. 3 検索結果の表示方法

検索結果が真となったデータを未整形のまま表示すると、GEOLIS FD 版本体のデータ内容がそのまま表示されるので、内容が把握しにくい (第 1 図)。ある程度の検索結果の整形が必要である。

またすべてのデータ項目の表示は必要ではない。例えば日本語論文の著者名のローマ字読み、英語論文の日本語資料名などがあてはまる。このようなものは省いたほうがよい。これらのことを念頭におきながら以下の原則をもとに表示を行なう。

(1) 参考文献リスト等に使用される必要最小限のデータを表示する。

日本語論文は日本語項目のみ、また英語論文は英語項目のみを表示する。またキーワードは文献検索の際必要になるが、表示の際には必要ない場合が多いので表示しない。

- A
[6]
福岡 敏夫, 藤田 和夫(1986): 福知山盆地の中部更新統, 第四紀研究, vol.24,
no.4, p.263-281-
- B
[2]
著者 : 卜部 卓, 大見 士朗
発行年 : 1985
表題 : D R 方式による小型で安価な微小地震観測システム
資料名 : 九州大学理学部島原地震火山観測所研究報告
巻数 : 13
ページ : 27-36
発行者・社 : 九州大学理学部地震火山観測所, 島原
キーワード : D R 方式, 微小地震観測システム

第 3 図 「gsea」と「gs」の出力形式 A. gsea による出力結果 B. gs による出力結果

A>gsea 検索文字列 検索対象ファイル名 -o 出力ファイル名
第1パラメータ 第2パラメータ 第3パラメータ 第4パラメータ

第 4 図 プログラム実行の際のパラメータの与え方

- (2) 表示する際, 見やすさのための禁則処理を行う。
括弧などは行の最初や, 行の最後にくると見苦しい。
そこで最低限の禁則処理を行なうようにした。禁則対象文字は第 2 図の通りである。また英文の場合単語の 1 文字だけ残して改行したり, 1 文字だけを改行する場合も禁則処理を行う。
- (3) ディスプレイ表示の際, 1 画面に収まる範囲で一旦停止し, キー入力によって, 次の表示に移る。それ以外でのファイル出力, プリンター出力の場合は検索が終わるまで出力を続ける。

3. プログラムの使用方法

プログラム実行は MS-DOS のコマンドラインにコマンドとパラメータを与える方式をとっている。プログラム名は gsea と gs である。gsea は参考文献リスト形式であり, gs は項目名を先にだしデータを記述する形式である(第 3 図)。

ここでは gsea を例にして説明する。検索の場合には第 4 図に示すように第 1 パラメータに検索文字列, 第 2 パラメータに検索対象ファイル名が必要である。第 3 パラメータを -o とすることによって, 第 4 パラメータに検索結果出力ファイル名を与えることができる。パラメータを全て省略して起動するとディスプレイに使用方法が表示される。第 3 パラメータ以下を省略すると検索結果がディスプレイに出力される。出力ファイル名にプリンターのデバイス名, 一般的には PRN を与えるとプリンター出力になる。PRN 以外の場合は新しいファイルを作成し, 検索結果を作成されたファイルに出力する。例えば -o KEKKA とすれば検索結果はファイル名 KEKKA に出力される。検索文字列は原則としてアポロストロフィで文字列を囲む。検索文字列の

中には論理演算子を含ませて記述することができる。例えば '山形県/古生物' 'CHIBA PREFECTURE&GROUND WATER' の形式で記述する。

検索文字列の中に論理演算子を記述した場合の検索方法は 2. 2 に述べている。検索文字列数の制限及び論理演算子数の制限はない。注意すべきことは論理演算子の優先順位がなく記述順にしか検索しないという点である。論理演算子の組合せで検索結果が違って来る(第 5 図)。

論理演算子の 1 つである「!」の使用も注意が必要である。例えば単に「山形県」という文字列を持つデータを省

第 5 図 論理演算子の組合せによる検索結果の違い

A. '地質図/システム&情報地質'

「地質図」の文字列があった場合は全て真となり, なかった場合は「システム&情報地質」の文字列を探す。後者の場合 & があるので「システム」と「情報地質」の両方があった場合を真とする。

B. '情報地質 & 地質図/システム'

「情報地質」の文字列がない場合は結果は偽となる。しかし / があるので「システム」の文字列を探す。「システム」があった場合は結果は真となる。また「情報地質」と「地質図」があった場合は真となり「システム」を探さずに結果は真となる。

- [5]
大塚 道男(1985): 余震についての研究—その1—大森公式の物理的解釈とその適用についての注意. 九州大学理学部島原地震火山観測所研究報告, vol.13, p.11-20, 九州大学理学部島原地震火山観測所, 島原
- [6]
福間 敏夫, 藤田 和夫(1986): 福知山盆地の中部更新統. 第四紀研究, vol.24, no.4, p.263-281, 日本第四紀学会, 東京
- [7]
岩内 明子, 長谷 義隆(1986): 大分県阿蘇野層(中期更新世)の珪藻化石群集. 第四紀研究, vol.24, no.4, p.283-293, 日本第四紀学会, 東京
- [8]
野村 亮太郎, 田中 真吾(1986): 兵庫県東部の山間低地に発見された大山起原の火山灰層. 第四紀研究, vol.24, no.4, p.301-307, 日本第四紀学会, 東京
- [9]
大上 和良, 町田 瑞男, 吉田 充(1986): 岩手県, 胆沢扇状地地域に分布する火山灰のフィッシュトラック年代. 第四紀研究, vol.24, no.4, p.309-313, 日本第四紀学会, 東京
- [10]
岡村 聡(1986): 西北海道寿都半島の新第三紀火山岩類. 地質学雑誌, vol.92, no.2, p.91-108, 日本地質学会, 東京
- >next?(Y/N)

第6図 検索結果の表示 1画面を表示すると一旦表示をとめる. ---> next (Y/N)の後に Y か y を入力するとプログラムは検索を続行する. それ以外は検索をそこで終了する.

く場合は '!山形県' と記述する. しかし他の演算子と組み合わせる際は必ず「&」と「!」の組み合わせが必要である. 例えば, 「日本海」という文字列を持つデータから「山形県」の文字列を持つデータを省く場合, 「日本海&!山形県」と記述する. 検索対象ファイル名には GEOLIS FD 版のファイル名を記述する. ファイル名には MS-DOS のワイルドカード(複数ファイルを対象にする場合に使用する特殊文字)が使用できるのでフロッピー 1 枚に収められた全てのファイルを検索対象にすることができる. ハードディスクにファイルをすべて移してあれば GEOLIS の全てを検索対象とすることができる. 例えば geolis.1 ~ geolis.10 のファイルがディスクに存在する場合 geolis.? とすれば geolis.1 ~ geolis.9 が対象になり, geolis.1? とすれば geolis.10 ~ geolis.19 が対象となり, geolis.* とすればファイルの全てが対象となる. 出力ファイル名が省略されるとディスプレイに検索結果が表示される. ディスプレイ表示の際は, 一画面が表示されると一旦表示をとめる. そこで続けるかどうかを確認するメッセージを出す(第6図). 続ける場合は再度検索を続行する. 続けない場合は検索をそこで終了する. 出力ファイル名を与えると検索対象ファイルのすべてを検索するまで中断しない. プリンター出力の際も出力を続ける.

検索時間は複雑な論理式を使用しても GEOLIS FD 版 1 枚(データ件数約 3800 件)でおおよそ約 6 分である. 但し, プリンター出力時間はプリンターの印字性能に左右される.

4. 終わりに

この利用プログラムの作動確認は, 日本電気 PC9801RX・VX・VM・UV, 富士通 FMR-60・50, 東芝 J3100GT・SL, Dynabook286, およびエプソン PC286 で行なった. これらのすべての日本語 MS-DOS 上で完全に作動した.

このプログラムは GEOLIS FD 版の配布を受けた方々で希望者にはすでに無償配布を行なっている. 本稿を読んでプログラムご希望の方がおられたら, 地質調査所地質情報センター資料情報課までご連絡して下さい. 入手可能である. 利用してプログラムのバグや使い勝手などご意見があれば併せて筆者までご連絡いただきたい. 最後に利用プログラム作成にあたり当所情報解析課の野呂博士の適切な助言とご指導に深く感謝する.

文 献

- 本荘時江・菅原義明(1989) 日本地質文献データベース (GEOLIS) の紹介 -- GEOLIS FD 版の公開にあたって --. 地質ニュース, 420, 38-42.
- 本荘時江・菅原義明(1990) 日本地質文献データベース GEOLIS の構築およびそのフロッピーディスク試作版. 情報地質, 1, 51-57.
- 野呂春文・村田泰章(1989) GEOLIS フロッピーディスク公開版利用法と作成に関わる問題. 地質ニュース, 420, 26-32.
- 菅原義明・本荘時江(1990) 日本地質文献データベース GEOLIS フロッピーディスク試作版 -- パソコンによるリレーショナルデータベースの開発 --. 情報地質, 1, 59-65.

付 録

プログラムリスト (1)

```

/*
 * gsea-- G e o l i s のデータを検索・表示するためのプログラム
 *
 * 使い方例
 *
 * (1) gsea ファイル名
 * データが1つ表示されると止まるので
 * 次を見るには、何かキーを押す。
 *
 * (2) gsea ファイル名 -o 出力ファイル名
 * 何も表示されずに、結果が出力ファイルに
 * 書き込まれて終了する。
 *
 * 検索文字列
 *
 * * 検索文字列は、一重引用符で囲む : 例 '鈴木 昭元'
 * * ただし例外として空白を含まない検索文字列のみ
 * * 一重引用符で囲まなくてよい。
 *
 * * 検索文字列「A」の否定は、文字列「!A」である。
 * * 「A」を含まないデータを探索する。 : 例 !鈴木
 *
 * * 複数の検索文字列を、and または or 結合して
 * * 検索文字列とすることができる。
 * * and は、「&」, or は、「/」である。 : 例 '鈴木 昭元&北海道'
 * * '飛騨/愛媛'
 *
 * ファイル名
 *
 * * ファイル名は、ワイルドカード表現が使える : 例 b:geolis.?
 * * 例 b:geolis.*
 *
 */
#include <stdio.h>
#include <jstring.h>
#include <string.h>
#include <conio.h>

#define LENGTH4096
#define MAXFIELD 100
#define PATLEN 50
#define PATNUM 10

#define ANDCHAR '&'
#define ORCHAR '/'
#define NOTCHAR '!'
#define DELIMITER " "
#define LEN 50
#define INDENT 20
#define NAGASA 74

extern int logical_combination(int kekka, int akekka, int number);
extern int match(char *str1, char *str2, int revflag);
extern int amatch(char *str1, char *str2, int revflag);
extern int pattern_touroku(char *str);
extern int string_pattern_touroku(char *str, int n);
extern int operator_touroku(int op, int n);
extern int out_string(char *str, FILE *fout);
extern int out_string_format(int len, int indent, char *str, FILE *fout);
extern int str_field(char *str, char *delim, char **rstr);
extern int out_filename(int argc, char **argv);

char *substring[MAXFIELD]; /* 野呂・村田 (1989) 地質ニュース, /*
/* 420号, 26-32. の /*
/* 第2巻に記載されたn番目のフィールド /*
/* のデータは、substring[n-1]でポイント /*
/* される文字列に格納されている /*

struct Pattern{
    char pstring[PATLEN];
    int revflag;
    int match;
} pattern[PATNUM];

int number_of_pattern = 0;

struct Operator {
    int op;
    int left;
    int right;
} operator[PATNUM];

int number_of_and = 0;
int number_of_or = 0;

int find_number = 0;
int count=0;
int flag=0;

main(argc, argv)
int argc;
char *argv[];
{
    int i;
    int onum;
    int end;
    FILE *fp, *fout;

    if(argc < 2){
        help();
        exit(1);
    }

    if(argc == 2){
        help2();
        exit(1);
    }

    pattern_touroku(argv[1]);

    onum = out_filename(argc, argv);
    if(onum == (argc - 1)){
        end = argc - 2;
        if((fout = fopen(argv[onum], "w")) == NULL)
            exit(fprintf(stderr, "%s cannot open\n", argv[onum]));
        else{
            end = argc;
            fout = stdout;
        }

        if(argc < 3){
            fp = stdin;
            fgrep(fp, fout);
        }
        else {
            for(i = 2; i < end; i++){
                if((fp = fopen(argv[i], "r")) == NULL){
                    exit(fprintf(stderr,
                        "%s not found\n", argv[i]));
                    continue;
                }
                fprintf(stderr, "-- %s --\n", argv[i]);
                fgrep(fp, fout);
                fclose(fp);
            }
            fclose(fout);

            fprintf(stderr, "検索ボタン %s にマッチするデータは %d 件でした\n",
                argv[1], find_number);
        }
    }

    int pattern_touroku(str)
    char *str;
    {
        int num, i, j;
        char tmp[PATLEN];
        int loc[20];

        loc[0] = -1;
        j = 1;
        for(i = 0; str[i] != '\0'; i++){
            if((str[i] == ANDCHAR) || (str[i] == ORCHAR))
                loc[j++] = i;
            loc[j] = i;
            num = j;
            number_of_pattern = num;

            for(i = 0; i < num; i++){
                for(j = loc[i]+1; j < loc[i+1]; j++){
                    tmp[j] = (loc[i+1] == str[j]);
                    tmp[loc[i+1] - (loc[i] + 1)] = str[j];
                    string_pattern_touroku(tmp, i);
                }
            }
            for(i = 0; i < num; i++){
                if(loc[i+1] != '\0')
                    operator_touroku(str[loc[i+1]], i);
            }
            return(0);
        }

    int string_pattern_touroku(str, n)
    char *str;
    int n;
    {
        if(str[0] != NOTCHAR){
            strcpy(pattern[n].pstring, str);
            pattern[n].revflag = 0;
        }
        else{
            strcpy(pattern[n].pstring, str+1);
            pattern[n].revflag = 1;
        }
        return(n);
    }

    int operator_touroku(Op, n)
    int Op;
    int n;
    {
        operator[n].op = Op;
        if(Op == ANDCHAR)
            number_of_and++;
        else
            number_of_or++;
        return(n);
    }

    int logical_combination(kekka, akekka, n)
    int kekka;
    int akekka;
    int n;
    {
        int Op;
        int ret;

        Op = operator[n-1].op;
        if(Op == ANDCHAR)
            ret = kekka & akekka;
        else if(Op == ORCHAR)
            ret = kekka + akekka;
        return(ret);
    }

    int match(str)
    char *str;
    {
        int kekka;
        int akekka[PATNUM];
        int i;

        for(i = 0; i < number_of_pattern; i++){
            akekka[i] =
                amatch(str, pattern[i].pstring, pattern[i].revflag);
            if(akekka[i] && (number_of_and == 0))
                return(1);
            if((akekka[i] == 0) && (number_of_or == 0))
                return(0);
        }

        kekka = akekka[0];
        for(i = 1; i < number_of_pattern; i++)
            kekka = logical_combination(kekka, akekka[i], i);

        return(kekka);
    }

    int amatch(str1, str2, revflag)
    char *str1;
    char *str2;
    int revflag;
    {
        int res;

        if(!strcmp(str1, str2) != NULL){
            if(revflag == 0)
                res=1;
            else
                res=0;
        }
        else {
            if(revflag == 0)
                res = 0;
            else
                res = 1;
        }
        return(res);
    }
}

```


プログラムリスト (3)

```

case 0x41 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x42 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x43 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x44 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x45 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x46 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x47 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x48 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x49 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x60 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x66 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
case 0x68 :
    putc(outbuf[i+1],fout);
    i++;
    putc(outbuf[i+1],fout);
    i++;
    sjls=0;
    break;
default:
    break;
}
if(outbuf[i+1] & 0x83){
    switch(outbuf[i+2]){
        case 0x42 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x44 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x46 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x48 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x62 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x83 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x85 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        case 0x96 :
            putc(outbuf[i+1],fout);
            i++;
            putc(outbuf[i+1],fout);
            i++;
            sjls=0;
            break;
        default:
            break;
    }
}
if(sjls == 0){
    if(outbuf[i+2] == ' '){
        if(outbuf[i+2] == 'W0'){
            putc(outbuf[i+1],fout);
            i++;
        }
        putc('Wn', fout);
        count++;
        for(column = 0; column < 4; column++){
            putc(' ', fout);
            if(fout==stdout){
                if((count == 22)&&(flag == 1)){
                    telshi(fout);
                    flag=0;
                }
            }
        }
    }
    else {
        column++;
        i++;
    }
    return(i-1);
}
hyouji(buf, fout);
char *buf;
FILE *fout;
{
    int num;
    int i;
    char *loc;
    char dummy[100];
    char outbuf[1024];
    num = str_field(buf, DELIMITER, substring);
    if(fout==stdout){
        if((count == 22)&&(flag == 0)){
            telshi(fout);
        }
    }
    count++;
    sprintf(outbuf, "[Xd]Wn", ++find_number);
    flag=1;
    author(outbuf);
    year(outbuf);
    title(outbuf);
    magazin(outbuf);
    series(outbuf);
    volume(outbuf);
    number(outbuf);
    page(outbuf);
    where(outbuf);
    put_string(outbuf, fout);
    putc('Wn', fout);
    count++;
}
telshi(fout)
FILE *fout;
{
    int x;
    for(intf(stderr, "--->next?<Y/N>");
    while(!kbhit())
        ;
    x= getch();
    if((x==0x6e)||x==0x4e){
        exit(1);
    }
    printf("XsWn", fout);
    count=0;
}
author(outbuf, dummy);
char *outbuf;
char *dummy;
{
    int y=0;
    if(substring[1][0]){
        strcat(outbuf, substring[1]);
    }
    else if(substring[2][0])
        strcat(outbuf, substring[2]);
    else if(substring[3][0])
        strcat(outbuf, substring[3]);
    else if(substring[4][0])
        strcat(outbuf, substring[4]);
    else if(substring[5][0])
        strcat(outbuf, substring[5]);
    else if(substring[6][0])
        strcat(outbuf, substring[6]);
    else if(substring[7][0])
        strcat(outbuf, substring[7]);
    else if(substring[8][0])
        strcat(outbuf, substring[8]);
}
year(outbuf)
char *outbuf;
{
    if(substring[9][0]){
        strcat(outbuf, "(");
        strcat(outbuf, substring[9]);
        strcat(outbuf, ")");
    }
}
title(outbuf)
char *outbuf;
{

```

プログラムリスト (4)

```

    if(substring[10][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[10]);
    }
    else if(substring[11][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[11]);
    }
}

series(outbuf)
char *outbuf;
{
    if(substring[12][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[12]);
    }
    else if(substring[13][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[13]);
    }
}

magazin(outbuf)
char *outbuf;
{
    if(substring[14][0]){
        if(substring[10][0]||substring[11][0]){
            strcat(outbuf, " ");
        }
        else {
            strcat(outbuf, " ");
        }
        strcat(outbuf, substring[14]);
    }
    else if(substring[15][0]){
        if(substring[10][0]||substring[11][0]){
            strcat(outbuf, " ");
        }
        else{
            strcat(outbuf, " ");
        }
        strcat(outbuf, substring[15]);
    }
}

volume(outbuf)
char *outbuf;
{
    if(substring[16][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[16]);
    }
}

number(outbuf)
char *outbuf;
{
    if(substring[17][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[17]);
    }
}

page(outbuf)
char *outbuf;
{
    if(substring[18][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[18]);
    }
}

}

}

where(outbuf)
char *outbuf;
{
    if(substring[19][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[19]);
    }
    else if(substring[20][0]){
        strcat(outbuf, " ");
        strcat(outbuf, substring[20]);
    }
}

help()
{
    printf(stderr,
    "使い方 (1) gsea 検索文字列 ファイル名 %n";
    printf(stderr,
    "データが一画面出力されると止まるので %n";
    printf(stderr,
    "次を見るには、何かキーを押す。 %n";
    printf(stderr,
    "使い方 (2) gsea 検索文字列 ファイル名 -o 出力ファイル名 %n";
    printf(stderr,
    "何も出力されずに、結果が出力ファイルに %n";
    printf(stderr,
    "書き込まれて終了する。 %n";
    printf(stderr,
    "(プリンター出力の場合は、出力ファイルに PRN と指定して下さい %n");
    printf(stderr,
    "検索文字列の書式 %n";
    printf(stderr,
    "検索文字列は、一重引用符で囲む : 例 '鈴木 樹元' %n";
    printf(stderr,
    "ただし例外として検索文字列を含まない検索文字列のみ %n";
    printf(stderr,
    "一重引用符で囲まなくてもよい。 %n";
    printf(stderr,
    "★ 検索文字列 「A」の否定は、文字列「!A」である。 %n";
    printf(stderr,
    "「A」を含まないデータを検索する。 : 例 !鈴木 %n";
    printf(stderr,
    "★ 複数の検索文字列を、and または or 結合して %n";
    printf(stderr,
    "検索文字列とすることが出来る。 %n";
    printf(stderr,
    "and は、「&」、or は「/」である。 : 例 '鈴木 樹元&北海道' %n";
    printf(stderr,
    "★ ファイル名は、ワイルドカードが使える : 例 b:geolis.7 %n";
    printf(stderr,
    "例 b:geolis.* %n";
    printf(stderr,
    "★★対象ファイル名が指定されていないか、検索式が間違っています!★★ %n";
    printf(stderr,
    "使い方 gsea 検索文字列 ファイル名 -o 出力ファイル名 %n";
    printf(stderr,
    "★ ファイル名は、ワイルドカードが使用出来ます。 %n";
    printf(stderr,
    "例 b:geolis.7 あるいは b:geolis.* %n";
    }
}

```

要 旨

パーソナルコンピュータ用 GEOLIS 検索利用プログラムの開発とその利用 : 菅原 義明

GEOLIS FD 版は GEOLIS (日本地質文献データベース) の全データをフロッピーディスクに収めたもので、一般に配布されている。筆者は、GEOLIS FD 版のデータを検索するためのプログラムを開発し、その利用法について検討した。

そのプログラムは 1) 複数の KEYWORD の論理結合によってデータを検索できる 2) 検索結果出力を整型できる そして 3) 出力デバイスファイルを選択できる という特徴をもつ。また、プログラムは、ANSI C で書かれ、MS-DOS の標準システムコールを使用していないため、ほとんど全て日本語 MS-DOS が使用できるパーソナルコンピュータで作動する。

キーワード: パーソナルコンピュータ, GEOLIS, 文献検索, MS-DOS

(1990 年 10 月 25 日受付, 1990 年 12 月 19 日受理)