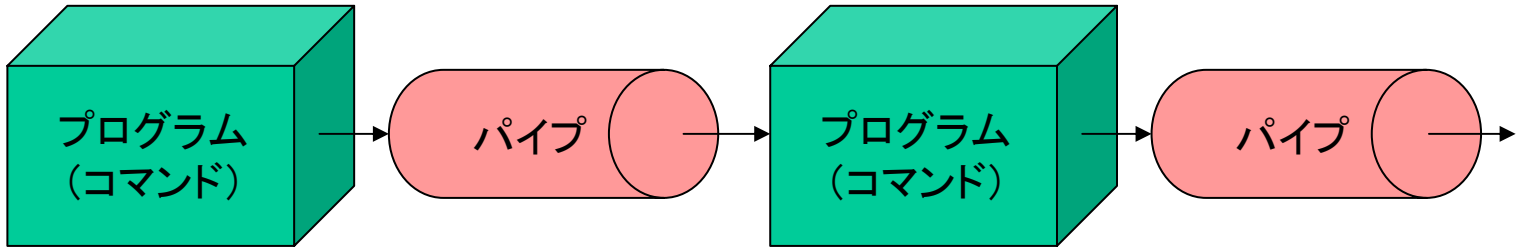
プログラミング１　第３回
復習（３）

- ファイル操作（プログラミング入門教科書P326～）

この資料にあるサンプルプログラムは
/home/course/prog1/public_html/2007/HW/lec/sources/
下に置いてありますから、各自自分のディレクトリにコピーして、コンパイル・実行してみてください。同時に **{in1,in2,in3,in4}.data** もコピーして使用してください

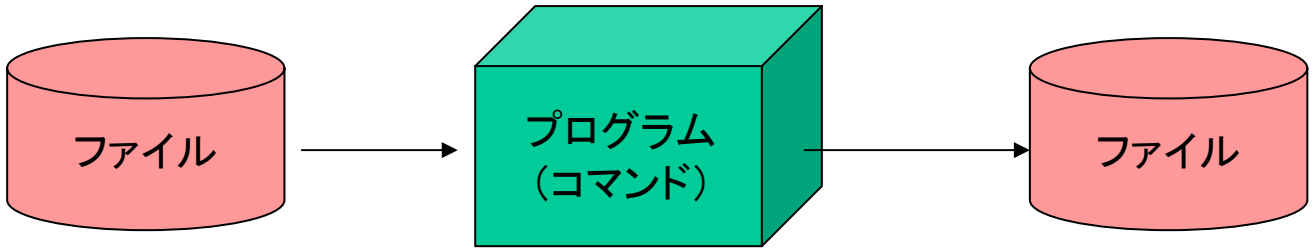
これまで習ったファイル操作の方法

- パイプ（プログラム同士）



プログラムの出力を次のプログラムの入力として使用する。
例：`cat file.data | wc -l`

- リダイレクト（プログラムとファイル間）



プログラムの標準入出力をファイルに切り替える
例：`ls -l > directory.data`

リダイレクト・パイプの例

- 読み込んだデータを表示し、文字数と行数も数えるcat+wcのようなプログラム

```
#include <stdio.h>

main()
{
  int  ccount = 0, lcount = 0, c;

  /* data input loop */
  while(1){
    c = getchar();
    if(c == EOF) break;
    putchar(c);
    ccount++;
    if(c == '\n') lcount++;
  }

  printf("\n%d characters\n",ccount);
  printf("%d lines\n",lcount);
}
```

文字数+1

改行なら
行数+1

getchar()：１文字読み込む

putchar(c)：１文字出力する

getcharの戻り値、putcharの引数はどちらもint型。ただ、int型でもchar型と同じように文字を扱う事が出来る。詳しくは前期教科書P139参照のこと。

パイプなら
cat in1.data | ./a.out

実行結果:

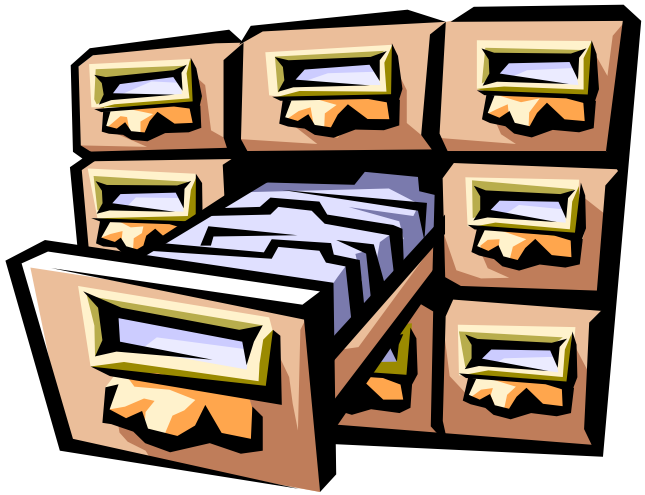
```
std1ss1{s1000000}1: ./a.out < in1.data
In 1984 Apple Computer introduced the Macintosh
desktop computer with a very "friendly" graphical
user interface. Graphical user interfaces(GUIs) began
to change the complexion of the software industry.

207 characters
4 lines
std1ss1{s1000000}2: wc in1.data
      4      28     207 in1.data
std1ss1{s1000000}3:
```

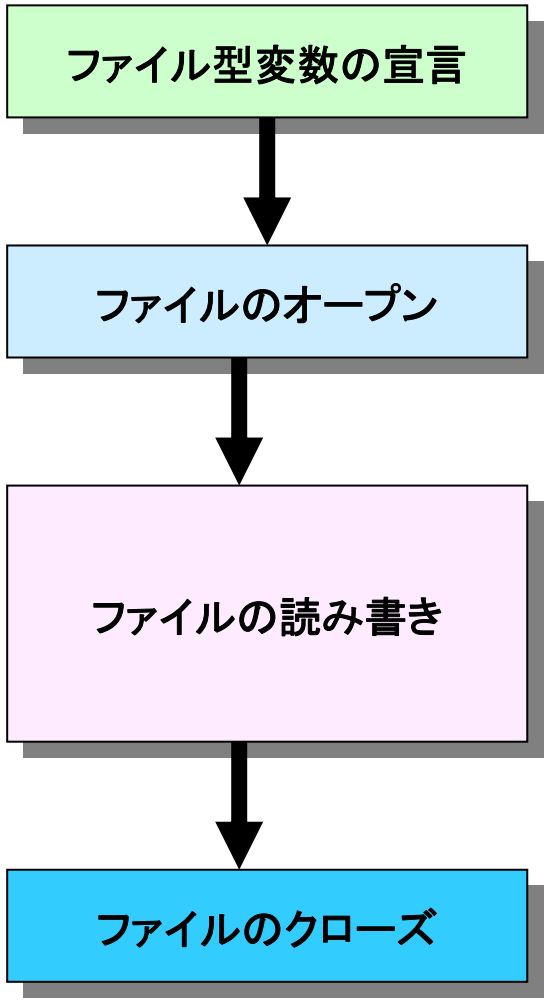
更に一般的ファイル処理へ

- これまで習ったパイプ・リダイレクトでも様々な事が出来る
- しかしパイプ・リダイレクトは標準入力・出力全部を変えてしまうものなので以下のような時は使用出来ない。
  - 入力や出力ファイルが複数個必要な場合
  - ファイルにあるデータの処理をディスプレー・キーボードで対話的に行いたい
- プログラムからファイルの読み書きが直接出来るような仕組みを「ファイル操作」と呼ぶ
- 以下でファイル操作を手順を追って紹介する

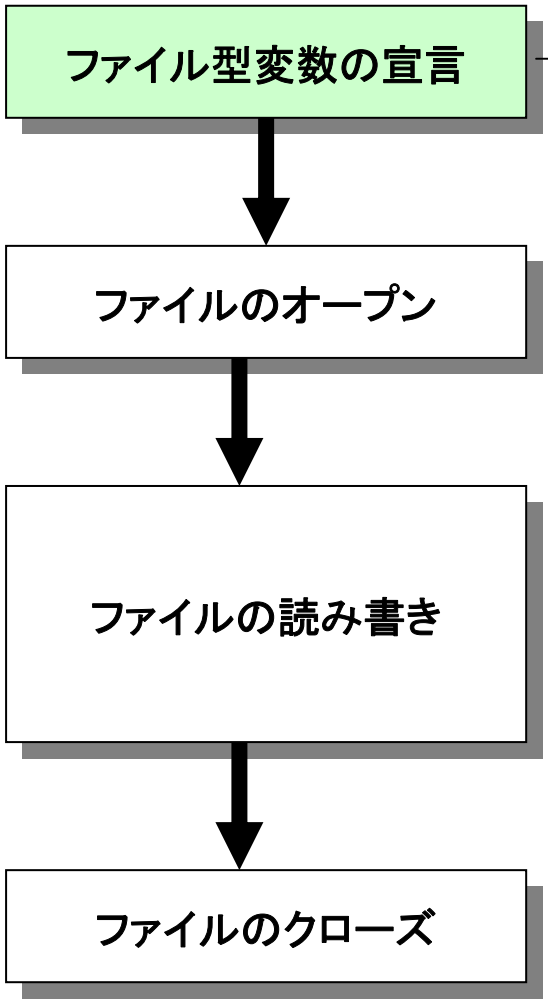
Cからのファイル操作の方法

- Cプログラム中からのファイル操手順は右の通り
- ファイル使用前にファイルのオープン
  使用後にファイルのクローズ
  を行う





宣言

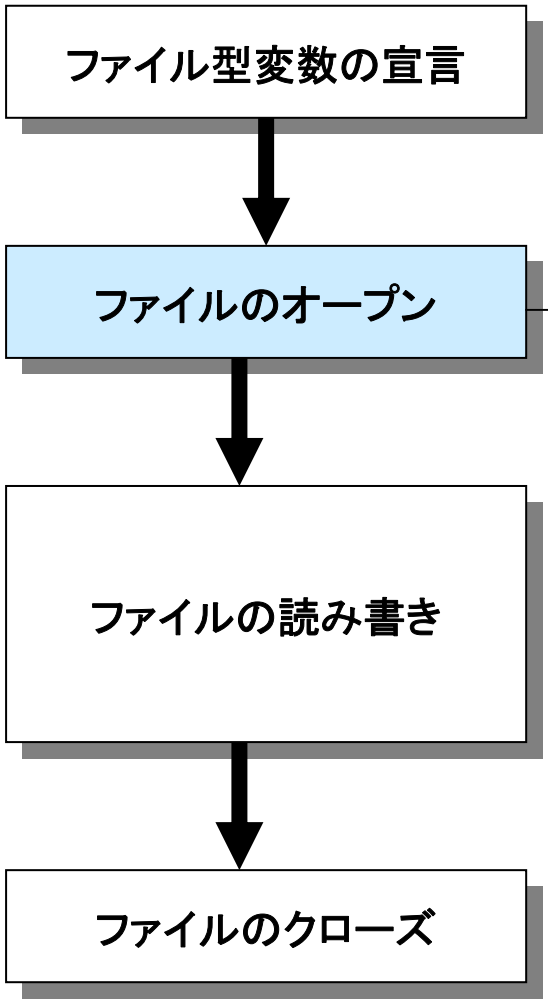


「FILE」は必ず大文字で！

ファイル変数は`FILE *fp` のように宣言する。(fpは変数名)
「*」の意味は今のところ知らなくても良い
「FILE *」で宣言された変数(この場合fp)を「ファイルポインタ」と呼ぶ。

以後の処理はこのファイルポインタに対して行う

ファイルオープン



ファイルのオープンは
`fp = fopen("in1.data","r")`のように書く

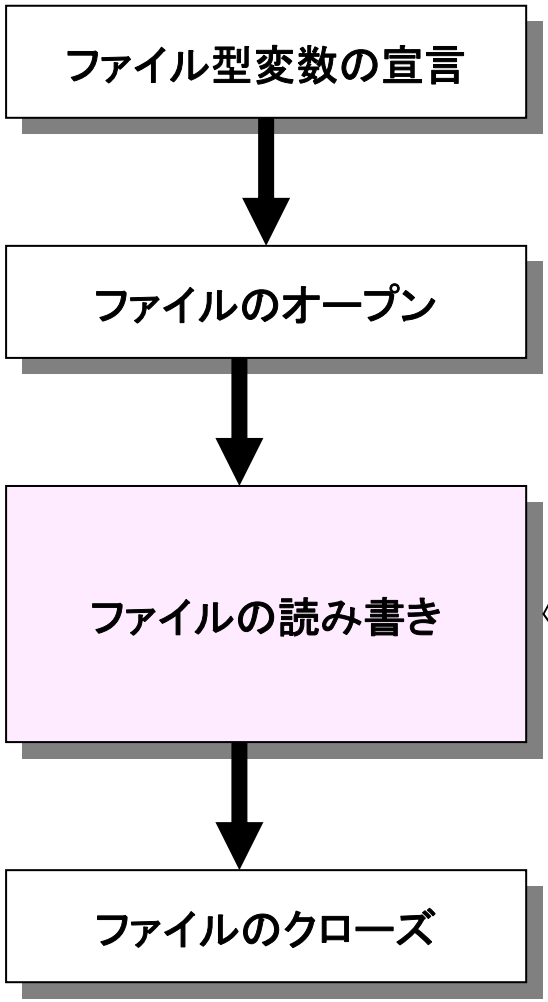
- fp：宣言したファイルポインタ
- ”in.data”：読み書き（オープン）するファイル名
- ”r”：読み出し指定（書き込む場合は”w”）
- fopen()の戻り値がNULLの場合は（ファイルがないなどの理由で）オープン失敗。
- エラー処理を含めて、以下のように使用する場合が多い

```
fp = fopen("in1.data","r");
if(fp == NULL){
  printf("エラーメッセージ");
  exit(1);
}
```

上2行は良く以下のようにも記述される（次ページ参照）

```
if((fp = fopen("in1.data","r"))== NULL){
```

ファイルの読み書き



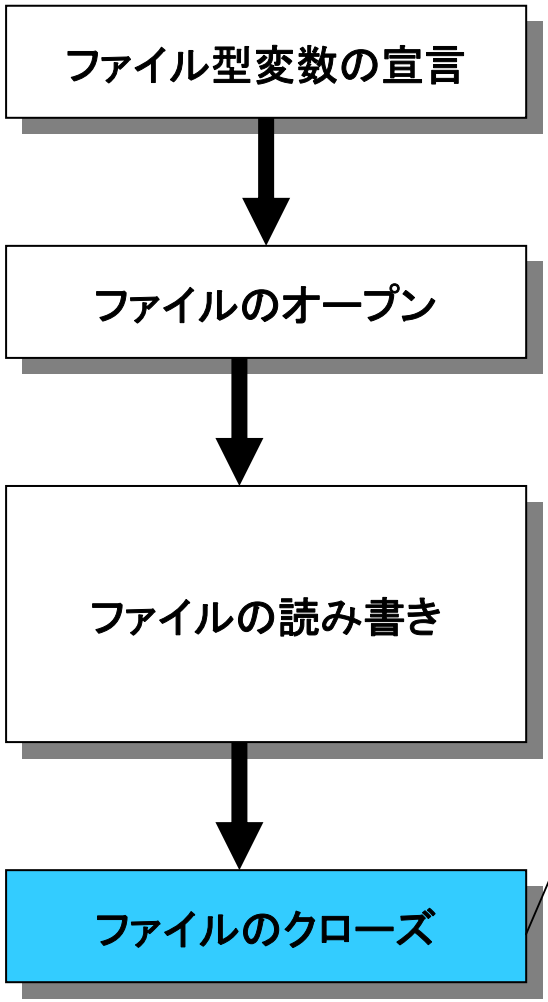
入力

```
fscanf(ファイルポインタ名,format,引数並び)
        例: fscanf(fi,"%d",&i);
fgetc(ファイルポインタ)
        戻り値はint型の文字コード

        例:(ファイルから1文字読んでディスプレーに表示)
            int c;
            c = fgetc(fi);
            putchar(c);
        (End Of Fileの場合はcに値EOFが入る)
```

出力

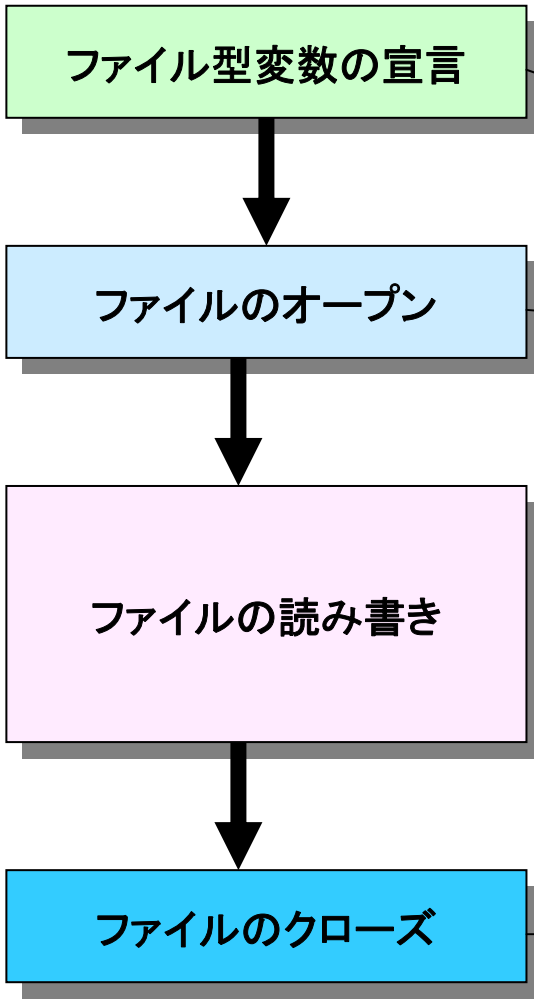
```
fprintf(ファイルポインタ,format,引数並び)
        例:  fprintf(fo,"%d¥n",i);
fputc(文字の入ったint型変数,ファイルポインタ)

        例:キーボードから1文字読んでファイルに書き込む
            int c;
            c = getchar();
            fputc(c,fo);
```

ファイルクローズ



オープンしたファイルをもう使用しない場合は、ファイルをクローズする（閉じる）。
ファイルのクローズは
`fclose(fp)` のように書く
（fpはオープンしたファイルポインタ）

ファイル操作の例1




```
#include <stdio.h>
#include <stdlib.h>
main()
{
  int  c;
  FILE *fp;

  /* file open & chek */
  fp = fopen("in1.data","r");
  if(fp == NULL){
    printf("Cannot open file in1.data\n");
    exit(1);
  }
  /* data input/output loop */
  while(1){
    c = fgetc(fp);
    if(c == EOF) break;
    putchar(c);
  }

  /* file close */
  fclose(fp);
}
```

ファイル操作の例2

- 読み込んだデータを表示し、文字数と行数も数えるプログラム（ファイルから読み込む）
- lec03-1.cのファイル入力版

実行結果:

```
std1ss1{s1000000}1: ./a.out
In 1984 Apple Computer introduced the Macintosh
desktop computer with a very "friendly" graphical
user interface. Graphical user interfaces(GUIs) began
to change the complexion of the software industry.

207 characters
4 lines
std1ss1{s1000000}2:
```

```c
#include <stdio.h>
#include <stdlib.h>

main()
{
  int  ccount = 0, lcount = 0, c;
  FILE *fp;

  if((fp = fopen("in1.data","r")) == NULL){
    printf("Cannot open file in1.data\n");
    exit(1);
  }

  while((c = fgetc(fp)) != EOF){
    putchar(c);
    ccount++;
    if(c == '\n') lcount++;
  }

  printf("\n%d characters\n",ccount);
  printf("%d lines\n",lcount);
  fclose(fp);
}
```

追加説明

- 複雑な条件式は一番中のカッコから見ていく(②ならまず代入)
- 複雑な条件式は見た目には複雑であるが、プログラムの構造自体はシンプルになる場合もあり、割と良く使用される。
- 例えばscanfで、エラーかEOFになったらループを脱出するプログラムは無限ループを使用すると①のように書ける。
- これを複雑な条件式を使って書き換えると②のようになるが、もう変数resultは必要無いので、③のように短くすっきりとする。
- 色々と自分で試してみると良い

```
while(1){
  result = scanf("%d",&data);
  if (result != 1) break;
  do_something(data);
}                                    ①
```

```
while((result = scanf("%d",&data)) == 1){
  do_something(data);
}                                    ②
```

```
while(scanf("%d",&data) == 1){
  do_something(data);
}                                    ③
```

ファイル操作の例3

- 2つのファイルを結合し、1つのファイルに書き込む
- 以下のコマンドと同じ働き
  cat in1.data in2data > out.data

```
#include <stdio.h>
#include <stdlib.h>

main(){
  int  c;
  FILE *fpin1, *fpin2, *fpout;

  /* 入力ファイル1 オープン */
  if((fpin1 = fopen("in1.data","r"))== NULL){
    printf("Cannot open file in1.data\n");
    exit(2);
  }
  /* 入力ファイル2 オープン */
  if((fpin2 = fopen("in4.data","r"))== NULL){
    printf("Cannot open file in4.data\n");
    exit(2);
  }
  /* 出力ファイル オープン */
  if((fpout = fopen("out.data","w"))== NULL){
    printf("Cannot open file out.data\n");
    exit(2);
  }

  /* 入力ファイル1/2からデータ入力,出力ファイルに書き込み */
  while((c = fgetc(fpin1)) != EOF)
    fputc(c,fpout);
  while((c = fgetc(fpin2)) != EOF)
    fputc(c,fpout);

  /* ファイルクローズ */
  fclose(fpin1);
  fclose(fpin2);
  fclose(fpout);
}
```

特殊なファイルポインタ

- 以下のファイルポインタは特別であるため、
  - 宣言が必要ない（暗黙に行ってくれる）
  - ファイルのオープン・クローズの指示が必要ない（自動的に行ってくれる）

- 特別なファイルポインタ名と機能
  - stdin：入力（キーボードからの標準入力）
  - stdout：出力（ディスプレーへの標準出力）
  - stderr：出力（ディスプレーへの標準エラー出力）

stdin/stdout/stderrの例

- 入力を全て大文字に変換する

```
#include <stdio.h>
#include <ctype.h>
main()
{
  int  c;
  while(1){
    c = fgetc(stdin);
    if(c == EOF) break;
    fputc(toupper(c), stdout);
  }
  fprintf(stderr,"Execution completed¥n");
}
```

toupper()は引数の文字コードを大文字のコードに変換する(ctype.h)

実行結果:

```
stdss1{s1000000}1: cat in2.data
This is a sample data
stdss1{s1000000}2: ./a.out < in2.data > out2.data
Execution completed
stdss1{s1000000}1: cat out2.data
THIS IS A SAMPLE DATA
stdss1{s1000000}3: ./a.out
Hello, how are you?
HELLO, HOW ARE YOU?
^D
Execution completed
stdss1{s1000000}4:
```

stdoutはファイルへ
stderrは画面

出力は全部画面へ

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
main()
{
  int  c;
  FILE *fin, *fout;
  fin = fopen("in2.data","r");
  if(fin == NULL){
    fprintf(stderr,"Cannot open in.data¥n");
    exit(1);
  }
  fout = fopen("out2.data","w");
  if(fout == NULL){
    fprintf(stderr,"Cannot open out.data¥n");
    fclose(fin);
    exit(1);
  }
  while(1){
    c = fgetc(fin);
    if(c == EOF) break;
    fputc(toupper(c), fout);
  }
  fprintf(stderr,"Execution completed¥n");
  fclose(fin);fclose(fout);
}
```

これは入力ファイル名
出力ファイル名を
指定したプログラム

ファイルからデータを読むリニアサーチ

```
#include <stdio.h>
#include <stdlib.h>
#define NUM 100
main(){
  int  data[2][NUM];
  int  count, i, ID, status;
  FILE *fp;

  /* file open */
  fp = fopen("in3.data","r");
  if(fp == NULL){
    printf("Cannot open file in3.data\n");
    exit(2);
  }
  /* data input loop */
  for(count = 0 ; count < NUM ; count++){
    status = fscanf(fp, "%d%d",
           &data[0][count], &data[1][count]);
    if(status == EOF) break;
    else if(status != 2){
      printf("File read ended abnormally\n");
      fclose(fp);
      exit(1);
    }
  }
  printf("%d data are in the file\n\n",count);

  /* file close */
  fclose(fp);

  /* data query infinite loop */
  while(1){
    printf("Enter student ID ->");
    status = scanf("%d",&ID);
    if(ID == 0 || status == EOF) break;
    else if (status != 1){
      printf("scanf ended abnormally\n");
      exit(3);
    }
    /* data query */
    i = 0;
    while ((i < count)&&(ID != data[0][i])) i++;
    if(i == count) printf("   Not found\n");
    else printf("   Found score : %d\n",
                                  data[1][i]);
  }
  printf("\n   Bye!\n\n");
}
```

プログラム実行結果

実行結果:

```
stdss1{s1000000}1: ./a.out
14 data are in the file

Enter student ID ->1002005
   Found score : 64
Enter student ID ->1004001
   Not found
Enter student ID ->1002009
   Found score : 88
Enter student ID ->0

   Bye!

stdss1{s1000000}2: ./a.out
14 data are in the file

Enter student ID ->1002014
   Found score : 72
Enter student ID ->x
scanf ended abnormally
stdss1{s1000000}3:
```

実行結果(続き):

```
stdss1{s1000000}3: cat in3.data
1002001 35
1002004 79
1002004 52
1002004 99
1002005 64
1002007 86
1002007 71
1002008 49
1002009 88
1002010 94
1002011 69
1002012 83
1002013 57
1002014 72
stdss1{s1000000}4:
```

文字配列

- intやfloat型の配列と同様に、文字型にも配列がある。
- 宣言:
  - `char word[10];` のように宣言する
  - この場合他の型の配列と同様にword[0]からword[9]までの10個の要素を持つ
- 初期化
  - 他の配列同様各要素を列挙することで配列の初期化を行うことが出来る
  - 例:`char abc[3] = {'a','b','c'};`
    この場合abc[0],abc[1],abc[2]は
    それぞれ'a','b','c'で初期化されている。

文字配列の処理例

```
stdss1{s1000000}1: cat in4.data
SPIM/SAL is a port, to the Apple Macintosh Personal Computer, of SPIM,
a MIPS R2000/3000 simulator that was written by James Larus for
instructional use at the University of Wisconsin Computer Sciences
Departement. Three other versions of SPIM exist: a character-cell
terminal version for Unix called "spim", an X windows version for
Unix called "xspim", and a version for PC's running WIN32S, also
called "SPIM/SAL." These versions are not identical, but they are
descended from a common ancestor and offer the same general
functionality. See the end of this document for information about
obtaining the different versions of SPIM.
stdss1{s1000000}2: ./a.out
.MIPS fo snoisrev tnereffid eht gniniatbo
 tuoba noitamrofni rof tnemucod siht fo dne eht eeS .ytilanoitcnuf
 lareneg emas eht reffo dna rotsecna nommoc a morf dednecsed
 era yeht tub ,lacitnedi ton era snoisrev esehT ".LAS/MIPS" dellac
 osla ,S23NIW gninnur s'CP rof noisrev a dna ,"mipsx" dellac xinU
 rof noisrev swodniw X na ,"mips" dellac xinU rof noisrev lanimret
 llec-retcarahc a :tsixe MIPS fo snoisrev rehto eerhT .tnemetrapeD
 secneicS retupmoC nisnocsiW fo ytisrevinU eht ta esu lanoitcurtsni
 rof suraL semaJ yb nettirw saw taht rotalumis 0003/0002R SPIM a
 ,MIPS fo ,retupmoC lanosreP hsotnicaM elppA eht ot ,trop a si LAS/MIPS
stdss1{s1000000}3:
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 1000
main(){
  char buf[MAX];
  int  i, n, c;
  FILE *fp;
  /* file open */
  fp = fopen("in4.data","r");
  if(fp == NULL){
    printf("Cannot open file in4.data\n");
    exit(2);
  }
  /* data input */
  for(n = 0 ; n < MAX ; n++){
    c = fgetc(fp);
    if(c == EOF) break;
    buf[n] = (char)c;
  }
  if(n == MAX){
    printf("Buffer overflow !\n");
    exit(3);
  }
  /* data output 逆順に出力 */
  for(i = n - 2 ; i >= 0 ; i--){
    fputc(buf[i],stdout);
  }
  printf("\n");
  /* file close */
  fclose(fp);
}
```

最後の改行は出力しない