担当： 富井尚志 (tommy@ynu.ac.jp)

第 6 回「ファイル入出力」

前回（第 5 回）「配列を扱うアルゴリズム(2)とポインタ」の復習

☆ 集合を扱う
・集合の表現，配列を使った表現，「要素の列」を「集合」にする，要素，積，和，差

☆ 文字列
・C 言語における文字列，文字列を扱うプログラム

☆ 計算機の記憶とポインタ
・計算機の記憶，データを計算機の記憶上に結びつけるには…，ポインタ

単項演算子 &　：変数のアドレスを与える
例） p = &c;　→　変数 p に c のアドレスが代入される

単項演算子 ＊　：間接演算子（逆参照演算子）．アドレスに適用すると変数を表す．
例） int *ip;　→　ip は int を指すポインタである　←！

◎ 前回の出席票の小テスト解答

練習問題：　２つの変数を渡すと，それぞれを２乗して返す関数 func1 を作成し，それを用いて 10 と 20 を２乗した値を表示する次のプログラムの［　　］の中に適切な記述を補って下さい．

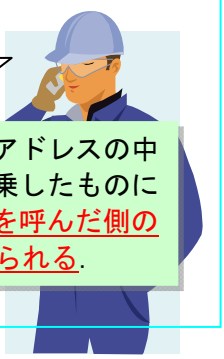
```
#include<stdio.h>

void func1( int *p1, int *p2 );

int main(void)
{
    int n1 = 10;
    int n2 = 20;

    printf("n1 = %d, n2 = %d¥n", n1, n2);
    func1( &n1, &n2 );
    printf("n1*n1 = %d, n2*n2 = %d¥n", n1, n2);
    return 0;
}

void func1( int *p1, int *p2 )
{
    *p1 = *p1  *  *p1;
    *p2 = *p2  *  *p2;
}
```

p1 および p2 はそれぞれ int 型変数へのポインタである．ここで p1, p2 は適当に与えた仮引数の名前で，func1 が呼ばれるときの変数（実引数）の名前とは無関係である．

２つの変数が入れてある箱の場所は，&n1 と &n2 なんで，ヨロシク！

int 型変数 n1 と n2 のアドレスを関数 func1 に渡している．

了解．じゃあ場所 p1(=&n1) と p2(=&n2) の箱の中身（*p1 と *p2）を書き換えとくんで，ヨロシク！



int 型変数へのポインタ p1, p2 が指すアドレスの中身（＝*p1, *p2）を，それぞれを 2 乗したものに置き換えている．これにより，func1 を呼んだ側の変数（n1 および n2）の値が書き換えられる．

第 6 回の始まり

ファイル入出力

☆ 動的データ構造としての「列構造」

- 「列（構造）」
  1. 「空列」 < > は「列」
  2. 値 x0 と「列」 < x1, x2, … > との連接 < x0, x1, x2, … > も「列」
- 配列に似ているが，**列の「長さ」は動的に変わり得る．**

☆ ファイル

- 大規模なデータを扱うためには，主記憶だけではなく，磁気ディスクや磁気テープなど**外部記憶装置**を使う．これらは列構造として考えられる．
- これらには，データの読みだし書き込み方法に関して**「制限」**がある．

- **「制限」＝「順アクセス」**
  - いかなる時でも直ちにアクセス可能な列内の要素は**「現在位置」にある一つだけ**
  - 「現在位置」は**「直後」の要素にのみ移動可能**（順番にデータを読むことしかできない）
  - 要素の**付加**は列の**「最後尾」のみ**．

- 磁気テープ　　→ 順アクセスのみ
  磁気ディスク　→ 各トラック（同心円上の記憶領域）は順アクセス
- これら記憶媒体は「列」の一つである**順ファイル(sequential file)**として位置付けられる．

☆ Unix/C におけるファイル

- **文字の列(text stream)**として統一されている．つまり，char の「列」である．
- ファイルに関する情報（「現在位置」など）は「　　」というデータ構造により表される．
- 特別なファイル --- **標準入力** と **標準出力**
  - プログラムにおける入出力を統一するための「ファイル」
  - 標準入力は入力専用，標準出力は出力専用
  - 常に使える状態にある．通常は，標準入力 =「キーボードからの入力」，標準出力 =「端末の画面」
- **標準入出力操作用関数**
  - int　getchar() : ファイル「　　」の「現在位置」を次の文字に進め，その文字を返す．int 型の値を返すのは，「ファイルの最後尾」を表すなど文字以外の情報を返す必要があるから．
  - int　putchar(int c) : 文字 c をファイル「　　」の「現在位置」＝「最後尾」に書き込み，「現在位置」を次に移動する．

この印の部分は講義中に自分で書いて下さい

あらかじめ用意されており，プログラマは自由に使える．

- 例 1　入力を出力に複写するサンプルプログラム　cat1.c
  - EOF は「ファイルの最後尾」(End Of File)を表す記号定数（通常 -1 という値）．getchar()はファイルの最後尾まで読み進むと，EOF という値を返す．
- 例 2　文字をカウントするサンプルプログラム　wc1.c
  - 改行文字（¥n，見えない）も 1 文字にカウントされていることに注意

☆ 一般的なファイルアクセス

- 標準入出力以外のファイルを利用する．
- ファイルへの入出力は「ファイルポインタ」を介して行なわれる．

- **ファイルの利用準備**
  - 関数 fopen を用い，ファイルを開く(オープンする)

    FILE *fopen(char *name, char *mode)

    name → ファイル名(パス名)，mode → "r" なら読みだし，"w" なら書き込み

    fopen：あらかじめ用意されている関数．
  - fopen の返す値＝「ファイルポインタ」(型 FILE *)
  - mode = "r" の時，すでにあるファイルの先頭に「現在位置」が設定される．
  - mode = "w" の時，新しくファイルが生成され（「空列」）「現在位置」が先頭に設定される．
- **ファイルの後片付け**
  - 関数 fclose を用い，すでに開かれているファイルを閉じる（クローズする）

    int　fclose(FILE *fp)

    fclose, getc, putc, fprintf：あらかじめ用意されている関数．
- **ファイルへの入出力**
  - int　getc(FILE *fp)：ファイルポインタ fp により示されるファイルの「現在位置」を次の文字に進め，その文字を値として返す．
  - int　putc(int c, FILE *fp)：ファイルポインタ fp により示されるファイルの「現在位置」＝「最後尾」に文字 c を書き込み，「現在位置」を次に移動する．
  - int　fprintf(FILE *fp, char *format, ...)：printf と同じ．違うのはファイルポインタ fp により示されるファイルの「　　」に書き込むこと．
- **ファイルに対する処理の一般形**

```
int main(void)
{
    FILE *fp;
    int c;

    fp = fopen(ファイル名,"r");
    while( (c = getc(fp) ) != EOF ) {
        文字 c に対する処理
    }
    fclose(fp);
    exit(0);
}
```

ファイルのオープン

ファイルポインタ fp の位置の文字を c として読み込み，それが EOF（＝ファイルの最後）になるまで，文字 c に対する処理を続ける．

ファイルのクローズ

シェルに値を返す標準関数．0：正常終了，0 以外：異常終了．return 0; と結果的に同じ．

- 例 3　入力を出力に複写するサンプルプログラム（ファイル名指定版）cat2.c

☆ コマンド行からの実行について

- Unix などのオペレーティングシステムでは，コンパイラにより生成された実行ファイルをコマンドとして実行することができる．
- 一番外側の関数である main がコマンド実行との間の仲立ちをする．
- 関数 main の引数
  - プログラム実行時のパラメータ（引数）が渡される．

```
int main(int argc, char *argv[])
{
   . . . . . .
   . . . . . .
}
```

要注意！

  - argc ＝ パラメタの数（　　を含む）
    argv[0], arg[1],... ＝ パラメタの値（文字列）
  - 例えば，

    cat3　test1.txt　testout.txt Enter

    というコマンド行で実行された場合，プログラム cat3 内では以下のように参照できる．

```
argc = 3,
argv[0] = "cat3",  argv[1] = "test1.txt",  argv[2] = "testout.txt"
```

超重要！

- 関数 main の返り値
  - main の中で return 文に出会うと，プログラムが終了．
  - オペレーティングシステム側にその値が通知される．→ プログラムの実行状況を表す．
  - 慣例として，値 0 が正常終了を表す．
  - システム関数 exit を呼んでも同じ．

- 例 4 入力を出力に複写するサンプルプログラム（ファイル名引数指定版）cat3.c

- 例 5 入力ファイルの行数，単語数，文字数を数えるサンプルプログラム wc2.c
  - 改行文字（¥n，見えない）も 1 文字にカウントされていることに注意

cat1.c

```
/*************************************************************
    アルゴリズムとデータ構造
    サンプルプログラム cat1.c
    <<ファイルの例: 標準入力を標準出力にコピー>>
    copyright (c) 1995,96,97 T.Mori <mori@forest.dnj.ynu.ac.jp>
*************************************************************/
#include <stdio.h>

int main(void)
{
  int c; /* 入力文字 */

  while (  (  c = getchar( )  )  !=  EOF  )
    /* EOF が現れるまで標準入力から文字を読み */
    putchar(c); /* それを印刷 */

  exit(0);
}
```

EOF=-1 と別のところで定義されており，プログラムは自由に使える．

標準出力（画面）に表示

システム関数（main 関数の戻り値=0（正常終了））．
異常終了のときは exit(1)などとする場合が多い．

【実行例のための準備】

1. test1.txt というファイルを以下の中身で用意

```
-----<test1.txt はじまり>-----
This is a test file.
This file contains two lines.
-----<test1.txt おわり>-----
```

2. 以下のコマンドでプログラムをコンパイルし，cat1 という名前の実行ファイルにする

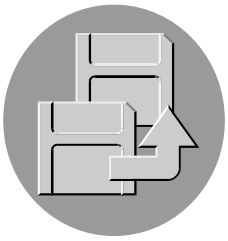
```
gcc  -o  cat1  cat1.c Enter
```

3. 以下のコマンドで実行（“<”により標準入力をファイル test1.txt に切替えている）

```
cat1  <  test1.txt Enter
```

【実行結果】以下の 2 行が画面に表示される．

```
This is a test file.
This file contains two lines.
```



wc1.c

```
/*************************************************************
     アルゴリズムとデータ構造
     サンプルプログラム wc1.c
     <<ファイルの例: 標準入力の文字数をカウントする>>
     copyright (c) 1995,96,97   T.Mori <mori@forest.dnj.ynu.ac.jp>
*************************************************************/
#include <stdio.h>

int main(void)
{
  int nc = 0; /* 文字数 */

  while (getchar() != EOF)
    /* EOF になるまで標準入力から文字を取得．その度に nc を 1 増加 */
    nc++;

  printf("%d¥n", nc);
  exit(0);
}
```

画面に文字数（=nc）を表示.

【実行例のための準備】

1. test1.txt というファイルを以下の中身で用意

```
-----<test1.txt はじまり>-----
This is a test file.
This file contains two lines.
-----<test1.txt おわり>-----
```

2. コンパイル（gcc　-o　wc1　wc1.c Enter）

3. 実行（wc1　<　test1.txt Enter）

【実行結果】次の数字が画面に表示される.

```
51
```

改行（¥n）など，見えない文字もカウントされている点に注意

cat2.c

```
/*************************************************************
     アルゴリズムとデータ構造
     サンプルプログラム cat2.c
     <<ファイルの例: 入力ファイルを出力ファイルにコピー>>
     copyright (c) 1995,96,97   T.Mori <mori@forest.dnj.ynu.ac.jp>
*************************************************************/
#include <stdio.h>

int main(void)
{
  int c;
  FILE *infp,*outfp;

  infp = fopen("test1.txt","r");
  /* test1.txt を読み出し用に開く．入力ファイル */
  outfp = fopen("testout.txt","w");
  /* testout.txt を書き込み用に開く．出力ファイル */

  while ((c = getc(infp)) != EOF)
    /* EOF が現れるまで入力ファイルから文字を読み */
    putc(c, outfp);                    /* 出力ファイルに書き込む */

  fclose(infp);                        /* 各ファイルを閉じる */
  fclose(outfp);

  exit(0);
}
```

読み込むファイルの名前

書き込むファイルの名前

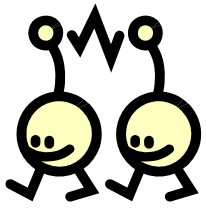
【実行例のための準備】

1. test1.txt を用意（内容はこれまでと同一）
2. コンパイル（gcc　-o　cat2　cat2.c Enter）
3. 実行（cat2 Enter）
4. testout.txt というファイルが生成される．

【実行結果】(testout.txt の中身)

```
This is a test file.
This file contains two lines.
```

test1.txt と同一．test1.txt → testout.txt とファイルがコピーされる．



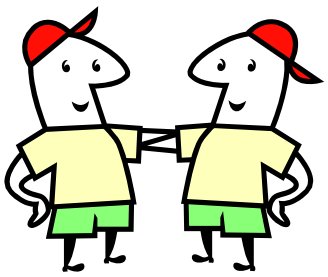
cat3.c

```
/*************************************************************
     アルゴリズムとデータ構造
     サンプルプログラム cat3.c
     <<ファイルの例: 入力ファイルを出力ファイルにコピー>>
     copyright (c) 1995,96,97   T.Mori <mori@forest.dnj.ynu.ac.jp>
*************************************************************/
#include <stdio.h>

int main(int argc, char *argv[])
{
  int c;
  FILE *infp,*outfp;

  if (argc != 3) { /* 引数の数が合わない時は，使い方を表示 */
    fprintf(stderr,"Usage: %s inputfile outputfile¥n", argv[0]);
    exit(1);
  }
  else {
    if ((infp = fopen(argv[1],"r")) == NULL) {
      /* 入力ファイルが開けない場合はエラー */
      fprintf(stderr,"%s: %s: No such file or directory¥n", argv[0],argv[1]);
      exit(1);
    }
    else if ((outfp = fopen(argv[2],"w")) == NULL) {
      /* 出力ファイルが開けない場合はエラー */
      fprintf(stderr,"%s: Cannot open %s¥n", argv[0],argv[2]);
      exit(1);
    }
    else {
      while ((c = getc(infp)) != EOF)
      /* EOF が現れるまで入力ファイルから文字を読み */
      putc(c,outfp);                    /* 出力ファイルに書き込む */
      fclose(infp);                      /* 各ファイルを閉じる */
      fclose(outfp);
      exit(0);
    }
  }
}
```

cat3 入力ファイル名 出力ファイル名 Enter

と入力して実行する．すなわち，
argc = 3,
argv[1] = "入力ファイル名",
argv[2] = "出力ファイル名"　　である．

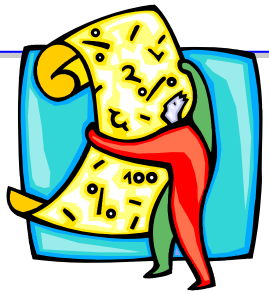
fail-safe(安全装置の)処理 : プログラムのユーザが誤った使用方法をしても大丈夫なようにすること．



48 【実行例のための準備】
49 1. test1.txt を用意（内容はこれまでと同一）
50 2. コンパイル（gcc　-o　cat3　cat3.c Enter）
51 3. 実行（cat3　test1.txt　testout.txt Enter）　実行方法に注意！
52 4. testout.txt というファイルが生成される.
53
54
55 【実行結果】(testout.txt の中身（＝test1.txt，コピーされている）)
56 This is a test file.
57 This file contains two lines.

応用問題（やりたい人だけやってみて下さい．難しくはありません.）:

英数字だけのテキストファイルがあるとします．そのファイル中の全ての文字を，改行(¥n)以外は全て 128 からそのアスキーコードを引いた値に変換して保存するプログラム（暗号化プログラム code.c）と，逆に暗号化されたファイルを通常のテキストファイルに戻すプログラム（復号化プログラム decode.c）を作成して実行してみて下さい．（なお，暗号化ファイルは普通の文字が制御記号などに置き換わるため，普通に画面に表示することはできなくなります.）



ファイルの入出力は，ある程度決まった処理ですので覚えてしまいましょう．特に，
・指定したファイルがないときの処理
・ EOF が来るまで処理を繰り返す部分
などは，これから皆さんが作るプログラムで有効に利用することができます.

wc2.c

```
/*************************************************************
アルゴリズムとデータ構造
サンプルプログラム wc2.c
<<ファイルの例: 入力ファイルの行数，単語数，文字数をカウント>>
copyright (c) 1995,96,97   T.Mori <mori@forest.dnj.ynu.ac.jp>
*************************************************************/
#include <stdio.h>

#define TRUE 1
#define FALSE 0

int main(int argc, char *argv[])
{
  FILE *infp; /* 入力ファイル用ファイルポインタ */
  int c;      /* 入力文字 */
  int nl = 0, nw = 0, nc = 0; /* 行数，単語数，文字数 */
  int whitespace = TRUE;
      /* 空白文字を読んでいる最中は真(単語を読んでいる間，偽) */

  if (argc == 1) { /* 引数なしの場合は*/
    infp = stdin;  /* 標準入力を入力ファイルとする */
  }
  else if (argc == 2) { /* 引数がある場合は */
    if ((infp = fopen(argv[1],"r")) == NULL) {
      /* 入力ファイルの名前とし，開く */
      fprintf(stderr,"%s: Cannot open %s¥n", argv[0],argv[1]);
      /* 開けない場合はエラー */
      exit(1);
    }
  }
  else {   /* 引数の数がおかしい場合は */
    fprintf(stderr,"Usage: %s [file]¥n", argv[0]); /* 使い方を表示 */
    exit(1);
  }

  while ((c = getc(infp)) != EOF) {
  /* EOF が現れるまで 1 文字づつ読みとり */
    nc++;                          /* 文字数を増やす */
    if (c == '¥n')                 /* 改行なら行数を増やす */
      nl++;
    if (c == ' ' || c == '¥n' || c == '¥t') /* 空白文字なら whitespace を真に */
      whitespace = TRUE;
    else if (whitespace) { /* 空白文字でなく，whitespace が真だった場合 */
      whitespace = FALSE;
      /* つまり，新たな単語が現れた場合，whitespace を偽にし */
      nw++;                /* 単語数を増やす */
    }
  }
```

異常終了で強制的に終了させる．これにより
プログラムの実行は終了する．

fail-safe 処理

```
  if (argc == 2) { /* ファイル名が与えられた場合 はそれも印刷 */
    printf("%8d%8d%8d %s¥n", nl, nw, nc, argv[1]);
    fclose(infp);
  }
  else
    printf("%8d%8d%8d¥n", nl, nw, nc);

  exit(0);
}
```

結果を画面に表示

【実行例のための準備】

1. test1.txt を用意（内容はこれまでと同一）
2. コンパイルし（gcc　-o　wc2　wc2.c Enter）
3. 実行（wc2　test1.txt Enter）

【実行結果】

次の数値・文字が画面に表示される（この場合はファイル名を test1.txt と入力した場合）

```
       2       10       51 test1.txt
```

変数 whitespace によって単語の切れ目を検出して，カウンタ nw をインクリメントしている．

infp		→								
ファイル中の文字		T	h	i	s		i	s		a
whitespace	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
nw	0	1	1	1	1	1	2	2	2	3

ファイル入出力に関する標準ライブラリ

（「プログラミング言語 C 第 2 版」B.W.カーニハン／D.M.リッチー著，石田晴久訳（共立出版）より）

ファイル入出力に関する標準ライブラリ中の代表的な関数を紹介しますのでプログラミングの際の参考にして下さい．これらは stdio.h で定義されています．また，ストリームとはデータの送出元および行先です．

◎ FILE *fopen(const char *filename, const char *mode)

指定されたファイルをオープンし，ストリームを返す．オープンできなければ返されるのは NULL である．Mode は "r", "w", "a", "r+", "w+", "a+"のいずれか．ただし "b"が mode の最初の文字の次に付いている場合はバイナリファイルを表す（ "rb", "wb"など）．

◎ int fclose(FILE *stream)

まだ書き出されていないデータを stream にはき出し，まだ読み込まれていないバッファ内の入力を捨て，自動的に割り当てられたバッファをすべて解放し，ストリームをクローズする．エラーが起きると EOF が，さもなければゼロが返される．

◎ int fprintf(FILE *stream, const char *format, ...)

format による制御のもとで，出力が変換されて stream への出力が行われる．返される値は書き出された文字の数で，エラーが起きると負の数となる．

◎ int sprintf(char *s, const char *format, ...)

¥0 を最後に付けた形で出力が文字列 s に書かれる点を除けば printf と同じ．s は結果を保持するのに十分大きくなければならない．返されるカウントには¥0 は含まれない．

◎ int fscanf(FILE *stream, const char *format, ...)

format の制御のもとに stream から読込みを行い，変換した値を後続の引数を通して代入する．各引数はポインタでなければならない．変換の前にファイルの終わりがくるか，エラーが起きると EOF が返される．そうでないときは，変換され，代入された入力項目数が返される．

◎ int sscanf(char *s, const char *format, ...)

入力文字が文字列 s から取られる点を除けば scanf と同じである．

◎ int fgetc(FILE *stream)

stream の次の文字を符号なし文字（int に変換した上で）として（あるいはファイルの終わりもしくはエラー発生時には EOF を）返す．

◎ char *fgets(char *s, int n, FILE *stream)

最大 n-1 文字を配列 s に読み込む．改行がくるとストップし，配列に含められ，その後に¥0 が付く．fgets は s を返すが，ファイルの終わりあるいはエラー発生時には NULL が返される．

◎ int fputc(int c, FILE *stream)

文字 c（を unsigned char に変換して）stream に書き込む．負ではない（エラー時は EOF）を返す．

◎ int fputs(const char *s, FILE *stream)

文字列 s(¥n を含む必要はない)を stream に書き込む．負ではない（エラー時は EOF）を返す．