プログラム言語C(3)

配列型の変数
文字列
文字列，ファイル入出力，応用

アンケート調査

- WebCTに準備してあるアンケートに回答をしてください。よろしくお願いします。
- URLは次の通り
  http://rupus.i.kyushu-u.ac.jp/~suga/Ipclass/
- アカウントとパスワードは次の通りです。
  - ユーザ名：zengaku
  - パスワード：jouhoushori

今日の内容

1. 配列の必要性
2. 配列の宣言
3. 配列変数のイメージ
4. 配列変数を使用した例
5. 範囲を超えた添字を使うと？
6. 多次元配列変数
9. 多次元配列変数を使用した例
10. データのソーティング
11. 文字列
12. 文字列の長さ
13. ファイル入出力
14. fgets の振る舞い
15. 応用

配列の必要性

多数のデータ処理

(要求)　10,000人分の成績データをもらって，
合計点の大きい順に並べ替えて出力したい．

(方法)　ひとまず，全部のデータを変数に入れておく．

(問題)　しかし，`int d1,d2,d3,..,d10000;`
のように，10,000個も変数を書けるか？

(解決策)　配列を使おう!!

```
int d[10000];
```

と書くだけで，d[0]～d[9999] の 10,000 個の変数を
使うことができる．

配列の宣言

- 一般の変数を宣言する場所で配列も宣言
- 宣言の一般形

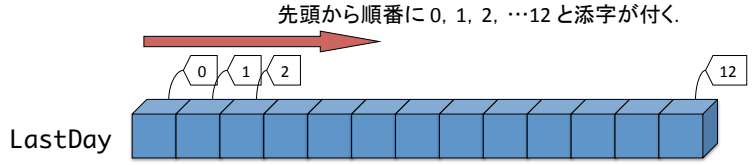
  型　配列名[配列の要素数];

  – 配列名は変数名と同様，アルファベット・数字・記号からなる．
    - 最初の文字はアルファベットであること．
    - 記号は一部しか使用できない．
  – 型は構成要素の型を表す．

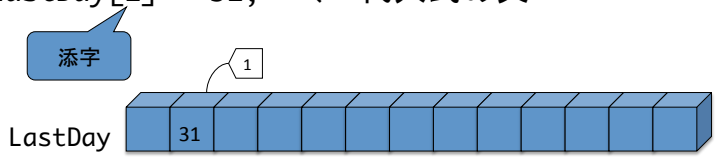
配列変数のイメージ

```
int LastDay[13];
```

13個の箱が並んでいると考える．

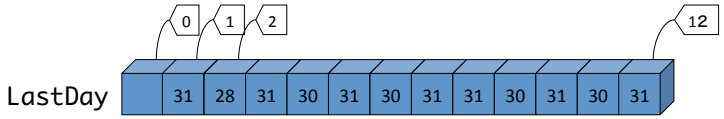



```
LastDay[1] = 31;
```
← 代入式の文



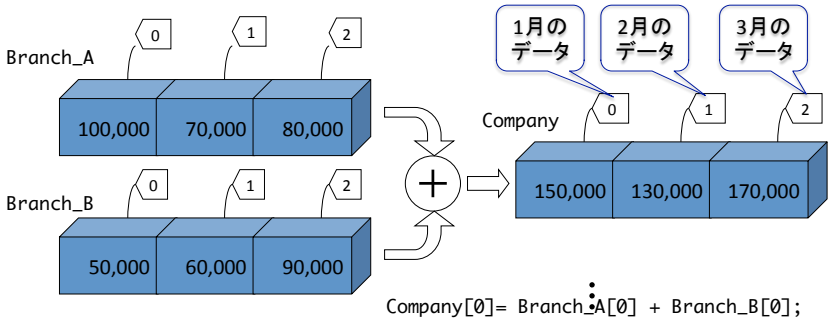
配列変数へのデータの代入と参照

```
int main(int argc, const char * argv[]) {
    int LastDay[13];
    int month;

    LastDay[1]=31;   LastDay[2]=28;   LastDay[3]=31;
    LastDay[4]=30;   LastDay[5]=31;   LastDay[6]=30;
    LastDay[7]=31;   LastDay[8]=31;   LastDay[9]=30;
    LastDay[10]=31;  LastDay[11]=30;  LastDay[12]=31;

    printf("月の日数を答えます．何月を知りたい？");
    scanf("%d", &month);
    printf("%d月は%d日です\n", month, LastDay[month]);
}
```



配列を使用した例

- ある会社に2つの支店（A支店とB支店）があるとする．
- 各支店で1月～3月の各月の売上額が下の表のようなデータとしてある場合，この会社の各月の売上額を計算する．



プログラム例

```
int main(int argc, const char * argv[]) {
    int Branch_A[3], Branch_B[3],Company[3];
    int month;

    Branch_A[0] = 100000;  Branch_A[1] = 70000;
    Branch_A[2] =  80000;  Branch_B[0] = 50000;
    Branch_B[1] =  60000;  Branch_B[2] = 90000;

    for(month = 0; month < 3; month++){
        // 各月で計算
        Company[month] = Branch_A[month] + Branch_B[month];
    }
    printf("この会社の売上額");
    for(month = 0; month < 3; month++){
        // 各月で売上額を表示
        printf("%d月は%d円", month + 1, Company[month]);
    }
}
```

配列を使用した例（2）

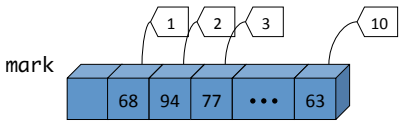
- テストの成績データがあったとして各学生の偏差値を計算するプログラムを作成してみよう.

学生番号	1	2	3	4	5	6	7	8	9	10
成績	68	94	77	52	73	85	79	71	86	63

出力例

学生番号	成績	偏差値
1	68	??
2	94	??
3	77	??
⋮	⋮	
10	63	??

上の表は下のような配列変数に格納されたデータと考えることができる.



偏差値の計算

- 偏差値は下に示す式で求まる.

$$学生i\text{の偏差値} = \frac{(学生i\text{の得点} - 平均)\times 10}{標準偏差} + 50$$

- 平均と標準偏差を求める.

$$平均 = \frac{個々の得点の総和}{学生数} = \frac{d_1 + d_2 + d_3 + \cdots + d_{10}}{10} = \frac{1}{10}\sum_{i=1}^{10} d$$

$$標準偏差 = \sqrt{\frac{(個々の得点 - 平均)^2\text{の総和}}{学生数}} = \sqrt{\frac{1}{10}\sum_{i=1}^{10}(d_i - m)^2}$$

- 総和を求める計算
  - 繰り返し制御構造を使う
  - 右のプログラム片で計算できる
    ループ終了後，変数waの値が総和

プログラム片

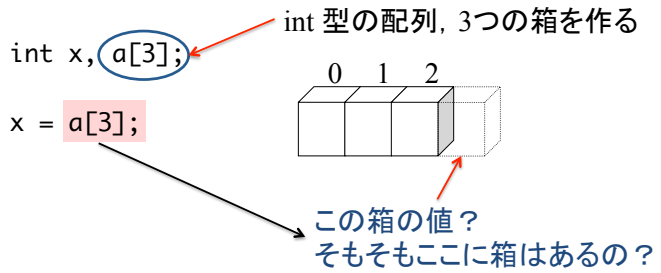
```
wa = 0;
for(i = 1; i <= 10; i++){
    wa = wa + d[i];
}
```

偏差値計算のプログラム例

```
#include <math.h>
int main(int argc, const char * argv[]) {
    const int N = 10;
    int mark[] = {0, 68, 94, 77, 52, 73, 85, 79, 71, 86, 63};
    int wa, i;
    double m, sigma, wa2, y;
    wa = 0;
    for(i = 1; i <= N; i++){
        wa = wa + mark[i];
    }
    m = wa / N;
    wa2 = 0;
    for(i = 1; i <= N; i++){
        wa2 = wa2 + pow((mark[i] - m), 2);
    }
    sigma = sqrt(wa2);
    printf("学生番号 成績 偏差値\n");
    for(i = 1; i <= N; i++){
        y = (10 * (mark[i] - m) / sigma + 50);
        printf("%4d      %2d   %2.1f\n", i, mark[i], y);
    }
}
```

範囲を超えた添字を使うと？

- 配列のサイズは宣言のときに決定する
  - 宣言時にメモリ領域を確保する
- サイズを超えて使用したら？



配列のオーバフロー

- C言語では，コンパイラは配列の添字の範囲のチェックは行わない
  - 配列の添字のチェックはプログラマの責任

```
int x, a[3];

x = a[3];
a[10] = 100;
```

この代入式では，何らかの値が変数 x に代入される

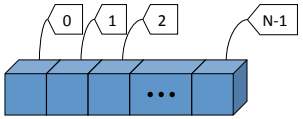
確保されていないメモリ領域に値 100 が書き込まれる

これらはエラーにならない

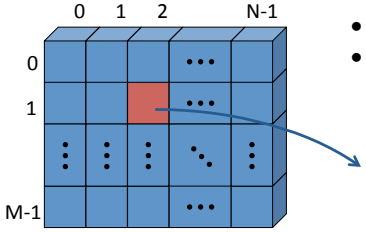
添字の範囲チェックを怠ると，おかしな動作をするプログラムになる

多次元配列変数

■ 1次元配列：一列（直線的）に並べた変数



■ 2次元配列：平面的に並べた変数



- 行と列を指定して1つの箱を特定する．
- 添字として2つの値を持つ．

B[M][N]

多次元配列変数の数学との対応

意味を与えた添字付きの変数を使う計算

例 $c_{ij} = a_{ij} + b_{ij}$ (行列の和の計算)

C言語		数学	
1次元配列	a[i]	数列	a_i
2次元配列	b[i][j]	行列	b_{ij}
3次元配列	c[i][j][k]	????	c_{ijk}
⋮			

C言語の配列では，何次元配列でも作ることができる．

例) `int b[n][m];`
配列 b は，n × m 行列

多次元配列を使用した例

- ある会社に2つの支店（A支店とB支店）があるとする．
- 各支店で1月～3月の各月の商品0，商品1の売上数が下の表のようなデータとしてある場合，この会社の各月の各商品の売上数を計算する．

Branch_A

	0	1	2
0	190	140	80
1	120	75	155

Branch_B

	0	1	2
0	80	95	150
1	70	100	140

＋ ⇒

Company

	0	1	2
0	270	235	230
1	190	175	295

```
Company[0][0]= Branch_A[0][0] + Branch_B[0][0];
```

月別，商品別の売り上げの計算（main関数の中）

```
int Branch_A[2][3], Branch_B[2][3], Company[2][3];
int good, month;

Branch_A[0][0]=190;  Branch_A[0][1]=140;  Branch_A[0][2]= 80;
Branch_A[1][0]=120;  Branch_A[1][1]= 75;  Branch_A[1][2]=155;
Branch_B[0][0]= 80;  Branch_B[0][1]= 95;  Branch_B[0][2]=150;
Branch_B[1][0]= 70;  Branch_B[1][1]=100;  Branch_B[1][2]=140;

for(good = 0; good < 2; good++){         //各商品で処理
    for(month = 0; month < 3; month++){ //各月で処理
        Company[good][month] =
                Branch_A[good][month] + Branch_B[good][month];
    }
}
printf("この会社の売上数\n");
for(good = 0; good < 2; good++){  //商品ごとに1行で表示
    printf("商品%d\n", good);
    for(month = 0; month < 3; month++){
        printf("%2d月 : %4d個\n", month + 1, Company[good][month]);
    }
}
```

3行3列行列の和の計算（main関数の中）

```
const int N = 3;
int a[N][N], b[N][N], c[N][N];
int i, j;

// 行列Aの成分を入力
printf("行列A=?\n");
for(i = 0; i < N; i++){
    for(j = 0; j < N;j++){
        scanf("%d", &a[i][j]);
    }
}

// 行列Bの成分を入力
printf("行列B=?\n");
for(i = 0; i < N; i++){
    for(j = 0; j < N; j++){
        scanf("%d", &b[i][j]);
    }
}
```

```
// 和C=A+Bを計算
for(i = 0; i < N; i++){
    for(j = 0; j < N; j++){
        c[i][j] = a[i][j]+b[i][j];
    }
}

// 和C=A+Bを出力
printf("A+B=\n");
for(i = 0; i < N; i++){
    for(j = 0; j < N; j++){
        printf("%3d", c[i][j]);
    }
    printf("\n");
}
```

データのソーティング

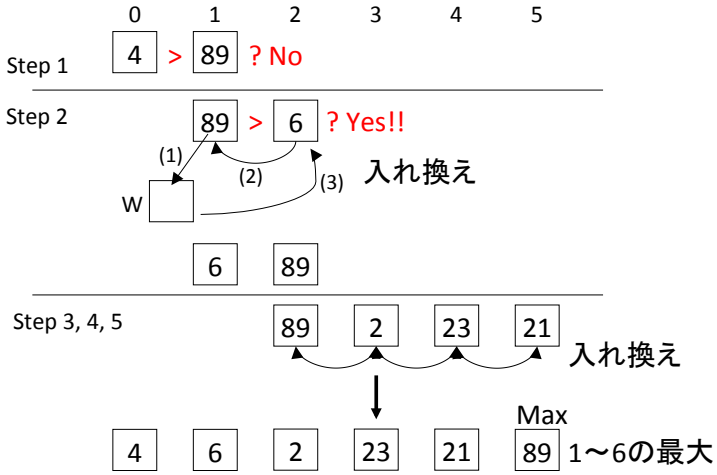
並べ替えイメージ(その1)

小さい順

	0	1	2	3	4	5	
並べ替え前	4	89	6	2	23	21	
						89	0～5の最大
					23	89	0～4の最大
				21	23	89	0～3の最大
			6	21	23	89	0～2の最大
並べ替え後	2	4	6	21	23	89	

並べ替えイメージ（その2）

隣り合った数を比べ，前の数が大きければ入れ替える．



プログラム例（main関数の中）

```
const int N = 6;
int d[]={4,89,6,2,23,25}, sd[N];
int i,j,w;
// 配列dを配列sdにコピー
for(i = 0; i < N; i++){
    sd[i] = d[i];
}
for(i = 0; i < N; i++){
    for(j = 0; j < N - i - 1; j++){
        if(sd[j] > sd[j+1]){
            w=sd[j];
            sd[j] = sd[j+1];
            sd[j+1] = w;
        }
    }
}
printf("Sorted data=\n");
for(i = 0; i < N; i++){
    printf("sd[%d]=%d\n", i, sd[i]);
}
```

文字列

- 文字列データを扱うときは，文字の配列を使う
  - 配列には，名前とサイズがある
  - 配列を使うために，配列の使用をコンピュータに伝えること（宣言）が必要

宣言

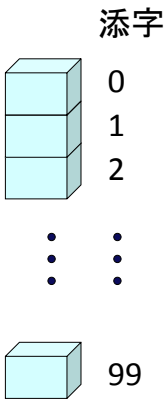
```
char x[100];
```

文字データ　名前はx　配列のサイズは100

文字列用の配列のサイズ

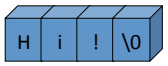
- 配列の添字は0から（サイズー1）
  例）char x[100]; と宣言したら，
  サイズは 100，添字は 0 から 99
- 文字列を格納する場所として実際に使えるのは，0 から（サイズー 2）まで
  例）char x[100]; と宣言したら，
  実際に文字列の格納のために使えるのは，添字 0 から添字 98 まで（最大 99 文字まで）

  「文字列の末尾」を表す特別な文字定数'\0'（ヌル文字，値は 0）を入れるのに，配列要素が1つ使われる



文字列の代入

文字列はダブルクォートで囲む




```
char s[]= "Hi!";
```

初期化

このように書くことはできない

```
char s1[100],s2[100];
strcpy(s1, "Hi!");
s2[0]= 'H';
s2[1]= 'I';
s2[2]= '!';
s2[3]= '\0';
```

× `s1= "Hi!";`

文字列用のライブラリ関数を使う
（string.hのインクルードが必要）

文字列の終わりを示す文字定数（ヌル(NULL)文字）

文字列の入出力文

- 「書式」と読み込むべき変数名を書く

```
char x[100];
scanf("%s", x);
```

配列名はポインタを表す

「&x[0]」と同じ

書式　読み込むべき変数名

配列名の前には「&」を付けない

```
printf("strlen(%s) = %d\n", x, len);
```

書式　表示すべき変数名

文字列の長さ

- 文字列を読み込んで，長さを表示するプログラムを作る

  例）“Computer” の長さは 8

  “プログラミング演習Ｉ” の長さは 20

  – ここでの文字列は，半角の「空白文字」を含まないものとする
  – 半角文字は1，全角文字は2として数える
    - 半角文字は1バイト
    - 全角文字は2バイト

文字列の長さを表示するプログラム

```
#include <stdio.h>
#include <string.h>
int main()
{
    char x[100];
    int len;

    printf("string=");
    scanf("%s", x);
    len = strlen(x);
    printf("strlen(%s) = %d\n", x, len);
    return 0;
}
```

文字列を格納するための配列の宣言
（「char」とあるのは「文字」という意味）

文字列データの読み込み
- 「%s」は文字列の意味
- 配列名「x」は，配列の先頭要素のメモリアドレスの意味

文字列の長さを求める標準関数
（string.h をincludeしておく）

文字列データと長さの表示
- 「%s」は文字列の意味

文字列用のライブラリ関数

- 文字列用のライブラリ関数を使うときには，次の1行をプログラムに含めること

```
#include <string.h>
```

- コピー: strcpy(s, ct) バッファオーバフローに注意
- 長さ取得: strlen(cs)
- 連結: strcat(s, ct) バッファオーバフローに注意
- 比較: strcmp(cs, ct)
- 検索: strstr(cs, ct)

など (cs, ct は const char *, s は char *)

文字列が変わらない

文字列が変わる

例題：文字列の連結

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

int main()
{
    char s1[80], s2[80];

    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    strcat(s1, s2);
    printf("s1=%s, s2=%s\n", s1, s2);
    return 0;
}
```

標準入力から文字列を受け取る

2つの文字列を連結する関数

例題：文字列の比較

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

int main()
{
    char s1[80], s2[80];
    int n;

    printf("s1=");
    scanf("%s", s1);
    printf("s2=");
    scanf("%s", s2);
    n = strcmp(s1, s2);
    if (n < 0) printf("%s < %s\n", s1, s2 );
    else if (n == 0) printf("%s == %s\n", s1, s2 );
    else if (n > 0) printf("%s > %s\n", s1, s2 );
    return 0;
}
```

標準入力から文字列を受け取る

文字列を比較する関数

入力文字列を受け取る時の注意

「文字列の連結」のプログラムに下の部分があるが，これは良いのだろうか？

```
char s1[80];

printf("s1=");
scanf("%s", s1);
```

↓

```
char s1[80];

printf("s1=");
fgets(s1, 80, stdin);
```

バッファオーバーフロー

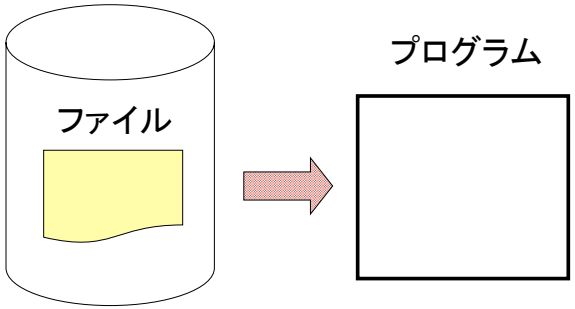
- 入力される文字列が80文字以上の場合は，確保した配列の領域を超えて書き込まれる
- 関数 scanf は s1 の添字の範囲をチェックしない

↓

- 入力文字数の上限を指定できる入力関数を使用する

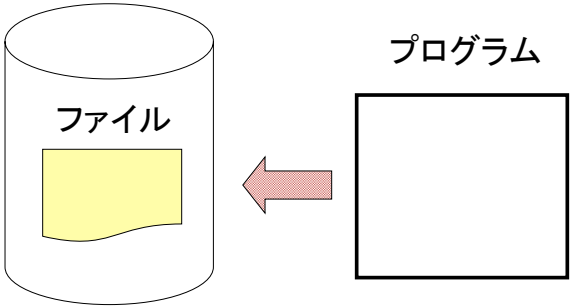
ファイル入出力

ファイルからの読み込み



- ファイルの中身は変わらない

ファイルへの書き出し



- ファイルの中身が変わる
- ファイルは伸び縮みすることがある

ファイルの種類

- テキストファイル
  - 行単位での読み書きに意味がある
  - 人間が「目で見て」読むことができるファイル

```
MPL 40
   pattern1 = 786
   pattern2 = 1
   pattern3 = 979
   pattern4 = 0
```

テキストファイルの例

```
<FF>0<FF><E0>^@^PJFI
F^@^A^B^A^@<96>^@<96
>^@^@<FF><E0>~JFXX^@
^P<FF>00<FF>U^@^C^@^
```

テキストファイルでない
バイナリファイルの例
（画像のファイル）

例題：ファイルからデータ読み込み

- 次のような名簿ファイル（テキストファイル形式）を読み込んで，1列目の氏名と，3列目の住所だけを表示するプログラムを作る
  - 各データは，半角の空白文字（1つまたは複数）で区切られる

```
九大太郎 1200/01/01 福岡市東区箱崎３丁目
情報次郎 1300/12/31 福岡市東区貝塚団地
福岡花子 1800/05/31 福岡市東区香椎浜１丁目
```

3行のテキストファイル

プログラム例

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

データファイルを準備する
（テキストファイル形式）

```
九大太郎 1200/01/01 福岡市東区箱崎３丁目
情報次郎 1300/12/31 福岡市東区貝塚団地
福岡花子 1800/05/31 福岡市東区香椎浜１丁目
```

半角の
空白文字

Book1.txt の例
（全角の空白文字が混ざっていると
動かないことがあるので注意）

ファイルから文字列を読み込む

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルオープンに失敗したときのみ実行される部分

whileによる繰り返し部分

NULLの意味

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

fopen 関数では「ファイルオープンの失敗」

fgets 関数では「ファイルの終わり」

「==」は等しいという意味
「!=」は等しくないという意味

条件を満たす場合の処理

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

int main()
{
    char line[100], nam
    FILE *in_file;

    in_file = fopen("Book1.txt"
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルオープンに失敗したら，プログラムが終わる

fopen 関数では「ファイルオープンの失敗」

fgets 関数では「ファイルの終わり」

ファイルの終わりになるまで，fgets，sscanf，printf を繰り返す

ファイル操作のライブラリ関数

- ファイルのオープンとクローズ
  - fopen　ファイルの読み書きを行う前に，ファイルをオープンしなければならない
  - fclose　ファイルの読み書きが終わったら，ファイルをクローズしなければならない
- ファイルから読み込み
  - fgetc　1文字単位の読み込み
  - fgets　1行単位の読み込み
  - fread　バイト単位の読み込み（1バイト，複数バイト）
- ファイルへ書き出し
  - fputc　1文字単位の書き出し
  - fputs　1行単位の書き出し
  - fwrite　バイト単位の書き出し（1バイト，複数バイト）
  - fprintf　整形した文字列の書き出し

オープンモード

```
in_file = fopen("Book1.txt", "r");
```

ファイル名（文字列）　オープンモード（文字列）

- "r" モード
  - 読み込みモード
  - 引数で指定したファイルが存在しないか，読み込み不可能な場合には，オープンすることができない
- "w" モード
  - 書き出しモード
  - 引数で指定したファイルが存在しない場合には，ファイルが新たに作成される
  - ファイルがすでに存在した場合は，ファイル中のデータはすべて捨てられる（ファイルの長さは0になる）

ファイルポインタの使用例

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

ファイルポインタ

ファイルのオープン

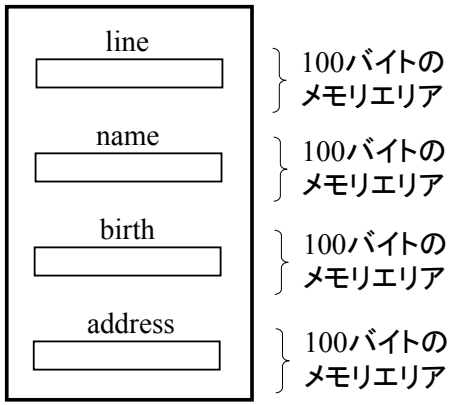
ファイルの読み込み（1行単位）

ファイルのクローズ

whileによる繰り返し部分

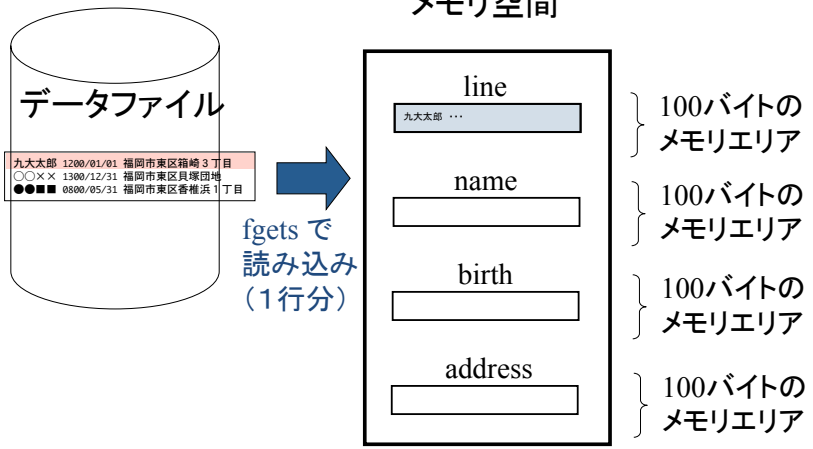
プログラムが行っていること

プログラムが使う
メモリ空間



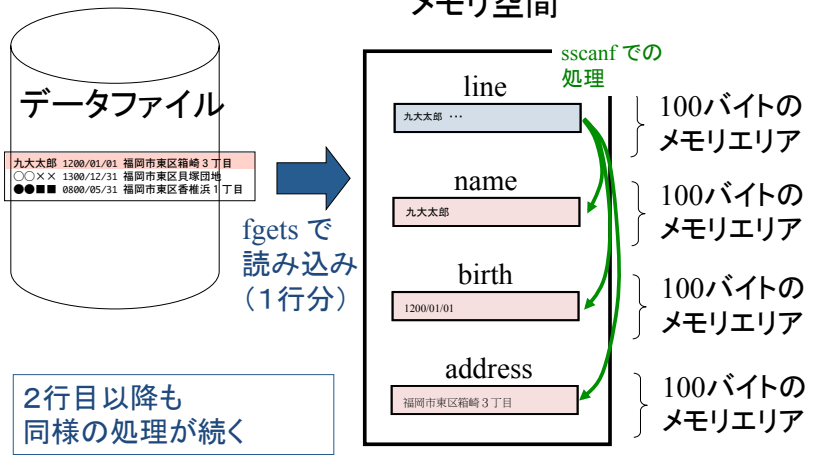
プログラムが行っていること

プログラムが使う
メモリ空間



プログラムが行っていること

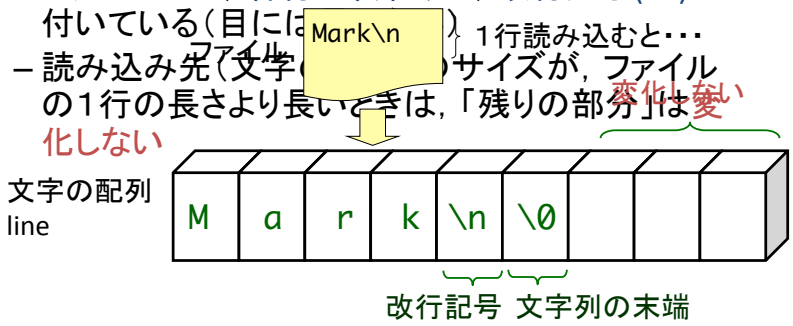
プログラムが使う
メモリ空間



2行目以降も
同様の処理が続く

fgets の振る舞い

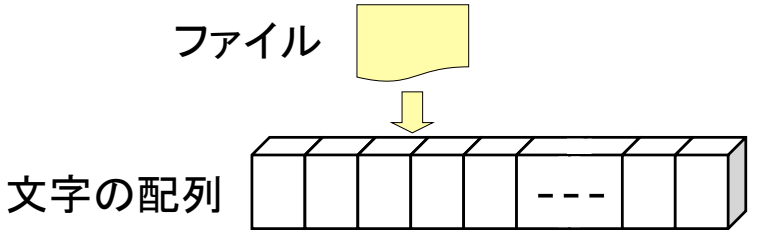
- ファイルの1行読み込み
  - ファイルの一行分を読み込んで，末端の¥0を付ける
  - ファイルには，各行の終わりに，改行文字(¥n)が付いている(目には
  - 読み込み先(文字
    の1行の長さより長いときは，「残りの部分」は変化しない



fgets の「100」の意味

```
fgets( line, 100, in_file )
```

「\0」を含めて100文字に達したら
行末になっていなくても読み込みを終了せよ



配列のサイズが100ならば，読み込める文字は99文字まで
（最後に必ず「\0」が付く）

文字列の書式指定文字

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

int main()
{
    char line[100], name[100], birth[100], address[100];
    FILE *in_file;

    in_file = fopen("z:\\Book1.txt", "r");
    if ( in_file == NULL ) {
        return 0;
    }
    while( fgets( line, 100, in_file ) != NULL ) {
        sscanf( line, "%s %s %s", name, birth, address );
        printf( "name=%s, address=%s\n", name, address );
    }
    fclose(in_file);
    return 0;
}
```

char 型の配列については，
sscanf，printf，fprintf
では「%s」を使う決まりになっている

それぞれ対応

それぞれ対応

例題：ファイルのコピー

```
#include <stdio.h>
int main()
{
    char line[100];
    FILE *in_file,*out_file;
    in_file = fopen("input.txt","r");
    if ( in_file == NULL ) {
        printf( "fopen_in error" );
        return 0;
    }
    out_file = fopen("output.txt","w");
    if ( out_file == NULL ) {
        printf( "fopen_out error" );
        return 0;
    }
```

ファイルポインタ変数の宣言

入力ファイルのオープン

出力ファイルのオープン

例題：ファイルのコピー（続き）

```
    while( fgets(line,100,in_file) != NULL){
        fputs(line,out_file);
    }
    fclose(in_file);
    fclose(out_file);
    return 0;
}
```

ファイルの終わりに達していないかを調べている

ファイルの1行読み込み

ファイルのクローズ

ファイルの1行書き出し

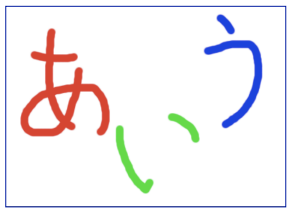
応用
（画像ファイルを扱う）

これまでに学んだ条件分岐，繰り返し，配列，ファイル入出力を使って，画像を扱うプログラムにチャレンジしてみよう

特定色の画素の検出

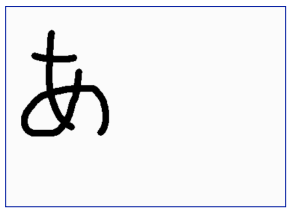
- 画像データが保存されているファイルを読み込んで，特定色の画素の位置を検出するプログラムを作成しなさい

元画像



赤色ピクセルの検出

生成画像（結果の画像）



元画像で赤色の画素だけ黒にし，それ以外の画素は白にした画像を生成する

特定色の画素の検出

- 画像は平面上に配置された画素の集まり
  - 二次元配列に画像データを入れる
- 配列の各要素に対して同様の処理を施す場合，繰り返しを使ったプログラムを記述する
  - 二次元配列を扱うので，二重ループを作成する
- ある画素の色が特定色であれば，見つけたときの処理，そう出なければ，見つけなかったときの処理を実行するので，条件分岐を使う
  - ピクセルの値（配列に入っている値）で判断する

プログラムの流れ

1. 定数の定義
2. 各種変数，配列の宣言
3. ファイルから画像データを読み込む
4. 赤色画素の検出，結果の画像の生成（ループ）
5. ファイルへ結果画像の書き出し

画素の値で処理を変える

```
if ((R >= 200) && (G <= 50) && (B <= 50)) {
    「赤い画素」の処理（見つけたときの処理）
} else {
    「赤い画素でない」の処理
}

// ただしR，G，Bは画素の各色の画素値を示す
```

赤色はRGBでどういう値になるのか？

R >= 200
G <= 50
B <= 50
} このようにしてみる

すべての画素で処理を行う

```
for (i = 0; i < height; i++) {
    for (j = 0; j < width; j++) {
        if ((R >= 200) && (G <= 50) && (B <= 50)) {
            「赤い画素」の処理（見つけたときの処理）
        } else {
            「赤い画素でない」の処理
        }
    }
}
```

画素の値で処理を変える

すべての画素を対象に値のチェックを行う

二次元配列 → 二重ループ

定数の定義と変数の宣言

```
#include <stdio.h>

int main (int argc, const char * argv[]) {
    const int MAX_WIDTH = 640;
    const int MAX_HEIGHT = 480;
    int width, height;

    /* 画像を入れる2次元配列の宣言 */
    unsigned char SrcImageR[MAX_HEIGHT][MAX_WIDTH];
    unsigned char SrcImageG[MAX_HEIGHT][MAX_WIDTH];
    unsigned char SrcImageB[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageR[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageG[MAX_HEIGHT][MAX_WIDTH];
    unsigned char ExImageB[MAX_HEIGHT][MAX_WIDTH];

    const char InFileName[] = "ColorChars.ppm"; /* 入力ファイルの名前 */
    const char OutFileName[] = "ExtRed.ppm";    /* 出力ファイルの名前 */
    FILE *f;

    const int LEN = 100;
    char line[LEN];
    int i, j;
```

データの読み込み

```
/* ファイルからの読み込み */
 f = fopen(InFileName, "r");
 if ( f == NULL ) {
     fprintf(stderr, "ファイル %s のオープンに失敗しました", InFileName);
     return -1;
 }
 fgets(line, LEN, f);  /* 1行目は読み飛ばす */
 fgets(line, LEN, f);
 while (line[0] == '#') {  /* コメントは読み飛ばす */
     fgets(line, LEN, f);
 }
 /* 画像の幅と高さ */
 sscanf(line, "%d %d", &width, &height);
 fgets(line, LEN, f);  /* データの3行目は読み飛ばす */
 for ( i = 0; i < height; i++ ) {
     for ( j = 0; j < width; j++) {
         fread( &(SrcImageR[i][j]), 1, 1, f);
         fread( &(SrcImageG[i][j]), 1, 1, f);
         fread( &(SrcImageB[i][j]), 1, 1, f);
     }
 }
 fclose(f);
```

条件を満たす画素の検出

```
for (i = 0; i < height; i++) {
    for (j = 0; j < width; j++) {
        if ((SrcImageR[i][j] >= 200) &&
            (SrcImageG[i][j] <= 50) && (SrcImageB[i][j] <= 50)) {
            ExImageR[i][j] = 0;
            ExImageG[i][j] = 0;
            ExImageB[i][j] = 0;
        } else {
            ExImageR[i][j] = 255;
            ExImageG[i][j] = 255;
            ExImageB[i][j] = 255;
        }
    }
}
```

見つけたときの処理
画素を黒色にする

画素を白色にする

ファイルへのデータの書き出し

```
    /* ファイルへの出力 */
    f = fopen(OutFileName, "w");
    fprintf(f, "P6¥n");  /* 1行目 */
    fprintf(f, "%d %d¥n", width, height);  /* 2行目 */
    fprintf(f, "255¥n");  /* 3行目 */
    for ( i = 0; i < height; i++ ) {
        for ( j = 0; j < width; j++) {
            fwrite( &(ExImageR[i][j]), 1, 1, f);
            fwrite( &(ExImageG[i][j]), 1, 1, f);
            fwrite( &(ExImageB[i][j]), 1, 1, f);
        }
    }
    fclose(f);

    return 0;
}
```

データの並べ替え

```
#include <stdio.h>
#define SIZE 100

int main (int argc, const char * argv[]) {

int n, x;
int d[SIZE];
int i,j,temp;

n = 0;
while (scanf("%d", &x) != EOF) {
    d[n] = x;
    n++;
    if (n>=SIZE) break;
}

for(i = 0; i < n; i++){
    for(j = 0; j < n - i - 1; j++){
        if(d[j] > d[j+1]){
            temp = d[j];
            d[j] = d[j+1];
            d[j+1] = temp;
        }
    }
}

printf("Sorted data=\n");
for(i = 0; i < n; i++){
    printf("d[%d] = %d\n", i, d[i]);
}

return 0;
}
```

課題

- キーボードからデータを読み込んで、大きい方から小さい方へ並べ替えて表示するプログラムを作成せよ。
  - データの終わりはCntl+Dで検出
  - データ個数を確認
  - 5個のデータを印刷したら改行
- データを20個程度入力して実行し、その結果を貼付けて提出せよ。
- プログラムのファイルを回答欄に貼り付けよ。

授業アンケートの提出

- 授業クラスコード：XXXX（XXXX学科）
- 専用の回収ボックスに投函
  - センター1号館1階情報学習室
  - センター1号館2階全学教育課
  - センター2号館1階ピロティ電子掲示板前
  - センター2号館4階おう鳴天空広場
  - 体育館事務室前
- 8月6日までに投函