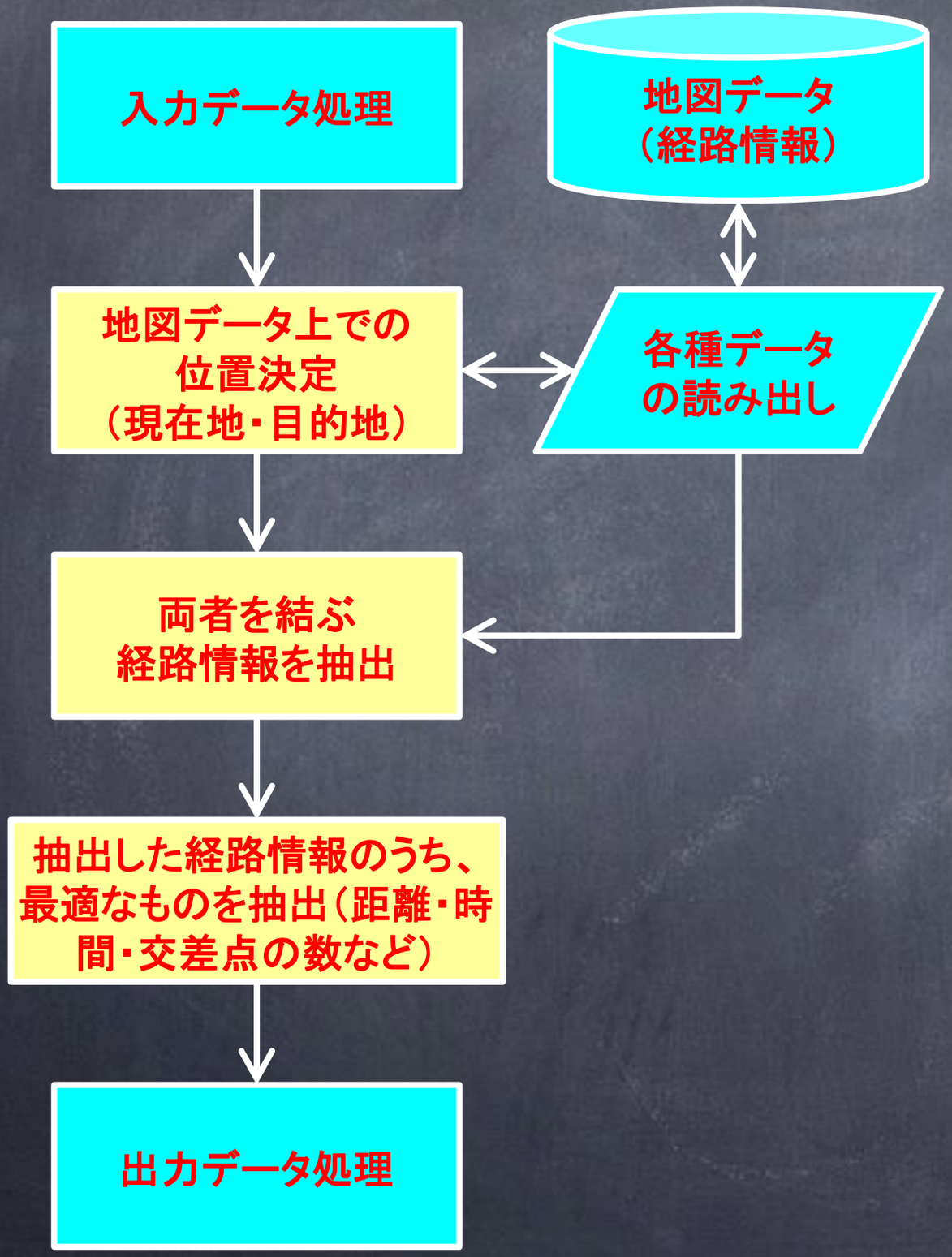
コンピュータ実習
第8回 2025年11月27日

カーナビゲーション2
サーチングとソーティング

担当：相澤　直人，宍戸 博紀

コンピュータ実習（第8回）

カーナビ作成に向けて



地図データの検索と抽出
→ 入出力・変数・構造体

抽出したデータを元に経路を割り出す
→ 条件検索（サーチ）

割り出した経路のうち、最適なものを選ぶ
→ 条件並び替え（ソート）

コンピュータ実習（第8回）

今後の授業の進行

1. 地図データを元に、目的地を決定
   - 地図データの読み込み
     - 前回第7回講義
   - **交差点（地図データ）の検索** ← 本日の講義
2. 経路を探索
   - 条件に基づく経路の設定
     - 第9回講義(簡易版)
   - 最適経路の計算（距離・時間等）
     - 第10回講義
3. 経路をドライバーに分かりやすく提示
   - 画面上で車をアニメーション表示
     - 第9回、第11回講義

毎回の講義で作成するプログラムを組合せば、最低限のカーナビプログラムは完成できます

コンピュータ実習（第7回おさらい）

カーナビゲーションに必要なデータ構造

交差点情報を扱うには

情報をまとめて取り扱うために、構造体を使用する

交差点情報

- 交差点のID（識別番号）
- 交差点名
- 位置座標（x, y）
- 平均待ち時間
- 交差道路数
  （隣接交差点数）
- 隣接交差点ID

構造体”Crossing”の定義

```
typedef struct{
    id;
    char name[50];
    double x, y;
    double wait,
    int points;
    int next [5];
} Crossing;
```

コンピュータ実習（第7回おさらい）

カーナビゲーションに必要なデータ構造

交差点情報を扱うには

構造体には構造体を入れ子にして使うことも可能

構造体”Crossing”の定義

```
typedef struct{
    int id;
    char name[50];
    double x, y;
    double wait,
    int points;
    int next [5];
} Crossing;
```

→

```
typedef struct{
    int id;
    char name[50];
    Position pos;
    double wait,
    int points;
    int next [5];
} Crossing;
```

コンピュータ実習（第7回おさらい）

カーナビゲーションに必要なデータ構造

交差点情報を扱うには

構造体には構造体を入れ子にして使うことも可能

構造体”Crossing”の定義

```
typedef struct{
    int id;
    char name[50];
    Position pos;
    double wait,
    int points;
    int next [5];
} Crossing;
```

構造体”Position”の定義

```
typedef struct{
    double x,y;
} Position;
```

位置情報はひとまとめで扱った方が分かりやすい

要するに、ひとまとめで扱う変数は、構造体で定義しておく

コンピュータ実習（第7回続き）

Web教材の第7回のページの
「4．課題2,3：交差点情報構造体を使用した
プログラミング2」
を開いてください．

コンピュータ実習（第7回続き）

課題3： 交差点情報のファイルからの入力

webテキストの課題3（readfile.c）について、交差点情報データ(map1.dat)をファイルから読み取り、ターミナルに表示できるように、完成させてください。

ヒント

- このプログラムは大まかに2つに分かれます
  - ファイル入力と画面（ターミナル）表示
- main(void)以前はファイルへの出力を”map_read”という関数として定義した部分です。
  ユーザー定義のmap_read関数を完成させましょう。
- writefileとreadfileは対になっています
  - fprintf(fp,”%d¥n”,crossing_number);
  - fscanf(fp,”%d”,&(crossing_number));

完成したプログラムはreadfile.cとして保存しておくこと

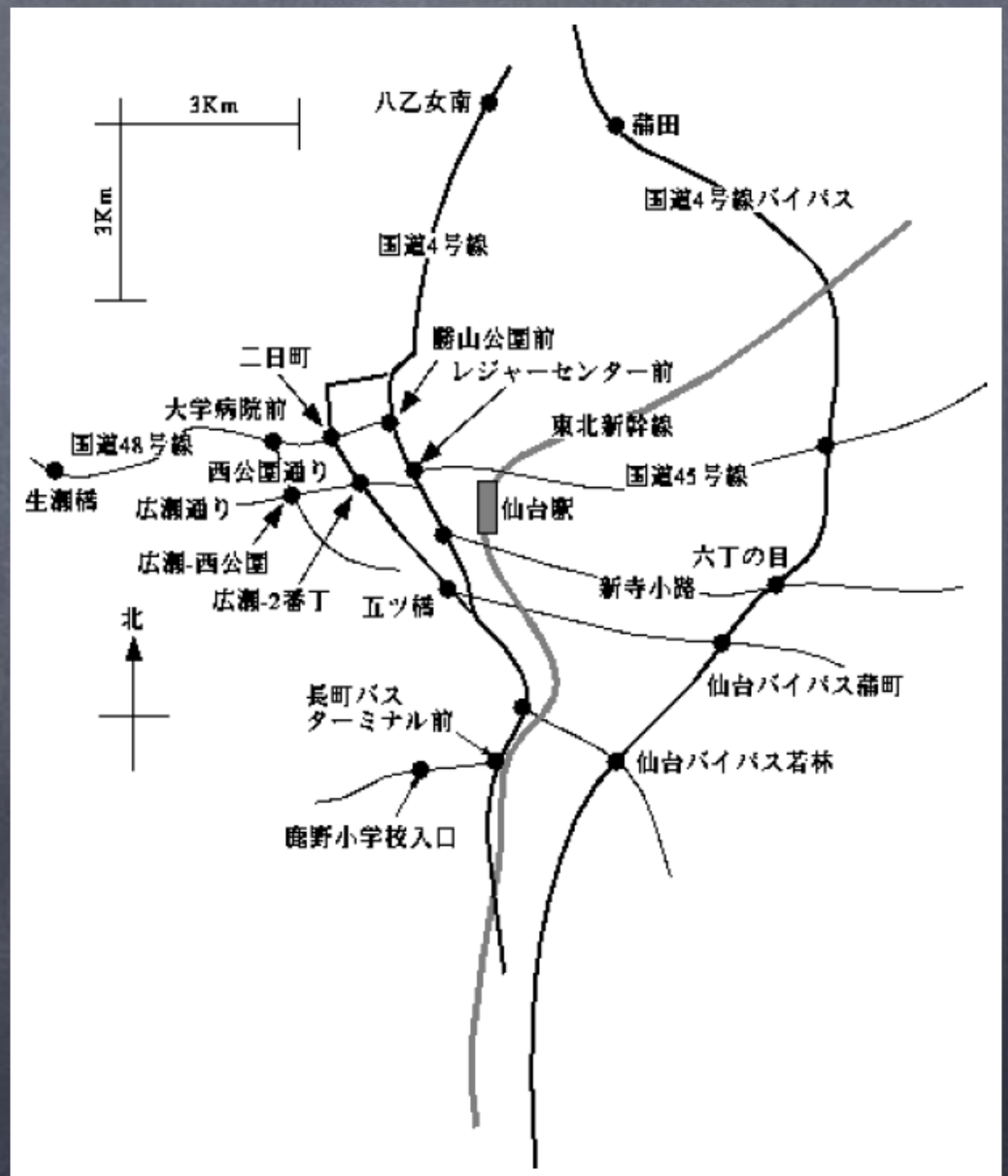
コンピュータ実習（第7回続き）

課題3： 交差点情報のファイルからの入力

地図データについて、本講義では
2種類の仙台市中心部の概略
地図のデータを用意しています
（かなり古い情報ですが・・・）

map1.dat
仙台市内の交差点データ（90個）
交差点名は日本語表記

map2.dat
仙台市内の交差点データ（90個）
交差点名は日本語表記と
ローマ字表記2通りのデータが
格納されています。



コンピュータ実習（第8回）

サーチングとソーティング

今日の課題：サーチ（search）とソート（sort）

- サーチ（検索）とは，
  - 大量にあるデータの中から，目的に合致したものを 探し出す作業
    - ピッタリ一致するものを探す
    - だいたい一致するものを探す
    - 一番近いものを探す(含む、ピッタリ)

- ソート（並び替え）とは，
  - 大量にあるデータを，指定の条件で順番に並べ替える作業
    - 並べる基準
    - 並べる方法

コンピュータ実習（第8回）

サーチングとソーティング

サーチ

- サーチはカーナビのデータ操作に必ず必要な要素
- データの検索の一致方式には「全文一致，部分一致，前方一致，後方一致」がある．
- 例えば，AA・AAA・AAAB・CAAAというようなデータ群からサーチする
  - 検索文字：AA
    - 全文一致：AA
    - 部分一致：AAA・AAAB・CAAA
    - 前方一致：AA・AAA・AAAB
    - 後方一致：AA・AAA・CAAA
- 一致の判定にも完全（”A”は，”A”）かあいまい（”A”は，”A”か”a”か”あ”）かがある

コンピュータ実習（第8回）

サーチングとソーティング

サーチ　全文一致

```
void search_cross(void)
{
    int i;
    char buff[256];
    int f=0; /* 見つかったかどうかのフラグ */
    printf("交差点名を入力してください (ローマ字)->");
    scanf("%s",buff);
    for(i=0; i<crossing_number; i++) /* データを最後までループ　*/
    {
      if(strcmp(cross[i].ename, buff)==0) /* 一致したら*/
      {
        printf("交差点名 %s(%s) 座標 %.2lf,%.2lf 駅からの距離 %.2lf¥n",
              cross[i].jname,cross[i].ename, cross[i].pos.x,cross[i].pos.y,
              hypot(cross[i].pos.x,cross[i].pos.y)); /* hypot は sqrt(x*x+y*y) を計算します */
        f=1;
      }
    }
    if(f==0)
    puts(“見つかりませんでした”); /* puts は 文字列を表示します */
}
```

コンピュータ実習（第8回）

サーチングとソーティング

サーチ　全文一致

```
void search_cross(void)
{
   int i;
   char buff[256];
   int f=0; /* 見つかったかどうかのフラグ */
   printf("交差点名を入力してください (ローマ字)→");
   scanf("%s",buff);
   for(i=0; i<crossing_number; i++) /* データを最後までループ */
   {
     if(strcmp(cross[i].ename, buff)==0) /* 一致したら*/
     {
       printf("交差点名 %s(%s) 座標 %.2lf,%.2lf 駅からの距離 %.2lf\n",
              cross[i].jname,cross[i].ename, cross[i].pos.x,cross[i].pos.y,
              hypot(cross[i].pos.x,cross[i].pos.y)); /* hypot は sqrt(x*x+y*y) を計算します */
       f=1;
     }
   }
   if(f==0)
   puts(“見つかりませんでした”); /* puts は 文字列を表示します */
}
```

strcmp関数

文字列の比較を行う関数
一致する場合に戻り値0

※#include <string.h>が必要

hypot関数

各引数の2乗の和の平方根を計算する→平面2点間の距離を計算する

puts関数

標準出力 (standard output) に指定した文字列を書き込みます

コンピュータ実習（第8回）

サーチングとソーティング

サーチ　部分一致

```
void search_cross_aprox(void)
{
    int i;
    char buff[256];
    int f=0;
    printf("交差点名を入力してください (ローマ字)->");
    scanf("%s",buff);
    for(i=0; i<crossing_number;i++)
    {
      if(strstr(cross[i].ename, buff)!=NULL) /* 含んでいたら */
      {
        printf(“交差点名 %s(%s) 座標 %.2lf,%.2lf 駅からの距離 %.2lf¥n”,
                cross[i].jname,cross[i].ename, cross[i].pos.x,cross[i].pos.y,
                hypot(cross[i].pos.x,cross[i].pos.y));
        f=1;
      }
    }
    if(f==0)
    puts("見つかりませんでした");
}
```

コンピュータ実習（第8回）

サーチングとソーティング

サーチ　部分一致

```
void search_cross_aprox(void)
{
   int i;
   char buff[256];
   int f=0;
   printf("交差点名を入力してください (ローマ字)->");
   scanf("%s",buff);
   for(i=0; i<crossing_number;i++)
   {
     if(strstr(cross[i].ename, buff)!=NULL) /* 含んでいたら */
     {
       printf(“交差点名 %s(%s) 座標 %.2lf,%.2lf 駅からの距離 %.2lf¥n”,
              cross[i].jname,cross[i].ename, cross[i].pos.x,cross[i].pos.y,
              hypot(cross[i].pos.x,cross[i].pos.y));
       f=1;
     }
   }
   if(f==0)
   puts("見つかりませんでした");
}
```

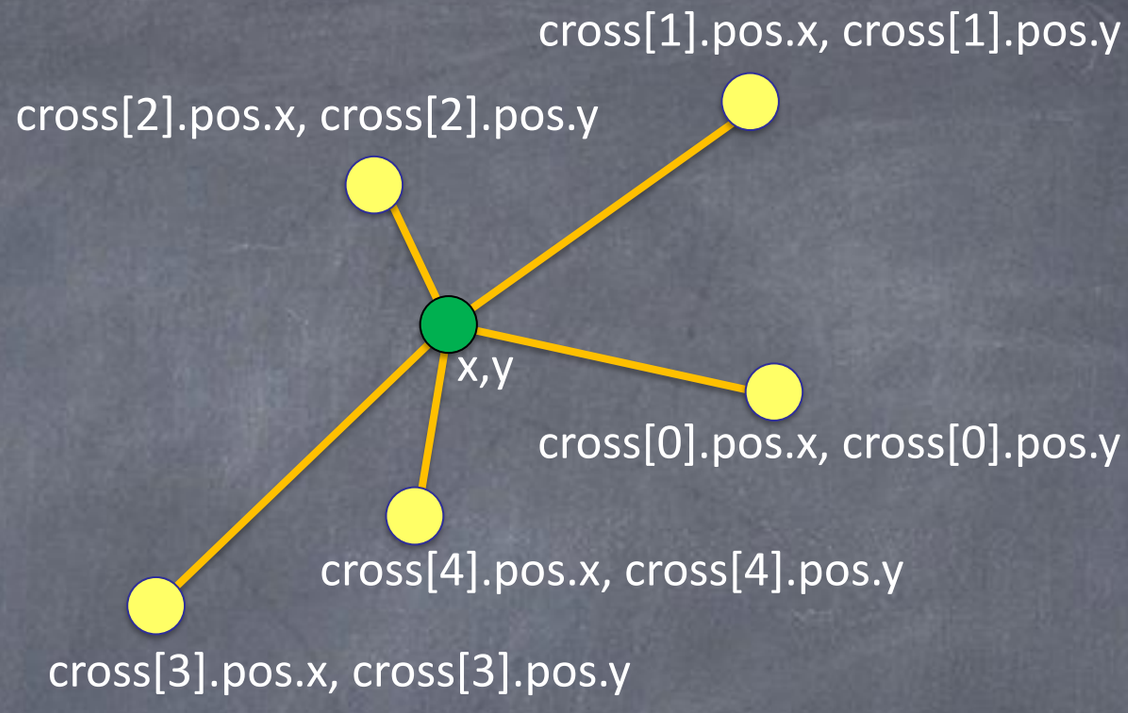
strstr(*s1,s2*)関数

s1 に s2 が含まれている場合、s1 の中で s2 が見つかった場所(ポインタ)を返す

コンピュータ実習（第8回）

サーチングとソーティング

サーチ　最短距離検索




```
void search_min_distance(void) /* 最短距離検索 */
{
    int i;
    int minindex=0; /* 暫定トップの位置 */
    double dist, mdist; /* 評価数値 */
    double x,y; /* 検索地点座標 */
    printf("検索地点座標を入力してください(%%lf %%lf 形式) -> ");
    scanf("%lf %lf",&x,&y);
    for(i=0;i<crossing_number;i++) /* データ最後までループ */
    { /* 現在の暫定トップの評価数値 */
      mdist=hypot(cross[minindex].pos.x-x,cross[minindex].pos.y-y);
     /* 挑戦者の評価数値 */
      dist=hypot(cross[i].pos.x-x,cross[i].pos.y-y);
      if(dist<mdist) /* 挑戦者の方が近い */
        minindex=i;
    }
    printf(“座標( %.2lf, %.2lf )の地点から最も近い交差点 (距離 %.2lf):¥n”,
           x,y,hypot(cross[minindex].posx-x,cross[minindex].pos.y-y));
    printf("交差点名 %s(%s) 座標 %.2lf,%.2lf 駅からの距離 %.2lf¥n",
          cross[minindex].jname, cross[minindex].ename, cross[minindex].pos.x, cross[minindex].pos.y,
          hypot(cross[minindex].pos.x,cross[minindex].pos.y));
}
```

ある地点からの交差点の距離を総当たりで比較している

コンピュータ実習（第8回）

サーチングとソーティング

サーチの必要性

- カーナビで出発地や目的地を指定する際に必要となります.

要求事項

- ローマ字入力，日本語入力両対応にしたい！
- 不正確な入力内容から交差点名を推察したい！
- 現在地から最寄りの交差点を探したい！
- ect...

コンピュータ実習（第8回）

サーチングとソーティング

演習1：検索演習(ex8-1)

1. キーボードから交差点名のローマ字名を入力して，map2.datに格納されている名前が完全に一致した交差点の情報を表示するプログラムを作成してください．
2. キーボードから座標数値を入れると，そこにもっとも近い交差点情報を表示するプログラムを作成してください．
3. これらの課題ができた人は，名前が部分的に一致した交差点の情報を表示するプログラムを作成してください．

コンピュータ実習（第8回）

サーチングとソーティング

ソート

- ソートはカーナビにおいて交差点間の距離の近い順に並べる操作等に必ず必要な要素

- ソートのアルゴリズムには「バブルソート，単純選択，クイックソート，マージソート法」などがある

- 最悪計算時間，アルゴリズムの単純さ，使用メモリ量などを考えて決める

コンピュータ実習（第8回）

サーチングとソーティング

バブルソート法

- データの配列をひたすら隣り同士で 比較し、望む順番でなければ入れ替える
- 遅いがアルゴリズムが単純

「58374916 20」を値の小さい順にソート

58374916 20 → 5837491620
5837491620 → 5387491620
5387491620 → 5378491620
5378491620 → 5374891620
5374891620 → 5374891620
5374891620 → 5374819620
…
5374816290 → 5374816209　一回目の並べ替え終了

コンピュータ実習（第8回）

サーチングとソーティング

バブルソート法

- 二回目の並べ替え開始

5374816209 → 3574816209

3574816209 → 3574816209

…

3547162089 → 3547162089　二回目の並べ替え終了

三回目　3547162089 → 3451620789

四回目　3451620789 → 3451206789

・・・と要素数-1回繰り返す（この場合は最悪9回）

最悪でも（並べる数の個数n-1）回の繰り返し

比較回数は全体でn(n-1)/2回

計算時間：$\mathcal{O}(n^2)$

コンピュータ実習（第8回）

サーチングとソーティング

バブルソート法

バブルソート法によるソーティングを体験してみよう！

sort0.cファイルをダウンロードして実行する

コンピュータ実習（第8回）

サーチングとソーティング

バブルソート法

```
void sort0(void) /* バブルソート */
{
    int i,j;
    int lo=0, up=N-1;
    while(up>lo)
    {
      j=lo;
      for(i=lo;i<up;i++)
      {
        if(compare_data(i,i+1)>0) /* 順序が正しくなければ */
        {
          exchange_data(i,i+1); /* 交換 */
          j=i;
        }
      }
      up=j;
    }
}
```

配列要素[i]と [i+1]を比較し、[i]が大きいなら正を、[i+1]が大きいなら負を返す

要素[i]と [i+1]を交換します

※compare_data/exchange_data関数はコード内で別途定義している

コンピュータ実習（第8回）

サーチングとソーティング

さまざまなソーティングアルゴリズム

- 単純比較（選択ソート）
  - 「一番小さなもの(小さな順の場合)を 探して，それを先頭に持ってくる」というアイデア
  - 1回目は全部 [0]～[N-1]から最小のものを探して先頭にもってきます．
  - 2回目は先頭を除いた配列[1]～[N-1]から，この範囲で最小のものを探して ここでの先頭，すなわち[1]に置きます．
  - つまり，小さな順に探しては 置いていく，という方法です．

コンピュータ実習（第8回）

サーチングとソーティング

さまざまなソーティングアルゴリズム

- 単純比較（選択ソート）

「5837491620」を値の小さい順にソート

5837491620 → 0837491625

0837491625 → 0137498625

0137498625 → 0127498635

0127498635 → 0123498675

0123498675 → 0123498675

...

最悪でも（並べる数の個数n-1）回のループ

比較回数は全体でn(n-1)/2回　　計算時間：$\mathcal{O}(n^2)$

バブルソート比べると交換回数が少ないため高速

コンピュータ実習（第8回）

サーチングとソーティング

さまざまなソーティングアルゴリズム

- クイックソート
  1. 適当な数（ピボットという）を選択する（この場合はデータの総数の中央値が望ましい）
  2. ピボットより小さい数を前方、大きい数を後方に移動させる（分割）
  3. 二分割された各々のデータを、それぞれソートする.

- 左半分をソートし，右半分もソートすると，与えられた 配列はソート完了です.

- 第2段階で左半分と右半分をソートするために，それぞれに対して同じ方法を使います．この段階で「与えられた配列」は左半分，右半分となります．どんどん小分けにしていくと、最後には要素一つの配列に なってしまいますが，その部分はソートが完了したことになります.

計算時間：$O(n^2)$

コンピュータ実習（第8回）

サーチングとソーティング

さまざまなソーティングアルゴリズム

- マージソート
  - クイックソートが左右に分けた上で，それぞれをさらにソートする，というアルゴリズムだったのに対して，マージソートは，とりあえず 二つに分けて，それぞれを並べてから，それを組み合わせる＝マージする，というアルゴリズム

1. データ列を分割する（通常、等分する）
2. 各々をソートする
3. 二つのソートされたデータ列をマージする

- 2番目のステップでは、マージソートを再帰的に適用する。
- マージは、データ列の先頭同士を比べ小さい方をデータ列から取り出し、残りのデータ列に対して再帰的に適用する。

計算時間：$\mathcal{O}(n \log n)$

コンピュータ実習（第8回）

サーチングとソーティング

さまざまなソーティングアルゴリズム

理解に役立つサイト

https://visualgo.net/en/sorting

他にもたくさんの参考になるサイトがあります．
単にコードをコピペをするのではなく，アルゴリズムを理解・実装できることが大事です．

コンピュータ実習（第8回）

サーチングとソーティング

演習2：ソート演習(ex8-2)

- map2.datのデータに対して入力された座標から近い交差点順にデータを並べ替えて出力させよ.

※アルゴリズムはバブルソートに限定しない.