コンピュータ実習
第7回 2025年11月20日

カーナビゲーション1
データ構造とファイル入出力

担当：相澤　直人，宍戸博紀

コンピュータ実習（第7回）

今日の実習課題を行う前に・・・

とりあえず、デモプログラムを動かしてみよう
（今日の授業ページから、「カーナビデモ」という
ページを開いて下さい。）

1. 最低限の機能を備えたプログラム
2. 過去の優秀作品のプログラム

この2つの動作イメージがあります。
優秀作品を目指して、頑張りましょう。

難しいな、と思う人は、最低限の機能のプログラムに
自分なりの工夫を加えたプログラムを目指しましょう。

コンピュータ実習（第7回）

一般的なカーナビの動作について

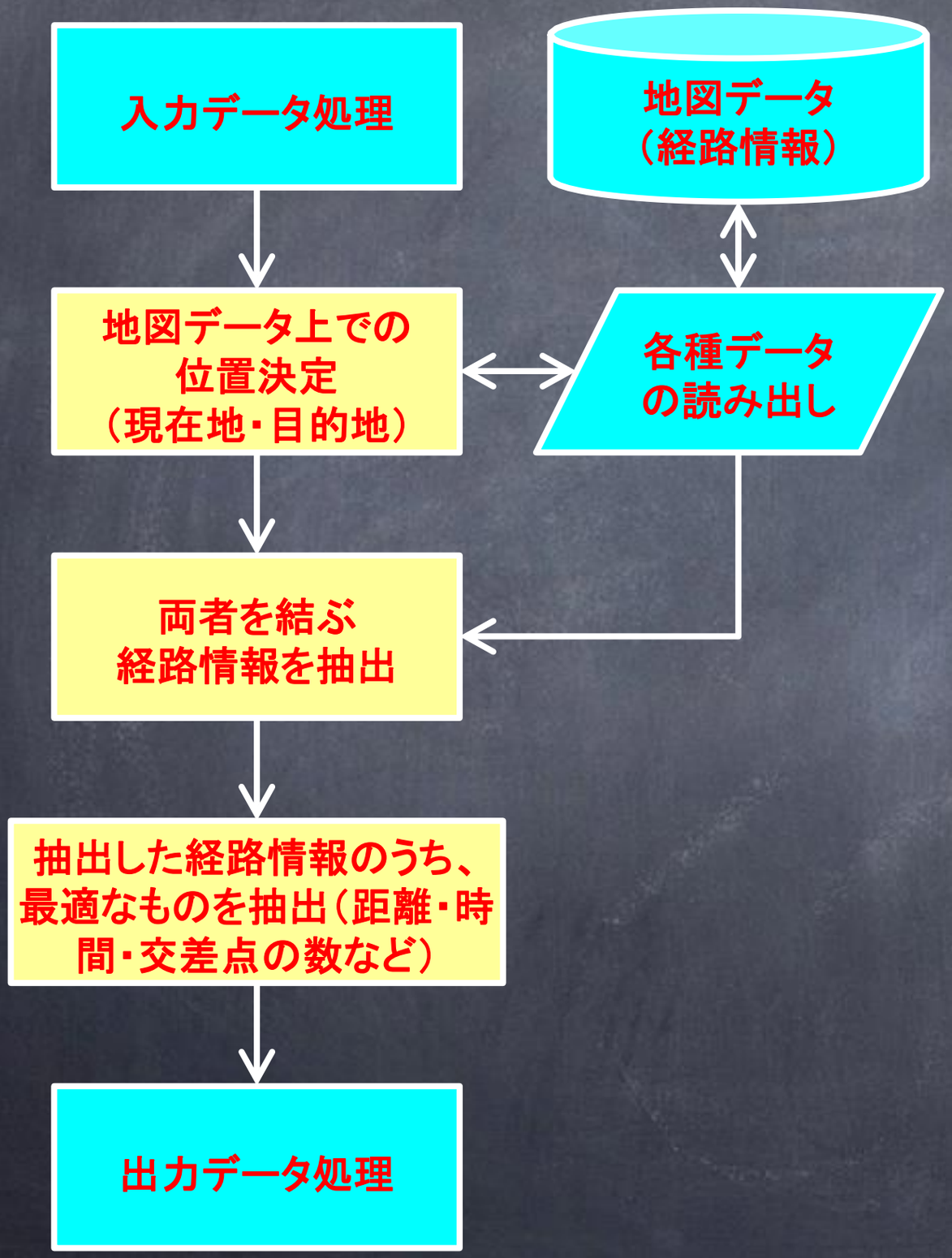
1. 地図データを元に、目的地を決定
   - 地図データの読み込み
   - 目的地（地図データ）の検索

2. 経路を探索
   - 条件に基づく経路の設定
   - 最適経路の計算（距離・時間等）

3. 経路をドライバーに分かりやすく提示
   - 画面上でアニメーション表示

コンピュータ実習（第7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7回）

今後の授業の進行

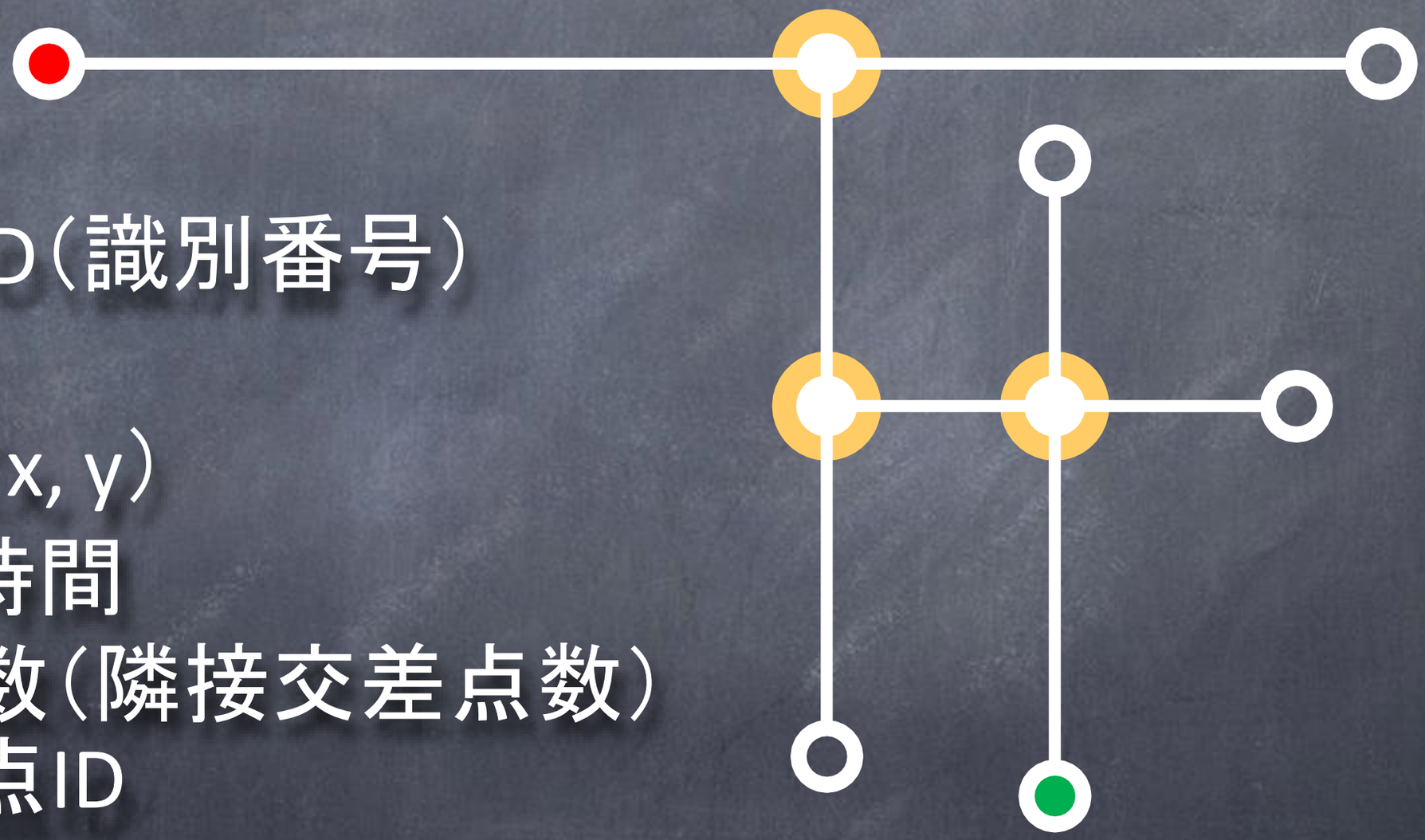
1. 地図データを元に、目的地を決定
   - **地図データの読み込み** ← 本日の講義
   - 目的地（地図データ）の検索
     - 次回(11/27)講義
2. 経路を探索
   - 条件に基づく経路の設定
     - 12/4講義(簡易版)
   - 最適経路の計算（距離・時間等）
     - 12/11講義
3. 経路をドライバーに分かりやすく提示
   - 画面上で車をアニメーション表示
     - 12/4 (12/25)講義

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）

カーナビゲーションに必要なデータ構造

道（経路）とは何か？　→　始点と終点を結ぶ線



交差点情報

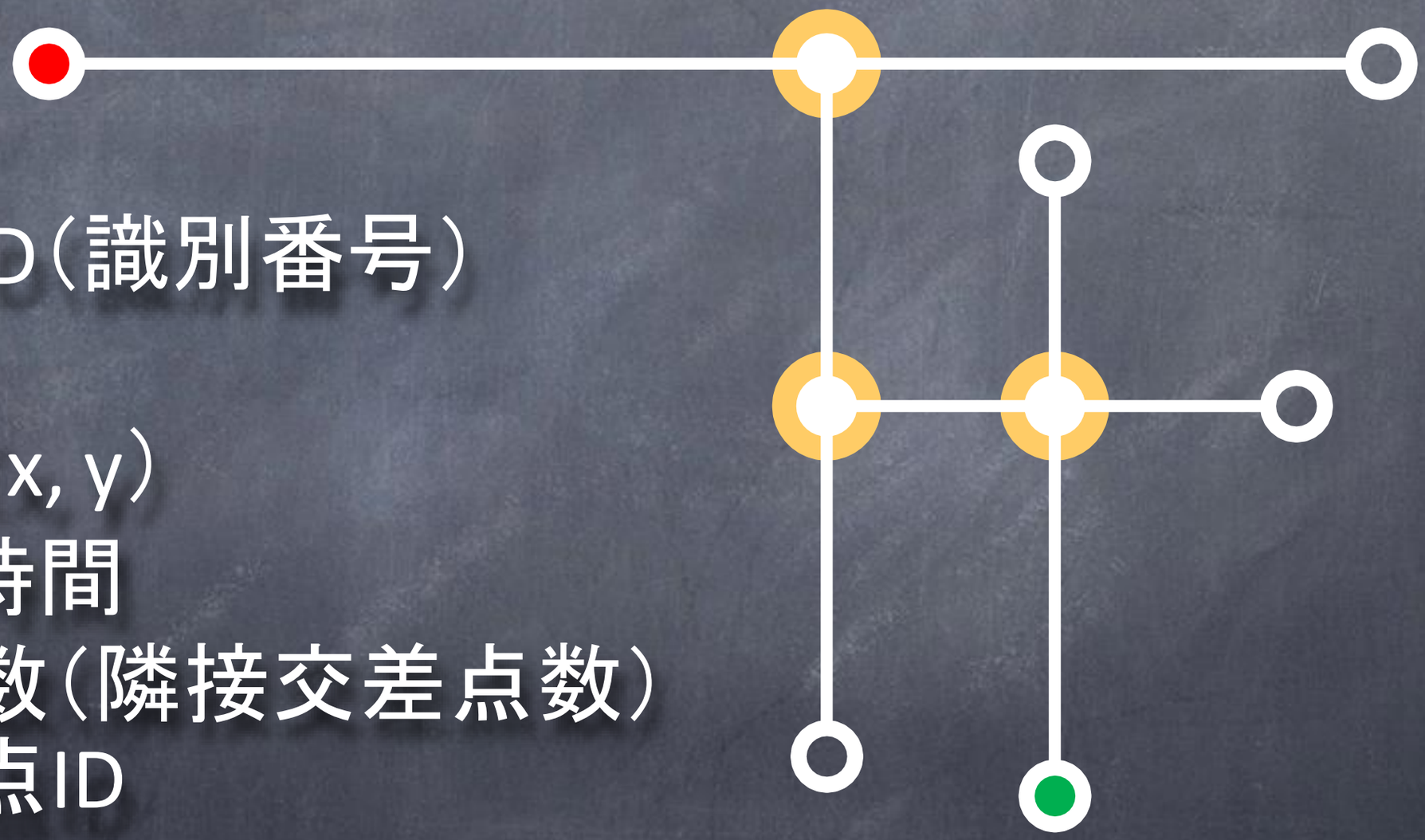
- 交差点のID（識別番号）
- 交差点名
- 位置座標（x, y）
- 平均待ち時間
- 交差道路数（隣接交差点数）
- 隣接交差点ID

隣接点が1カ所のみと2カ所以上の点
これらを繋いだ（リンクした）ものが「経路」

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造

道（経路）とは何か？　→　始点と終点を結ぶ線



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

始点・終点（ノード）、　経路（エッジ）
数学的にはグラフ理論と呼ばれるもの

コンピュータ実習（第7回）

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

情報ごとに1個1個、新たな交差点情報を追加しようとすると面倒

交差点情報	必要となる情報の変数
• 交差点のID（識別番号）	• int id[100];
• 交差点名	• char name[100][50];
• 位置座標（x, y）	• double x[100], y[100];
• 平均待ち時間	• double wait[100],
• 交差道路数（隣接交差点数）	• int points[100];
• 隣接交差点ID	• int next[100][5];

コンピュータ実習（第7回）

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）

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）

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）

カーナビゲーションに必要なデータ構造

構造体の型定義について（参考）

以前の講義では、以下のような表記をしていました

```
struct info{
    int id;
    char name[50];
};
typedef struct info Info
```

今回出てきた以下の表記は、省略形です

```
typedef struct {
    int id;
    char name[50];
} Info;
```

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造

構造体の使い方は、前に講義で出てきた通り

構造体は型を定義しているだけなので、その型を使用する変数の宣言をすること。
構造体の定義に変数が入っているが、これはこんな変数を使いますよ、
と言っているだけです。

構造体”Crossing”の定義

```
Crossing cross;          // Crossing型の構造体変数crossを宣言

cross.id = 1;            // IDに1を設定
cross.pos.x = 10.5;      // (x,y)座標を設定
cross.pos.y = -9.2;

scanf(“%lf”, &(cross.wait));   //待ち時間をキーボードから入力
scanf(“%s”, cross.name);       //交差点名をキーボードから入力
```

交差点は複数あるので、構造体を配列として活用

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造

入力方法（配列）

―プログラムに直書き＆キーボード入力

Crossing cross[100]

```
cross[0].id = 1;
cross[0].pos.x = 10.5;
cross[0].pos.y = −9.2;

scanf(“%lf”, &(cross[0].wait));
scanf(“%s”, cross[0].name);
```

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造
入力方法（配列） ―ファイルから読み込み

```
Crossing cross[100]
fp = open(“inputfile”, “r”);
for(i=0;i<100;i++){
    fscanf(fp, “%d”, &(cross[i].id));
    fscanf(fp, “%lf”, &(cross[i].pos.x));
    fscanf(fp, “%lf”, &(cross[i].pos.y));
    fscanf(fp, “%lf”, &(cross[i].wait));
    fscanf(fp, “%s”, cross[i].name);
}
```

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造

出力方法（配列） －ファイルへ書き出し例

```
fp = open(“outputfile”, “w”);
for(i=0;i<100;i++){
    fprintf(fp, “%d”, cross[i].id);
    fprintf(fp, “%lf”, cross[i].pos.x);
    fprintf(fp, “%lf”, cross[i].pos.y);
    fprintf(fp, “%lf”, cross[i].wait);
    fprintf(fp, “%s”, cross[i].name);
}
```

コンピュータ実習（第7回）

カーナビゲーションに必要なデータ構造

%[^,]の取り扱いについて

“,”(カンマ)が検出されるまでを1つの文字列として読み込むという意味

（入力データ例）

0,1.2,2.3,30,駅前青葉,3,41,38,40

（読み込みプログラム例）

```
fscanf(fp, “%d,%lf,%lf,%lf,%[^,],%d”,
    &(cross[i].id), &(cross[i].pos.x), &(cross[i].pos.y),
    &(cross[i].wait),cross[i].name, &(cross.points));
```

（あとの3つの読み込み方は考えてみよう）

%s は連続する文字列やカンマや数字を、別のものだと認識してくれません

コンピュータ実習（第7回）

課題1： キーボードから交差点情報を入力する

webテキストの課題1（crossing2.c）について、
未完成のひな型を参考に完成させてください。

ヒント

- 入力されたデータをターミナルへ出力して終了するプログラムです
- 入力も出力も複数回繰り返すので、ループ処理が必要となります
- 出力の際は、ループ処理回数の設定が必要です

＜補足＞

- 「判定文」の箇所にある”strcmp関数”は、カッコ内の2つの引数が一致するかを判定するための関数です。詳細は次回講義にて説明があります。

コンピュータ実習（第7回）

課題2：　交差点情報のファイルへの出力

webテキストの課題2（writefile.c）について、
未完成のひな型を参考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
  - データ入力とファイル出力
- main(void)以前はファイルへの出力を”map_write”という関数として定義した部分です。
  ユーザー定義のmap_write関数を完成させましょう。
- ファイル出力は以前に講義で説明したとおりです

完成したプログラムはwritefile.cとして保存しておくこと

コンピュータ実習（第7回）

課題2： 交差点情報のファイルへの出力

交差点データ”map.dat”の保存形式

```
//Total交差点数
  crossing_point
//cross[0]のデータ
  id[0], x[0], y[0], wait[0], points[0], next[0], … next[points[0]-1]
                                  ▪
                                  ▪
                                  ▪
//cross[crossing_point-1]のデータ
//便宜上、以下は“n=crossing_point-1”としている
  id[n], x[n], y[n], wait[n], points[n], next[0], … next[points[n]-1]
```

コンピュータ実習（第7回）

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

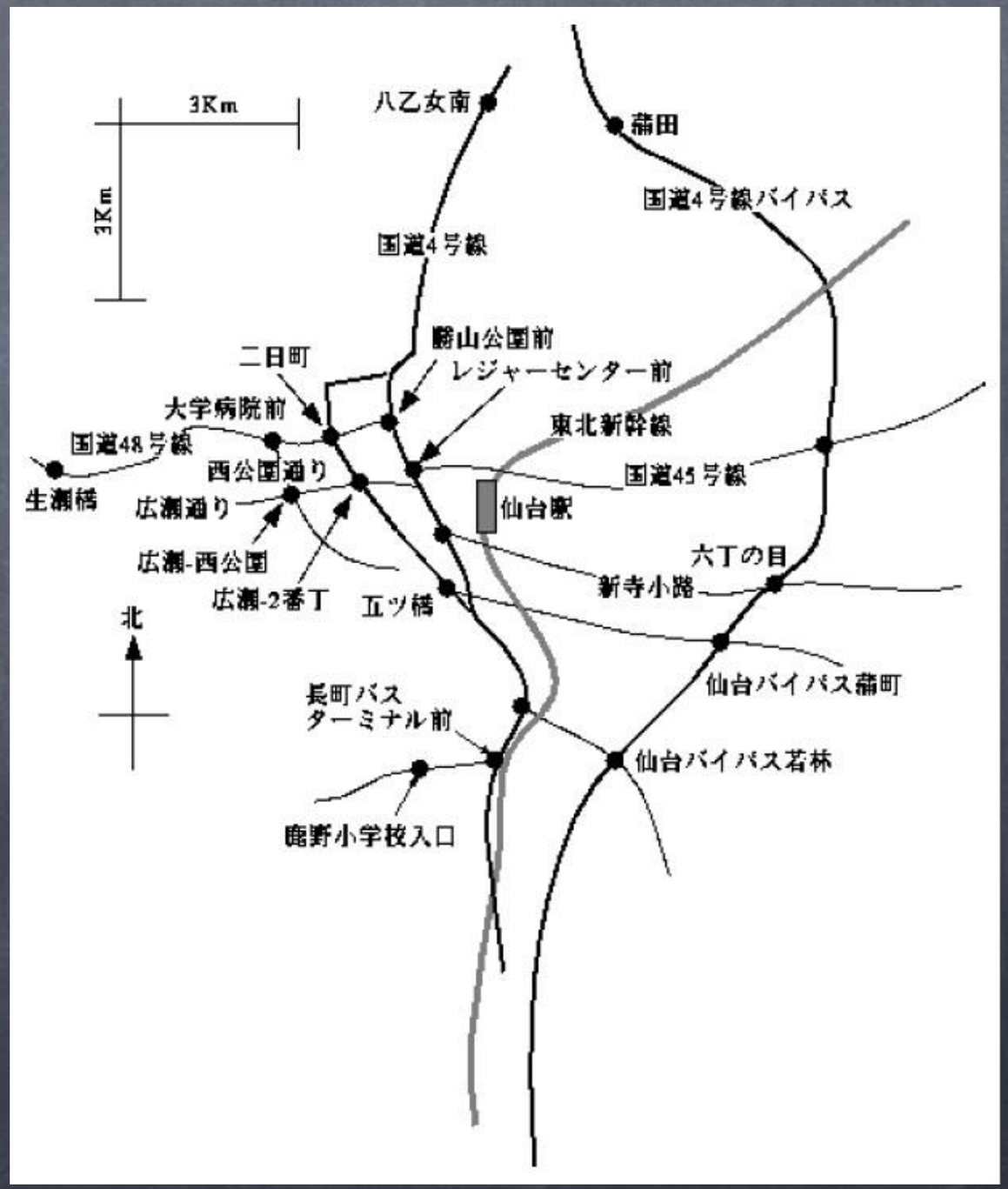
コンピュータ実習（第7回）

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7回）

課題3－2：　交差点情報のファイルからの入力

課題3（readfile.c）を改造して、map2.datを読み込めるようなmap_read関数を作成してください

これをカーナビの地図データ読み込み部として、次回以降に使用します

map_write関数とmap_read関数は重要なので、よく理解しておくように！