ソフトウェア２
第３回
（2012/12/20）

鶴岡 慶雅

連絡用ページ

- URL

  http://www.logos.t.u-tokyo.ac.jp/~tsuruoka/lecture/software2/

  ユーザ名:　 soft2

  パスワード:　ee2012

- 資料
  - 講義スライド(ppt, pdf)
  - サンプルプログラム

今日の内容

- C言語入門
  - 構造体
    - 宣言、メンバ
    - 構造体を指すポインタ
    - 初期化
  - コマンドライン引数
- 物理シミュレーション
  - 多体問題

構造体

- 複数のデータ型をまとめ、あらたなデータ型を定義

例

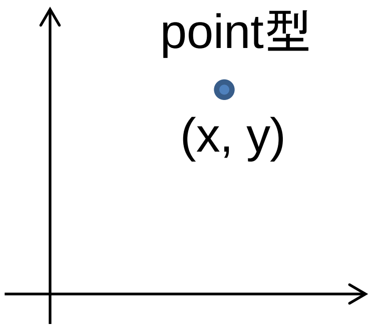
学生	
学生証番号	整数
名前	文字列
年齢	整数
身長	浮動小数点数
体重	浮動小数点数

```
struct student
{
        int id;
        char *name;
        int age;
        double height;
        double weight;
};
```

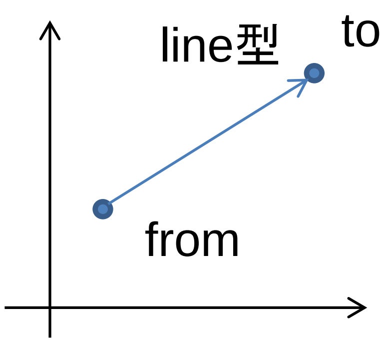
メンバ

構造体

- 構造体をメンバに持つ構造体




```
struct point
{
        int x;
        int y;
};
```




```
struct line
{
        struct point from;
        struct point to;
}
```

構造体

- 構造体の宣言、メンバへのアクセス

```
struct point
{
        int x;
        int y;
};

int main()
{
        struct point p1;

        p1.x = 10;
        p1.y = 20;
}
```

構造体

- 構造体の初期化・代入

```
struct point
{
        int x;
        int y;
};

int main()
{
        struct point a = { 10, 20 };
        struct point b;

        b = a;
}
```

構造体

- 構造体を指すポインタ

```
struct point
{
        int x;
        int y;
};

int main()
{
        struct point *pp1 = malloc(sizeof(struct point));

        (*pp1).x = 10;
        pp1->x = 10;
}
```

どちらでも同じ

構造体

- 構造体へのポインタのよくある使い方

```
void increment_age(struct student* ps)
{
	ps->age += 1;
}

void somefunc()
{
	struct student a = { 1, “Mike”, 21, 175, 72 };

	increment_age(&a);
}
```

  – 関数に渡すのはポインタだけなので高速
    - 値渡しにすると構造体のメンバをすべてコピー

コマンドライン引数

- プログラムの実行時にパラメータやオプションを指定できる

コレ

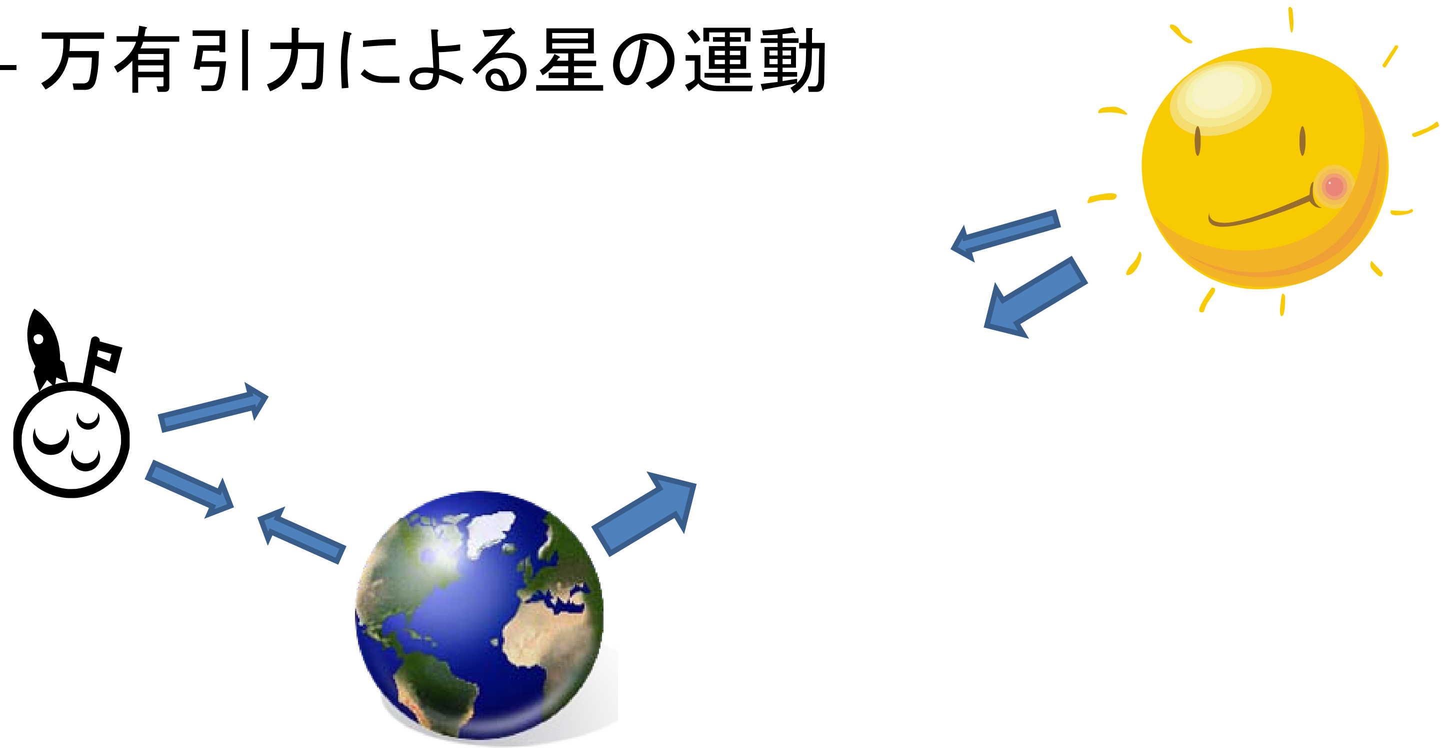
./a.out 0.1

  - 文字列へのポインタの配列として得られる
  - 数値に変換したいときは atoi, atof 等を利用

```
int main(int argc, char *argv[])
{
        int i;
        for (i = 0; i < argc; i++) {
                printf("%d %s¥n", i, argv[i]);
        }
}
```

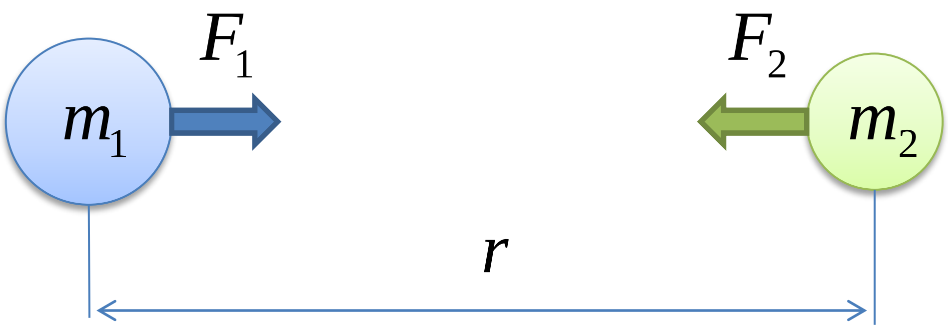
物理シミュレーション

- 多体問題（*n*-body problem）
  - 万有引力による星の運動



万有引力

- 質量を有する2つの物体に働く引力



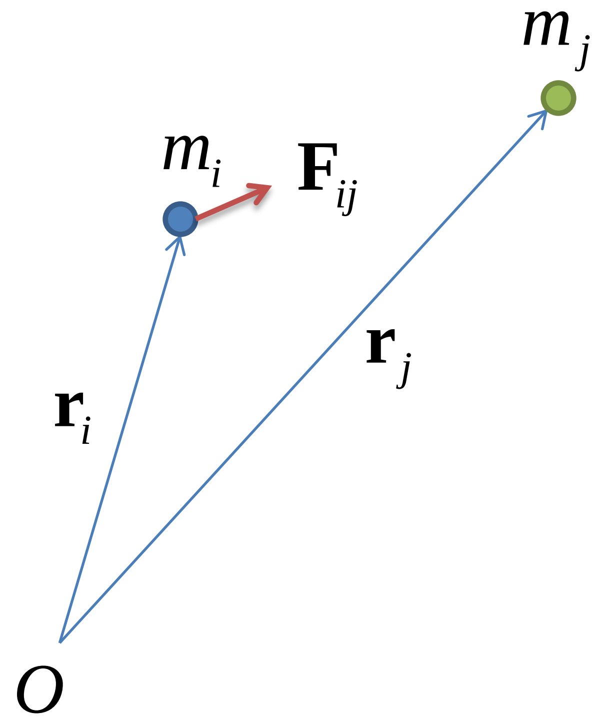
$$F_1 = F_2 = G\frac{m_1 m_2}{r^2}$$

重力定数　$G \approx 6.674 \times 10^{-11} \left[\mathrm{m^3 s^{-2} kg^{-1}}\right]$

重力多体系

- 粒子 i が粒子 j から受ける重力（ベクトル）

$$\mathbf{F}_{ij} = G\frac{m_i m_j}{\left|\mathbf{r}_j - \mathbf{r}_i\right|^2}\frac{\mathbf{r}_j - \mathbf{r}_i}{\left|\mathbf{r}_j - \mathbf{r}_i\right|} = G\frac{m_i m_j}{\left|\mathbf{r}_j - \mathbf{r}_i\right|^3}\left(\mathbf{r}_j - \mathbf{r}_i\right)$$



- 多数の粒子から受ける重力

$$\mathbf{F}_i = \sum_{j\neq i}\mathbf{F}_{ij}$$

運動方程式

- 物体の運動を記述する微分方程式

位置

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i$$

速度

$$m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{F}_i$$

$$= G m_i \sum_{j \neq i} \frac{m_j}{\left|\mathbf{r}_j - \mathbf{r}_i\right|^3} \left(\mathbf{r}_j - \mathbf{r}_i\right)$$

$$\Downarrow$$

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = G \sum_{j \neq i} \frac{m_j}{\left|\mathbf{r}_j - \mathbf{r}_i\right|^3} \left(\mathbf{r}_j - \mathbf{r}_i\right)$$

オイラー法（Eurer’s method）

- １階常微分方程式の数値解法

関数 $x(t)$ に関して以下が成り立つ

$$x'(t) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

$$x(t+\Delta t) \approx x(t) + x'(t)\Delta t$$

$$\begin{cases} \mathbf{a}_i^{(n)} = G \sum_{j \neq i} \dfrac{m_j}{\left|\mathbf{r}_j^{(n)} - \mathbf{r}_i^{(n)}\right|} \left(\mathbf{r}_j^{(n)} - \mathbf{r}_i^{(n)}\right) \\ \mathbf{v}_i^{(n+1)} = \mathbf{v}_i^{(n)} + \mathbf{a}_i^{(n)} \cdot \Delta t \\ \mathbf{r}_i^{(n+1)} = \mathbf{r}_i^{(n)} + \mathbf{v}_i^{(n+1)} \cdot \Delta t \end{cases}$$

サンプルプログラム gravity.c

- コンパイル＆実行

% gcc -O2 gravity.c

% ./a.out

最適化オプション（プログラムの実行速度が速くなる）

- ターミナルをもうひとつ開く（表示用）

% tail –f space.txt

ターミナルのサイズをマウスで調整して ---------- が左上にくるように

課題（締め切り12/26）

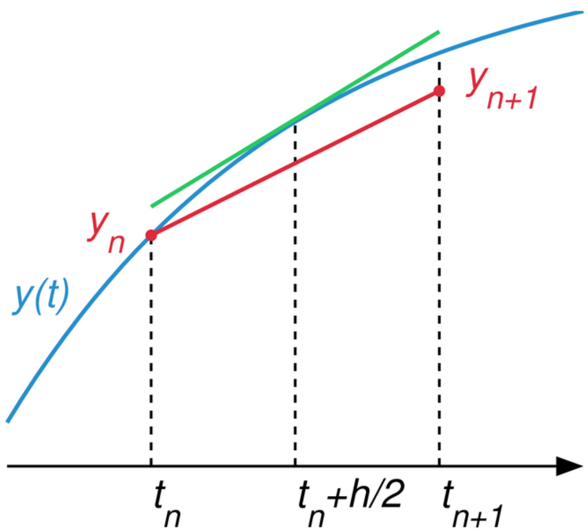
1. 時間の刻み幅 Δt をコマンドライン引数で指定できるように拡張せよ
   - プログラムを添付すること（ファイル名は “gravity1.c”）
2. ２次元空間を扱えるように拡張せよ
   - 星の数を増やして動作を確認すること
   - プログラムを添付すること（ファイル名は “gravity2.c”）
3. 星と星との衝突（融合）現象を実装せよ
   - 星同士の距離がある値以内になったら融合（運動量保存）
   - プログラムを添付すること（ファイル名は “gravity3.c”）
4. ［発展課題］ ルンゲ・クッタ法などを利用してシミュレーション精度を向上させよ
   - プログラムを添付すること（ファイル名は “gravity4.c”）
   - 手法の簡単な説明と、精度が向上したこと（理論値とのずれが小さくなった、Δt を小さくするのと同じ効果が得られた、など）がわかる実例も記述すること

課題の提出方法

- 宛先
  - software2@logos.t.u-tokyo.ac.jp
- Subject
  - 形式: SOFT-MM-DD-NNNNNNX
    - MM: 月
    - DD: 日（授業が行われた日）
    - NNNNNNX: 学籍番号
- 本文
  - 冒頭に学籍番号、氏名を明記

中点法（Midpoint method）

- ２次のルンゲ･クッタ法（Runge-Kutta method）



http://en.wikipedia.org/wiki/Midpoint_method

$$\mathbf{r}_i^{\left(n+\frac{1}{2}\right)} = \mathbf{r}_i^{(n)} + \mathbf{v}_i^{(n)} \cdot \frac{\Delta t}{2}$$

$$\mathbf{a}_i^{\left(n+\frac{1}{2}\right)} = G \sum_{j \neq i} \frac{m_j}{\left| \mathbf{r}_j^{\left(n+\frac{1}{2}\right)} - \mathbf{r}_i^{\left(n+\frac{1}{2}\right)} \right|^3} \left(\mathbf{r}_j^{\left(n+\frac{1}{2}\right)} - \mathbf{r}_i^{\left(n+\frac{1}{2}\right)} \right)$$

$$\mathbf{v}_i^{(n+1)} = \mathbf{v}_i^{(n)} + \mathbf{a}_i^{\left(n+\frac{1}{2}\right)} \cdot \Delta t$$

$$\mathbf{r}_i^{(n+1)} = \mathbf{r}_i^{(n)} + \frac{\left(\mathbf{v}_i^{(n+1)} + \mathbf{v}_i^{(n)} \right)}{2} \cdot \Delta t$$