

# ソフトウェア2

## [第2回]

(2020/11/26)



齋藤 大輔

# 講義用ページ

---

- URL

[https://www.gavo.t.u-tokyo.ac.jp/~dsk\\_saito/lecture/software2](https://www.gavo.t.u-tokyo.ac.jp/~dsk_saito/lecture/software2)

- 進行はスライドを中心に行うが適宜ページも参照

# 成績評価

---

- 全6回のレポート課題により評価
  - 基本課題については全て採点対象
  - 発展課題については全6回中上位3回分を採用
- レポート期限は締切日の23:59:59（日本時間）

# 本日のメニュー

---

- C言語入門編
  - 構造体
- 今日の題材
  - 物理シミュレーション
- プログラム解説
- 実習
- レポート課題について

# 本日のメニュー

---

- C言語入門編

  - 構造体

- 今日の題材

  - 物理シミュレーション

- プログラム解説

- 実習

- レポート課題について

# 構造体

- 複数のデータ型をまとめ、新たなデータ型をつくる

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[100];  
    int age;  
    double height;  
    double weight;
```

```
};
```

メンバ

# 構造体

---

## ■ 構造体の宣言、定義、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()
{
    // 初期化時にはこの形式がおすすめ
    struct point a = {
        .x = 10,
        .y = 20
    };
    struct point b;

    b = a;
}
```



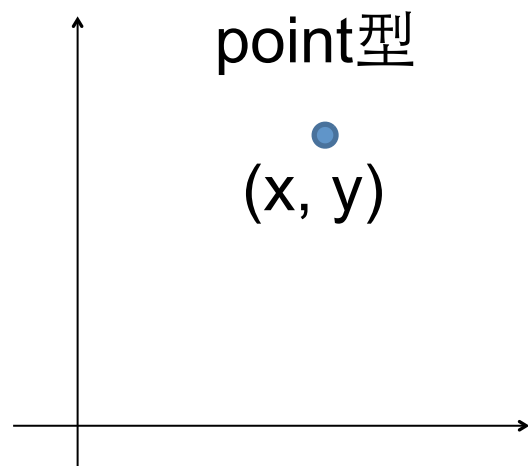
# struct\_init.c

---

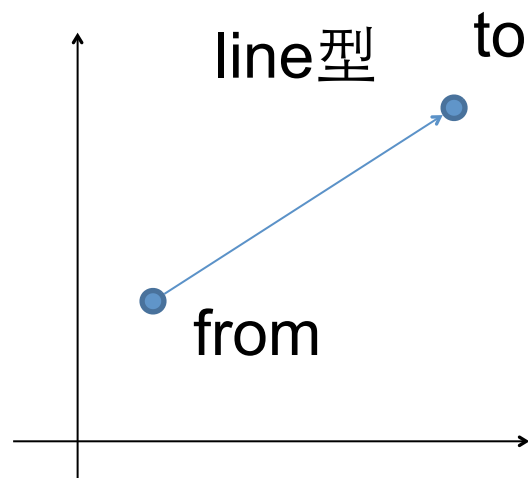
- 構造体の初期化や代入についてまとめているので一読すること
- 構造体は座標を表す単純なもの

# 構造体

- 構造体をメンバに持つ構造体もつくれる



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



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

# 構造体

## ■ 構造体を指すポインタ

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

```
int main()
{
    struct point a;
    struct point *p = &a;

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

どちらでも同じ  
(下の方[アロー演算子]が一般的)

# 構造体

- 構造体へのポインタのよくある使い方
  - 関数呼び出し

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

ポインタ変数で構造体のアドレスを受け取る

```
void some_function ()
{
    struct student a = { 1, "Mike", 21, 175, 72 };

    increment_age (&a);
}
```

構造体aが格納されているメモリの  
アドレスを渡す

# 構造体

- もちろんポインタではなく値渡しでもできる


```
void print_age(struct student s)
{
    printf("age = %d\n", s.age);
}
```

- ただし、大きな構造体を渡すときには注意

- 関数を呼び出すたびに構造体がコピーされるので無駄が大きい

- 大きな構造体を渡すときは原則としてポインタで

構造体の内容を変更しない場合は安全のため `const` をつける



```
void print_age(const struct student *s)
{
    printf("age = %d\n", s->age);
}
```

# 構造体

## ■ 構造体のサイズ

```
struct point1
{
    int x;      ← 4バイト
    int y;      ← 4バイト
};
```

```
struct point2
{
    int x;
    int y;
    char c;     ← 1バイト
};
```

```
int main()
{
    printf("%zu\n", sizeof(struct point1));
    printf("%zu\n", sizeof(struct point2));
}
```

## 実行結果の例

```
$ ./a.out
8
12
```

point2 のサイズが9ではない（！）

4バイト境界へのアラインメントのため

0x600d0	
0x600d1	int x
0x600d2	
0x600d3	
0x600d4	
0x600d5	int y
0x600d6	
0x600d7	
0x600d8	
0x600d9	
0x600da	
0x600db	

} パディング  
14

# 実習

---

- `struct_alignment.c` を修正し、定義された構造体のサイズを調べる (`struct_alignment.c`)
- 基本型の変数のアドレスを複数表示してみる
- `struct_init.c` をもとに、`struct point` の配列1000個の平均 (=重心) を計算するプログラムを書く
  - 各点の座標はランダムに初期化してよい

# 本日のメニュー

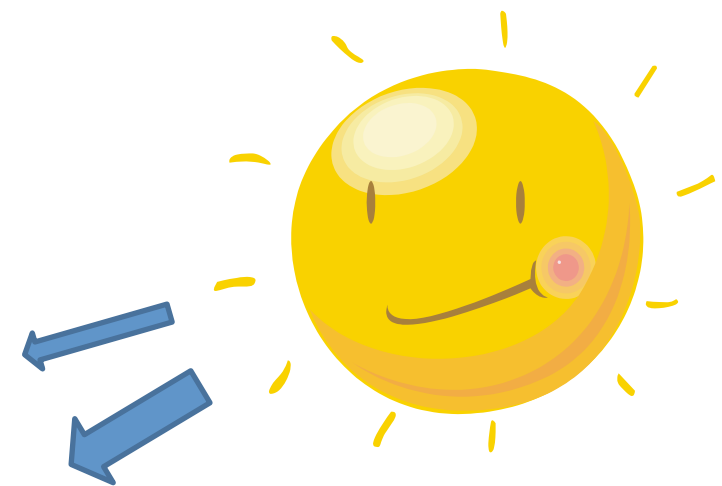
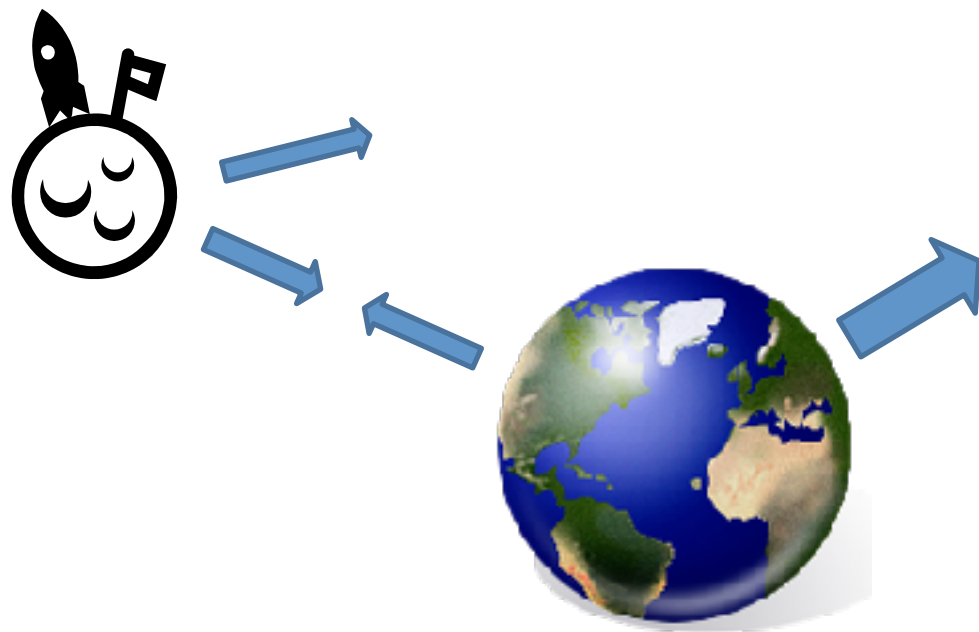
---

- C言語入門編
  - 構造体
- 今日の題材
  - 物理シミュレーション
- プログラム解説
- 実習
- レポート課題について



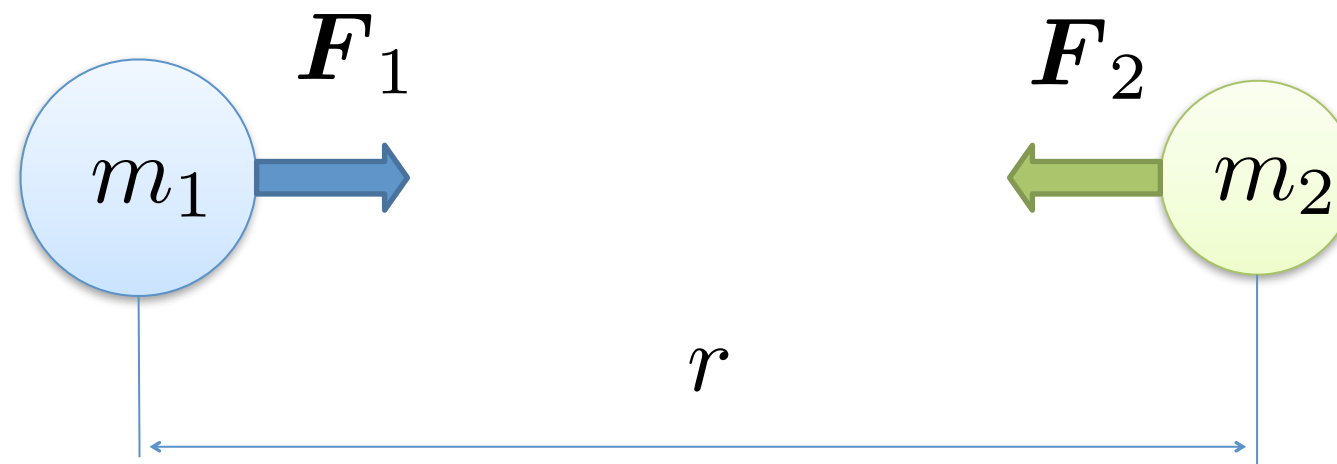
# 物理シミュレーション

- 物理法則を条件として計算機シミュレーション
- 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  $G \approx 6.674 \times 10^{-11} [\text{m}^3 \text{s}^{-2} \text{kg}^{-1}]$

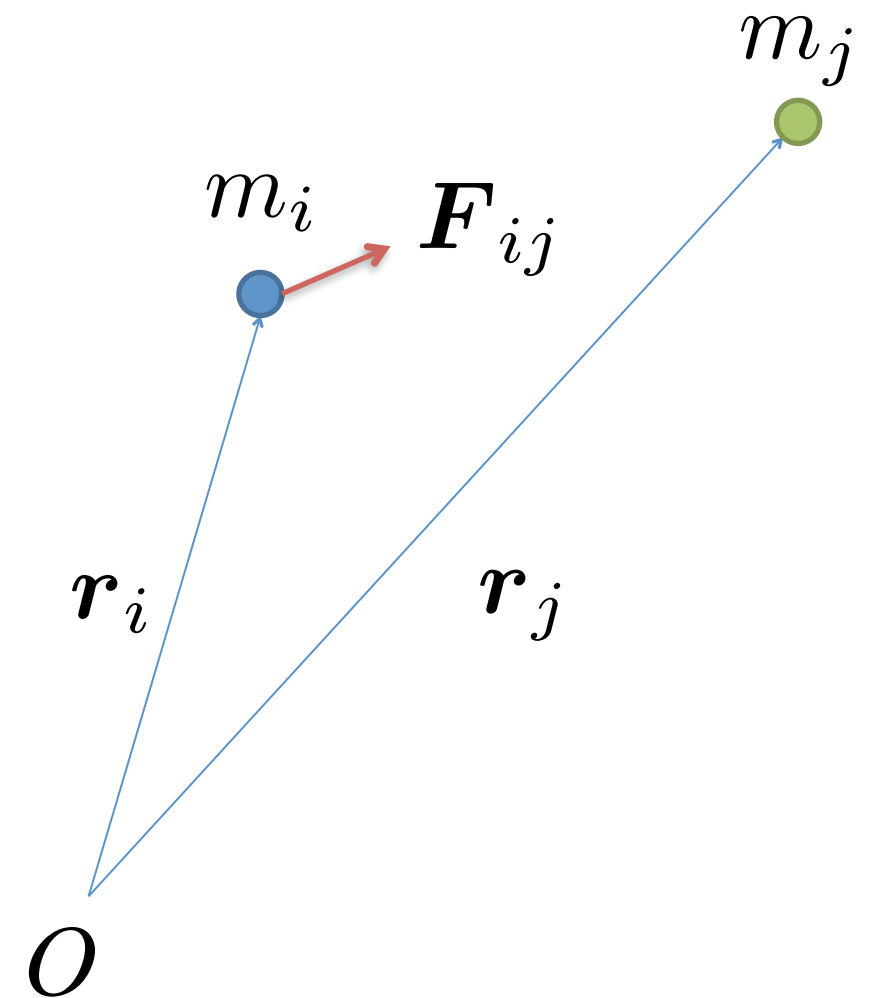
# 重力多体系

- 粒子  $i$  が粒子  $j$  から受ける重力 (ベクトル)

$$\begin{aligned} \mathbf{F}_{ij} &= G \frac{m_i m_j}{|\mathbf{r}_j - \mathbf{r}_i|^2} \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|} \\ &= G \frac{m_i m_j}{|\mathbf{r}_j - \mathbf{r}_i|^3} (\mathbf{r}_j - \mathbf{r}_i) \end{aligned}$$

- 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

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i$$

速度

$$\begin{aligned} m_i \frac{d\mathbf{v}_i}{dt} &= \mathbf{F}_i \\ &= Gm_i \sum_{j \neq i} \frac{m_j}{|\mathbf{r}_j - \mathbf{r}_i|^3} (\mathbf{r}_j - \mathbf{r}_i) \end{aligned}$$



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

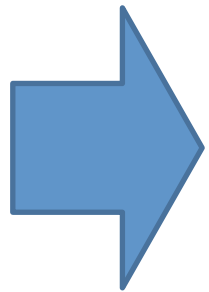
# オイラー法 (Euler's method)

## ■ 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

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

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



$$\begin{aligned} \mathbf{a}_i^{(n)} &= G \sum_{j \neq i} \frac{m_j}{|\mathbf{r}_j^{(n)} - \mathbf{r}_i^{(n)}|^3} \left( \mathbf{r}_j^{(n)} - \mathbf{r}_i^{(n)} \right) \\ \mathbf{v}_i^{(n+1)} &= \mathbf{v}_i^{(n)} + \mathbf{a}_i^{(n)} \times \Delta t \\ \mathbf{r}_i^{(n+1)} &= \mathbf{r}_i^{(n)} + \mathbf{v}_i^{(n)} \times \Delta t \end{aligned}$$

# 基本的な手順

---

- 微小時間 $dt$  に対する加速度の更新式に基づき  
加速度の値を更新
- 加速度と速度、微小時間の関係から速度の値を更新
- 上記で求めた値と微小時間の関係から位置を更新

# 本日のメニュー

---

- C言語入門編
  - 構造体
- 今日の題材
  - 物理シミュレーション
- **プログラム解説**
- 実習
- レポート課題について

# 事前準備

---

- サイトの記述をもとに以下をすませること
  - ダウンロード
  - 環境毎にシンボリックリンクを設定



# サンプルコードfalling.c

---

- コンパイル&実行（プログラム名をfallingとする）

```
% gcc -o falling -Wall falling.c -L. -  
lfall -lm  
% ./falling
```

- 丸が自由落下していきます
- ライブラリを用いている点は前回と同じ
- Linux ユーザは特に、最後の-lm を忘れないこと

# サンプルコード **bouncing.c**

---

- コンパイル（プログラム名をbouncingとする）

```
% gcc -o bouncing -Wall bouncing.c -L.  
-lbounce -lm  
% ./bouncing
```

- 今度は壁があります
- ライブラリを用いている点は前回と同じ
- Linux ユーザは特に、最後の-lm を忘れないこと

# 全体像 (falling)

---

- physics.h:
  - 今回の構造体の定義と関数プロトタイプ宣言
- libfall.a
  - プロトタイプ宣言で宣言されている
- falling.c
  - main 関数の記述
  - プログラムの全体像

# 全体像 (bouncing)

---

- physics2.h:
  - 今回の構造体の定義と関数プロトタイプ宣言
- libbounce.a
  - プロトタイプ宣言で宣言されている
- falling.c
  - main 関数の記述
  - プログラムの全体像

# やるべき順序

---

- まずはmainのあるソースを読む
- その上でヘッダファイルを読む
- 全体像を理解した上で  
実装すべき関数をmy\_??????? で実装
- ヘッダファイル内の使わなくなった  
プロトタイプ宣言の削除
- -L. および -l???? を除いてコンパイル
  - ただし -lm は残しておくこと

# 本日のメニュー

---

- C言語入門編
  - 構造体
- 今日の題材
  - 物理シミュレーション
- プログラム解説
- **実習**
- レポート課題について

# 実習

---

- まずは物理パラメータや座標、速度などを変化させてコンパイルし、遊んでみる
- main のある関数内で使われていると同様の機能を持つ my\_\*関数を完成させる

# レポート課題（締切12/02）

---

1. struct\_alignment.c の3つの構造体について、構造体中のメンバ変数に割り当てられているアドレスを表示するプログラムを作成せよ
2. boucing.c をもとに今回の物理シミュレーションについて2次元空間を扱えるように拡張せよ
3. 第一引数にオブジェクトの個数、第二引数に2次元座標と質量が記載されたファイルから物体の設定を読み取れるようにせよ。
4. 物体同士の衝突（融合）現象を実装せよ
5. [発展課題] 本アプリケーションをさらに発展させよ

**詳細はWebページを確認すること**



# 課題の提出方法 (ITC-LMS)

---

- 形式: ファイルアップロード
  - 全てのプログラム/ファイルをまとめ、zipやtar.gzで圧縮
  - git archive コマンドやzipコマンド等を用いる
    - git での提出はもう少々お待ちください
  - SOFT-11-26-NNNNNNNNNN.zip または  
SOFT-11-26-NNNNNNNNNN.tar.gz
    - NNNNNNNNNN 部分は学籍番号 (ハイフン除く)
    - JについてはJ???????? のようにしてください。
- 課題について
  - [基本課題] 毎回提出
  - [発展課題] 成績計算に全6回中上位3回分を採用する