

ソフトウェア2

[第2回]

(2018/12/06)



齋藤 大輔

講義用連絡ページ

■ URL

https://www.gavo.t.u-tokyo.ac.jp/~dsk_saito/lecture/software2

■ ユーザ名: ee2018

■ パスワード: software

■ 講義スライドやサンプルプログラムを適宜ダウンロードすること

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
 - main関数におけるコマンドライン引数の扱い
- 今日の題材
 - 多体問題の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

- main関数におけるコマンドライン引数の扱い

- 今日の題材

- 多体問題の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()
{
    struct point a = { 10, 20 };
    struct point b;

    b = a;
}
```


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("%d\n", sizeof(struct point1));
    printf("%d\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（！）

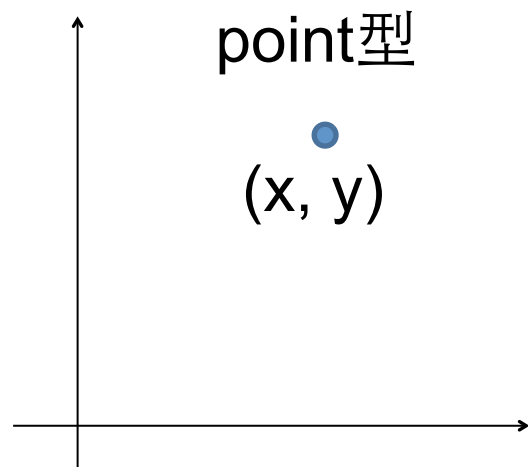
4バイト境界へのアラインメントのため

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

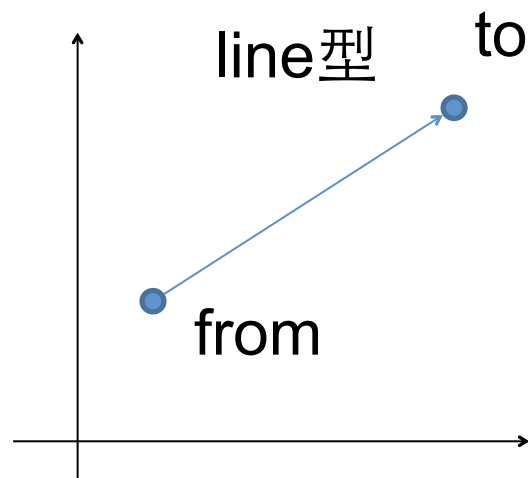
} パディング
9

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);
}
```

コマンドライン引数（復習）

- 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

`% ./a.out 0.1`

コレ

- 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

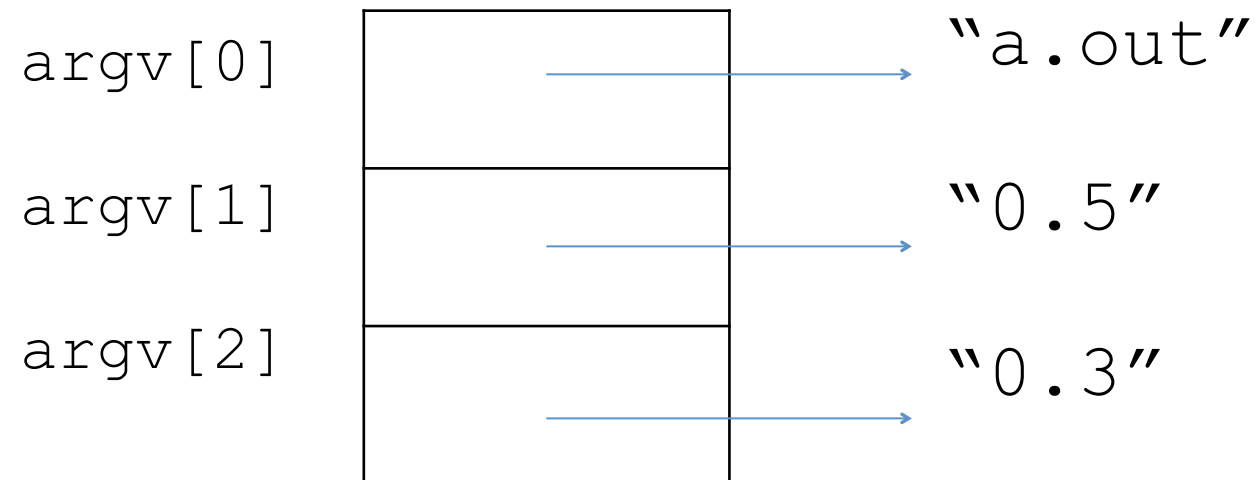
（“a.out”も含めた）引数の数

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

char *argv[] とは？

■ 文字（列）へのポインタの配列

```
% ./a.out 0.5 0.3
```



■ よくある別な書き方

```
int main(int argc, char **argv)
```

文字（列）へのポインタのポインタ

本日のメニュー

- C言語入門編

 - 構造体

 - main関数におけるコマンドライン引数の扱い

- 今日の題材

 - 多体問題の物理シミュレーション

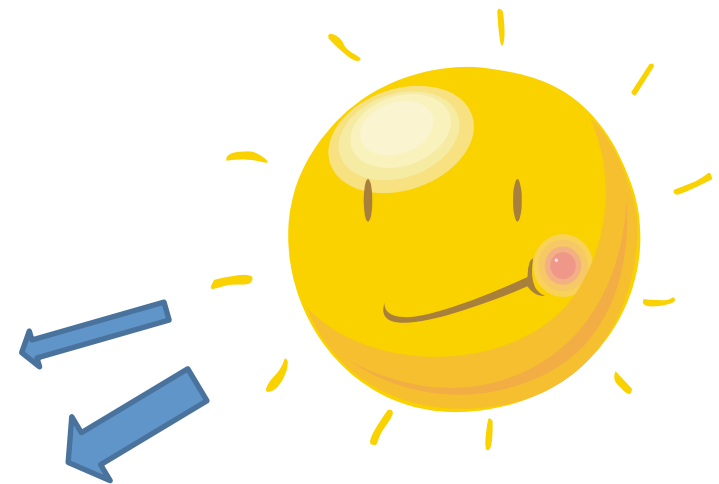
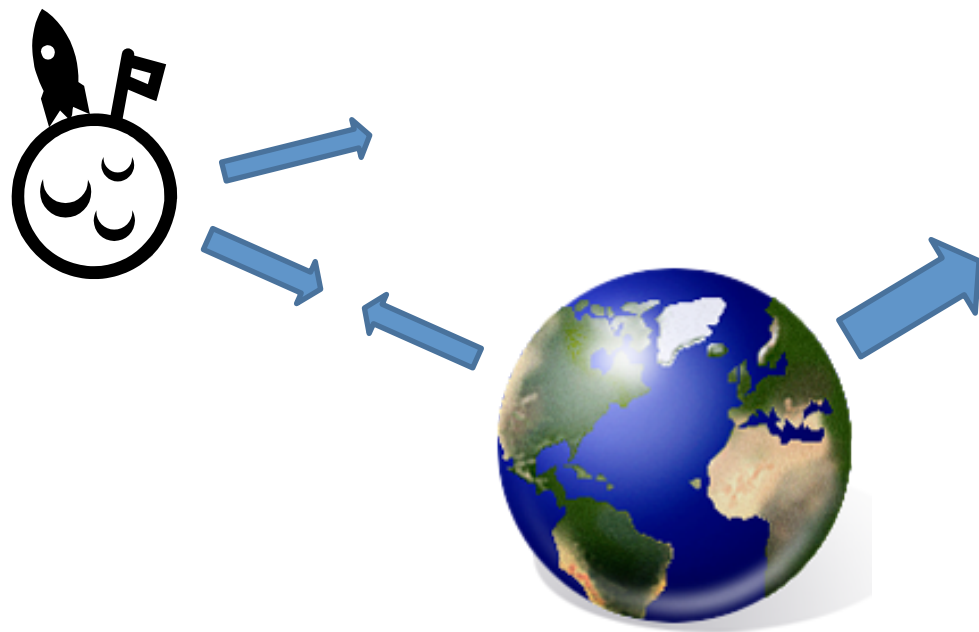
- プログラム解説

- 実習

- レポート課題について

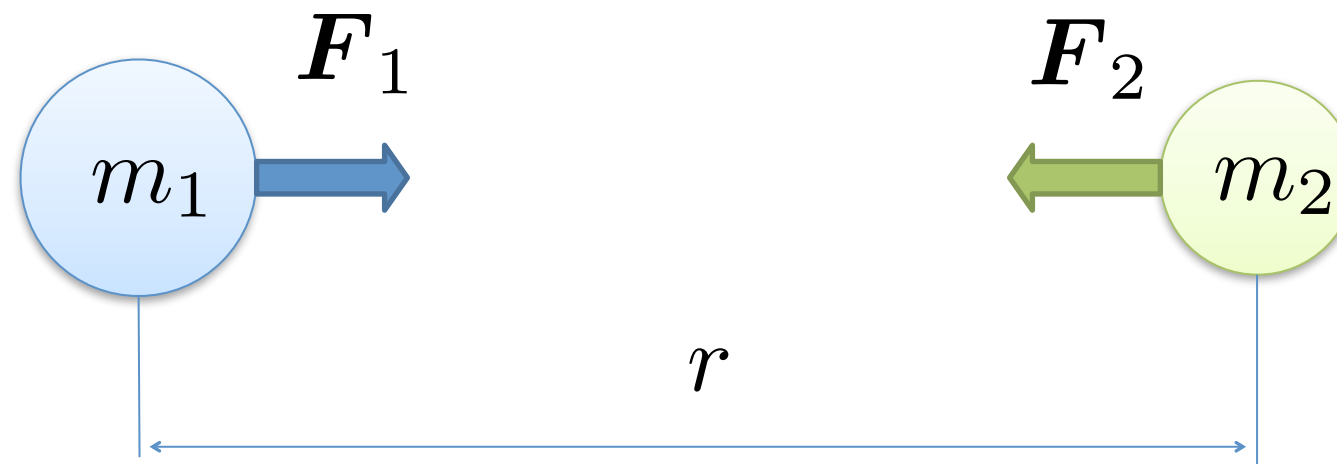
物理シミュレーション

- 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 $G \approx 6.674 \times 10^{-11} [\text{m}^3 \text{s}^{-2} \text{kg}^{-1}]$

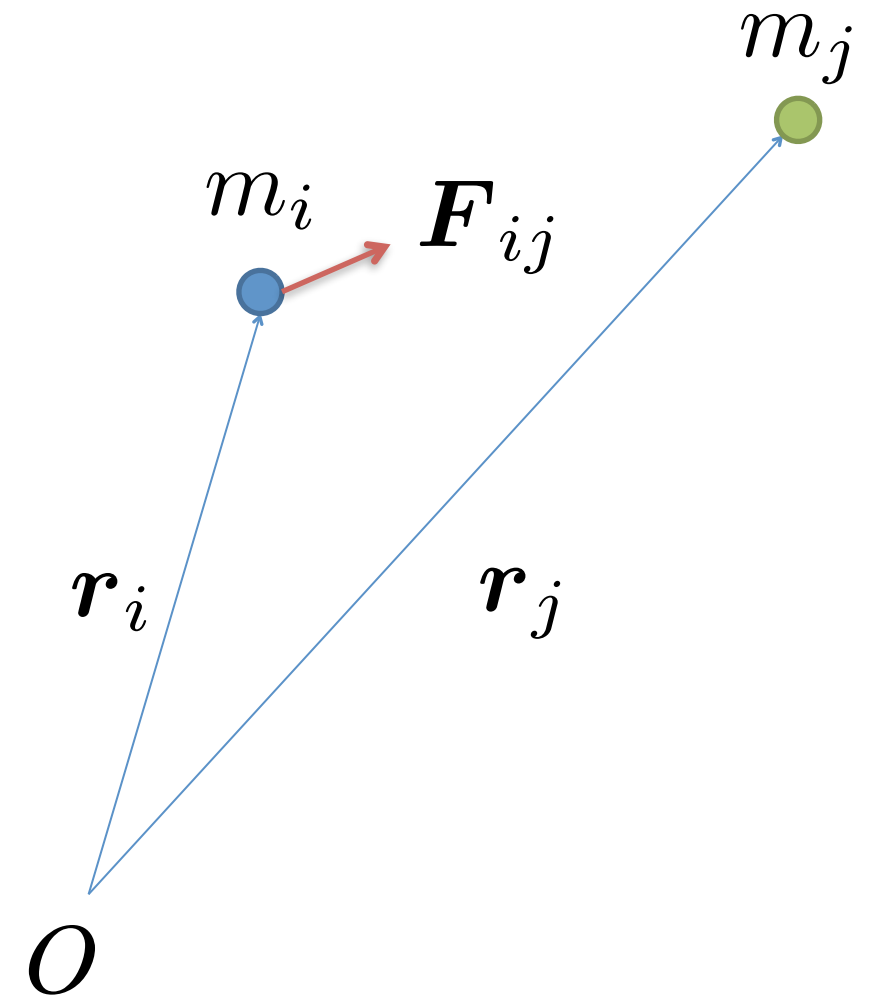
重力多体系

- 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

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



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

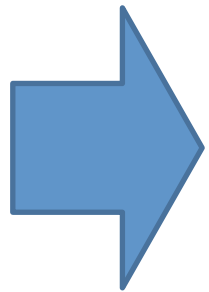
オイラー法 (Euler's method)

■ 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

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

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



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

本日のメニュー

- C言語入門編

 - 構造体

 - main関数におけるコマンドライン引数の扱い

- 今日の題材

 - 多体問題の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

サンプルコード gravity.c

- コンパイル&実行（プログラム名をgravityとする）

```
% gcc -o gravity -Wall gravity.c  
% ./gravity
```

- ターミナルをもうひとつ開く（結果表示用）

```
% tail -f space.txt
```

ターミナルのサイズをマウスで調整（縦に大きく）して
----- が左上にくるように

- 注意

- 何度も実行していると space.txt ファイルのサイズが大きくなるので適当なタイミングで初期化すること

```
% :> space.txt
```

gravity.c 冒頭

■ 構造体の宣言

```
struct star
{
    double m;    // mass
    double x;    // position_x
    double vx;   // velocity_x
};
```

■ 構造体を使った配列の初期化

```
struct star stars[] = {
    { 1.0, -10.0, 0.0 },
    { 0.5,  10.0, 0.2 }
};
```

2つの星のデータを用意

重さ	x 座標	速度 (x成分)
1.0	-10.0	0
0.5	10.0	0.2

gravity.c 冒頭

■ sizeof について

- 変数、型、配列などの大きさバイトを単位として返す

```
const int nstars = sizeof(stars) / sizeof(struct star);
```

48バイト

24バイトの構造体が2つ
格納されているので

24バイト

double がひとつで8バイト
(64ビット) なので

■ 割り算すると、結局 nstars が星の数になる

- 静的配列のサイズを測る方法のひとつ（動的配列では使えない）

plot_stars()関数

■ memset() について

■ 連続領域のメモリを指定したバイトで埋める

```
void plot_stars(FILE *fp, const double t)
{
    // 中略

    memset(space, ' ', sizeof(space));
}
```

開始位置
&(space[0][0]) と等価

埋める内容
(空白文字)

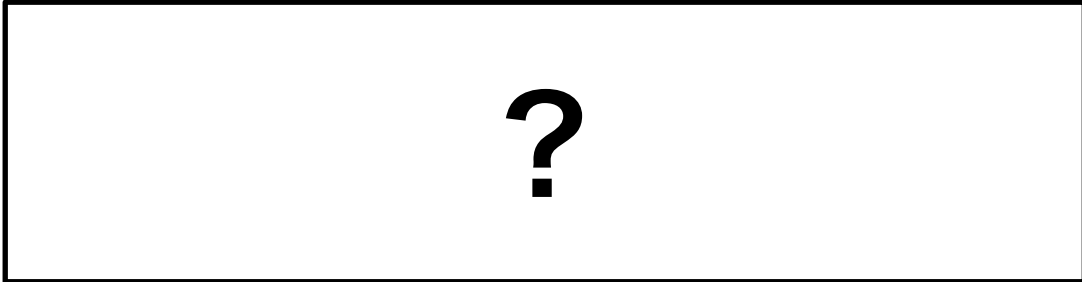
バイト数

■ char型の2次元配列を空白文字で初期化している

- もちろん for 文の2重ループで初期化しても構わない

update_velocities() 関数

■ 星の速度を更新する

```
void update_velocities(const double dt)
{
    int i, j;
    for (i = 0; i < nstars; i++) {
        double ax = 0;
        
        stars[i].vx += ax * dt;
    }
}
```

他の星から受ける引力を
もとに加速度を計算

gravity.c その他

■ usleep() 関数

- スリープする時間をマイクロ秒で指定

■ fabs() 関数

- 浮動小数点数の絶対値を返す
- update_velocities() 関数で使う

本日のメニュー

- C言語入門編

 - 構造体

 - main関数におけるコマンドライン引数の扱い

- 今日の題材

 - 多体問題の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

- `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`)
- `update_velocities` 関数を完成させる (`gravity.c`)
- 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する (`gravity.c`)

レポート課題（締切12/6）

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
 - プログラムを添付すること（ファイル名は“struct_alignment.c”）
 - プログラムのコメントにどのようなアラインメントが起きているかについての考察を記せ
2. 多体問題について2次元空間を扱えるように拡張せよ
 - 星の数を3つ以上に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“gravity1.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するとしてよい（運動量保存）
 - プログラムを添付すること（ファイル名は“gravity2.c”）
4. [発展課題] 本アプリケーションをさらに発展させよ
 - プログラムを添付すること（ファイル名は“gravity3.c”）
 - 発展のさせ方は任意
 - 例）中点法等によるシミュレーションの精度向上、3次元空間に拡張、地球と月と太陽の配置を再現してみる、衝突による破壊・散乱を考慮、電界・磁界を考慮、etc.

課題の提出方法（メール）

■宛先

- software2@gavo.t.u-tokyo.ac.jp

■メールのSubject

- 形式: SOFT-MM-DD-NNNNNNNNNN

- MM: 月 / DD: 日（授業が行われた日） / 1桁の場合は01, 02
- NNNNNNNN: 学籍番号（ハイフンを除いた8桁）
- 今回はSOFT-12-06-NNNNNNNNNN となる

■本文

- 冒頭に学籍番号、氏名を明記

■発展課題について

- 必ずしも毎回提出しなくてもかまわない
- 成績の計算には全6回のうち上位3回分を採用する

課題提出の注意

- 提出するプログラムをtar.gzでかためること
 - 展開後のディレクトリ名がSOFT-12-06-NNNNNNNNNN（学籍番号）となるようにする（下記参照）

```
% mkdir SOFT-12-06-NNNNNNNNNN
```

```
% cp struct_alignment.c gravity*.c SOFT-12-06-NNNNNNNNNN
```

```
% tar cvzf SOFT-12-06-NNNNNNNNNN.tar.gz SOFT-12-06-NNNNNNNNNN
```

- 追加のデータファイル等ある場合はそれも同梱せよ