# 情報構造 2023
# 第 1 回

ガイダンス

# 自己紹介

- 大村英史（おおむらひでふみ）
- 理工学部 情報計算科学科 講師
- 専門分野：音楽情報科学，人工知能，認知科学

# 今日の予定

- ガイダンス
  - 授業の形態
  - 成績
  - 授業で扱う内容
  - 大学の計算機環境

# 授業の形態

- 授業方法
  - 対面（K702）
  - もし何かがあったらzoomやほかの方法をとるかも
- 全14回（4/17 - 7/17）
- 授業 + 演習
  - 授業後半に，計算やC言語の実装をしたりします
  - ノートパソコンなどがあるとよいですが，持ち帰って自宅で行ってもかまいません
  - C言語は，簡単なフォローのみしかしません（すでに習得済みとみなします）

# 成績評価方法

- 試験
- 演習問題をあつめるかも
- 試験と演習問題から総合的に判断します

- 出欠はとらない（予定）

# 授業で扱う内容

# 情報構造の授業内容予定

- ガイダンス（1）
- データ型，抽象データ型（2, 3）
- アルゴリズム（4）
- ソーティング（5）
- リスト，スタック，キュー（6, 7）
- 木，順序木，二分木（ 8, 9, 10, 11）
- 集合，辞書（12, 13）
- 二分探索木，AVL木，B-木（14, 15）

# 授業の目的（シラバス）

- コンピュータにおける情報（データ）の扱い方を習得する
  - データ構造の概念
  - アルゴリズム

- データ構造の概念を学ぶことにより，各コンピュータ分野での洗練された効率のよいアルゴリズム記述する能力を得る。

# データとは

- データ【data】（広辞苑 第五版）
  1. 立論・計算の基礎となる、既知のあるいは認容された事実・数値。資料。与件。「実験―」
  2. コンピューターで処理する情報。
- データ - しょり【―処理】（広辞苑 第五版）
  - 必要な情報を得るためにデータに対して行う一連の作業。例えばコンピューターによって、大量の資料について集計・分類・照合・翻訳などの**算術的・論理的処理**を行うこと。
- データの語源
  - 「与える」意のラテン語ダーレ（dare）の受身形からでたもの。

  => 「**情報を与える・共有する**」という意図も含んでいる

# アルゴリズムとは

- アルゴリズム【algorithm; algorism】（広辞苑 第五版）
  - アラビア記数法。
  - **問題を解決する**定型的な手法・技法。コンピューターなどで、演算手続きを指示する規則。算法。
  - アラビアの数学者アル=フワリズミーの名に因む
- アルゴリズム(algorism)〔数学〕（現代用語の基礎知識）
  - あるタイプの一般的な問題が与えられたとき、その**解を求める計算の手順**のこと。
  - もっとも古いものは、紀元前三世紀の「ユークリッドの互除法」。
  - 二つの数の最大公約数を求める方法
  - 順にたがいを割っていき最後の商を最大公約数とする。
  - アルゴリズムの語源は、代数の国アラビアの第一の数学者アル・コワリズミ（九世紀）の名がなまったものといわれている。コンピュータ時代になり、その命令（算法）手順として用いられるようになった。
  - アルゴリズムを箇条書きの形で書いたものは複雑なうえ、操作の流れがわかりにくいためこれを図で示すことが多い。この図を「流れ図」という。

# アルゴリズムとは

- さんぽう【算法】（岩波国語事典）
  - 計算の方法。(ア) 算術。(イ) 演算手続きを指示する規則。
  - 特に、同類の問題一般に対し、有限回の基本的操作を、指示の順を追って実行すれば、解がある場合にはその解が得られ、解がない場合にはそのことが確かめられるように、はっきりと仕組んである手順。
  - <イ>は数値計算や数学の問題には限らない。algorithmの訳語
- アルゴリズム（マイペディア―小百科事典）
  - アラビア数字を用いる筆算法を意味する英語で，アラビアの数学者アル・フワリズミが語源。
  - 今日の数学では，問題を解くための演算の手順を，足す，引く，掛ける，割るという四則演算の単位操作の指示の系列で表わしたものをいう。
  - 2数の大小の比較，位置の入れ替え，指示から指示への移行などの補助的操作も含む。⇒プログラミング

# プログラミングとは

- プログラミング（マイペディア―小百科事典）
  - コンピュータ用語。機械に行なわせる作業の指示を順序だてて細かく記述したものをプログラムと呼び，これを作ることをいう。
  - **アルゴリズムを基礎とする**。できたプログラムはコード化されて記憶装置に送りこまれる。⇒自動プログラミング
- 自動プログラミング（マイペディア―小百科事典）
  - コンピュータのプログラミングを**容易かつ能率的**にするための技法。プログラミングを直接機械に入力可能な機械語で行なわず，一般の人の使用しやすい形の**プログラミング用言語**を用いて行なうこと。現在は人間のやりたいことをモニターと対話し，メニューの選択などの指示により，その内容をコンピュータが理解し，プログラミングを行なってくれることまでを意味する。人間とコンピュータのこのようなインターフェイスを実現するソフトウェアの開発は，システムエンジニア，プログラマーの不足に対応し，コンピュータ普及のキーとなっている。

# データ構造とアルゴリズム

- データ構造
  - コンピュータでデータを扱いやすくするためのデータの関係規則
- アルゴリズム
  - 計算問題を解くための実行可能な手続き
- プログラム
  - アルゴリズムに基づいた，コンピュータでの処理手順

=> 効率のよいデータ構造やアルゴリズム記述
  - 計算時間が短い，メモリーの使用量が少ない

# 最大公約数（greatest common divisor）のアルゴリズムを考えてみよう

**2つの正の整数の最大公約数を求めよ.**

一般的な求め方

と

Euclid互除法

# 最大公約数を求める

GCD: greatest common divisor

**2つの正の整数の最大公約数を求めよ.**

1. 問題を理解する
   - 最大公約数: 2つの数を共に割り切る整数のうち、最大の数
   - 例: $GCD(9,6) = 3$
2. アルゴリズムのアイデアを得る
   - 2つの数の約数の集合を求めればよいのでは?
   - 約数の集合が求められれば、その共通集合も計算できる
   - 共通集合が求められれば、その最大値も計算できる

=> 公約数の定義に基づいた方法

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm: GCD**
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数

1. $x$の約数の集合を計算する
2. $y$の約数の集合を計算する
3. $x$の約数の集合と$y$の約数の集合の共通集合を計算する
4. 共通集合の最大値を計算する

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 **GCD**
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd_{x,y}}$
1. $X_d \leftarrow \mathrm{Divisor}(x)$ $\{x$の約数の集合$\}$
2. $Y_d \leftarrow \mathrm{Divisor}(y)$ $\{y$の約数の集合$\}$
3. $Z \leftarrow X_d \cap Y_d$
4. $\mathrm{gcd_{x,y}} \leftarrow \max(Z)$

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 GCD
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd}_{x,y}$
1. $X_d \leftarrow \mathrm{Divisor}(x)$ $\{x$の約数の集合$\}$
2. $Y_d \leftarrow \mathrm{Divisor}(y)$ $\{y$の約数の集合$\}$
3. $Z \leftarrow X_d \cap Y_d$
4. $\mathrm{gcd}_{x,y} \leftarrow \max(Z)$

Divisorは$x$の約数の集合を計算する関数（サブルーチン）
- 関数の計算は未定義

{}はコメント

maxは集合Zの最大値を求める関数
- 擬似コードは簡単に書けるので省略

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 **GCD**
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd_{x,y}}$
1. $X_d \leftarrow \mathrm{Divisor}(x)$ $\{x$の約数の集合$\}$
2. $Y_d \leftarrow \mathrm{Divisor}(y)$ $\{y$の約数の集合$\}$
3. $Z \leftarrow X_d \cap Y_d$
4. $\mathrm{gcd_{x,y}} \leftarrow \max(Z)$

$x = 9, y = 6$の場合
1行目: $X_d = \{1, 3, 9\}$
2行目: $Y_d = \{1, 2, 3, 6\}$
3行目: $\mathrm{Z} = X_d \cap Y_d = \{1, 3\}$
4行目: $\mathrm{gcd}_{x,y} = \max(\{1, 3\}) = \mathbf{3}$

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 約数の計算 **Divisor**
**Input:** 整数 $x\ (> 0)$
**Output:** $x$の約数の集合 $D$

1. $D \leftarrow \emptyset$
2. **for** $i \leftarrow 1$ **to** $x$ **do**
3. **if** $x \bmod i = 0$ **then**
4. $D \leftarrow D \cup \{i\}$
5. **end if**
6. **end for**

- １から$x$までの各整数が$x$で割り切れるかチェックする

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 約数の計算 **Divisor**
**Input:** 整数 $x\ (> 0)$
**Output:** $x$の約数の集合 $D$
1. $D \leftarrow \emptyset$
2. **for** $i \leftarrow 1$ **to** $x$ **do**
3. **if** $x \bmod i = 0$ **then**
4. $D \leftarrow D \cup \{i\}$
5. **end if**
6. **end for**

1 から$x$までの各整数が$x$で割り切れるかチェックする

$\emptyset$は空集合

for文: $i$は1から$x$までの値をとる

$x \bmod i$ は$x$を$i$で割ったときの剰余を計算する
→ 剰余が0ならば、 $i$は$x$の約数であるので、$D$に追加する

# 公約数の定義に基づいた方法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 約数の計算 **Divisor**
**Input:** 整数 $x\ (>0)$
**Output:** $x$の約数の集合 $D$

1. $D \leftarrow \emptyset$
2. **for** $i \leftarrow 1$ **to** $x$ **do**
3. **if** $x \bmod i = 0$ **then**
4. $D \leftarrow D \cup \{i\}$
5. **end if**
6. **end for**

$x = 6$の場合
1行目: $D = \emptyset$
2から5行目:

- $i = 1 : 6 \bmod 1 = 0$ なので$D = \{1\}$
- $i = 2 : 6 \bmod 2 = 0$ なので$D = \{1, 2\}$
- $i = 3 : 6 \bmod 3 = 0$ なので$D = \{1, 2, 3\}$
- $i = 4 : 6 \bmod 4 = 2$ なので$D = \{1, 2, 3\}$
- $i = 5 : 6 \bmod 5 = 1$ なので$D = \{1, 2, 3\}$
- $i = 6 : 6 \bmod 6 = 0$ なので$D = \{1, 2, 3, 6\}$

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

- 最大公約数を計算するアルゴリズム
  - アレクサンドリアのエウクレイデス(紀元前3 世紀ごろ?)が著したとされる数学書「原論（Στοιχεία）」第7巻に載っている**世界最古のアルゴリズム**！
- 巧妙かつ高速で、現代でも通用する方法

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

- 次の事実を利用する

任意の非負整数$x$と正の整数$y$に対し、$x$と$y$の最大公約数は $y$ と$x \bmod y$の最大公約数と等しい

$$GCD(x, y) = GCD(y, x \bmod y)$$

- $GCD(9, 6) = GCD(6, 9 \bmod 6) = GCD(6, 3)$
- $GCD(6, 3) = 3$
- つまり、 $GCD(9, 6) = 3$

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

- $GCD(9,6)$
- $\rightarrow GCD(6, 9 \bmod 6) = GCD(6,3)$
- $\rightarrow 6 \bmod 3 \ = 0$ なので、$GCD(9,3) = 3$

- $x$と$y$の値を変更しながら、$GCD$を繰り返し呼び出すことで最大公約数の計算が可能

→ **再帰**を使う!

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

- $GCD(1071, 1029)$
- $\rightarrow GCD(1029, 1071 \bmod 1029) = GCD(1029, 42)$
- $\rightarrow GCD(42, 1029 \bmod 42) = GCD(42, 21)$
- $\rightarrow 42 \bmod 21 = 0$ なので、$GCD(42, 21) = 21$

- $x$と$y$の値を変更しながら、$GCD$を繰り返し呼び出すことで最大公約数の計算が可能

→ **再帰**を使う!

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

- $GCD(12, 5)$
- $\rightarrow GCD(5, 12 \bmod 5) = GCD(5, 2)$
- $\rightarrow GCD(2, 5 \bmod 2) = GCD(2, 1)$
- $\rightarrow 2 \bmod 1 = 0$ なので、$GCD(2, 1) = 1$

- $x$と$y$の値を変更しながら、$GCD$を繰り返し呼び出すことで最大公約数の計算が可能

→ **再帰**を使う!

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 Euclid
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd}_{x,y}$

1. **if** $y = 0$ **then**
2. $\quad \mathrm{gcd}_{x,y} \leftarrow x$
3. **else**
4. $\quad \mathrm{gcd}_{x,y} \leftarrow \mathrm{Euclid}(y, x \bmod y)$
5. **end if**

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 Euclid
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd}_{x,y}$

1. **if** $y = 0$ **then**
2. $\quad \mathrm{gcd}_{x,y} \leftarrow x$
3. **else**
4. $\quad \mathrm{gcd}_{x,y} \leftarrow \mathrm{Euclid}(y, x \bmod y)$
5. **end if**

再帰呼び出しの停止条件
- $GCD(6,3)$の計算と同値

再帰呼び出し

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 Euclid
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\mathrm{gcd}_{x,y}$

1. **if** $y = 0$ **then**
2. $\quad \mathrm{gcd}_{x,y} \leftarrow x$
3. **else**
4. $\quad \mathrm{gcd}_{x,y} \leftarrow \mathrm{Euclid}(y, x \bmod y)$
5. **end if**

- $\mathrm{Euclid}(9, 6)$
- $\rightarrow \mathrm{Euclid}(6, 9 \bmod 6) = \mathrm{Euclid}(6, 3)$
- $\rightarrow \mathrm{Euclid}(3, 6 \bmod 3) = \mathrm{Euclid}(3, 0)$
- $\rightarrow \mathrm{Euclid}(3, 0) = 3$

# Euclidの互除法

**2つの正の整数の最大公約数を求めよ.**

**Algorithm:** 最大公約数の計算 Euclid
**Input:** 整数$x, y$ $(0 < x, y)$
**Output:** $x, y$ の最大公約数 $\gcd_{x,y}$

1. **if** $y = 0$ **then**
2. $\quad \gcd_{x,y} \leftarrow x$
3. **else**
4. $\quad \gcd_{x,y} \leftarrow \mathrm{Euclid}(y, x \bmod y)$
5. **end if**

- $\mathrm{Euclid}(134, 24)$
- $\rightarrow \mathrm{Euclid}(24, 134 \bmod 24) = \mathrm{Euclid}(24, 14)$
- $\rightarrow \mathrm{Euclid}(14, 24 \bmod 14) = \mathrm{Euclid}(14, 10)$
- $\rightarrow \mathrm{Euclid}(10, 14 \bmod 10) = \mathrm{Euclid}(10, 4)$
- $\rightarrow \mathrm{Euclid}(4, 10 \bmod 4) = \mathrm{Euclid}(4, 2)$
- $\rightarrow \mathrm{Euclid}(2, 0) = 2$

# ユークリッド互除法でわかること

- アルゴリズムの良し悪しによって，解に至るまでの難易度が変わる

# プログラミング言語の歴史

- 1940年代：機械語，アセンブラ

---

- 1950-60年代：FORTRAN, LISP, ALGOL（Pascalの祖）
- 1970年代：C言語，Pascal

---

- 1980年代：C++, Objective C, Small Talk
- 1990年代：Java，JavaScript, Python


# データ型

- 機械語のデータ型
  - レジスタ値とその番地
- 原子データ型
  - char, int, long short…
- 構造データ型
  - 配列，構造体
- 基本的データ構造
  - リスト，スタック，キュー
- 先進的データ構造
  - 木，ハッシュ

機械語
型なし

昔の言語
C, Pascal

最近の言語
C++, Java…

# データ構造の違い

- データの型によって，性能が変わる

- 例：配列を使わずに数列を扱えますか？
- 例：配列よりリストの方が便利？

# 大学の計算機環境（C言語）を使うために

- VPN (Virtual Private Network)
- ssh (Secure Shell)
  - ssh linux.ed.tus.ac.jp -l ユーザ名
  - 鍵がコロコロ変わるので鍵の消し方
  - ssh-keygen -R  linux.ed.tus.ac.jp
- テキストエディタ: emacs/vi (vim)
- ~~TUSストレージ~~
- SCPコマンド
  - https://qiita.com/chihiro/items/142ebe6980a498b5d4a7

# VPN

- 大学のvpnに接続すると，仮想的に大学の環境から計算機を操作していることになります．これにより，大学内に限定されたwebページにアクセスできるようになります．
  多くの授業ページは学内からのアクセスに限定されているため，VPN接続が必要になります．
- 大学のVPNについては，教育環境のコンピュータ利用案内のマニュアルに説明がありますが．学外からは見られません．
  ※ 学外から参照する場合はCLASSにログイン後「コンピュータ設備情報」の「VPN接続の利用方法」を参照してください．
  または以下で簡易的に説明します．
- https://www.ed.tus.ac.jp/

# VPN クライアントソフトウェア

- vpn接続のために，以下のURLにアクセス・教育アカウントでログインし，自分の環境（自宅のOS）にあったソフトウェアをダウンロードして下さい.
  https://tusuas1.tus.ac.jp/download/vpn
- インストールは，各OSの利用方法に従ってインストールをして下さい.
- 接続先サーバーは「tusvpn1.tus.ac.jp」です.
  教育環境で利用しているメールアドレス (ユーザ名@ed.tus.ac.jp) とパスワードを入力してください

# ssh (secure shell)

- プログラミングだけならばCUIのみで十分なのでSSHを紹介します.
- 大学の環境に自宅からssh接続します． sshとは，セキュア シェル（Secure Shell）の略で， ネットワークに接続された計算機に接続し操作することです． ssh接続によって，大学の教育環境を自宅で操作することができるようになります.
- 最新のwindows環境からは，コマンドプロンプトやPowerShellからsshと入力すれば，OpenSSHを利用してssh接続できます.
- 大学のwindows環境にもインストールされているTera Termは文字化けなど回避できます.
  - https://ja.osdn.net/projects/ttssh2/releases/
  - TeraTermを起動して，ホストを「tusedls00.ed.tus.ac.jp」と打ち込んで「OK」ボタンを押します.
  - ssh認証の画面が表示されるので，ユーザ名とパスフレーズを入力して「OK」ボタンを押すと大学の環境に接続されます.
- コマンドプロンプトやPowerShellの場合は，「ssh ユーザ名@linux.ed.tus.ac.jp」と打ち込んでエンターを押すとパスワードが求められます．パスワードを入力して（表示されない）エンターキーを押すと接続されます.
- MacやLinuxを使っている人は，ターミナルから「ssh ユーザ名@linux.ed.tus.ac.jp」と入力すれば，接続することができます.
- 【注意】sshで大学の環境に接続する場合は，かならずvpn接続する必要があります.

# Emacs/vi (vim)

- ターミナル環境でつかえるテキストエディタにEmacsやvi(vim)があります.
- どちらのエディタも操作やショートカットが特殊ですが慣れると便利です.
- これらを使ってプログラミングして下さい.

- nanoというエディタは下にショートカットが表示されていて分かりやすいです.

# TUSストレージ

- TUSストレージを利用すると，大学環境のファイルにアクセスできます．
  - https://tusols.ed.tus.ac.jp/

- 作成したコードを手元のコンピュータに持ってこれます．

# C言語のコンパイラー：gcc

- コードを作成したらコンパイルをします
- gcc（GNU c compiler）をつかいます
- 作成したコードファイルをabc.cとすると,

  gcc abc.c
- a.outという名前の実行ファイルが作成されます．以下で実行できます．

  ./a.out
- 実行ファイルに名前（ここではtest）をつけたいときは
  - gcc -o test abc.o

# まとめ

- ガイダンス
  - 授業で扱う内容
  - 授業形態
  - 成績
  - 大学の計算機環境