# 環境科学基礎プログラミング

化学生物環境学科・環境科学コース
高須夫悟　たかすふうご
takasu@es.nara-wu.ac.jp

- **C言語の基礎の習得**
- **環境科学基礎プログラミングは講義形式（G302）**
- **対応する環境科学基礎プログラミングは G401 で行う**
- **成績は、数回のレポート+学期末試験で決定**





# 参考書

・新・明解C言語　入門編　柴田望洋著
　– SB クリエイティブ　2,300円
・例題で学ぶ　はじめてのC言語　大石弥幸/朝倉宏一共著
　– ムイスリ出版 2,100円

特に参考書は指定しませんが、最低1冊は手元にあると効率よく学習できると思います

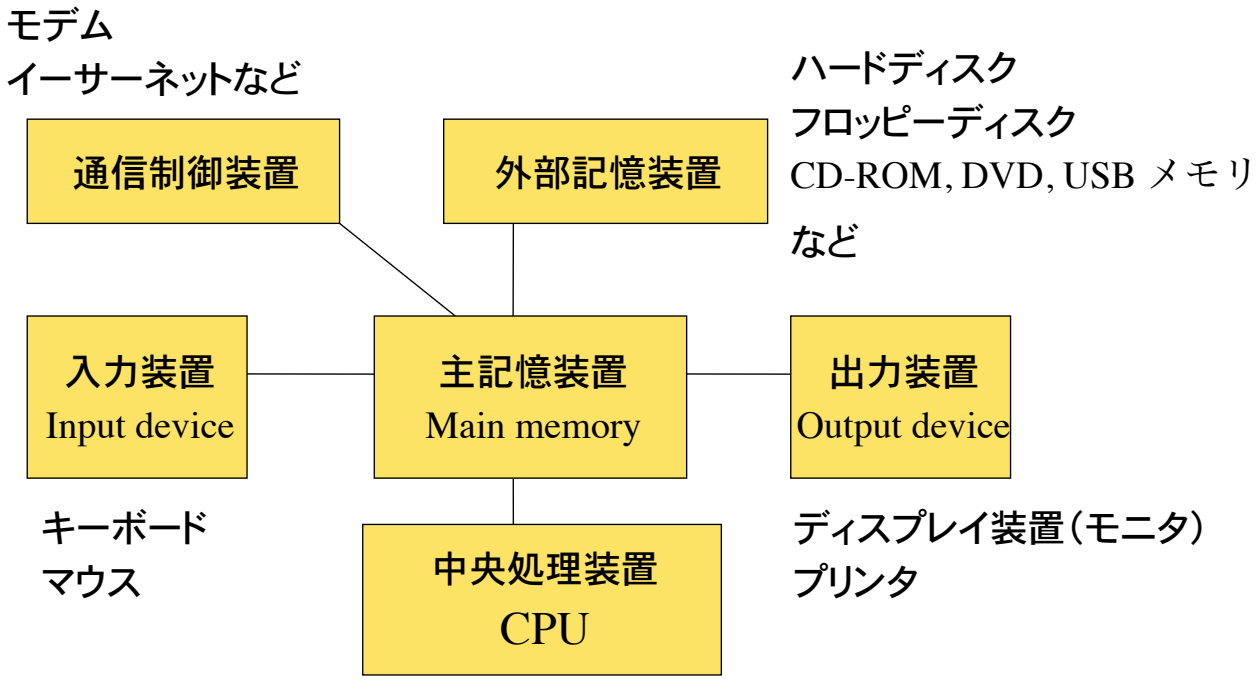
# 自習のための環境整備

- 手元にPCがある場合、C言語プログラミング環境を設定することを勧めます。下記のいずれも無料です
- macOSの場合 —> App Store から Xcodeをインストール
- Windowsの場合 —> Cygwin をインストール、もしくは、Windows Subsystem for Linux をインストール
- cygwinとWSLのインストール方法についてはネットの情報参照





# コンピュータ（計算機）の構成



# コンピュータの種類の概観

**大型計算機**

奈良女子大情報処理センター

**ワークステーション**

G401 旧計算機システム

**パーソナルコンピュータ**

G401 計算機システム

**スマートフォン**

大規模、複数ユーザの同時使用

小規模、個人ユーザ使用

近年の技術発展により、上記の区別はあまり明確ではなくなってきている。





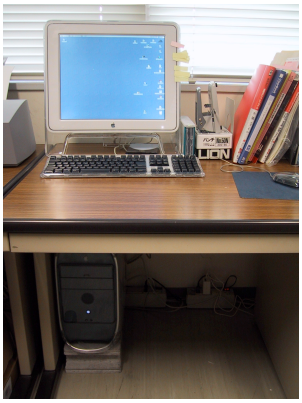
# ワークステーション（1990年代）

DEC 社製 AlphaStation

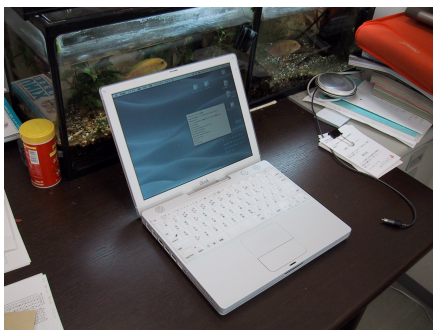
HIT 社製 SS

パソコン

Apple 社製
PowerMac



Apple 社製 iBook



# コンピュータとネットワーク

最近のコンピュータは、電話回線、CATV、光ファイバー、Ethernet などを経由して互いに接続可能（ネットワークの形成）。

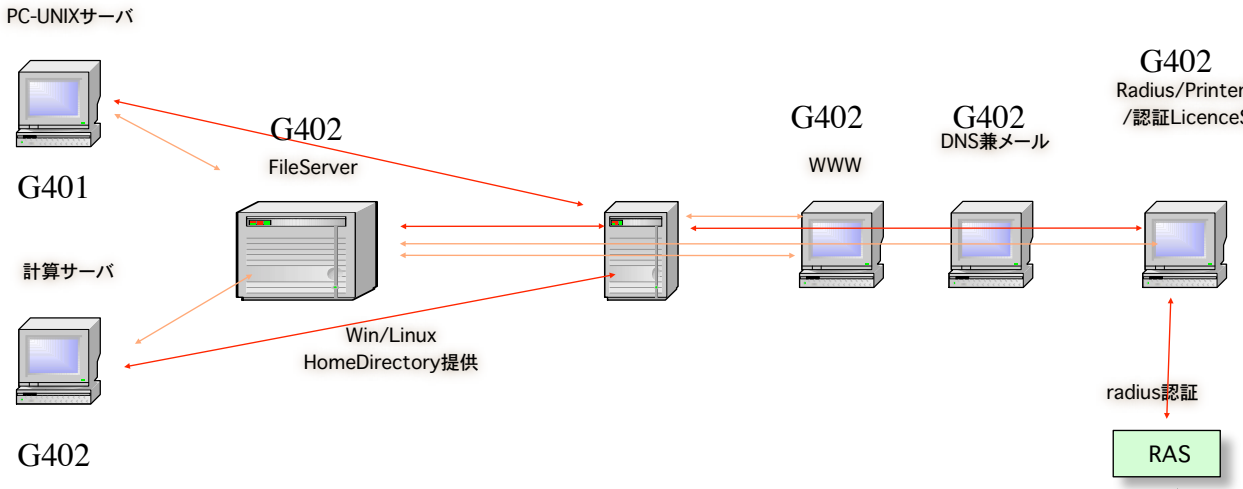
ローカルネットワーク（LAN）が多数接続することで広域ネットワーク（WAN）が形成される。インターネット（Internet）は WAN の1つ。

Internet を介して、電子メール、ウェブ情報などの膨大なデータの転送・共有が可能になる。





# 奈良女子大学様システム NW構成概案




ファイルサーバ：ユーザのファイルを管理収納する為の専用マシン

メールサーバ：メールの送受信をする為の専用マシン

ウェブサーバ：ウェブコンテンツの配信を行う為の専用マシン

理学部G棟計算機システムは百数十台の計算機から構成される

# 奈良女子大学・計算機システム

## メールサーバ、ウェブサーバ、計算サーバ、など






# ハードウェアとソフトウェア

計算機の物理的な装置をハードウェア（hardware）という

Apple 社製 MacBook Pro などはハードウェアの固有名詞

ハードウェアの動作を利用して実際の処理を行うプログラムのことを
ソフトウェア（software）という

マイクロソフト社製の Word, Excel などのアプリケーションソフト

ソフトウェアの中でも、アプリの実行を手助けするものとして基本ソフトウェアがある。基本ソフトウェアはオペレーションシステム（Operation System）とも呼ばれる。

Windows 8, 10 , macOS, Linux, iOS, Android など

# OS の種類

ワークステーションでは UNIX という OS が一般的

UNIX では複数ユーザの利用を前提としており、個人ファイルの保護、プログラム実行時のプロセス管理がしっかりしている。

UNIX にもいくつかの種類（系列）がある。Linux はその1つ。

パソコンの OS としては

Windows 10, macOS などがある。

スマートフォン（ほとんどパソコン）のOSとしては、

Android, iOS などがある。





# プログラムとは？

プログラム＝コンピュータへの指令書

コンピュータに、こういう手順に従って、こういう動作をせよ、という指令書がプログラムである。

1つの結果を得るにも、一般に幾通りもの方法がある。
プログラムで用いる方法（計算手順）のことをアルゴリズムという。

アルゴリズムをコンピュータで実行可能な命令の集合として記述することをプログラミングという。命令の列をプログラムという。

# プログラミング言語

コンピュータが直接理解できる命令（instruction）を機械語（machine language）という。

機械語は 0 と 1 の列、すなわちビット列 (bit)、で表現され、人間には取り扱いが非常に困難。

ビット列：2 進法で表現される 0 と 1 の列。

4 ビット列の例：

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, .... 1111

2 の 4 乗＝ 16 通りの表現が可能





# 機械語の表現

通常、ビット列は 8 ビットを単位として 16 進法で表す。8 ビットの単位を 1 バイト (byte) と呼ぶ。

1バイトは、00000000, 00000001, ... 11111111 までの 256 通りを表現
2^8 = 256

機械語は通常、0, 1, 2, ..., 9, A, B, C, D, E, F の 16 個の記号を用いた 16 進数で表現する。

```
00, 01, 02, 03, ..., 09, 0A, 0B, 0C, 0D, 0E, 0F
10, 11, 12, 13, ..., 19, 1A, 1B, 1C, 1D, 1E, 1F

F0, F1, F2, F3, ..., F9, FA, FB, FC, FD, FE, FF
```

# 高級言語

機械語は人間にとって取り扱いにくい。そこで、人間が理解しやすいプログラミング言語として高水準（高級）言語が考案されている。

FORTRAN, BASIC, COBOL, Pascal, C, C++, Java など。

FORTRAN：古くから科学技術計算の分野で使用

COBOL：事務処理系で広く使われている

BASIC：入門用プログラミング言語

Pascal：教育用言語として開発

C：1972年代、アメリカベル研究所で開発

Java：インターネット時代の仮想計算機上の言語

Python：汎用プログラミング言語





# 実例

```
{ '17 4/11 }
{Pascal 言語}

program Example1;

begin
  writeln('Hello!');
end.
```

```
/*  '22 4/13 C 言語 */
#include <stdio.h>

int main()
 {
  printf("Hello!\n");
  return 0;
 }
```

```
C '17 4/11
C FORTRAN 言語
WRITE(*,*) 'Hello!'
END
```

```
# python 言語
print( “Hello!” )
```

画面（標準出力）に Hello! と表示するプログラムの例

# コンパイル

高級言語で書かれたプログラムを実行するためには、プログラムをコンピュータが理解できる機械語に変換（翻訳：コンパイル compile）する必要がある。

この変換を行うプログラムのことをコンパイラ（compiler）と呼ぶ。

高級言語で書かれたプログラムをソースプログラム（source）、
コンパイラが変換して出力したものをオブジェクトプログラム（object）と呼ぶ。

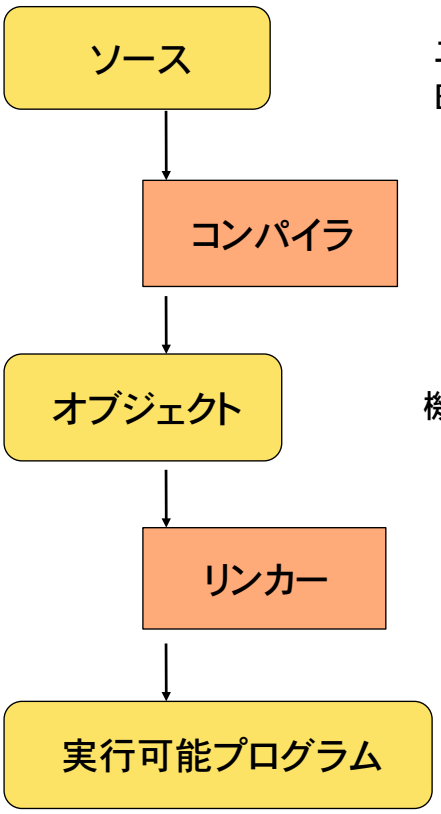
C 言語で書かれたプログラムをコンパイルするプログラムのことを、
C コンパイラと呼ぶ。

FORTRAN のプログラムには FORTRAN コンパイラ、
Pascal のプログラムには Pascal コンパイラ、、、





# プログラム実行までの流れ




エディタでソースプログラムを作成
Emacs や vi などの好みのエディタを使う

コンパイラにソースを入力してコンパイル

機械語に翻訳されたオブジェクトが生成される

標準ライブラリから必要なライブラリをオブジェクトに追加し、実行可能なプログラムを仕立てる作業をリンク（Link）という。
リンクを行うプログラムをリンカー（Linker）と呼ぶ。

# 言語処理系

コンパイルとリンクは通常 1 つの過程と見なすことが出来る。通常はコンパイルとリンクを合わせて、単に「コンパイルする」という。

コンパイラとリンカーを合わせて**言語処理系**と呼ぶ。

FORTRANでプログラムを書いて実行する為には、FORTRAN 言語処理系が、C 言語のプログラムをするには C 言語処理系が必要になる。

UNIX 系の OS には、たいていの場合 C 言語処理系が含まれている。
Xcode (macOS)

パソコン系の OS では、C 言語処理系などのアプリケーションを別途インストールする必要がある。Visual Studio (Windows)





# プログラミング言語まとめ

コンピュータが直接理解可能
（人間には分かりにくい）

**機械語**

言語処理系（コンパイラ）が
両者の仲立ちをする



プログラムし易い
（コンピュータは直接実行不可能）

**高級言語**：FORTRAN, C など

# C プログラミングの一般手順

1)ソースファイルの作成(編集)

エディターを用いてソースプログラムを書く。
ソースプログラムをソースファイルとして保存。

C 言語処理系では、ソースファイル名の最後に拡張子 .c をつける。
例)my_first_program.c など。ファイル名に全角文字は使わない!

2)ソースプログラムのコンパイル

コンパイル途中でソースプログラムに文法の誤りがあれば、コンパイルエラーとなり、コンパイルは中断。

ソースプログラムの欠陥や過ちをバグ bug という。バグを見つけて修正する作業をデバッグ debug という。

バグが無くなるまでデバックを続ける。

文法の過ちによるバグとアルゴリズムのバグは別物であることに注意。





# 初めてのC言語

1)ソースプログラムの作成

エディタでプログラムを書き、sample.c というファイル名で保存

```
/* はじめてのプログラム */
#include <stdio.h>
int main(void)
{
  printf("Hello!\n");
  return 0;
}
```

/* と */ で囲まれた部分はコメント文になる。コンパイラに無視される。注釈文とも言う。

この部分は、C 言語のプログラムの(とりあえず)決まり切った型。

記号の読み方:

/ スラッシュ, * アスタリスク, # シャープ, \ バックスラッシュ
" ダブルクォーテーション, ; セミコロン, { } 中カッコ

# 初めてのコンパイル

## 2)ソースプログラムのコンパイル

UNIX (Linux)系の OS では、C 処理系は cc コマンド(もしくは gcc)で呼び出す。cc コマンドに続いてコンパイルするソールファイル名を指定。

```
% cc sample.c
%
% ./a.out
```

コンパイルエラーが無ければ、a.out という名前の実行可能プログラムが作成される。

実行可能プログラムを実行するには
./a.out と入力(エンターを忘れずに)。

./ とは今自分が作業しているディレクトリを意味する

もしコンパイルエラーがあればエディタに立ち戻ってソースファイルの修正を行う。
修正したらソースの保存 save を忘れずに!





# C言語のプログラムの構成

C では関数を基本単位としてプログラムを構成する

単純なプログラムは main 関数のみから成る

```
#include <stdio.h>

int main(void)
{
  文1
  文2
  ...
}
```

stdio.h というヘッダファイルをこの場所に挿入する

stdio.h には標準的に使われる関数の定義が書かれている

main 関数内に書かれた文 (statement) は上から下へと順番に実行される

文はセミコロン ; で終わる

# 標準出力

```
#include <stdio.h>

int main(void)
{
  printf("Hello!\n");
  return 0;
}
```

文字の列を文字列という

文字列をプログラム中で表現するには " " で囲む。これを文字列リテラルという

左の場合、"Hello!\n" が文字列リテラル

\n は特殊文字の一つで、改行を表す

文字列を表示するには printf 関数を用いる

printf 関数に文字列リテラルを引き渡すと、文字列が標準出力（画面）に表示される。printf 関数は標準ライブラリ関数の一つ。ヘッダファイル stdio.h で定義されている





# printf を用いた例

```
#include <stdio.h>

int main(void )
{
  printf("Hello!\n");
  printf("How are you?\n");
  printf("I am fine. And you?\n");
  return 0;
}
```

2 つのプログラムを実行すると、まったく同じ結果を得る。

```
#include <stdio.h>

int main(void)
{
  printf("Hello!\nHow are you?\nI am fine. Are you?\n");
  return 0;
}
```

# プログラムの記述形式

C 言語では原則としてソースファイルの自由な位置にプログラムを記述することが出来る（自由形式という）

```
#include <stdio.h> int main(void ){printf("1行に書いてもいいです\n");return 0;}
```

```
#include <stdio.h>
int main(void)
{
printf(
"こんなのもOKよ\n"
);return 0;
}
```

通常、プログラムを見易くするために段付け（インデント）をつける（タブキー）。Emacsが自動的にやってくれる。

```
#include <stdio.h>
int main(void)
{
  printf("読み易くない？\n");
  return 0;
}
```





# 特殊文字について

\n：改行
\'：シングルクォーテーション '
\"：ダブルクォーテーション "
\?：疑問符 ?
\a：警報文字（alert）

---

**問題：**　1）標準出力に自分の氏名を表示するプログラムを作れ。

2）特殊文字を含む文字列リテラルを表示するプログラムを作れ。

3）printf の後のセミコロンが抜けているとどうなるか確認せよ。

4）#include <stdio.h> の行をコメントアウトするとどうなるか？

プログラムの一部分を /*　*/ で囲ってコメント文にすることをコメントアウトという。

# UNIX の基本コマンド 1

| | |
|---|---|
| pwd | 今自分がいるディレクトリの表示 |
| ls | ディレクトリ内容の一覧 |

詳細な使い方は、オンライン・マニュアルを参照。

```
% man ls
```

```
% pwd
```
カレントディレクトリの絶対パスの表示。(Working directory とも言う)

```
% ls
```
カレントディレクトリの一覧

```
% ls -F
```
オプション `-F` を付けると、ディレクトリの後にスラッシュ `/` が、
実行形式のファイルの後にアスタリスク `*` が表示される。

```
% ls -l
```
オプション `-l` を付けるとファイル・ディレクトリの更新日時とサイズを表示。





# 基本コマンド 2

| | |
|---|---|
| cat, more | ファイルの内容の表示 |
| mv | ファイル名の変更・ファイルの移動 |
| rm | ファイルの削除 |
| cc, gcc | C コンパイラ呼び出し |

```
% cat test.c
```
ファイル `test.c` の内容を表示。

```
% more test.c
```
ファイル `test.c` の内容を表示。

```
% mv file.c file_new.c
```
既存のファイル `file.c` を `file_new.c` という名前に変更する。

```
% mv file.c programming
```
既存のファイル `file.c` を既存のディレクトリ `programming` に移動する。
`programming` というディレクトリが存在しない場合はファイル名の変更になる。

```
% rm file.c
```
既存のファイル `file.c` を削除(元に戻りません!)

```
% cc test.c
```
ファイル `test.c` をコンパイル。

# 基本コマンド 3

| | |
|---|---|
| cd | ディレクトリの移動 |
| mkdir | ディレクトリの新規作成 |
| rmdir | ディレクトリの削除 |

```
% cd
```
ホームディレクトリへ移動

```
% cd ..
```
上位ディレクトリへ移動

```
% cd programming
```
ディレクトリ `programming` へ移動。`programming` が存在しないとエラー。

```
% mkdir my_dir
```
ディレクトリ`my_dir`を新規に作成。

```
% rmdir my_dir
```
ディレクトリ`my_dir`を削除。`my_dir`の中にファイルが存在すると削除できない