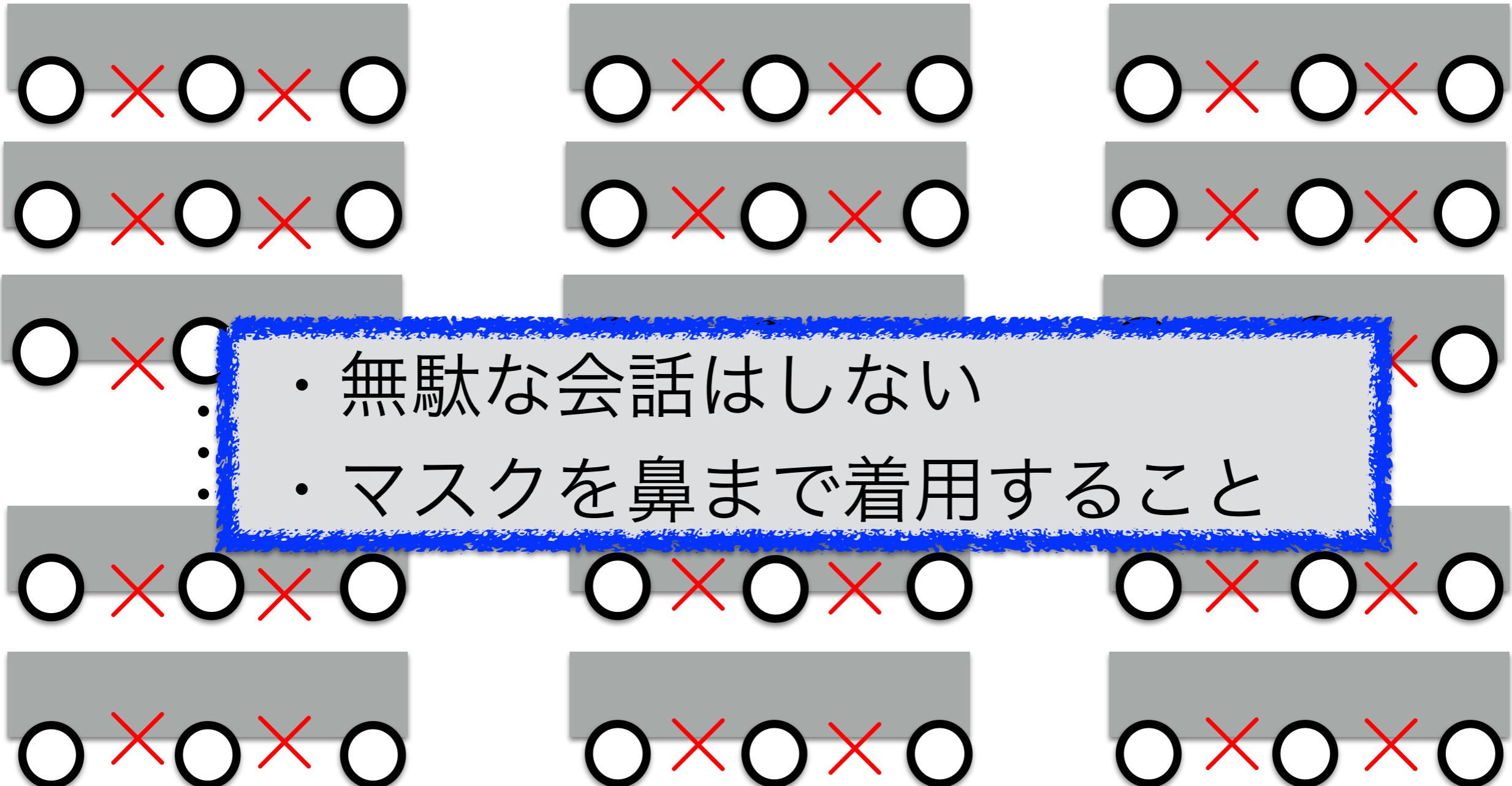


Lecture 2

プログラミング演習Ⅰ その2

教室での座席について

前



後

- ・ 無駄な会話はしない
- ・ マスクを鼻まで着用すること

注意事項

1. 授業のスライドはHPからダウンロードすることができます、毎回の演習問題の回答例などは、自分で打ってください。
2. スライド以外の授業内容もあるので、ノートを取ること。
3. 授業中にスマートフォンやLineなどは使用禁止（演習問題の回答の写真撮影も不可！）
4. 偽造出席や出席名簿改竄（両者）＝ F.

授業のホームページ

<http://amth.mind.meiji.ac.jp/courses/PE1/>

C言語で“Hello World” プログラムを書いてみよう

Cのプログラムは左上から右へと読み、改行があれば次の行の左端からまた右へと読む。つまり、左から右へ、上から下へと読むことになる。

行

```
1 #include <stdio.h>
2
3 int main(void){
4     printf("Hello World!\n");
5     return 0;
6 }
```

C言語で“Hello World” プログラムを書いてみよう

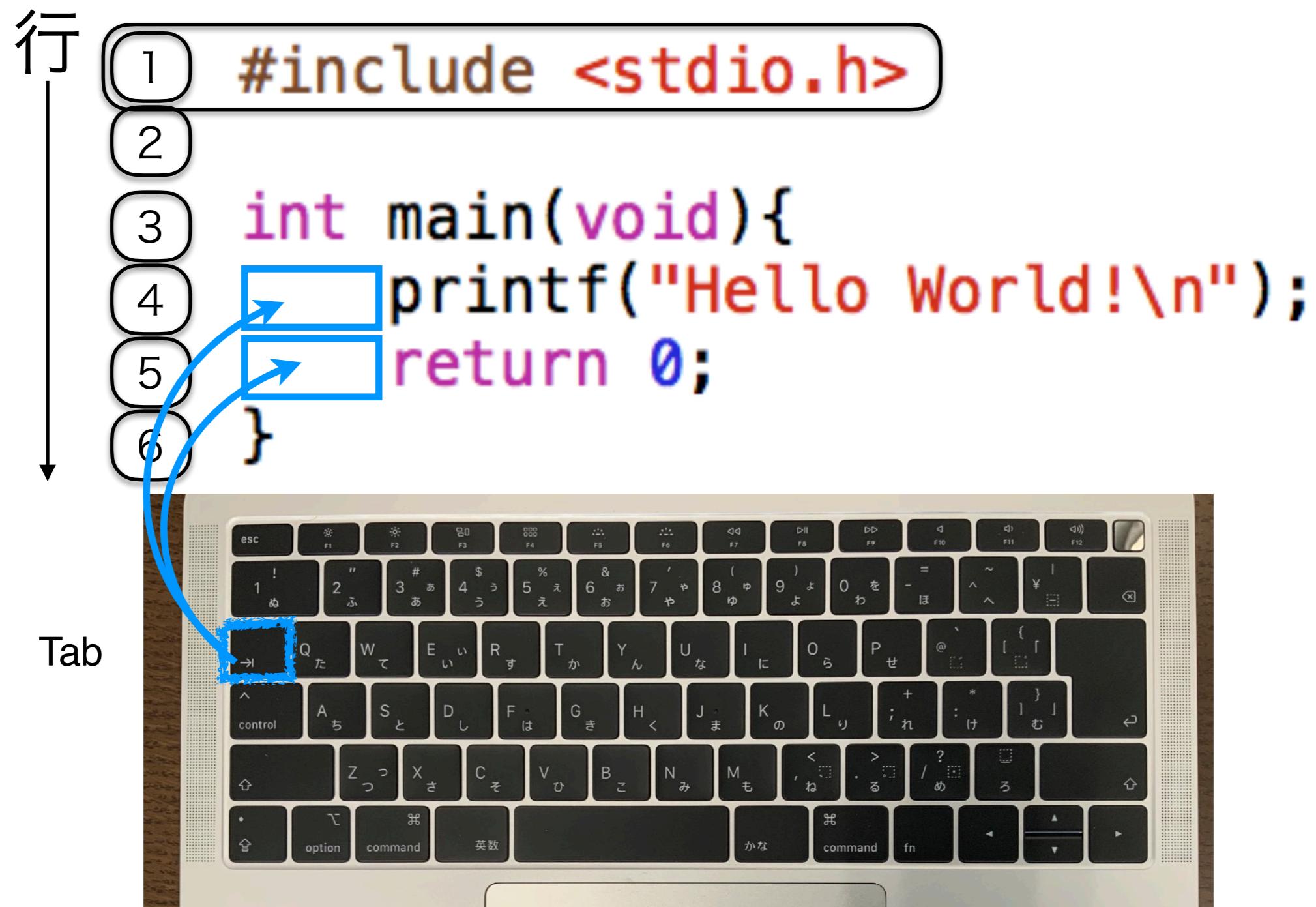
Cのプログラムは左上から右へと読み、改行があれば次の行の左端からまた右へと読む。つまり、左から右へ、上から下へと読むことになる。

行

```
1 #include <stdio.h>
2
3 int main(void){
4     printf("Hello World!\n");
5     return 0;
6 }
```

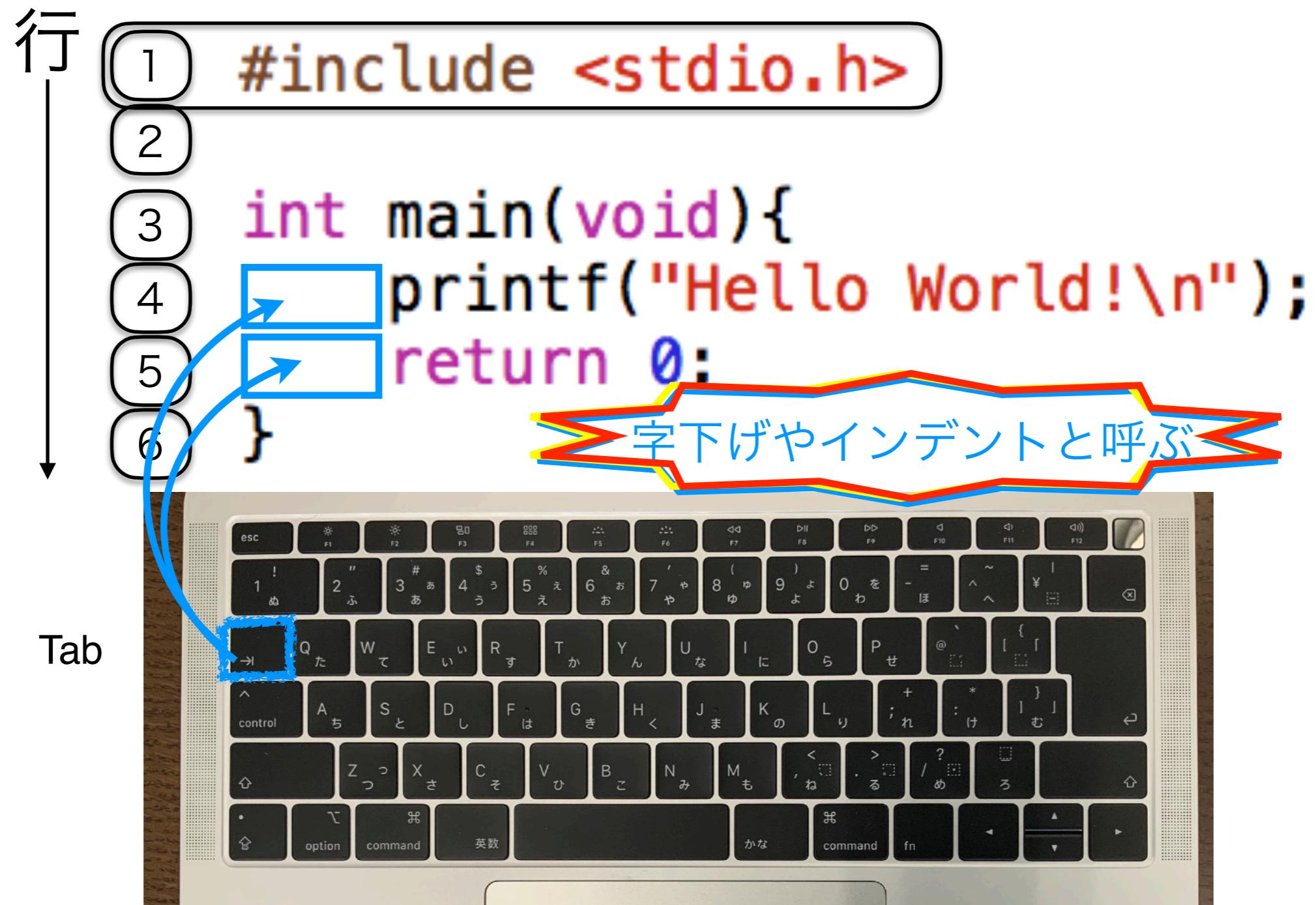
C言語で“Hello World” プログラムを書いてみよう

Cのプログラムは左上から右へと読み、改行があれば次の行の左端からまた右へと読む。つまり、左から右へ、上から下へと読むことになる。



C言語で“Hello World” プログラムを書いてみよう

Cのプログラムは左上から右へと読み、改行があれば次の行の左端からまた右へと読む。つまり、左から右へ、上から下へと読むことになる。



コメントについて

プログラムの理解を補助するために、ソースコード中に記述された注釈文のことをコメント（comment）という。

1a

次のプログラムを一字一句正確に作成してみよう。

```
1 #include <stdio.h>
2 int main(void){
3     printf("Hello world!\n"); // This is a comment
4     // comments are for the programmer
5     printf("Hello again.\n"); // The compiler does not read them
6     return 0;
7 }
```

コメントについて



ここ！

Pythonで「Hello World」

1b

以下のように hello.py を作成して、保存し、「python hello.py」で実行してみよう。

```
1 | print("Hello World") ← hello.py
```

出力

```
[imath@sougousuurino MacBook-Air ~ % python report1.py  
Hello World
```

NOTE Python ではコメントは#を先頭に付けて記述する。

```
1 | print("Hello World") # this is a comment
```

「;」は不要

コメントについて

2a 以下のようなCプログラム作成し、実行せよ。

```
1 #include <stdio.h>
2 int main(void){
3     printf("Hello.\n");
4     printf("My name is Elliott.\n"); // your name
5     printf("I like math.\n\n\n");    // something you like
6     printf("I do not like traffic.\n"); // something you don't like
7     return 0;
8 }
```

コメントについて

2b

以下のプログラムをPythonで作成せよ。

```
1 #include <stdio.h>
2 int main(void){
3     printf("Hello.\n");
4     printf("My name is Elliott.\n");    // your name
5     printf("I like math.\n\n\n");        // something you like
6     printf("I do not like traffic.\n"); // something you don't like
7     return 0;
8 }
```

変数の宣言とデータの型指定

プログラム中で変数を使用するときには、そのことをプログラム中で宣言しなければならない。次のように、変数名とその変数の型 (type) を指定する。

型指定子　変数名1；

例: int a;

複数の変数名を「,」で区切り、同時に宣言できる。

型指定子　変数名1,　変数名2,　・・・,　変数名n；

例: int a,b;

型の名称	型	変換指定文字	表現できる範囲
整数	int	%d	-32768 ~ 32767
倍長整数	long	%li	-21474883648~2147483647
実数	float	%f	$\pm 3.4 \times 10^{-38} \sim \times 3.4 \times 10^{38}$
倍精度実数	double	%lf	$\pm 1.7 \times 10^{-308} \sim \times 1.7 \times 10^{308}$
文字	char	%c	1 文字

変数の宣言とデータの型指定

プログラム中で変数を使用するときには、そのことをプログラム中で宣言しなければならない。次のように、変数名とその変数の型 (type) を指定する。

型指定子　変数名1；

例: int a;

複数の変数名を「,」で区切り、同時に宣言できる。

型指定子　変数名1,　変数名2,　・・・,　変数名n；

例: int a,b;

型の名称	型	変換指定文字	表現できる範囲
整数	int	%d	-32768 ~ 32767
倍長整数	long	%li	-21474883648~2147483647
実数	float	%f	$\pm 3.4 \times 10^{-38} \sim \times 3.4 \times 10^{38}$
倍精度実数	double	%lf	$\pm 1.7 \times 10^{-308} \sim \times 1.7 \times 10^{308}$
文字	char	%c	1 文字

変数の宣言とデータの型指定

3

以下は、整数型変数 `x` を宣言し、その初期値を5に設定するプログラムである。コンパイルして、実行せよ。

```
#include <stdio.h>
```

```
int main(void){  
    int x;  
    x = 5;  
    printf("the value of x is %d", x);  
    return 0;  
}
```

- ＊ 整数型の値を画面に出力したい場合には、`%d`を使用すること。

scanf関数の使い方（ユーザーからの入力について）

入力用の関数としてはscanfやfscanfなどがあります。

scanf キーボードから入力を受ける

fscanf ファイルから入力を受ける

例

```
1 #include <stdio.h>
2 int main(void){
3     int x;
4     printf("What is your favorite number?\n");
5     scanf("%d",&x);
6     printf("Your favorite number is %d\n",x);
7     return 0;
8 }
```

注意：scanfのフォーマット部分（“で囲まれた部分）と実際の入力データの型が一致しないと、出力がおかしくなる可能性があります。

scanf関数の使い方（ユーザーからの入力について）

入力用の関数としてはscanfやfscanfなどがあります。

scanf キーボードから入力を受ける

fscanf ファイルから入力を受ける

例

```
1 #include <stdio.h>
2 int main(void){
3     int x;
4     printf("What is your favorite number?\n");
5     scanf("%d",&x);
6     printf("Your favorite number is %d\n",x);
7     return 0;
8 }
```

注意：scanfのフォーマット部分（“で囲まれた部分）と実際の入力データの型が一致しないと、出力がおかしくなる可能性があります。

scanf関数の使い方（ユーザーからの入力について）

4

次のプログラムは、変数 `x` に対して、キーボードから入力した整数値を読み込み、その整数値を表示するコードである。一字一句正確に作成してみよう。

```
1 #include <stdio.h>
2 int main(void){
3     int x = 0;
4     printf("x=");
5     scanf("%d", &x);
6     printf("x=%d\n", x);
7     return 0;
8 }
```

整数の代わりに、実数 `0.12345` を入力すると、`0.12345` を表示する場合のプログラムを書いてみよう。

ヒント：`int` と `%d` の代わり、`double` と `%lf` を使用する。

複数のデータを入力にういて

入力するデータの数が比較的少ない場合、ひとつのscanf関数でよみとることができます。

- 5 いかのプログラムの出力を確認してください。

```
1 #include <stdio.h>
2 int main(void){
3     int x,y,z;
4     printf("Give me three integers\n");
5     scanf("%d %d %d",&x,&y,&z);
6     printf("Your input: %d %d %d\n",x,y,z);
7     return 0;
8 }
```

計算演算子について

6a

以下は、変数 x, y に対して、キーボードから入力した数値を読み込み、加算の結果を表示するコードである。整数 int 型の計算演算子を全て調べ、それぞれの演算結果を表示するように修正せよ。

```
1 #include <stdio.h>
2
3 int main(void){
4     int x, y;
5     printf("x y = \n");
6     scanf("%d %d", &x, &y);
7     printf("%d + %d=%d\n", x, y, x+y);
8     return 0;
9 }
```

ヒント

加算：+

減算：-

乗算：*

除算：/

余り：%

計算演算子について



ここ！

計算演算子について

6b

Pythonの入力、出力、変数（ラベル）と計算演算子の使い方は以下の通りです。確認しましょう。

```
1 x = input("what is x?\n")
2 y = input("what is y?\n")
3 print("{} + {} = {}".format(x,y,x+y) )
4 print("{} - {} = {}".format(x,y,x-y) )
5 print("{} * {} = {}".format(x,y,x*y) )
6 print("{} / {} = {}".format(x,y,x/y) )
7 print("{} = {} (mod {})".format(x,x%y,y) )
```

what is x?
4
what is y?
3
4 + 3 = 7
4 - 3 = 1
4 * 3 = 12
4 / 3 = 1
4 = 1 (mod 3)

出力

NOTE: pythonでは、代入時に自動的に型を判別している。
`type()`を使うと変数の型を調べることができる。

計算演算子の優先順位について

1つの式に複数の演算記号があるときには、次のように優先順位が決められている。

第1順位

* , / , %

第2順位

+ , -

7

以下のコードを実行し、出力を確認しなさい。

```
1 #include <stdio.h>
2 int main(void){
3     int a;
4     a = 5 + 3 * 4 - 2;
5     printf("%d\n",a);
6     a = (5 + 3) * (4-2);
7     printf("%d\n",a);
8     return 0;
9 }
```

Python の対話型シェル(interactive shell)について

Pythonは電卓として、簡単な計算ができる。特にターミナルで「python」を打てば、Pythonの対話型シェルが使える。式を入力するとPythonが計算を行う。ENTERを押せば、結果がすぐ表示される。試してみよう。



終了には quit() か ctrl-d

2

python

```
Python 2.7.10 (default, Feb 7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

数の足し算と引き算は、加算演算子「+」と減算演算子「-」を使う。次の通りである。

```
>>> 1+3
4
>>> 1+3.5
4.5
>>> -1+2.5
1.5
```

Python の対話型シェル(interactive shell)について

2 (continued)

掛け算には、乗算演算子「*」を使う。

```
[>>> 3*2  
6  
[>>> 3.5*1.5  
5.25
```

割り算には、除算演算子「/」を使う。

```
[>>> 3.0 / 2.0  
1.5  
[>>> 3 / 2  
1  
[>>> 6 / 4  
1
```

Python の対話型シェル(interactive shell)について

2 (continued)

余りだけが必要な場合は、乗余演算子（%, モジュロ, modulo）を使う

```
[>>> 9 % 2  
1
```

累乗（べき乗）の計算は「**」を使う。

```
[>>> 2 ** 2  
4  
[>>> 2 ** 10  
1024  
[>>> 1 ** 10  
1
```

Python の対話型シェル(interactive shell)について

2 (continued)

この例から分かるように、括弧を使って計算演算子を組み合わせて、より複雑な式を扱うことができる。

```
>>> 5 + 5 * 5  
30  
>>> (5+5)*5  
50
```

Python標準ライブラリのmathモジュール (module) で一般的な関数 (`sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`) が入っている。ネイピア数`e`も定義されている。

```
[>>> import math  
[>>> math.e  
2.718281828459045  
>>> math.sin(math.pi/2)  
1.0
```

Python の対話型シェル(interactive shell)について

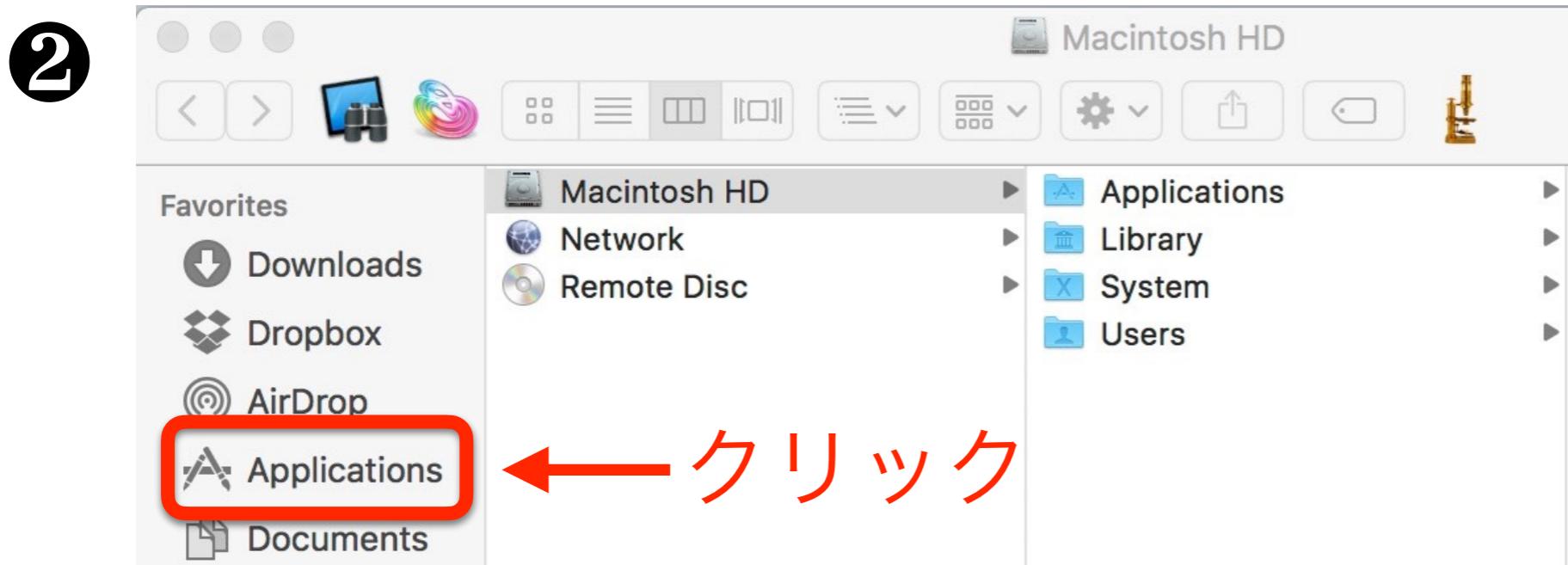
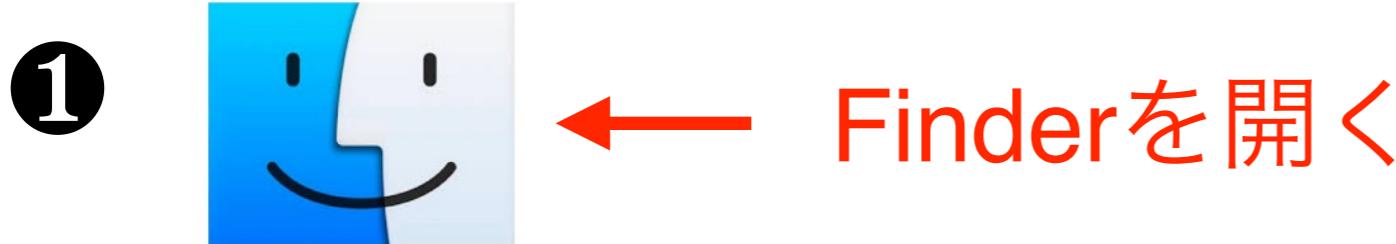
2 (continued)

さらに複雑なPythonプログラムを設計するために、変数（ラベル (label) とも呼ぶ）に数を代入する。

```
>>> a = 3
>>> a + 1
4
>>> a
3
>>> a = a + 5
>>> a
8
>>> math.sin(a)
0.9893582466233818
```

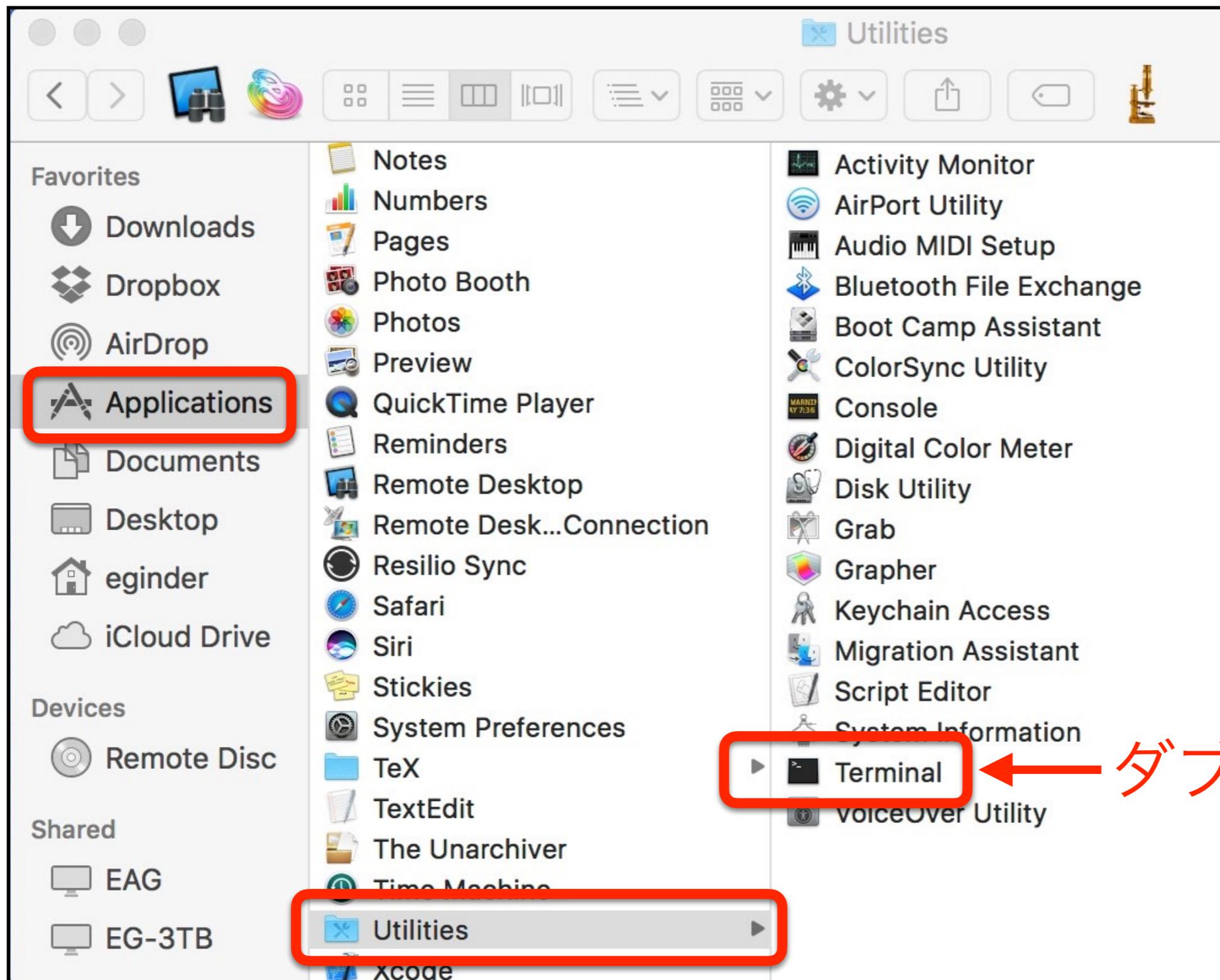
付録

端末 (terminal) の開き方

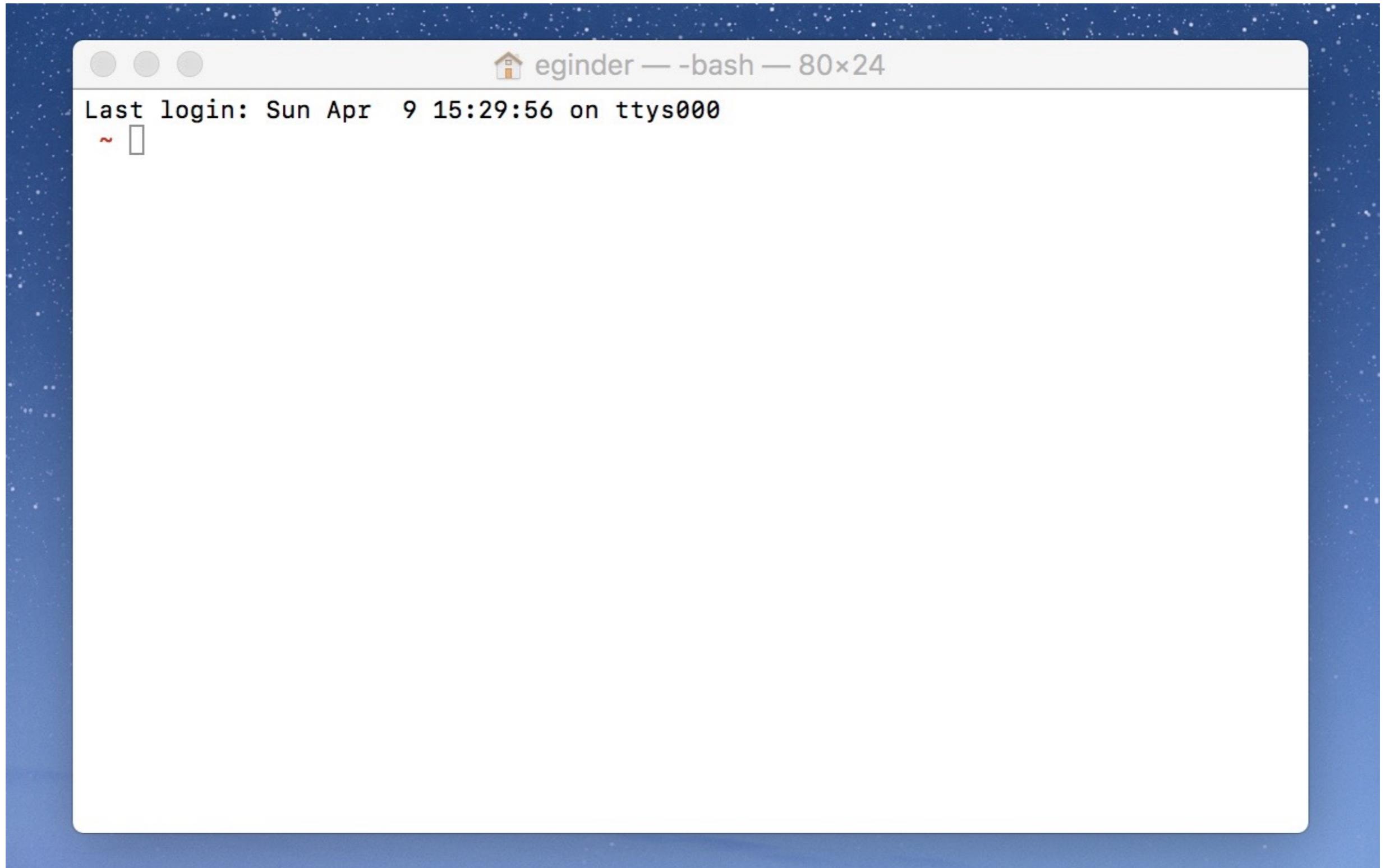


端末 (terminal) の開き方

③



端末 (terminal) の開き方



端末の基本的なコマンド

ls ファイル、ディレクトリ(フォルダのこと)一覧を表示する。

pwd 現在作業中のワーキングディレクトリのパス(位置)を表示する。

open ファイルやアプリなどを開く。

more ファイルの中見を閲覧する。

mkdir ディレクトリ(フォルダ)を作成する。

cd 現在のワーキングディレクトリを移動する。

cp ファイル、ディレクトリを移動する

mv ファイルディレクトリを移動する。

rm ファイルを削除する。

cd .. 前のディレクトリへ戻る

rmdir ディレクトリを削除する。

詳しい使い方は、端末から **man** コマンド名と打つことで、マニュアルが表示されるので、それを参考とする。

manコマンドを終了する場合は、上記のようにマニュアルが表示されている状態で、"q"キーを押します。

プログラミング: 一連の流れ

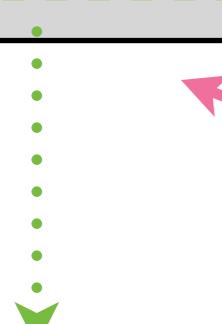
ソースファイルの作成

1

```
#include <stdio.h>  
  
int main(void){  
    printf("Hello World!\n");  
    return 0;  
}
```

prog.c

人に読める形で、言語の文法に従い、計算機への命令を書いたファイル。テキストエディタを使って作成する。



cot, atom, emacs, etc.

ソースファイルのコンパイル

2

cc prog.c

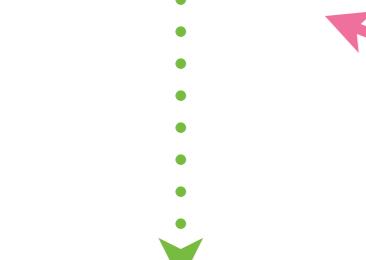
ソースファイルを計算機が読み込んで実行できる形に変換する。コンパイラによって行う。

~~~ a.out

計算機が読み込める形で、命令が書かれたファイル。

コンパイルエラーが出たら修正

-----



## プログラムの実行

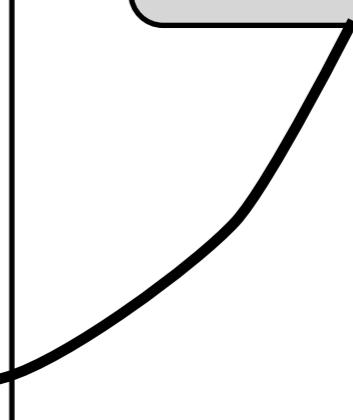
3

./a.out

結果が思った通りのものかチェックすること。

cc proc.c -o mypro.out

出力結果が正しくなければ修正



# C言語で“Hello World” プログラムを書いてみよう

1) ターミナル（端末）を開く



2) CotEditor を開く



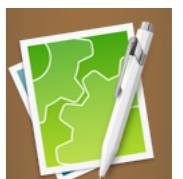
3) 

```
#include <stdio.h>
```

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



一字一句を書く



4)⌘-s (ファイルを保存する)



5) cc hello.c (コンパイル → a.outを作成)



6) ./a.out (プログラムを実行する)

