情報工学演習 1
第 6 回

プログラミング入門の復習 2

講義内容

- 関数
- ファイル入出力

関数

関数について

- ライブラリ関数
- 自分で作成する関数

ライブラリ関数

- よく使う機能の関数を提供
- あるライブラリにある関数を利用をしたい場合
- 例：string.h にある関数を利用

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

ライブラリ関数の例：数学処理の標準ライブラリ

- `math.h`
  - 三角関数
  - 指数
  - 対数
  - 平方根
  - べき乗，累乗
  - 数学定数 (円周率 π，ネイピア数 e など)
- 使い方は web などで各自調べてみよう！

数学処理のサンプルプログラム（１）

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

int main(void) {
  float a,b;
  printf("Input a number: ");
  scanf("%f", &a);
  b = sqrt(a);
  printf("sqrt of %e = %e¥n", a, b);
  return 0;
}
```

←― sqrtを使うため

数学処理のサンプルプログラム（２）

- 三角関数の角度はラジアン

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

int main ( void ){
    float a, s, c, t;
    printf( “Type an angle: ¥n” );
    scanf( “%f”, &a );
    s = sin( a );
    c = cos( a );
    t = tan( a );
    printf( “sin(a): %f, cos(a): %f, tan(a): %f” , s, c, t );
    return 0;
}
```

乱数を発生させるサンプルプログラム

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

int main( void ){

  int a, i;
  for( i=0; i<10; i++ ){
     a = rand();
     printf( "%d¥n", a );
  }
  return 0;
}
```

乱数を何度発生させても同じ？

- 乱数を初期化していないため
- 初期化をすると，毎回異なる乱数を発生させることが可能

初期化を加えたサンプルプログラム

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

int main( void ){

   int a, i;
   srand((unsigned)time(NULL));
   for( i=0; i<10; i++ ){
      a = rand();
      printf( "%d¥n", a );
   }
   return 0;
}
```

time 関数を使うため

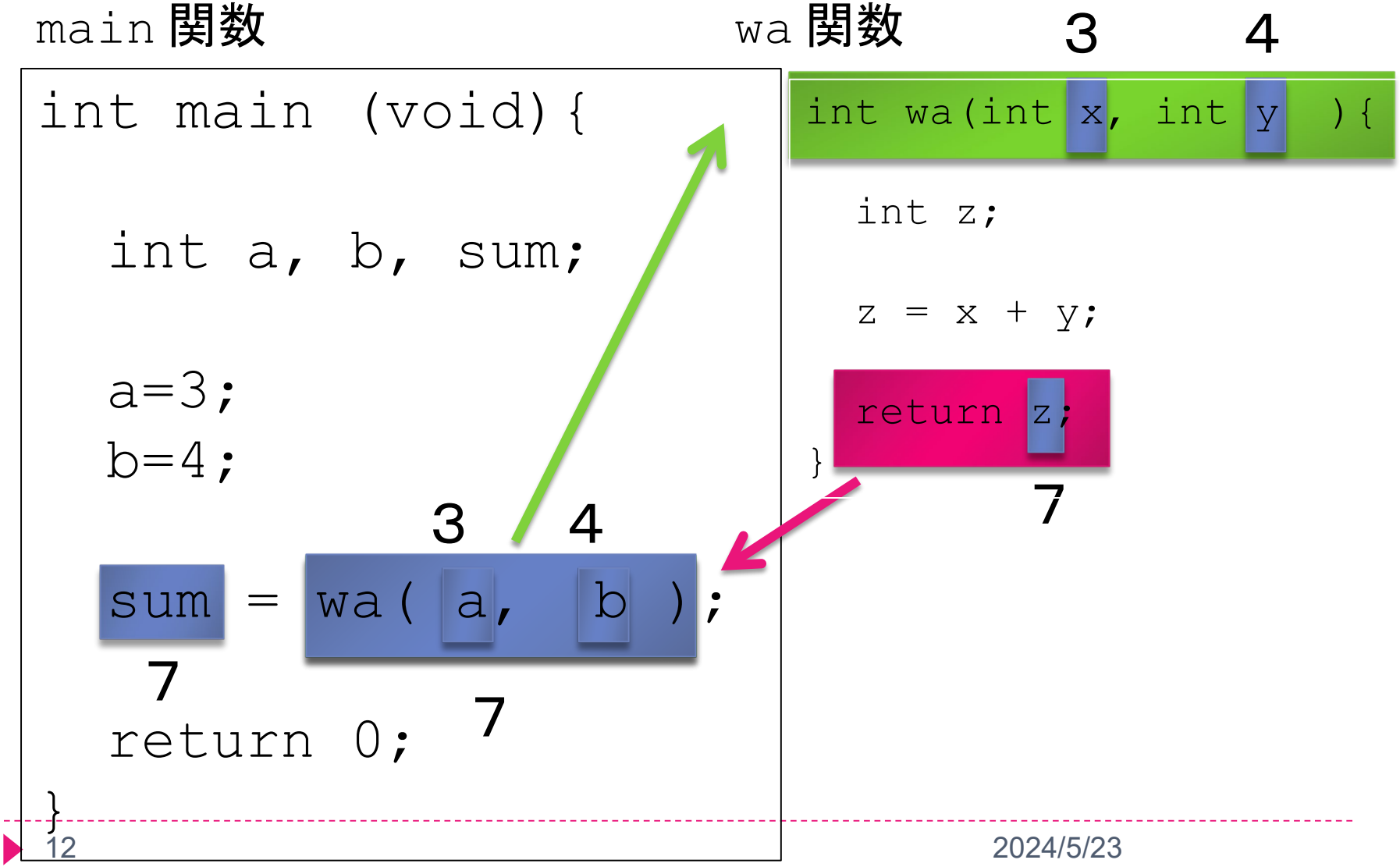
1970年1月1日の0時0分0秒からの秒数 (UNIX時間)

乱数の発生の初期化

関数を自分で定義することの意義

- `main` 関数を短く，読みやすいプログラムにする
- 同じような処理を関数化することで，繰り返しを減らす

関数を用いた処理の流れ



関数内の変数の呼び方

main 関数

```
int main (void){

   int a, b, sum;

   a=3;
   b=4;

   sum = wa( a, b );

   return 0;
}
```

実引数

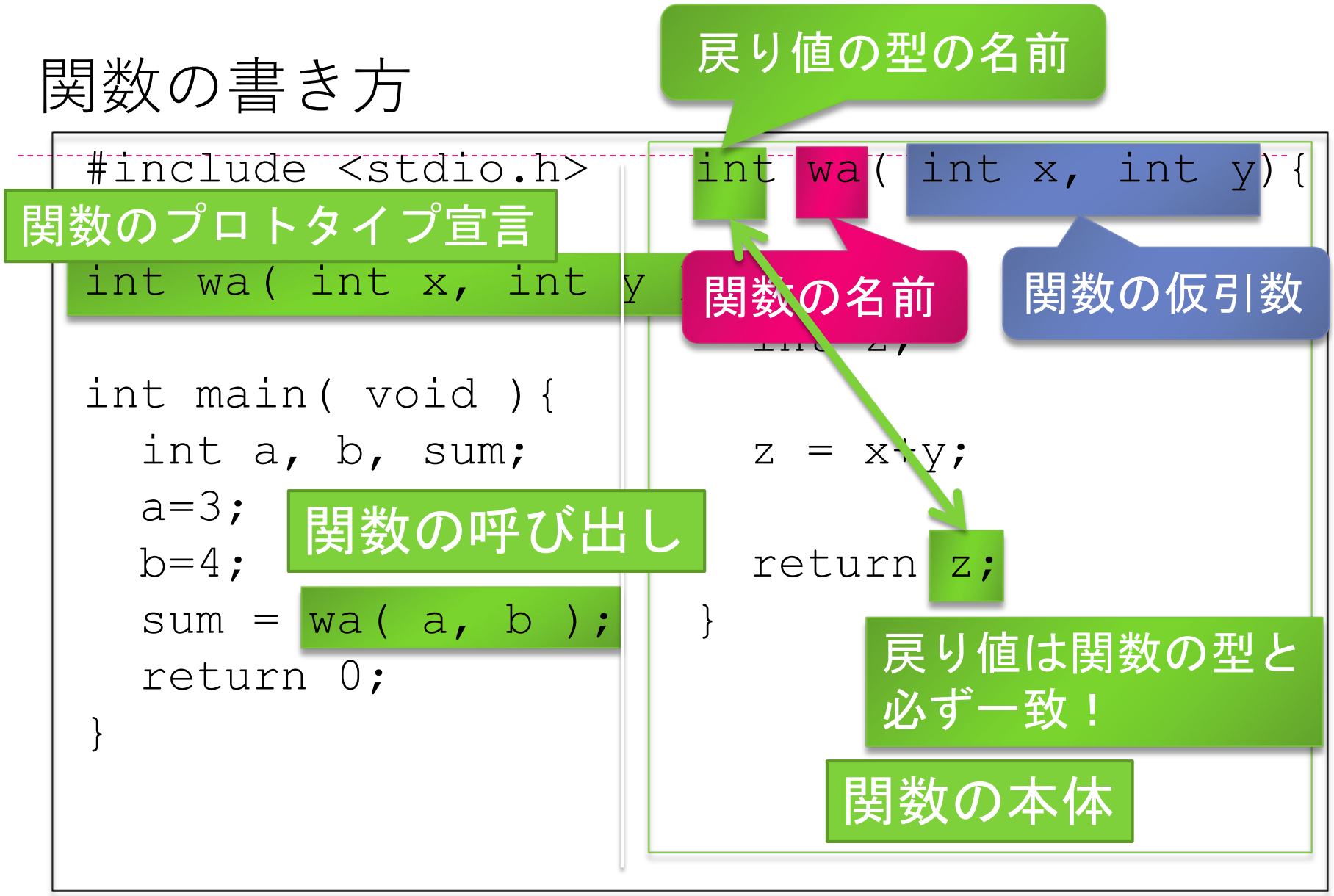
wa 関数

```
int wa( int x, int y ){

   int z;

   z = x + y;

   return z;
}
```

仮引数

戻り値

関数の書き方


戻り値の型の名前
関数のプロトタイプ宣言
int wa(int x, int y
関数の名前
関数の仮引数
#include <stdio.h>
int main(void){
int a, b, sum;
a=3;
b=4;
sum = wa(a, b);
return 0;
}
関数の呼び出し
int wa(int x, int y){
z = x+y;
return z;
}
戻り値は関数の型と必ず一致！
関数の本体

関数の引数のルール

- 仮引数は必要なだけ記述可能
  - 例：`sumpfunc ( int a, doube b, float c);`
- 仮引数と実引数の型は同じ
  - 型が異なる場合，可能であればキャストされる
    例：仮引数 `double` -> 実引数 `int`
- 仮引数は他の関数の仮引数や，変数と同じでもよい
  - 例：
```
int big( int a, int b );
int small( int a, int b );
```
- 引数がない場合は `(void)` と記述
  - 例: `int random_num(void);`
- 引数に構造体を使ってもよい

関数の名前のルール

- 変数と同様のルール
  - 先頭の文字は英字または下線 (_)
  - 2 文字目以降は英字，下線，数字
  - 大文字と小文字を区別
  - 先頭から最低 31 文字まで有効
  - 予約語と同じものは利用不可

戻り値のルール

- 戻り値と関数の型は一致させる
- 戻り値に数式を書いてもよい
  - 例： `return a+b;`
  - この場合，`a+b` の計算結果が戻り値
- 戻り値がない関数の型は `void`
  - 例：`void show_name( int n );`
  - `return` を最後の行に書いてはいけない

配列を引数にしたい場合

- 1 次元配列の場合
  - 仮引数（プロトタイプ，関数本体）

```
double dot_product( double a[], double b[]);
double dot_product( double a[3], double b[3]);
```

  - 実引数（他の関数内での呼び出し）

```
dot_product( a, b );
```

- 2 次元配列の場合
  - 仮引数（プロトタイプ，関数本体）

```
void matrix_sum( double a[][3], double b[][3] );
void matrix_sum( double a[3][3], double b[3][3] );
```

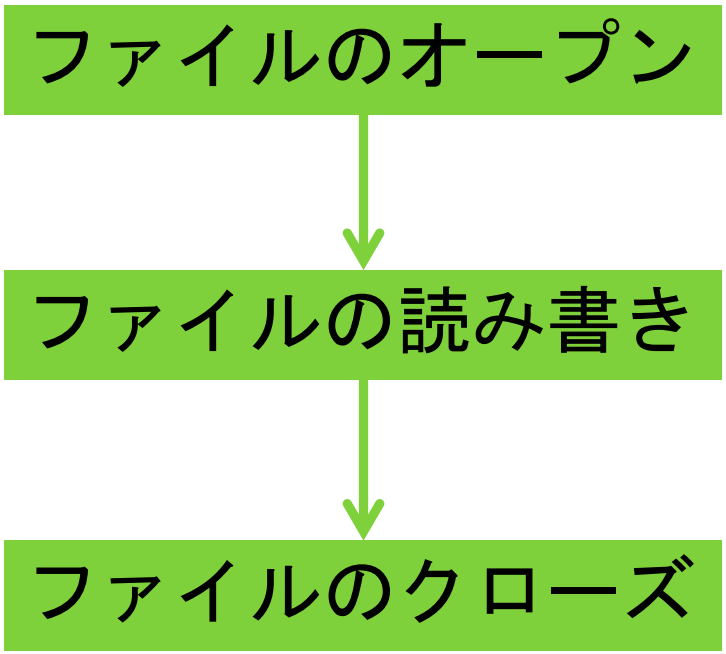
  - 実引数（他の関数内での呼び出し）

```
matrix_sum( x, y );
```

ファイル入出力

2024/5/23

ファイル処理の手順



fopen 関数を使って
ストリームポインタを取得

ストリームポインタを使って
データを読み書きする

fclose 関数を使って
ファイル処理の後始末をする

簡単な解説

どうしてch は int 型なの？

```
FILE   *fp;  /* ストリームポインタ */
int    ch; /* ファイルから読み込んだ文字を代入する用 */

fp = fopen("myfile1.txt", "r"); /* myfile1.txtを読み込み用にオープン */
if (fp == NULL) {
    /* もしmyfile1.txtのオープンに失敗したら */
    printf("Cannot open myfile1.txt¥n");
    exit(1);
}

ch = fgetc(fp); /* fpから1文字取得してchに代入 */
printf("ch = %c¥n", ch); /* chを画面に表示 */

fclose(fp); /*fpをクローズ*/

return 0;
```

fgetc の返り値が int 型である理由

- fgetc の結果，ファイルの中に何も残っていなかった場合，それを知らせなければならない
- ファイルの終端であれば，EOF（End of File）が返り値になる
- EOF は char型で表現される文字以外の値で定義されなければならないため，int型を用いる

fopen 関数のオープンモード(抜粋)

モード	説明
"r"	読み取り ファイルがないとエラーになる
"w"	書き込み ファイルがあればサイズ0にする ファイルがなければ新規に作る
"a"	追加書き込み ファイルがあればファイルの最後に追加する ファイルがなければ新規に作る
"rb"	"r" のバイナリモード
"wb"	"w" のバイナリモード
"ab"	"a" のバイナリモード

サンプルプログラム

ファイルを読み取り用にオープン
失敗したら NULL が返る

```
if( ( fp = fopen(filename, "r") ) == NULL ) {
  printf("Cannot open %s./n", filename);
  exit(1);
}
```

ファイル入出力関数

- ファイルからの読み込み
  - `fgetc`：１文字ずつ
  - `fgets`：１行ずつ
  - `fscanf`：書式指定
- ファイルへの書き込み
  - `fputc`：１文字ずつ
  - `fputs`：１行ずつ
  - `fprintf`：書式指定

ファイル入力のサンプルプログラム

```
#include <stdio.h>
#include <stdlib.h>
int main( void ){
    FILE *fp;
    int c;
    char row[256]
    if( (fp = fopen( “in.txt”, “r” )) == NULL ){
        printf( “cannot open the file.¥n” );
        exit( -1 );
    }
    c = fgetc( fp );
    fgets( row, sizeof( row ), fp );
    fscanf( fp, “%d”, c );

    fclose( fp );
    return 0;
}
```

int 型

読み込む最大の文字数を配列の要素数で指定
（sizeof演算子は配列の要素数を返す）

読み込む形式とバッファを指定

ファイル出力のサンプルプログラム

```
#include <stdio.h>
#include <stdlib.h>
int main( void ){
    FILE *fp;
    int dt = 12345;
    if( (fp = fopen( “out.txt”, “w” )) == NULL ){
        printf( “cannot open the file.¥n” );
        exit( -1 );
    }
    fputs( “文字列を出力¥n”, fp );
    fputc( ‘A’, fp );
    fputc( ‘¥n’, fp );
    fprintf( fp, “dt=%d¥n”,  dt );

    fclose( fp );

    return 0;
}
```

コマンドラインから引数を渡す

- main関数の引数部分を、以下のように書く

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

  - argc ： 引数の数（コマンド名自身を含む）
  - argv ： 引数文字列の配列
    - argv[0] ： 0番目の引数文字列（コマンド名 ./a.exe等）
    - argv[1] ： 1番目の引数文字列

コマンドラインから引数を渡す（サンプルコード）

```c
#include <stdio.h>

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

```
> gcc -o main main.c
> ./main aa bbb cccc ddd
argc = 5
argv[0] = ./main
argv[1] = aa
argv[2] = bbb
argv[3] = cccc
argv[4] = ddd
>
```

演習課題（１）

1. 一桁の奇数をランダムに出力するプログラムを作成せよ
2. ある `int` 型一次元配列に格納されている値の平均を求める関数を作成し，`main` 関数内で挙動を確認せよ
   - 関数の仮引数に配列，戻り値を平均とする
   - 配列の要素数は `10` に固定すること
   - `main` 関数内でコマンドラインに配列の要素と平均を出力すること
3. ２つの文字列を入力したときに，２つの文字列が等しいか異なるかを示すプログラムを作成せよ
   - 文字列を処理するライブラリ関数を使用すること

演習課題（２）

4. ${}_aC_b$を計算する関数を作成し，main 関数の中で挙動を確認せよ．
   - 引数はa, b とし，戻り値は${}_aC_b$の計算結果とする
   - a,b に不適切な値が入力された場合は，警告を出してプログラムを終了すること．
5. テキストファイルを読み込み，指定したアルファベットの小文字を大文字に置き換えて新しいファイルに出力するプログラムを作成せよ．
   - アルファベットの小文字と読み込むファイル名，出力ファイル名はコマンドラインの引数から指定すること
   - デバグ用に emperors_new_clothes.txt を授業支援システムに用意しているので，必要な人は使って下さい

提出に関して（1）

- 提出するもの
  - ソースファイル(.c ファイル)
    - ファイル名は kadai6-学籍番号-課題番号.c
    - ソースファイルにはコメントアウトで学籍番号と氏名を記入
  - 出力ファイル
    - out6-学籍番号-課題番号 .txt
  - 実行結果と講義に関するコメント
    - tex で作成した .pdf ファイルで、学籍番号、氏名を含む
    - ファイル名は report6-学籍番号.pdf とする
  - 全提出ファイルを1つのフォルダに格納し，そのフォルダを zip 圧縮したファイルを提出せよ.
    - フォルダ名：report6-学籍番号
    - Zip 圧縮したファイル名 : report6-学籍番号.zip

提出に関して（２）

- 提出期限
  - **5 月 29 日（水）23:59 JST.**
- 提出方法
  - Moodle から提出
- 注意点
  - ファイル名の命名規則が間違っているものは減点する
  - Repl.it の Cモードで動作しないものは採点しない (0 点)