プログラミング入門1
第11回

芝浦工業大学 工学部 情報工学科
菅谷みどり, 井尻敬

今日の内容

- タイピング(10分) & UNIXテスト(10分)
- プログラミング実技テスト(20~30分)
- 前回の復習
- 関数
- 分割コンパイル
- プログラミング課題
  - 基本課題 3問
  - 発展課題 2問

プログラミング実技テスト

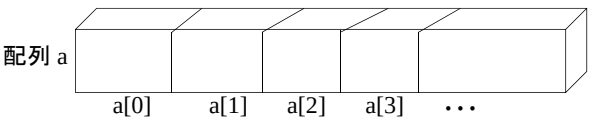
- 準備
  - UNIXにログイン
  - マウスとキーボード以外はしまう
  - 不要なウィンドウは消す
- テストの手順
  - 問題を出題するのでemacsでプログラムを作成する.
  - プログラムが完成したらコンパイル，および実行してみて正しく動作するか自分で確認する.
  - ソースファイルに指定のファイル名を付け, メールに添付して担当TAへ送る.

※ 正しいプログラムを制限時間内にTAへ送信できたら合格.

前回の復習

復習： 大量のデータを処理するために‥配列

- 配列とは，仕切られた長い箱である
- 各仕切りを「配列要素」と呼ぶ
- 配列要素には，0から始まる番号が振られており，配列名[番号]で参照できる



- 配列要素は，普通の変数と「全く同じ」方法で扱うことができる.

復習：配列要素は普通の変数と同じように扱える

- 値の代入

```
a[3] = 4;
scanf ("%d", &a[3] );
```

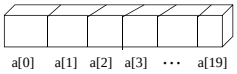
- 値の参照

```
x = a[2] * 3;
printf ("a[5]の値は %d です", a[5] );
```

復習： 配列を使うための準備（宣言）

- 配列を使うときは，普通の変数と同じように，プログラムの最初に配列名と要素数を宣言する必要がある．

例： `int a[20];`



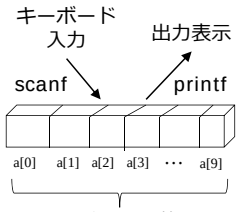
配列 a は20個(a[0]～a[19])の仕切りを持った箱(配列)である

復習： 複数データを入出力するプログラム例

```
#include <stdio.h>

int main(void)
{
    int a[10];
    int i;
    for (i = 0; i < 10; i++) {
        scanf ("%d", &a[i]);
    }
    for (i = 0; i < 10; i++) {
        printf ("%d¥n", a[i]);
    }
    return 0;
}
```

この 2行は 1行にまとめることもできる
`int a[10], i;`



for文の繰り返し範囲は
0≦i<10
（または 0≦i≦10-1)

関数

例題： 順列・組合せを計算する

- n 個のものから r 個を取り出す組合せの数

$$nCr = \frac{n!}{r! \times (n-r)!}$$

- プログラムを作成してみよう
  - 階乗計算が3回あるので..
  - ここで　n! = 1×2×…×n （階乗計算）．

例題： 順列・組合せを計算する

```
#include <stdio.h>

int main(void)
{
  int i, n,  r, answer;
  int fact_answer[3];
  printf ("n?  "); scanf("%d" , &n );
  printf ("r?  "); scanf("%d" , &r );

  // n の階乗計算
  for ( answer = i = 1; i <= n; i++ ) {
    answer = answer * i;
  }
  fact_answer [0] = answer;

  //  r の階乗計算
  for ( answer = i = 1; i <= r; i++ ) {
    answer = answer * i;
  }
  fact_answer [1] = answer;
```

```
  // n – r の階乗計算
  for ( answer =  i = 1; i <= (n-r); i++ ) {
    answer = answer * i;
  }
  fact_answer [2] = answer;

  // 計算結果表示
  printf ("nCr = %d¥n",
          fact_answer [0] /
          fact_answer [1] /
          fact_answer [2]));
  return 0 ;
}
```

例題： 順列・組合せを計算する

```
#include <stdio.h>

int main(void)
{
  int i, n,  r, answer;
  int fact_answer[3];
  printf ("n?  "); scanf("%d" , &n );
  printf ("r?  "); scanf("%d" , &r );

  // n の階乗計算
  for ( answer = i = 1; i <= n; i++ ) {
    answer = answer * i;
  }
  fact_answer [0] = answer;

  //  r の階乗計算
  for ( answer = i = 1; i <= r; i++ ) {
    answer = answer * i;
  }
  fact_answer [1] = answer;
```

```
  // n – r の階乗計算
  for ( answer =  i = 1; i <= (n-r); i++ ) {
    answer = answer * i;
  }
  fact_answer [2] = answer;

  // 計算結果表示
  printf ("nCr = %d¥n",
          fact_answer [0] /
          fact_answer [1] /
          fact_answer [2]));
  return 0 ;
}
```

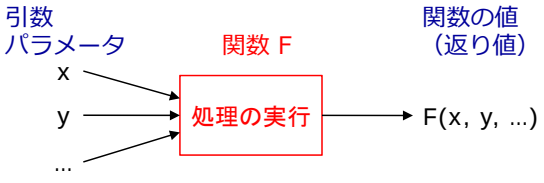
似たような処理が何度もある
→ 関数を使ってコードを整理

プログラミングにおける関数とは？

（数学で言う「関数」と必ずしも同じでないので注意！）

特定の処理を行うプログラムのかたまり

- **引数（ひきすう）**としてデータを受け取り
- データを利用して特定の処理を行ない
- 処理の結果として得られたデータを返す**（返り値）**



関数の役割

- 入力データから出力データを求める処理を 1つのまとまりとして表現する.
- プログラム中に同じ処理が何回も出てくるとき，その処理を関数にすれば，プログラムが簡潔になる.

- 関数をうまく利用すると…
  - 可読性向上：プログラムが短く読みやすくなる
  - 再利用性向上: よく使う処理を関数にしておくと，次からはそれを呼び出すだけでよい
  - 管理しやすさ向上：同じ処理が複数の場所に存在しその部分にバグが発見されたら？バグを見つけたときにはプログラム全体を見る必要が…

ライブラリ関数と自作の関数

- ライブラリ関数
  - あらかじめシステムに用意された関数
  - プログラム中で自由に呼び出して使える
    例：printf関数，scanf関数，sin関数，log関数など

```
x = sin(y) ;
```

sin関数の呼び出し
（sin関数の定義はシステムに組込まれている）

- 自作の関数
  - システムに用意されていない関数を，自分で独自に定義して，それを呼び出して使うことができる.

例題： 順列・組合せを計算する

- n 個のものから r 個を取り出す組合せの数

$$nCr = \frac{n!}{r! \times (n-r)!}$$

- プログラムを作成してみよう
  - 階乗計算が3回あるので..
  - ここで　n! = 1×2×…×n （階乗計算）.

実習： 組み合わせの数の計算 samp6-1.c

下のプログラムを作成・実行せよ

```
#include <stdio.h>
int fact(int n)
{
    int answer , i ;
    answer = 1 ;
    for ( i = 1; i <= n; i++ ) {
        answer = answer * i ;
    }
    return answer ;
}
int main(void)
{
    int n, r ;
    printf (”n? ”) ;    scanf(”%d” , &n ) ;
    printf (” r? ”) ;    scanf(”%d” , &r ) ;
    printf (”nCr = %d¥n” , fact(n) / fact(r) / fact(n-r) ) ;
    return 0;
}
```

fact関数の定義

fact関数の呼び出し

C言語における関数の定義方法

```
返り値の型　関数名 ( 引数の型　引数名 )
{
    変数の宣言
    実行する処理
    return 関数が返す値 ;
}
```

「変数の宣言」，「実行する処理」は省略される場合もある

例： F(x) = x*x*x*x の場合

```
int F (int x)
{
    return x*x*x*x ;
}
```

続き

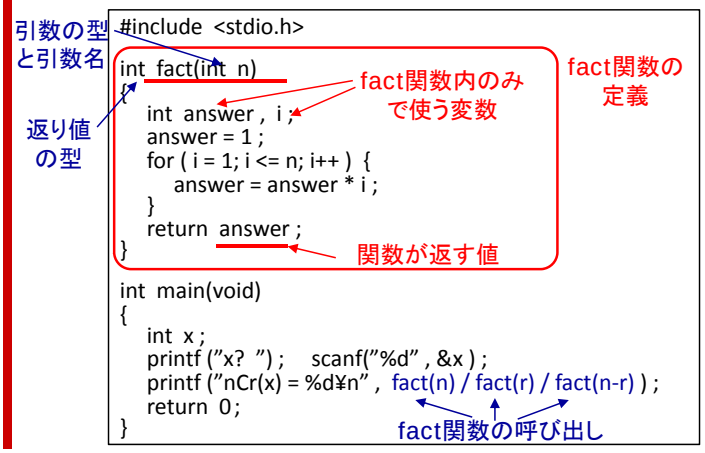
- 引数は 2個以上あってもよい.
- 例： xとyの平均値を求める関数 average(x, y)

```
double  average (double  x , double  y)
{
   return  ( x + y ) / 2 ;
}
```

これを次のように定義することもできる.

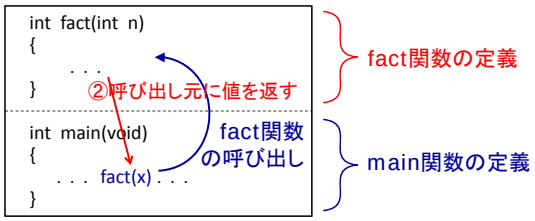
```
double  average (double  x , double  y)
{
   double  sum , ave ;        変数の宣言
   sum = x + y ;
   ave = sum / 2 ;
   return  ave ;
}
```

組み合わせの数(nCr) の計算



main関数とその他の関数

- main関数は，その他の関数と同じ形式で関数定義される（返り値の型は int型．引数は無し（void））．
- プログラムが実行を開始すると，まずmain関数が呼び出される．その他の関数はmain関数から呼び出される（またはmain関数から呼び出される関数から呼び出される，など）．
- 1つのプログラムには，必ず1つのmain関数が含まれる．



数学における「関数」との違い

1. 引数のない関数を定義することもできる.

例： main関数 `int main(void)`

処理の実行・値の計算 → 返り値 int

2. 返り値のない関数を定義することもできる.

例： 次の実習 `void print_binary(int n)`

引数 n → 処理の実行・値の計算

まとめ：関数

- 関数とは特定の処理を行なうプログラムのかたまり
  - 引数を受け取り（省略可）
  - 処理を行い
  - 返り値を返す（省略可）

```
返り値の型  関数名 ( 引数の型  引数名 )
{
    変数の宣言
    実行する処理
    return  関数が返す値 ;
}
```

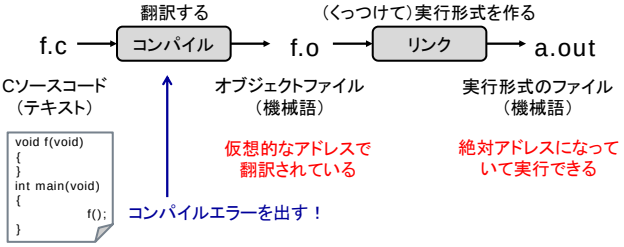
- 関数化のメリット
  - コードが簡潔で読みやすくなる
  - バグ発見やデバッグが楽になる
  - 関数名をうまくつけると処理の流れが明確に
  - 複数人で作業を分担できる → 分割コンパイル

コンパイルとリンク

コンパイルとリンク

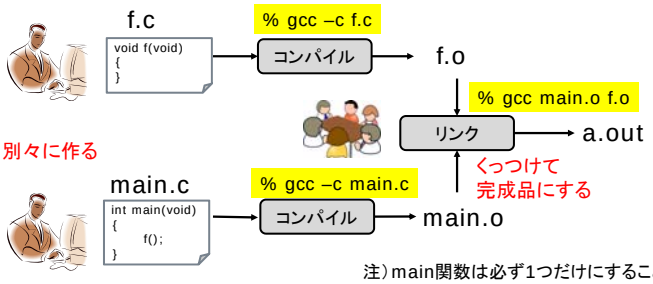
gccコマンドは，『コンパイル』と『リンク』という２つのことを実行している

```
%  gcc f.c
```



分割コンパイル

- 異なるファイルを使って，関数ごとに開発できる
- 開発後，別々にコンパイルしたものをリンカでくっつけて実行ファイルを作ることができる



注）main関数は必ず1つだけにすること

実習：分割コンパイル

ファイル名：fact.c

```
int  fact(int  n)
{
 int  answer,  i;
 answer = 1;
 for ( i = 1; i <= n; i++ )  {
   answer = answer * i;
 }
 return  answer;
}
```

ファイル名：main.c

```
#include <stdio.h>

int fact(int); //プロトタイプ宣言（コンパイラに呼び出す関数の情報を伝える）

int  main(void)
{
 int  n,r;
 printf ("n?  "); scanf("%d", &n);
 printf ("r?  "); scanf("%d", &r);
 printf ("nCr = %d¥n",
fact(n)/fact(r)/fact(n-r));
 return  0;
}
```

- コンパイルの手順
  1. gcc –c fact.c main.c ➔ オブジェクトモジュール生成
  2. gcc fact.o main.o ➔ リンクしてプログラムを生成
  2’. gcc –o fact fact.o main.o ➔fact という名前でプログラムを生成

実習：整数を2進数に変換して表示する関数

- 関数 print_binary(int n)
  - 「整数 n の2進数表現を求めて表示する」 という処理を行う関数
  - 0≦n≦255 の範囲を想定し，8桁の符号無し2進数表現を求めることにする．
  - 処理手順

    n を 2 で割る処理を 8回繰り返し，それぞれの「余り」を右→左の順に並べればよい．ただし画面に出力する際は左の桁から順に出力する必要があるので，配列を利用して，桁の順番を反転させる．

```
100÷2 = 50・・・0
 50÷2 = 25・・・0
 25÷2 = 12・・・1
 12÷2 =  6・・・0
  6÷2 =  3・・・0
  3÷2 =  1・・・1
  1÷2 =  0・・・1
  0÷2 =  0・・・0
⇒ 01100100
```

```
#include <stdio.h>

void print_binary(int  n)
{
   int  a[8] , i ;
   for ( i = 0 ; i < 8 ; i++ ) {
      a[i] = n % 2 ;
      n = n / 2 ;
   }
   for ( i = 7 ; i >= 0 ; i-- ) {
      printf ("%d" ,  a[i] ) ;
   }
}

int  main(void)
{
   int  x, y ;
   printf ("x? ") ;  scanf ("%d" ,  &x ) ;
   printf ("y? ") ;  scanf ("%d" ,  &y ) ;
   printf ("   x = ") ;  print_binary(x);  printf (" %d¥n" , x ) ;
   printf ("   y = ") ;  print_binary(y);  printf (" %d¥n" , y ) ;
   printf ("x+y = ") ;  print_binary(x + y);  printf(" %d¥n" , x+y) ;
   return  0 ;
}
```

実習6-2. samp6-2.cとして入力･実行

実習6-3. ファイル分割し，分割コンパイルしてみる

ロボットPBL

- 12回，13回はロボットPBLを行います
  - 基本課題などが未提出の人はこの期間に提出
  - 未提出課題の有無は担当TAに確認してください
- 12回目
  - 競技：規定
- 講義時間外のロボット貸し出し
  - 日程：7/12(火), 14(木), 15(金)
  - 16:10 – 18:30
- 13回目
  - 競技：フリー
  - ショートプレゼンテーション

今日の課題

プログラミング課題について

- 毎回 4～5問を出題するので，授業の後半の時間を使って各自プログラムを作成し，担当 TA のチェックを受ける （作ったプログラムの内容と，実行の様子を見てもらう）．
- 授業時間内に完成しなかった課題は，原則として翌週の授業までに各自で完成させて，翌週の授業時間中に TA にチェックしてもらう．
- 課題の種類： 基本課題，発展課題
  - まず基本課題に取り組む
  - 基本課題が完成したら，発展課題をやる

基本課題1：三角形の面積 （ex6-1.c）

- 底辺の長さ（x）と高さ（y）を入力すると三角形の面積を求めるプログラムを，関数を使って作りなさい．

```
#include <stdio.h>

double triangle(double x, double y)
{
    （ここに処理を書く）
}

int main(void)
{
    double x, y;
    printf("底辺? "); scanf("%lf", &x);
    printf("高さ? "); scanf("%lf", &y);
    printf("三角形の面積は%f¥n", triangle(x, y));
    return 0;
}
```

底辺x, 高さyの三角形の面積を計算して，結果を返す関数

基本課題2： 大きい方の数の出力（ex6-2.c）

- 2つの整数 x, y を入力すると，大きい方の数を出力するプログラムを，関数を使って作りなさい．
  - 次のような関数を定義して使うこと．

```
int max(int x, int y)
{
    （ここに処理を書く）
}
```

x, y のうち大きい方の数を返す関数

もちろん，この他にmain関数を用意する必要がある．

- 実行例

```
$ ./a.out
100
200
大きい方の数は200です
```

キーボードからの入力

プログラムの出力

基本課題3： べき乗の計算（ex6-3.c）

- 実数 x と自然数 n を入力すると，べき乗 x^n を計算して出力するプログラムを，関数を使って作りなさい．
  - ライブラリ関数powを使うのではなく，次のような関数を自作して使うこと．値の計算には for文を使うこと．

```
double power(double x, int n)
{
    （ここに処理を書く）
}
```

べき乗 x^n = x * x * ･･ * x（n個の積）を計算して返す関数

この他にmain関数を用意する必要がある．

- 実行例

```
$ ./a.out
x? 2
n? 10
2.000000の10乗は1024.000000
```

キーボードからの入力

プログラムの出力

ヒント： べき乗の計算方法

- べき乗 x * x * ･･ * x の計算は，

n個の積

階乗 1 * 2 * ･･ * n の計算（samp6-1.c）

n個の積

とほぼ同様にできる （階乗の場合，i 回めの繰り返しで answer に i をかけるが，べき乗の場合は i の代わりに x をかければよい）

注： 本問では，べき乗の計算で n≧0 の場合のみ考えればよい．したがって上記のやり方で済む．（もし n<0 の場合も扱うならば，除算が必要となる）

発展課題1： 最大公約数を求める （ex6-4.c）

- 2つの正の整数 x, y を入力すると，ユークリッドの互除法を用いて x, yの最大公約数 gcd(x, y) を求めて出力するプログラムを作りなさい.
- 次ページに示すように，main関数から gcd関数と read_positive関数を呼び出す構成とすること.
- 実行例

```
$ ./a.out
x? 100
y? -50
正の数を入力して下さい 0
正の数を入力して下さい 65
gcd(100, 65) = 5
```

正でない数が入力された場合，再入力を要求する.

続き： 以下のようなプログラム構成とすること

```
#include <stdio.h>
int read_positive(void)
{
    （キーボードから正の数が入力されるまで繰り返し入力を要求する．正の数が入力されたら，その数を返す）
}

int gcd( int x , int y )
{
    （x と y の最大公約数を求めて返す）
}

int main(void)
{
    int x, y ;
    printf ("x? ") ;  x = read_positive() ;
    printf ("y? ") ;  y = read_positive() ;
    printf ("gcd(%d, %d) = %d¥n" , x , y , gcd(x, y) ) ;
    return 0 ;
}
```

ヒント： ユークリッドの互除法

正の整数 x, y (x≧y) の最大公約数を求める手順

1. r = x % y とする.
2. r が 0 なら，yを最大公約数として出力し終了.
3. x = y とする.
4. y = r とする.
5. r = x % y とする.
6. 2 に戻る.

計算例

```
 x    y      r
100 ÷ 65 = 1・・・35
 65 ÷ 35 = 1・・・30
 35 ÷ 30 = 1・・・5
 30 ÷  5 = 6・・・0
   最大公約数
```

発展課題2： 数値積分 （ex6-5.c）

- 台形公式を用いて定積分 $\int_a^b f(x)\,dx$ の近似値を求めるプログラムを作りなさい.
  - 関数 f(x) と積分区間 a, b の値はプログラム内に直接書き，積分区間の分割数 n の値のみ，実行時にキーボードから入力する
  - 例として次の 2つの定積分の値を計算してみること

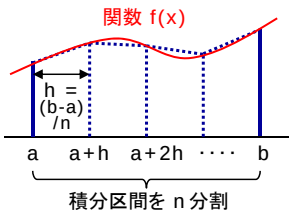
  1. $\int_0^1 x^2\,dx \left(=\frac{1}{3}\right)$　　2. $\int_0^\pi x\sin x\;dx\,(=\pi)$

- 実行例

```
[*** ~/pro1]$ ./a.out
n? 100        ← キーボードからの入力
0.333350      ← 定積分値の出力（①の場合）
```

数値積分と台形公式について

- 数値積分 ‥ 積分を解析的に解く（原始関数を求める）ことなく，数値計算で定積分の近似値を求める
- 台形公式 ‥ 数値積分の最も単純な計算法の一つ．積分区間を n 等分し，下図のような台形領域の面積の和を計算することで，定積分の近似値を求める



$$\int_a^b f(x)dx = \frac{(f(a)+f(a+h))\cdot h}{2} + \frac{(f(a+h)+f(a+2h))\cdot h}{2} + \Lambda$$