プログラミング基礎演習
第3回
C言語編
【再帰】

長谷川禎彦

気づいたこと

- インデントをしよう
  - 「Ctrl + x, h」 で全選択
  - 「Tab」で自動インデント
- コンパイルするとき，**-Wall**オプションを使う
  - **gcc -Wall -o kadai0x kadai0x.c**
  - 全警告を出力する＞バグが見つかりやすい
- 1行**if, for**でも中括弧**{ }**を使う

前回課題1

```c
#include <stdio.h>
#include <math.h>
int main() {
  double s = 0;
  double pi;
  int n;
  for(n = 1;n < 100;n++) {
    s += 1 / (n*n*pow(2,n-1));
  }
  s += log(2)*log(2);

  pi = sqrt(6 * s);
  printf("pi = %f\n", pi);
}
```

課題1の補足

- どこまで足すべきか？
- doubleの値xが表せる絶対値の範囲は

$$2.225074 \times 10^{-308} < |x| < 1.797693 \times 10^{+308}$$

なお，0は表すことが可能

- つまり，forループで10000まで足しても意味がない
  - $2^{-10000} = 10^{-3000}$
- 100までの和で誤差は~10^{-34}，1000まで足せば~10^{-307}となっている

前回課題2

```c
#include <stdio.h>
#include <math.h>
int zeller(int, int, int);
int main() {
  int year = 2016, month = 10, day = 5;
  int w;
  w = zeller(year, month, day);
  printf("%d/%d/%d is ",month, day, year);
  switch(w) {
  case 0: printf("Sat.\n"); break;
  case 1: printf("Sun.\n"); break;
  case 2: printf("Mon.\n"); break;
  case 3: printf("Tue.\n"); break;
  case 4: printf("Wed.\n"); break;
  case 5: printf("Thurs.\n"); break;
  case 6: printf("Fri.\n"); break;
  }
  return 0;
}
```

```c
int zeller(int year, int month, int day) {
  int m = month;
  int d = day;
  int year2 = year;
  int w;

  if (m < 3) {
    m += 12;
    year2--;
  }
  int h = (int) (year2 / 100);
  int y = year2 % 100;
  w = y + (y/4) + (h/4) - 2*h + (int)(13*(m+1)*0.2) + d;

  return w % 7;
}
```

キャストすると，少数は切り捨て

整数/整数 = 整数（少数は切り捨て）

前回課題3,4

```c
#include <stdio.h>
#include <math.h>
int is_prime(int); /*引数が素数の場合1, それ以外の場合0を返す*/
int main() {
    int i;
    double p = 1.0;
    for (i = 2; i <= 10000; i++) {
      if (is_prime(i) == 1) {
           p *= (1 - 1.0 / (i * i));
      }
    }
    printf("pi = %f\n", (sqrt(6 / p)));
    return 0;
}
```

```c
int is_prime(int n) {
    int i;
    for (i = 2; i < n; i++) {
      if (n % i == 0) {
           return 0;
      }
    }
    return 1;
}
```

前回課題5,6

```c
#include <stdio.h>
int gcd(int, int);
int main() {
  int x,y,z;
  int max = 1000;
  for (x = 1; x < max; x++) {
    for (y = x; y < max; y++) {
      for(z = y; z < max; z++) {
       if (x*x + y*y == z*z && gcd(x,y) == 1) {
         printf("%d^2 + %d^2 = %d^2\n",x, y, z);
       }
      }
    }
  }
  return 0;
}
```

ここで互いに素かどうか確かめる

```c
int gcd(int a, int b) {
    int t;
    while (b != 0) {
     t = b;
     b = a % b;
     a = t;
    }
    return a;
}
```

前回課題7

```c
#include <stdio.h>
#include <math.h>
int is_prime(int);
int goldbach(int);
int main() {
  int i;
  int n;
  int m;
  for (i = 2; i <= 50; i++) {
      n = 2 * i;
      m = goldbach(n);
      printf("%d = %d + %d\n", n, m, n - m);
  }
  return 0;
}
```

```c
int goldbach(int n) {
  int i;
  for (i = 2;i < n;i++) {
      if (is_prime(i) == 1 && is_prime(n - i) == 1) {
      return i;
      }
  }
}
int is_prime(int n) {
  int i;
  for (i = 2;i < n;i++) {
      if (n % i == 0) {
      return 0;
      }
  }
  return 1;
}
```

再帰

- 再帰呼び出しとは、関数fの中で関数fを呼び出すこと
- 慣れないと難しいが，簡潔なソースプログラムを書くことが可能

再帰の例:階乗

- n!は(n-1)!にnを掛けたもの
- factorial(n) = n * factorial(n-1)

```
int factorial(int n) {
   return n * factorial(n-1);
}
```

再帰:階乗

- 処理を終了する条件が必ず一つは必要
  - ないと無限ループに陥る

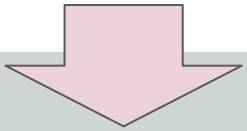
階乗の場合 1!=1 で止める(0!=1でもOK)

```
int factorial(int n) {
    if (n == 1) { return 1; }
    else { return n * factorial(n-1); }
}
```

漸化式と再帰

フィボナッチ数列

$$F(n+2) = F(n+1) + F(n),\ \ F(1)=1,\ \ F(0)=0$$



```
int F(int n) {
  if(n==0) {
    return 0;
  } else if(n==1) {
    return 1;
  } else{
    return F(n-1)+F(n-2);
  }
}
```

課題1（必須課題）

量子力学で出てきたLegendre多項式$P_n(x)$は

- $P_0(x) = 1$
- $P_1(x) = x$
- $(n+1)\,P_{n+1}(x) = (2n+1)\,x\,P_n(x) - n\,P_{n-1}(x)$

で定義される．再帰関数を用いることで

$P_{16}(0.5)$や$P_{32}(0.7)$を計算せよ．

なお，$P_{128}(0.5)$などは計算が終わらない（愚直に実装した場合）．理由も考えてみよ．

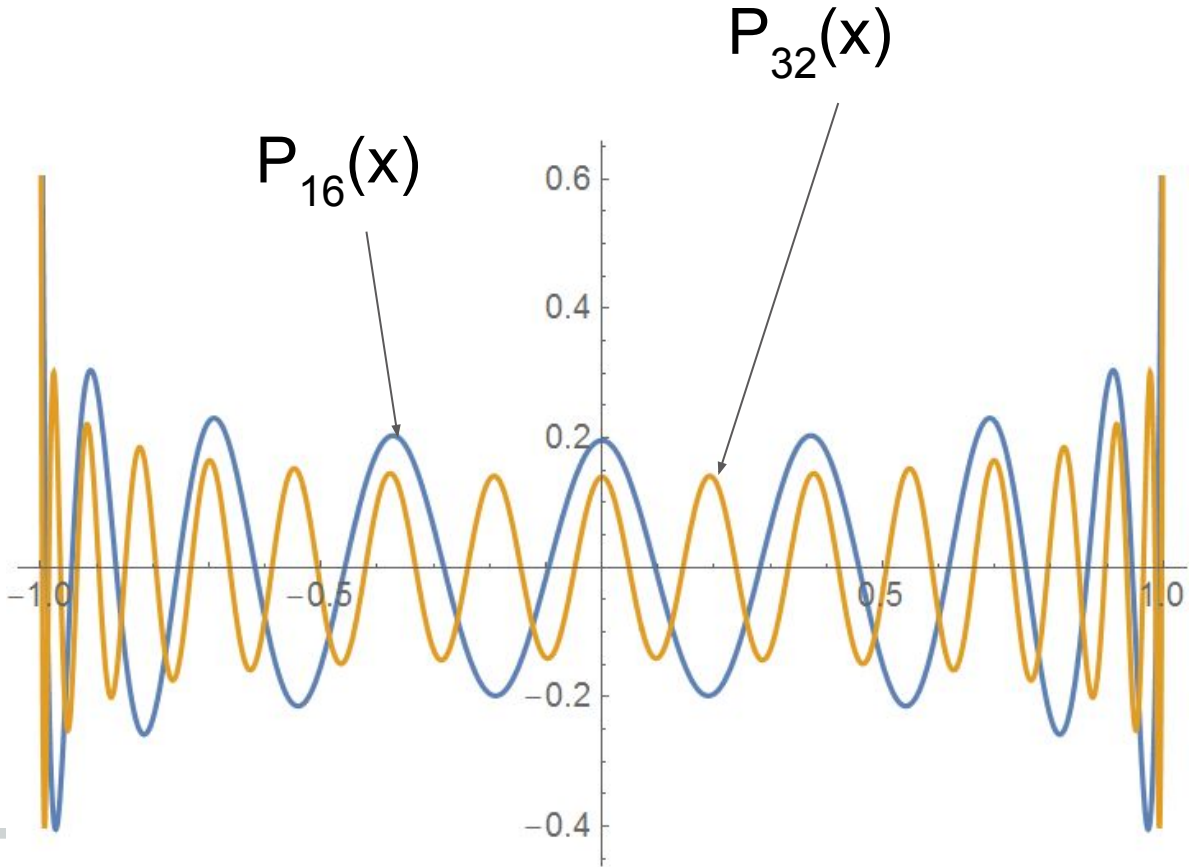
課題1

正しくプログラム出来ていれば

$P_{16}(0.5) = -0.1498...$

$P_{32}(0.7) = 0.1651...$

の値が計算される



```c
#include <stdio.h>

double legendre(int n, double x);
int main() {
  double v = legendre(16, 0.5);
  printf("%f\n", v);
  v = legendre(32, 0.7);
  printf("%f\n", v);
  return 0;
}

double legendre(int n, double x) {
  /*ここを埋める*/
}
```

「整数割る整数は整数」に注意！
1/2は0.5ではなく0になってしまう！

再帰を使っていればこのひな形を用いなくてもよい

課題2（必須課題）

以下の式は連分数（continued fraction）と呼ばれる.

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}$$

課題2

- 数の平方根は以下の連分数で求めることが出来ることが知られている

$$\sqrt{z} = 1 + \cfrac{z-1}{2 + \cfrac{z-1}{2 + \cfrac{z-1}{2 + \cfrac{z-1}{2+\cdots}}}}$$

- 連分数でzの平方根を計算するprogramを再帰で作れ
- 適当な深さで（深さ10くらい）で切る必要がある
  - これが再帰の止まる条件

課題2

再帰を使っていればこのひな形を用いなくてもよい

```c
#include <stdio.h>
double sqrt_cf(double);
double K(double, int);
int main() {
    double z = 2;
    double s = sqrt_cf(z);
    printf("%f\n", s);
    return 0;
}
double sqrt_cf(double z) {
    return 1.0 + K(z, 0);
}
double K(double z, int depth) {
    /*ここを埋める*/
}
```

$$\sqrt{z} = 1 + \underbrace{\frac{z-1}{2+\frac{z-1}{2+\frac{z-1}{2+\frac{z-1}{2+\cdots}}}}}_{K(z,\,\mathrm{depth})}$$

depth = 0

depth = 1

depth = 2

depthは，現在の連分数の深さ．

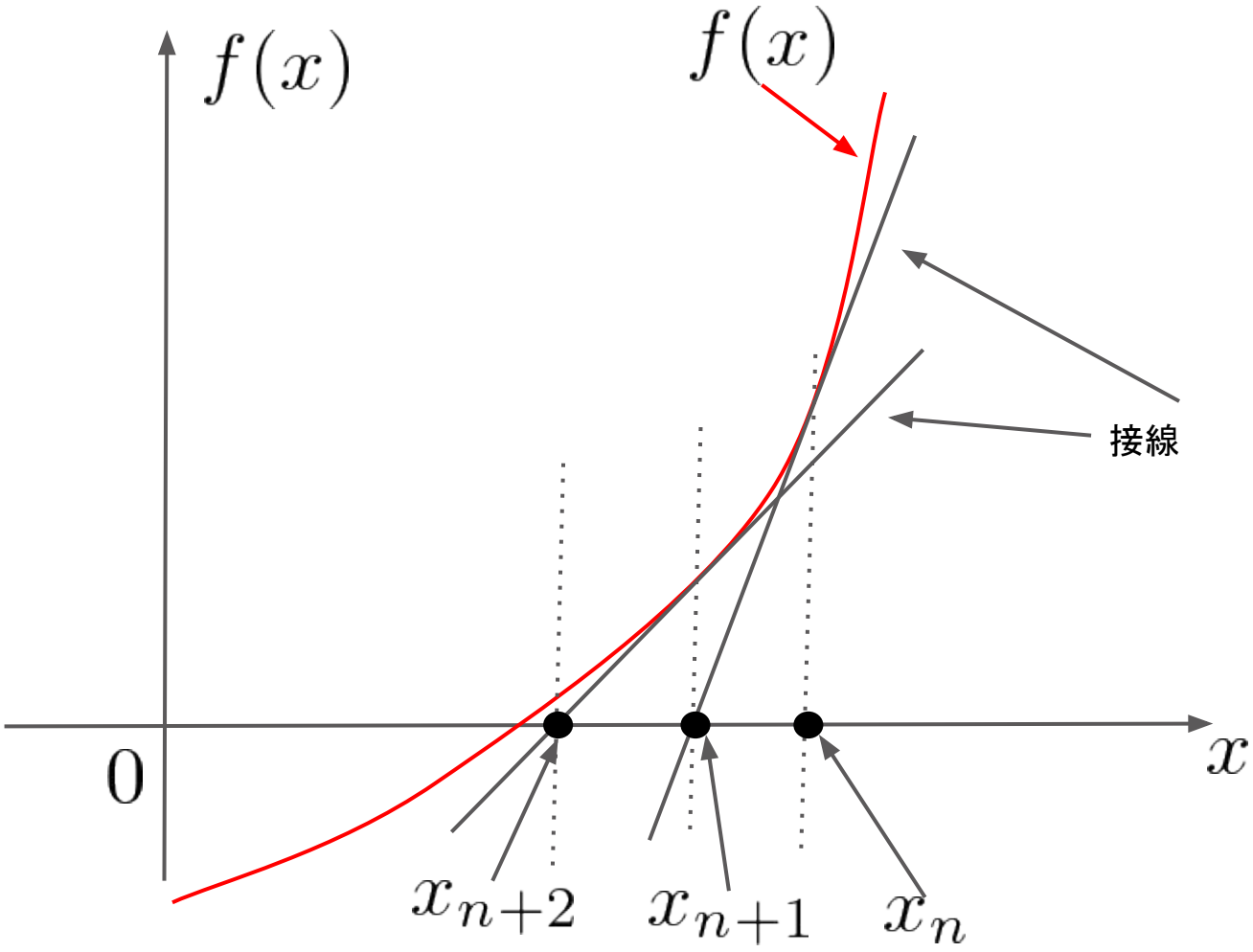
課題3 (自由課題)

ニュートン法を再帰で解く.
ニュートン法は数値計算によって方程式$f(x) = 0$の解を求める反復アルゴリズムである.

手順

1. 解に近いと思われる値を適当にとる
2. そこでのグラフの接線を考え，そのx切片を計算する. このx切片の値は解により近いものとなる. 再び2に行く.

課題3 (自由課題)



課題3 (自由課題)

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

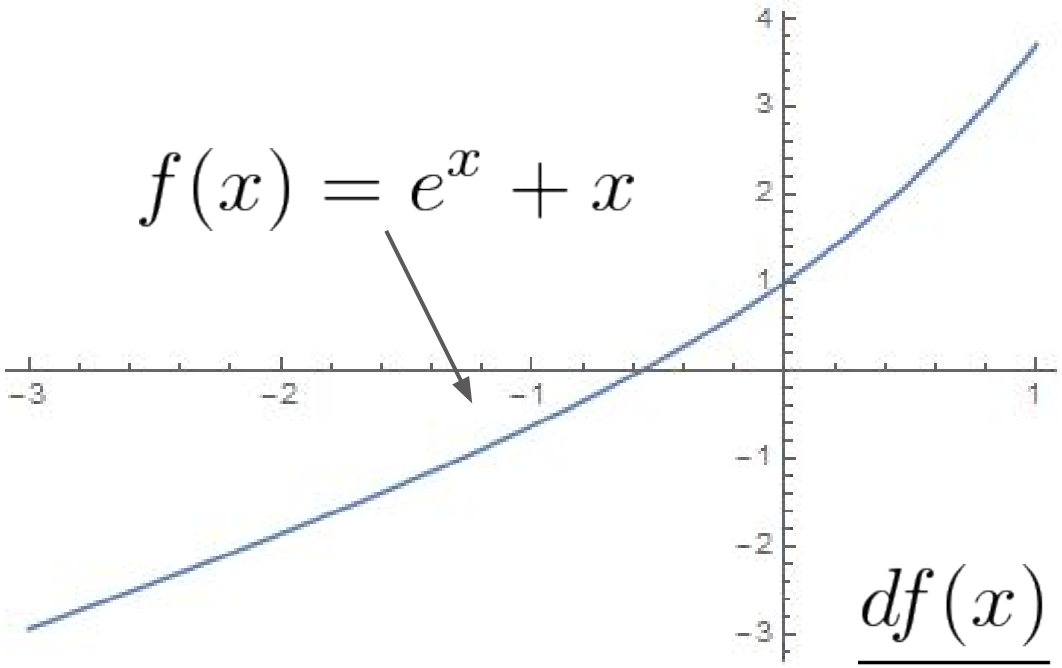
これは，点$(x_n, f(x_n))$における接線と，x軸との交点$(x_{n+1}, 0)$を求めている

- 再帰停止条件は $|x_{n+1} - x_n| < \varepsilon$
  - ε は適当に小さい値を選ぶ.
  - Cでの絶対値は**fabs**関数(**math.h**が必要)
    - コンパイルは**-lm**が必要

      gcc -Wall -o kadai0x kadai0x.c -lm
- 初期値 x_0 は，収束する値を適当に選ぶ.

課題3 (自由課題)

具体例として $f(x) = e^x + x$の $f(x) = 0$の解を求めてみよ.



$$\frac{df(x)}{dx} = e^x + 1$$

課題3 (自由課題)

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

double f(double);
double dfdx(double);
double newton(double);

int main() {
  double x0 = 1.0;
  double solution = newton(x0);
  printf("solution is %f\n", solution);
  return 0;
}
```

$f(x)$

$f'(x)$

newtonという関数は，まず x_{n}を受け取ってx_{n+1}を計算するように実装してみると分かりやすい

再帰を使っていればこのひな形を用いなくてもよい

初期値

課題4 (自由課題)

$\mathrm{mod\_apm}(a, p, m) = a^p \ \mathbf{mod}\ m$

の計算を行うプログラムを作成せよ．普通にプログラムしてしまうと，a^pで数が大きくなりすぎてオーバーフローしてしまうため，以下の関係式を用いて再帰関数で計算せよ．

$\mathrm{mod}(a^p, m) = \mathrm{mod}(a\ \mathrm{mod}(a^{p-1}, m), m)$

ここで

$\mathrm{mod}(a, b) = a\ \mathbf{mod}\ b$

課題4 (自由課題)

これを用いて,

$541^{1234} \bmod 1299709$

を計算してみよ(答えは487747).

(これは数字が大きいので, 数式の定義のまま計算することは出来ない)

課題一覧

- 課題1：必須課題
- 課題2：必須課題
- 課題3：自由課題
- 課題4：自由課題

課題提出方法

- 課題xの答えをkadai0x.cファイルに書く.
  - xは1,2,3,...
- 締め切りは日曜日の23:59
- ファイルをzipファイルにまとめる
  - ファイル名：学籍番号.zip
  - 例：学籍番号が641234の場合，641234.zipとする.
  - 圧縮方法はホームページに記載してある
- 学籍番号.zipファイルのみを提出する
  - 提出は https://goo.gl/QJ3LMP