プログラミング論I
(7) 標準関数

電子情報工学専攻　日浦 慎作

ログインしておいてください

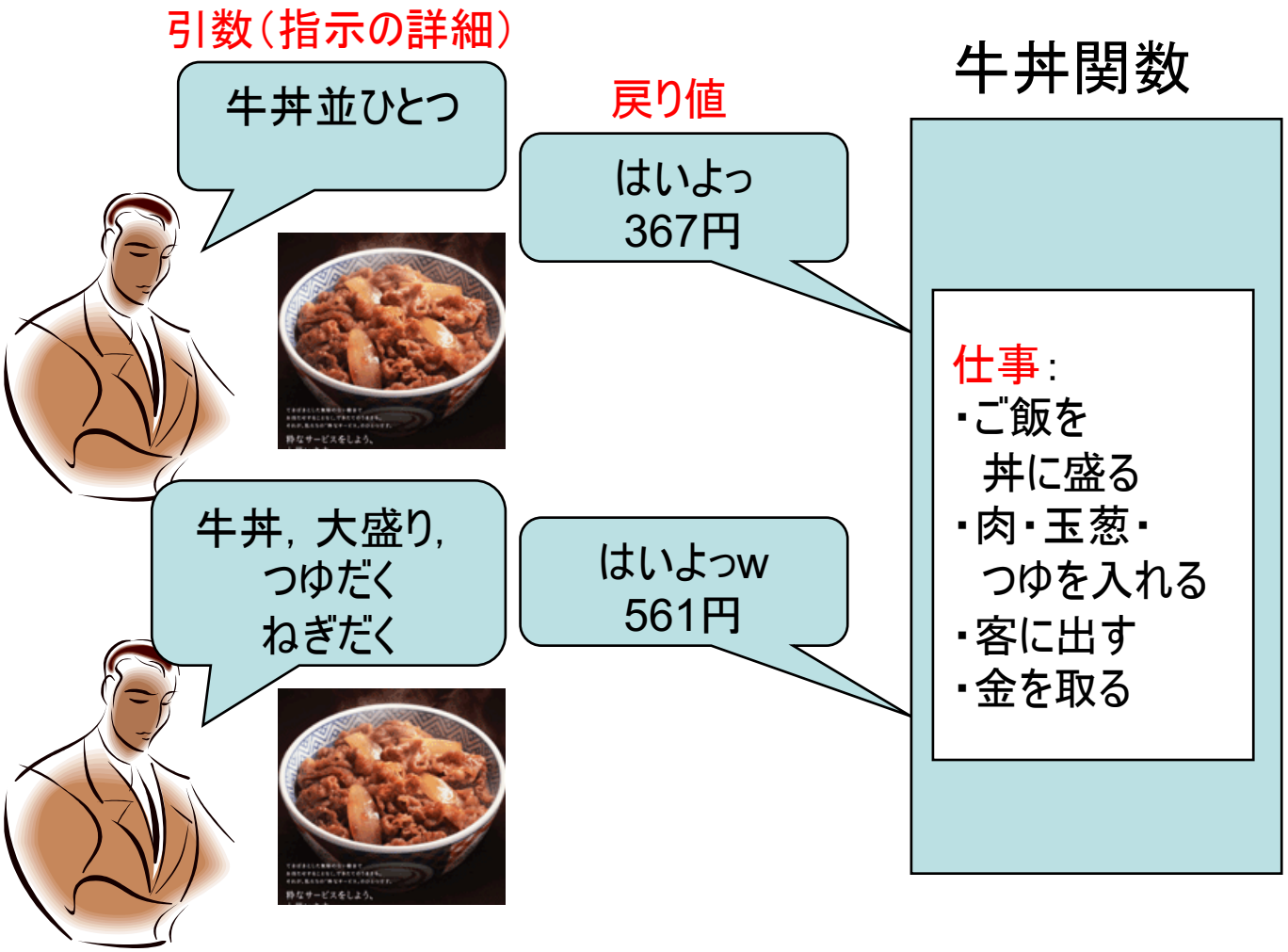
中間試験日程

- 再来週　11月26日　1限
- A棟A101教室（大きい教室）

- 前中先生の電気回路と連続

関数とは？

- 一連の仕事をひとつにまとめたもの



- 簡単な指示で複雑な仕事をさせることができる
  - 数学の「関数」は計算結果を返すだけだが，プログラミングでは計算以外の仕事もできる
- 数値を1つ，返すことができる（戻り値）
- 指示を出すほうは作業方法を知らなくてもよい
- あちこちから何度でも同じ仕事を依頼できる
- 少し違う要求を引数で指示することができ，それに応えることが出来る

C言語で使える関数

- これまでに習った関数

  `printf()`　　文字や数値を画面に表示する

  `scanf()`　　数値をキーボードから入力する

- もっと他にもあります
  - 今日の講義：いろんな関数の紹介と，使い方の説明
    - 三角関数，指数・対数・平方根，乱数など
  - 来週の講義：関数を自分で作る方法を学ぶ

標準関数ってなんだ？

- 標準Cライブラリ
  - C言語の標準規格で定められた，（ほぼ）どんな環境でも使うことができる関数などの集合
- この講義では
  - 数学関数（三角関数や指数など）
  - 乱数，絶対値
  - 日付，時間

  に関する関数を習います

数学関数(1)

- 【重要】プログラムの冒頭に，以下の1行が必要

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

- 例(三角関数 sin の場合)

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

ポイント

- 引数(上記のx)は，実数(double か float)
- 戻り値(上記のy)も，実数(double か float)
- xは，ラジアン単位(一周が2π)

数学関数(2)

- 用意されている関数
  - 三角関数 `sin(x) cos(x) tan(x)`
  - 三角関数の逆関数 `asin(x) acos(x) atan(x)`
  - 指数関数 `exp(x)` $= e^x$
  - 対数関数 `log(x)` $= \log_e x$, `log10(x)` $= \log_{10} x$
  - べき乗 `pow(x, y)` $= x^y$
  - 平方根 `sqrt(x)` $= \sqrt{x}$
  - 実数の絶対値 `fabs(x)` $= |x|$

すべて，引数も戻り値も実数です！

プログラム例（面積から円の半径を求める）

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

//円の面積から，半径を計算
int main(void) {
  double area, radius;

  printf("面積を入力してください:");
  scanf("%lf", &area);

  radius = sqrt(area / 3.14159);

  printf("面積 %f の円の半径は %f です¥n", area, radius);

  return 0;
}
```

実行結果

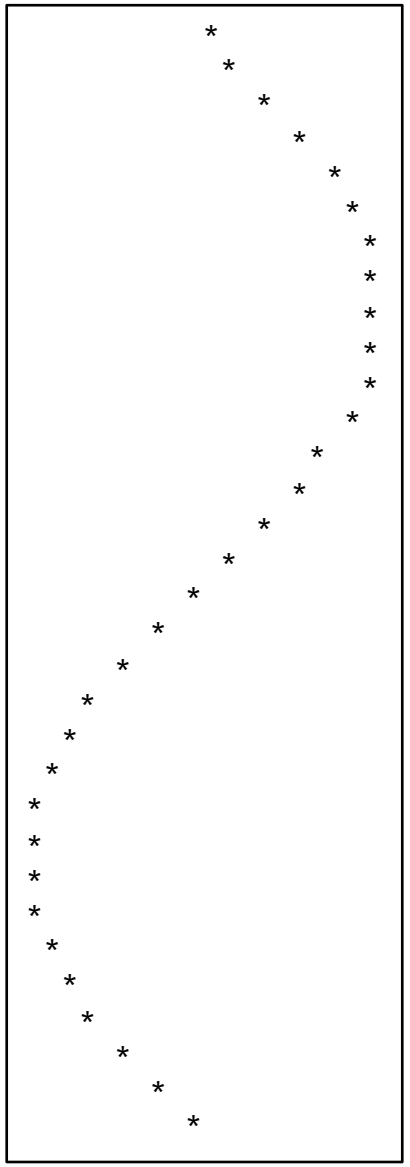
```
面積を入力してください:5
面積 5.000000 の円の半径は 1.261566 です
```

プログラム例（sin のグラフを描く）

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

int main(void) {
  double x;
  int space, i;

  for(x = 0; x < 3.14 * 2; x += 0.2) {
    space = (sin(x) + 1) * 10;
    for(i = 0; i < space; i++) {
      printf(" ");
    }
    printf("*¥n");
  }
  return 0;
}
```



実行結果

一般ユーティリティ関数(1)

- 【重要】プログラムの冒頭に，以下の1行が必要

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

- いろんな関数の雑多な詰め合わせ
  - プログラムを終了させる関数 `exit(x)`
  - 整数の絶対値 `abs(x)`
  - 乱数 `rand()` とその初期化 `srand(x)`

  など．他にもあるが，難しいものが多いので省略．

一般ユーティリティ関数(2)

- `exit(x)`

  いつでもプログラムを終了できる

  `exit(0);` 正常終了 `exit(1);` 異常終了

  （正常・異常は，他のプログラムと組み合わせない限り無意味なので，演習では `exit(0);` でよい．）

- `abs(x)`

  整数の絶対値．`fabs(x)` とは区別してください．

  `long` について使える `labs(x)` というのもある．

一般ユーティリティ関数(3)

- 乱数とは？

  予測が困難な数を計算で作り出す（疑似乱数）

- `rand()` の使い方
  - 引数なし `x = rand();`（カッコは省略不可）
  - 戻り値は整数 `0` ～ `RAND_MAX`（大きな整数）
- 剰余と組み合わせて使うことが多い

  `x = rand() % 6;`
  とすると，`0, 1, 2, 3, 4, 5` の6通りのどれか

乱数が乱数ではない？

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

int main(void) {
  int i;

  for(i = 0; i < 10; i++) {
    printf("%d¥n", rand() % 6);
  }
  return 0;
}
```

プログラム例（0～5の乱数を10個生成）

1回目	2回目	3回目
1	1	1
1	1	1
5	5	5
2	2	2
4	4	4
2	2	2
0	0	0
2	2	2
5	5	5
1	1	1

- 乱数は計算で作り出すので，毎回同じになる

乱数を毎回変えるには？

- 現在時刻を用いて乱数を初期化する

現在時刻の取得による乱数の初期化方法

- 【重要】プログラムの冒頭に，以下の1行が必要

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

初期化方法

```
srand((unsigned)time(NULL));
```

実行ごとに変化する乱数

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

int main(void) {
  int i;

  srand((unsigned)time(NULL));
  for(i = 0; i < 10; i++) {
    printf("%d¥n", rand() % 6);
  }
  return 0;
}
```

プログラム例（0～5の乱数を10個生成）

1回目	2回目	3回目
4	1	2
1	4	5
5	0	4
1	0	1
1	5	2
5	4	0
2	2	2
3	1	2
4	5	3
5	5	5

- 実行ごとに値が変わる

余談

- time() が返す値とは？
  - 1970年1月1日 00:00:00（世界標準時）からの経過秒数（UNIX時間とか，エポック秒という）
  - この値を年月日時分秒に変換する関数もあるが，使い方が難しいので，ここではやらない．
- さらに余談（2038年問題）
  - 2038年1月19日にエポック秒が 2^{31}-1 に達する
  - 32ビット符号付き整数（2の補数）に代入すると，負の数になる → バグるかも！怖い！！

関数の説明の読み方

- 関数について調べると，例えば

  ```
  double pow(double x, double y);
  ```

  のように書いてある．どういうことか？
- 意味
  - 前の double 戻り値の型が double である
  - カッコ内の double 引数の型が double である
  - 引数が2つある
    - 普通，関数は引数の数が決まっている．
      printf や scanf は例外．

double pow(double, double); という書き方もある．

関数呼び出し時の動作順序

- 次のプログラムの計算順序は？

```
double x = 2.0, y = 3.0, z;
z = sqrt(x + y);
```

1. x + y を計算する（値は 5.0 になる）
2. 関数 sqrt() を呼び出す.
3. sqrt(5.0)が実行され，5.0の平方根が計算される.
   1. 戻り値は √5 = 2.2360679 になる
4. 戻り値が変数 z に代入される

思い出してください！プログラムは，計算の手順を書いたもの.
変数の関係を決めたものではない.
数式に「実際の値を代入して計算」する作業しかできない.

演算子の優先順位

既に習った演算子

種類	演算子	結合規則	優先順位
関数，添字，構造体メンバ参照,後置増分/減分	() [] . -> ++ --	左→右	高
前置増分/減分，単項式※	++ -- ! ~ + - * & sizeof	左←右	
キャスト	(型名)		
乗除余	* / %	左→右	
加減	+ -		
シフト	<< >>		
比較	< <= > >=		
等値	== !=		
ビットAND	&		
ビットXOR	^		
ビットOR	|		
論理AND	&&		
論理OR	||		
条件	?:	左←右	
代入	= += -= *= /= %= &= ^= |= <<= >>=		
コンマ	,	左→右	低

a = -b;
のような，
正負を反転
するマイナス

※単項式とは演算子を適用する項が1つだけの式で、!(否定)、~(排他的論理和)、+(正)、-(負)、*(ポインタ)、&(アドレス)、sizeofが該当します

出典：https://www.grapecity.com/developer/support/powernews/column/clang/020/page05.htm

優先順位から考える

- 以下のプログラムはどういう意味か？

```
n = n + 1;
```

= の優先順位は + より低いので，優先順位としては

```
n = (n + 1);
```

ということになる

- よって，処理の順序は以下の通りとなる

1． n + 1 を計算する

2． その結果を n に代入する

（結果として，n の値が1増える）

数学の = とは違い，C言語の = は「右辺の値を左辺の変数に代入する」という動作を指示する命令（演算子）である．

もうちょっと練習

- 例1

  `if(a * 2 > 10 && b * 3 < 20) { … }`
  の赤字部分の優先順位をカッコで示せ.

  → `((a * 2) > 10) && ((b * 3) < 20)`

  `&&` の優先順位は比較演算子よりも低いので，最後に計算される

- 例2

  `a += 3 * -b + c;`
  の赤字部分の優先順位をカッコで示せ.

  → `a += ((3 * (-b)) + c)`

  ここの `-` は単項演算子（符号の反転）なので，`*` よりも
  優先順位が高い（先に計算される）

結合規則

- 同じ順位の演算子が続く場合，
  右から計算するか，左から計算するかを決める
- 例1

  ```
  a = b * 4 / c * d;
  ```

  → `a = (((b * 4) / c) * d);`

  `*` と `/` は同じ優先順位で，左から結合なので，左から計算される
- 例2

  ```
  a = b = c = 3;
  ```

  → `a = (b = (c = 3));`

  代入演算子 `=` は右から結合なので，右から代入される．

  代入演算子の値（計算結果）は代入された値そのものなので，
  この場合，`a, b, c` の全てに `3` が入る．

二項演算子と単項演算子

- 二項演算子とは　a　〇　b　の形式のもの
  - 加減乗除，比較，代入，論理演算など
- 単項演算子とは　〇a　や　a〇　の形式のもの
  - 符号の反転　–a　のほか，a++　（aを1増やす）も単項演算子
  - 識別子が関数であることを表す()　も単項演算子
    sqrt(2);　識別子　sqrt　の後ろに付く()は関数呼び出しを表す
- 前置の単項演算子，後置の単項演算子
  - 前置　-a　のように，前につけるもの
  - 後置　a++　や　sqrt(3)　のように，後ろにつけるもの

インクリメント ++ とデクリメント --

- ++ と -- は前置も後置もできる

  変数に1を加える(1を引く)ことは同じだが，式の値が異なる

  前置すると，変数の値を変更した後の値が式の値となる

  後置すると，変数の値を変える前の値が式の値となる

- 例

```
a = 1;
b = a++;
```

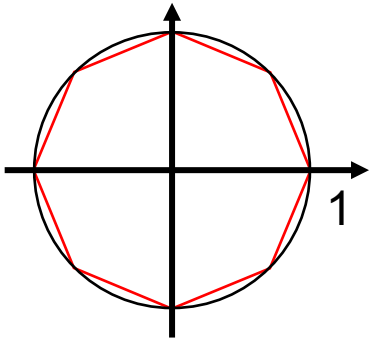
a に 1 を加える前の値が b に代入されるので，
実行後の値は a が 2，b が 1 となる

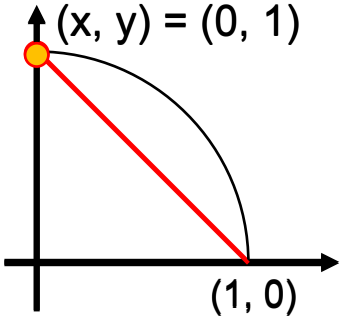
```
a = 1;
b = ++a;
```

a に 1 を加えた後の値が b に代入されるので，
実行後の値は a が 2，b も 2 となる

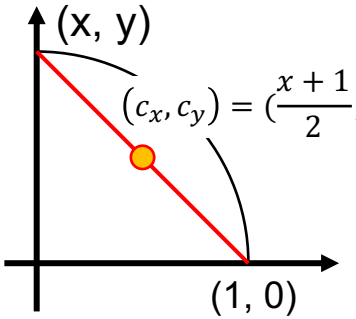
練習問題：円周率を求めてみよう！

- 円周率を求めるには
  - 半径1の円に内接する多角形の辺の長さを求めれば良い

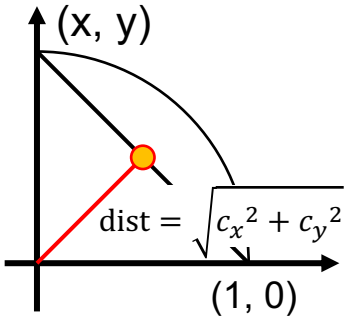




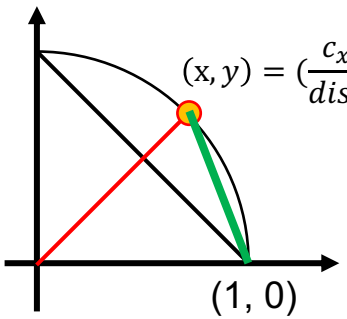
1. 正方形で近似し，
この辺を分割していく
一方の頂点を(x,y)とする



2. 辺の中点の座標
(cx,cy)を計算する



3. 中点と原点の距離
dist を計算する



4. (cx,cy)と原点を結んだ
直線と円の交点を求める

5. 繰り返す

- 辺の長さは，(x, y) と (1, 0) の間の距離 $l = \sqrt{(x-1)^2+y^2}$
- 正n角形を用いて求める円周率の近似値は，$(l * n)/2$

円周率を求めるプログラム

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

int main(void) {
  int n;
  double x = 0, y = 1, cx, cy, dist, pi; // step 1

  for(n = 8; n < 10000000; n*=2) {
    cx = (x + 1) / 2; // step 2
    cy = y / 2;       // step 2
    dist = sqrt (cx * cx + cy * cy); // step 3
    x = cx / dist; // step 4
    y = cy / dist; // step 4
    pi = (sqrt((x - 1) * (x - 1) + y * y) * n) / 2;
    printf("%.12f¥n", pi);
  }
  return 0;
}
```

```
3.061467458921
3.121445152258
3.136548490546
3.140331156955
3.141277250933
3.141513801144
3.141572940367
3.141587725277
3.141591421511
3.141592345570
3.141592576585
3.141592634339
3.141592648777
3.141592652387
3.141592653289
3.141592653515
3.141592653571
3.141592653585
3.141592653589
3.141592653590
3.141592653590
```

計算結果

- 円周率
  3.141592653589793..
- double を使うと15桁程度までしか計算できない（doubleが64bitと決まっているため）
- もっと精度の高い円周率を求めるには？
  - 多倍長演算（任意精度計算）を行う必要がある