情報通信工学演習 III

C 言語復習

濱本 雄治 [1]

情報工学部 情報通信工学科

2025 年 4 月 15 日

[1] hamamoto@c.oka-pu.ac.jp

問題 0

OS とコンパイラのバージョンを記せ

例 1:

- OS: macOS Sequoia 15.4
- コンパイラ: Apple clang version 17.0.0

例 2:

- OS: Chromium OS 19.0
- コンパイラ: gcc version 12.2.0

問題1

コマンドラインから複数の整数値を受け取り、その平均値と不偏分散を出力するプログラムを作成せよ。ただしコマンドライン引数の個数を変えても正しく動作するプログラムにすること。

実行例:

```
$ ./a.out 89 66 71 92 58 69 97 45 77
mean = 73.7778
var = 285.1944
```

コマンドライン引数 (教科書 p.228)

- コマンドラインからプログラムを実行する際に値を指定

```
$ ./a.out val1 val2 val3 ...
```

- main 関数の引数として argc、argv を定義

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

変数名	型	意味
argc	整数	引数の個数
argv[0]	文字列	プログラム名
argv[i]	文字列	i 番目の引数

- プログラム commandline.c

```
#include <stdio.h>
int main(int argc, char *argv[]){
  printf("%d\n", argc);
  for(int i=0; i!=argc; ++i)
    printf("%s\n", argv[i]);
}
```

- 出力

```
$ gcc commandline.c
$ ./a.out 1 2
3          # <= argc
./a.out    # <= argv[0]
1          # <= argv[1]
2          # <= argv[2]
```

文字列を数値に変換 (教科書 p.210)

- argv[i] は文字列なので、計算の際には数値に変換する必要

関数名	意味
atoi	文字列を整数に変換
atof	文字列を実数 (double) に変換

- プログラム convert.c

```
#include <stdio.h>
#include <stdlib.h> /* atof */
int main(){
  char s[] = "1.23";
  printf("%s\n", s);
  printf("%f\n", atof(s));
}
```

- 出力

```
$ gcc convert.c
$ ./a.out
1.23        # <= 文字列のまま出力
1.230000    # <= 実数に変換して出力
```

平均と分散

- 母平均

$$\mu \equiv \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} x_i$$

- 標本平均

$$\bar{x} \equiv \frac{1}{N} \sum_{i=1}^{N} x_i$$

- 母分散

$$\sigma^2 \equiv \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

- 標本分散

$$s^2 \equiv \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

- 標本分散の期待値

$$E(s^2) = \frac{N-1}{N}\sigma^2 < \sigma^2$$

- 不偏分散

$$\tilde{s}^2 \equiv \frac{N}{N-1} s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

問題2

2つのベクトル $\boldsymbol{a} = (103, 24, -32)^T, \boldsymbol{b} = (8, -55, -23)^T$ が与えられているとする。このとき外積 $\boldsymbol{a} \times \boldsymbol{b}$ を求めるプログラムを作成し、出力結果を示せ。ただし以下の条件を満たすこと。

- 各ベクトルの値は構造体 `struct vec3d` に格納する。
- 外積の計算は外部関数 `void vecprod` で行う。
- 計算結果は `main` 関数で標準出力に出力する。
- `main` 関数と関数 `void vecprod` は別々のファイルに作成し、分割コンパイルを行うことで実行ファイルを作成せよ。

構造体 (教科書 p.168)

- 変数をまとめて扱う
- 構造体の定義

```
struct MyStruct{
   メンバ変数の宣言;
};
```

- 構造体の初期化

```
struct MyStruct str = {val1, val2, ...};
```

- メンバ変数へのアクセス

```
str. メンバ変数名
```

プログラム struct.c

```
#include <stdio.h>
struct MyStruct{
  int i;
  double d;
  char s[10];
};
int main(){
  struct MyStruct str = {1, 2.0, "3.0"};
  printf("%d\n", str.i);
  printf("%f\n", str.d);
  printf("%s\n", str.s);
}
```

出力

```
$ gcc struct.c
$ ./a.out
1
2.000000
3.0
```

関数 (教科書 p.124)

- 関数の定義

```
戻り値の型 MyFunc(引数の宣言){
   変数と処理の宣言;
}
```

- 値渡し

```
MyFunc(var1);
```

  - var1 のコピーを渡す
  - 関数内の処理は元の var1 に影響しない

- アドレス渡し

```
MyFunc(&var2);
```

  - var2 のアドレスを渡す
  - 関数内の処理は元の var2 に影響する

- プログラム func.c

```
#include <stdio.h>
void add1(int *a){
  *a += 1;
}
int main(){
  int i = 1;
  printf("%d\n", i);
  add1(&i);
  printf("%d\n", i);
}
```

- 出力

```
$ gcc func.c
$ ./a.out
1
2
```

分割コンパイル (教科書 p.6)

- プログラムの開発効率や可読性の向上、コンパイル時間の短縮

- プログラム main.c

```
#include <stdio.h>
void hello(); /* プロトタイプ宣言 */
int main(){
  hello();
}
```

- プログラム hello.c

```
#include <stdio.h>
void hello(){
  printf("hello\n");
}
```

- コンパイル

```
$ gcc main.c hello.c
$ ./a.out
hello
```

問題 3

変数 x が範囲 $[-3.0, 3.0]$ を 0.1 間隔で変化するとき、x および関数

$$f(x) = \sin x + \frac{1}{9}\sin 3x + \frac{1}{25}\sin 5x + \frac{1}{49}\sin 7x$$

の値をテキストファイル `data_fx.txt` に出力するプログラムを作成せよ。さらに横軸を x、縦軸を $f(x)$ として結果をグラフとして図示せよ。離散点が分かるようにマーカーを表示すること。

Fourier 級数展開

- 任意の周期関数は三角関数に関する級数で展開可能

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx)$$

- 三角関数の規格直交性

$$\int_{-\pi}^{\pi} \cos nx \cos mx \, \mathrm{d}x = \int_{-\pi}^{\pi} \sin nx \sin mx \, \mathrm{d}x = \pi \delta_{nm},$$

$$\int_{-\pi}^{\pi} \cos nx \sin mx \, \mathrm{d}x = 0$$

- Fourier 係数

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx \, \mathrm{d}x, \qquad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx \, \mathrm{d}x$$

数学ライブラリ (教科書 p.205)

- 数学関数を利用するために math.h をインクルード
- コンパイル時に-lm オプションで libm をリンク

- プログラム func.c

```
#include <stdio.h>
#include <math.h>
int main(){
  printf("%f", 4*atan(1));
}
```

- 出力

```
$ gcc math.c -lm
$ ./a.out
3.141593
```

ファイル出力 (教科書 p.216)

- ファイル入出力の前後処理

```
FILE *f = fopen(ファイル名，オープンモード);
ファイル入出力;
fclose(f);
```

- プログラム func.c

```
#include <stdio.h>
int main(){
  int i = 1;
  double d = 2.0;
  FILE *fout = fopen("output.txt", "w");
  fprintf(fout, "%d %f\n", i, d);
  fclose(fout);
}
```

- 出力

```
$ gcc file.c
$ ./a.out
$ cat output.txt
1 2.000000
```

グラフの描画

- グラフ作成ソフトフェア (好きなものを用いてよい)
  - Google スプレッドシート
  - Excel
  - LibreOffice Calc
  - Gnuplot
- グラフを描画する際の注意
  - 縦軸と横軸の意味が分かるようにラベルをつけること
  - 縦軸と横軸に適切な目盛りをつけること
  - 離散点の位置をマーカーで明示すること

Gnuplot によるグラフ作成例

- output.txt

```
0.00 0.00
0.20 0.01
0.40 0.04
0.60 0.09
0.80 0.16
1.00 0.25
```

- Gnuplot の操作

```
$ gnuplot
gnuplot> set terminal postscript eps color
gnuplot> set output "output.eps"
gnuplot> set xlabel "x"
gnuplot> set ylabel "f(x)"
gnuplot> plot "output.txt" with linespoints
```

- 出力