演習問題

Koki Ukeba

October 21, 2024

はじめに

資料内の演習問題に対する, 参考コードは github.com の KokiUkeba にあります. 動作環境は gcc (Ubuntu 11.4.0-1ubuntu1 22.04) 11.4.0 です.

プログラミングの基本

文字列の出力

Q1.1

"Hello world"という文字列を出力するコードを作成せよ.

入力

出力

Hello world.

プログラミングの基本

文字定数

Q1.2

"\"?" を出力するコードを作成せよ.

入力

出力

\"?

プログラミングの基本

文字定数

Q1.3

16 進数の deadbabe を 10 進数で出力してください.

入力

出力

-559039810

プログラミングの基本

文字定数

Q1.4

16 進数の deadbabe を入力し,10 進数表示が-559039810 であることを確認せよ.

入力

deadbabe

出力

-559039810

プログラミングの基本

文字定数

Q1.5

10 進数の-559039810 を入力し,16 進数表示が deadbabe であることを確認せよ.

入力

-559039810

出力

deadbabe

変数〜繰り返しを添えて〜

Q-2

ライプニッツ級数

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

が $\frac{\pi}{4}$ に収束することを確かめてください．また, その計算結果と π とを比較してください.

入力

出力

calc = 0.7853956634
M_PI/4 = 0.7853981634
dif = 0.0000025000

演算

シフト演算子

Q2.1

"<<"及び">>"はシフト演算子と呼ばれ,

$$x << (>>)n$$

はxを左(右)へn[bit]シフトさせることを意味する.
この演算子を利用し, 入力した整数値aの2^b倍の値を出力するコードを作成せよ.

入力

1
5

出力

32

演算

モジュロ演算子, 否定演算子

Q2.2

"%" はモジュロ演算子と呼ばれ,

$$x\%y$$

はxをyで割ったときのあまりとなる.
"!" は否定演算子と呼ばれ,

$$!x$$

はxが0のときのみ1となる.
この演算子を利用し, 入力した整数値aがbを因数に持つときのみbを出力し,bを因数に持たない場合は0を出力するコードを作成せよ.

入力

1001
11

出力

11

演算

3 項演算子

Q2.3

"? :" は 3 項演算子と呼ばれ, ある式を $expr_i (i = 1, 2, 3)$ とすると,

$$expr_1 ? expr_2 : expr_3$$

は $expr_1$ が 0 ではない場合 (真の場合)$expr_2$ の値となり, そうでない場合は $expr_3$ の値となる.
この演算子を利用し, 整数値 a の絶対値を出力するコードを作成せよ.

入力

-18

出力

18

演算

ASCII

Q2.4

ASCII 文字セットのある 1 文字 (半角英語) を入力した際に, その文字の大文字小文字を変換するコードを作成せよ. 半角英語以外の値が入力されたときの処理は自由にしてよい.

入力

A

出力

a

演算

変数の型

Q2.5

浮動小数点型の数を入力し, その数の整数部と小数部を分けて表示するコードを作成せよ.

入力

3.141592

出力

3, 0.141592

条件分岐

Q3.1

実数 a,b,c が任意に与えられた際に, 二次方程式

$$ax^2 + bx + c = 0$$

の解を表示するコードを作成してください.(a=0 のときの処理は自由にしてください.)

入力

1
2
1

出力

$1.0x^2 + 2.0x + 1.0 = 0$
double root $-$ 1.0e + 00

条件分岐

出力例 2

$1.0x^2 + 1.0x + 2.0 = 0$
$-5.0e-01+I*1.3e+00, -5.0e-01-I*1.3e+00$

出力例 3

$0.0x^2 + 2.0x + 1.0 = 0$
single root $-5.0e-01$

出力例 4

$0.0x^2 + 0.0x + 1.0 = 0$
no solution

反復

for 文

Q4.1

任意の整数値を入力し, その 2 進数表記を出力するコードを作成せよ.

入力

70241

出力

00000000000000010001001001100001

反復

while 文

Q4.2

任意の正の整数値を入力したとき, その数が素数の場合は 1 を出力し, 素数でない場合は 0 を出力するコードを作成せよ.

入力

13

出力

1

反復

素数判定

Q4.3

1〜10000000 の間にある素数を全て出力するコードを作成せよ.
ただし実行時間が 1 分を超えるものは不正解とする.

入力

出力

(略)

Newton-method

Q-4

ニュートン法を用いて

$$f(x) = x^3 - 2x^2 - 3x + 3$$

の近似解を求めて下さい.
(初期値を 1 とし, 誤差の許容値 δ は 10^{-7} としてください.)
($f'(x)$ は自分で求めて大丈夫です.)
(ニュートン法の簡単な説明は次ページにありますが, 詳しい解説は各自で調べてください.)

入力

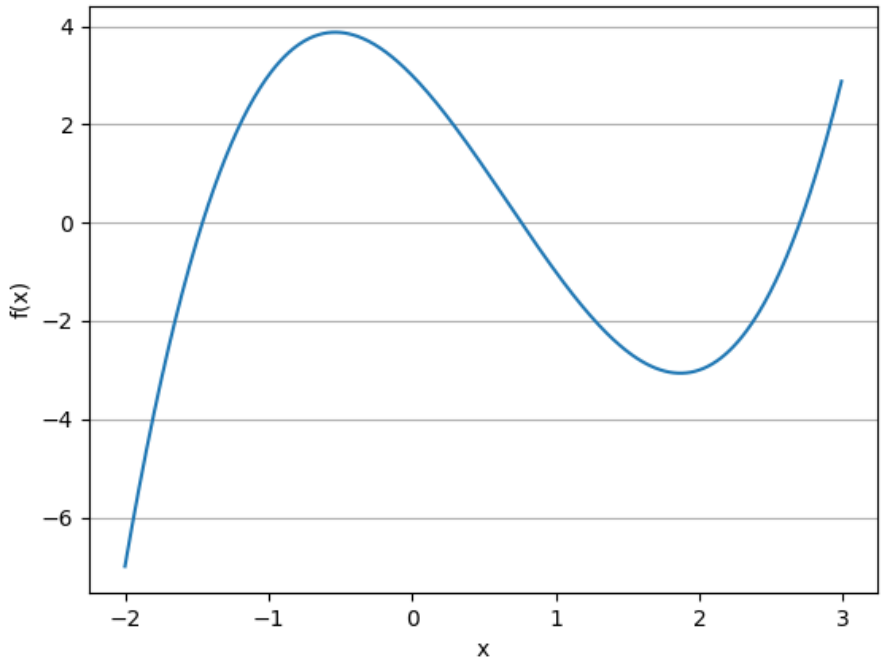
出力

7.608767e-01

ニュートン法

1. 初期値 x_a を求めたい解になるべく近い値にとる.
2. もし $|f(x_a)| \leq \delta$ の場合はこの x_a を近似解とする.
3. 上の条件が成り立たない場合は次の代入式により x_a を更新して 2. へ戻る.

$$x_a = x_a - \frac{f(x_a)}{f'(x_a)}$$



素因数分解

Q-5

変数に 2 以上の値を代入し, その数を素因数分解して得られる素数をすべて表示してください.
最終的に変数に 119028, 2146654199 の 2 数を代入し出力してください.

入力

出力例 1

```
429 =
3 ** 1
11 ** 1
13 ** 1
49238 =
2 ** 1
7 ** 1
3517 ** 1
```

出力例 2

```
7502751 =
3 ** 2
47 ** 1
17737 ** 1
5104981 =
7 ** 1
17 ** 1
42899 ** 1
```

出力例 3

```
9562 =
2 ** 1
7 ** 1
683 ** 1
515 =
5 ** 1
103 ** 1
```

ベクトルの内積

Q-6

大きさ N の double 型の配列 a,b は, すべての要素に値が入った状態で与えられているとします.

$$a = (1, 2, 3, \ldots, N)$$

$$b = (N, N-1, N-2, \ldots, 1)$$

さらに,$N = 1001$ とし, その内積となす角を求めよ.

ソートアルゴリズム

バブルソート (bubble sort)

Q7.1

バブルソートとは, 隣り合う要素の大小を比較しながら整列させるアルゴリズムである. これを C で実装せよ. 下に Python で実装したコードの一部を示した.

Listing 1: bubbleSort.py

```
1 data = []
2
3 for i in range(20):
4     data.append(randint(0, 100))
5
6 for i in data:
7     for j in range(len(data)-1, 0, -1):
8         if (data[j] < data[j-1]):
9             data[j], data[j-1] = data[j-1], data[j]
```

ソートアルゴリズム

選択ソート (selection sort)

Q7.2

選択ソートとは, 要素の最大値または最小値を探索し並び変えるアルゴリズムである. これを C で実装せよ.

Listing 2: selectionSort.py

```
1  data = []
2
3  for i in range(len(data)):
4      mini = i
5
6      for j in range(i + 1, len(data)):
7          if (data[j] < data[mini]):
8              mini = j
9
10     if (mini != i):
11         data[i], data[mini] = data[mini], data[i]
```

ソートアルゴリズム I

カウントソート (counting sort)

Q7.3

これを C で実装せよ.

Listing 3: countingSort.py

```
1  data = []
2  dmin = 0
3  dmax = 100
4  N = 100
5
6  for i in range(N):
7      data.append(random.randint(dmin, dmax))
8
9  mini = data[0]
10 maxi = data[0]
11
12 for i in range(len(data)):
```

ソートアルゴリズム II

カウントソート (counting sort)

```
13      if (data[i] < mini):
14          mini = data[i]
15      if (data[i] > maxi):
16          maxi = data[i]
17
18  countLen = maxi - mini + 1
19  count = []
20
21  for i in range(countLen):
22      count.append(0)
23
24  for i in range(len(data)):
25      count[data[i] - mini] += 1
26
27  index = 0
28  i = 0
29  while (index < len(data)):
30      if (count[i] != 0):
```

ソートアルゴリズム III

カウントソート (counting sort)

```
31              data[index] = i
32              index += 1
33              count[i] -= 1
34          else:
35              i += 1
```

ソートアルゴリズム

マージソート (merge sort)

Q7.4

これを C で実装せよ.

Listing 4: mergeSort.py

```
1  data = []
2
3  for i in range(len(data)):
4      mini = i
5
6      for j in range(i + 1, len(data)):
7          if (data[j] < data[mini]):
8              mini = j
9
10     if (mini != i):
11         data[i], data[mini] = data[mini], data[i]
```

ソートアルゴリズム

クイックソート (quiq sort)

Q7.5

これを C で実装せよ.

ソートアルゴリズム

ヒープソート (heap sort)

Q7.6

これを C で実装せよ.