2018 年度 情報領域演習第三
— 第 1 回 —

はじめに

本科目「情報領域演習第三」の目的は，前学期の「プログラミング通論」や今学期の「アルゴリズム論第一」において学習した内容を実践的なプログラミングを通じて習得することである．この演習では，最終的には具体例として国内の鉄道路線図のデータを用い，探索・整列や最短経路問題などに対するさまざまなアルゴリズムやデータ構造を実装することで，効率的な計算を行うためのプログラミング手法への理解を深めていく．

本演習ではプログラミング言語として C 言語を用いる．C 言語については，すでに「基礎プログラミングおよび演習」「情報領域演習第一・第二」「プログラミング通論」を通じて慣れ親しんできたと思うが，その知識を使いこなすための復習もしながら進めていくので，これまでついてこられなかった人も恐れずに挑戦してほしい．

第 1 回である今回は，出席の取り方やプログラムの提出方法の確認を主な目的とするが，簡単な C プログラム問題も用意したので締切りまでに提出すること．

出席やプログラムの提出について

課題ごとに出席確認やプログラムの提出を行うためのコマンドが用意されている．今回であれば，CED において以下のコマンド (青字の部分) を実行することで出席を表明できる．ただし，N は所属するクラス名 1，2，3 で置き換え，N の前には空白を入れないこと．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/01/checkerN
提出開始: 10 月  2 日 14 時 40分
提出締切: 10 月  9 日 14 時 39分
ユーザ: p1610999, 出席状況: 2018-10-02-14-58
 問題:    結果 | 提出日時           | ハッシュ値                        |
    1:  未提出 |                    |                                  |
    2:  未提出 |                    |                                  |
    3:  未提出 |                    |                                  |
    4:  未提出 |                    |                                  |
    5:  未提出 |                    |                                  |
```

上の出力結果からわかるように，このコマンドの実行によってその回の締切りや提出状況も確認することができる．クラス名を間違えると出席扱いにならないので注意すること．また，次回以降は授業の開始から必ず 30 分以内に実行しよう．

今回は 1 から 5 までの 5 問の問題を解くことになるが，提出も同じコマンドを用いて以下のように実行すればよい．ただし，N はクラス名 (1 から 3)，X は 1 から 5 のいずれかの問題番号であり，prog.c は提出する C プログラムのファイル名であり，X の前後には忘れずに空白を入れること．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/01/checkerN X prog.c
```

ここで，ファイル名として指定するのは，問題番号 X の問題を解く C プログラムのソースファイルであり，コンパイル済みの実行ファイルではない．ファイル名は特に指定しないが，「英数字からなる文字列.c」などとすることが望ましい．このコマンドを実行することにより，指定されたファイルがコンパイルされ，実行テストプログラムが自動的に起動される．コンパイルに失敗した場合にはエラーとなり，提出されたことにはならないので注意すること．

実行テストに成功すれば，以下のように「成功」と表示される．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/01/checkerN X prog.c
成功 (You win!)
```

実行テストに失敗すれば，以下のように反例とともに「失敗」と表示される．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/01/checkerN X prog.c
失敗
=================反例:入力==================
350
============================================
=================反例:出力==================
23
============================================
なんとか.in と，
かんとか.out に反例を書き出します．
```

反例というのは，提出したプログラムが正しく動作しないときの入力例のことなので，失敗した理由を見つけるための参考になる．ただし，今回の 1 問めのように入力のない問題の場合は反例は出力されない．

なお，実行テストで失敗すると「反例となる入力のファイルのパス (例:counterexamples/ce_1_xxx.in)」が表示される．反例というのは，失敗の原因となった入力であり，失敗の理由を見つけるヒントとなる．手元でコンパイルしたプログラム a.out に対して以下の例のように実行しよう．

```
[p1610999@blue00 ~]> ./a.out < counterexamples/ce_1_xxx.in
```

また，提出したソースコードの内容も見ることがあるので，テストに成功したからといってそれがそのまま得点になるとは限らない．また，複製などの不正は発覚した場合には，この科目だけでなく同じ学期に取得した全単位が無効になる．他人に見られないようにパーミッションを設定しておくこと．正しく提出できたか確認するためには以下のように出席の表明と同じコマンドを用いて確認できる．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/01/checkerN
提出開始: 10 月  2 日 14 時 40分
提出締切: 10 月  9 日 14 時 39分
ユーザ: p1610999, 出席状況: 2018-10-02-14-47
 問題:     結果 | 提出日時            | ハッシュ値                          |
    1:     成功 | 2018-10-02-14-58  | f9ee0dbf2f03aa88607bb0d7a47ffba8 |
    2:     失敗 | 2018-10-02-15-33  | 9b762c81932f3980cf03a768e044e65b |
    3:   未提出 |                   |                                  |
    4:   未提出 |                   |                                  |
    5:   未提出 |                   |                                  |
```

ハッシュ値は，提出した C プログラムのソースファイルの MD5 値である．CED では md5sum コマンドを用いて MD5 値を見ることにより，提出したファイルと手元のファイルが同じものであるかを確認することができる．

```
[p1610999@blue00 ~]> md5sum prog.c
9b762c81932f3980cf03a768e044e65b
```

なお，一度成功した問題に対して失敗するプログラムを送信すると，結果が「失敗」になってしまうので注意すること．

問題 1

標準出力に Hello, UEC! と出力するプログラムを作成せよ．

出力例

```
Hello, UEC!
```

問題 2

標準入力から与えられた整数 n に対し，標準出力に $n+1$ の値を出力するプログラムを作成せよ．ただし，入力として与えられる文字列は `int` 型の整数を表すものとする．

入力例

```
2018
```

出力例

```
2019
```

問題 3

標準入力から与えられた文字列をそのまま標準出力に出力するプログラムを作成せよ．ただし，入力となる文字列は 128 文字以下であると仮定してよい．

入力例

```
%\ca[,aD0ABD,FcFc!-cf.$C\C9
```

出力例

```
%\ca[,aD0ABD,FcFc!-cf.$C\C9
```

問題 4

標準入力から与えられた 2 つの整数に対し，それらの最大公約数を標準出力に出力するプログラムを作成せよ．ただし，入力として与えられる文字列は 2 つの `int` 型の非負整数が空白で区切られた文字列である．

入力例

```
777 296
```

出力例

```
37
```

問題 5

標準入力から与えられた 10 個の整数値のうち，2 番めに大きい数値を標準出力に出力するプログラムを作成せよ．ただし，入力として与えられる文字列は 10 個の `int` 型の整数が空白で区切られた文字列であり，最大の整数が 2 つ以上含まれる場合にはそれを 2 番目に大きい値とする．

入力例

```
3 -10 1 7 9 4 5 0 3 -2
```

出力例

```
7
```