プログラミングおよび実習Ⅱ・第 1 回

2003 年 9 月 24 日

科目概要	ソーティングや検索などの基本的なアルゴリズムを計算機上で実行するためのプログラミング技法について学習し、実習を通して実際のプログラミングに応用する能力を養います。また、実用的なプログラムを作成するための開発技法、テスト技法、完成したプログラムの実行速度などの評価法について考えます。
身につくポイント	プログラミング能力、論理的思考能力、アルゴリズムに対する基本的な理解。
授業方法	毎週 2 講時 (講義 1 講時と実習 1 講時) の授業が、 2 つのクラスに分かれて開講されます。クラス分けについては、学期始めに案内しますので掲示に注意しておいてください。受講者が「プログラミングおよび実習 I」と「データ構造とアルゴリズム I」をすでに受講済みであること、また「データ構造とアルゴリズムⅡ」をすでに受講したか、あるいはこの科目と並行して受講中であることを前提として授業を行います。講義部分では、教科書や配布資料を使って、基礎となるアルゴリズムと実現の方針、必要となるプログラミング技法を解説した後、プログラミングの演習問題が出題されます。実習部分では、講義部分で解説した例題や演習問題のプログラムを実際に計算機上で動かしてみることで理解を深めます。また、ティーチングアシスタントの助言を得ながら、作成したプログラムをよりよいものへ変更していきます。
成績評価	演習問題 (小テスト) の解答、実習で作成したプログラム、定期試験 (筆記) の結果で総合的に評価します。詳細は授業中に明示します。
授業計画	(1) 最大値、最小値、平均値、標準偏差の計算 (2) 選択ソートと実行トレース (3) ソーティングプログラムのテスト (4) バブルソートとアサーション (5) クイックソート (6) 処理時間の測定とバッチファイル (7) 実験データの整理 (8) 分割コンパイルと make (9) ファイル入出力 (10) 線形探索 (11) 2 分探索 (12) テキスト処理 (13) ハッシュ法
系統的履修科目	プログラミングおよび実習 I、データ構造とアルゴリズム I、データ構造とアルゴリズムⅡ
テキスト	C 言語によるアルゴリズムとデータ構造入門 (東野勝治ら著、森北出版) 効率よく学ぶ C プログラミング (宇土顕彦著、コロナ社)
参考文献	アルゴリズムとデータ構造 (茨木俊秀著、昭晃堂) コンピュータ・アルゴリズム (小澤孝夫著、昭晃堂) プログラミング言語 C (第 2 版) (カー二ハン、リッチー著、石田晴久訳、共立出版)

「プログラミングおよび実習Ⅱ」の中野クラスでは次のように運営していく予定です。ただし、今後の状況に合わせてやり方を変えていくかも知れません。

水曜日 3 講時 4 号館 107 教室 — 講義と紙の上での演習　配布したプリントを使って紙の上でのプログラミングの演習を行います。 C に関するの教科書あるいは参考書を必ず持参するようにしてください。このような演習と平行して、プログラミングにおけるいろいろな考え方や技法について中野が解説します。また、アルゴリズムとデータ構造 I、Ⅱで学んだ各種のアルゴリズムの復習を行うこともあります。

水曜日 4 講時 1 号館 542 実習室 — 計算機を使った実習　配布したプリントの「演習問題」に対して、計算機上でプログラミングを行います。 TA の皆さんが受講者の質問等に答えてくれます。各自のホー

[1] 中野クラスでは `http://www602.math.ryukoku.ac.jp/~nakano/Prog2/index.html` から入手できる配布資料を中心に講義と実習を進めていきます。

ムディレクトリに Prog2 というディレクトリを作り、そこにできあがったプログラムのソースファイルを置いていただきます。

1 基本的なプログラミングの復習

1.1 最大値を求める

標準入力 (キーボード) からいくつかの整数をつぎつぎと読み込んで、最後にそれらの最大値を出力する C プログラム prog1-1.c を作ってみます。ただし、整数でない入力や、EOF (ファイルの終り)[1]を検出したら、そこで入力を終了するものとします。また、整数が 1 つも入力されなかった場合は、プログラムは何も出力せずに終了するものとします。

このようなプログラムを作成するための実現の方針やアルゴリズムは以下のようなものとなります。

実現の方針 (1) 常に「これまでの最大値」を保持しておく。

(2) 最初のデータを読み込んで、その値を「これまでの最大値」とする。

(3) その後、新しくデータを読み込む度に「これまでの最大値」と読み込んだデータを比較し、必要なら「これまでの最大値」を更新して行く。

(4) すべてのデータを読み終えた段階で「これまでの最大値」を全体の最大値として出力する。

アルゴリズム (1) 整数型変数 max と new を用意する。

(2) 入力された整数を max に読み込む。読み込めなければ (もう入力がなければ) プログラム全体を終了。

(3) 次の 2 つの作業を順に繰り返す。

(3-1) 入力された整数を new に読み込む。読み込めなければ (もう入力がなければ) 繰り返しを終了する。

(3-2) new と max を比較し、もし、new > max が成り立っていれば、

(3-2-1) new の値を max へ代入する。

(4) max の値を求める最大値として出力する。

ポイント (1) scanf や printf などの標準入出力ライブラリ関数を利用するためには、プログラムの冒頭に次の 1 行が必要となる。

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

(2) プログラム全体は main 関数。 その骨格は

```
int main ()
{
```

[1]Unix (Linux) でのキーボードからの入力では、 Control-D (コントロールキーを押しながら D のキー) を押すと、それがファイル (入力) の終りと見なされます

```
        ⋮
    return 0;
}
```

(3) 整数型変数は int で宣言。

(4) 入力された整数を変数に格納するには scanf を使う。整数 (int 型) なので入力書式文字列は "%d" となる。変数の前には & が必要。scanf は書式文字列中の % に対応して実際に読み込んだデータの数を返すので、scanf("%d", &...) が 1 を返したかどうかで、整数が入力されたかどうかを判定できる。整数でない入力の場合には scanf は 0 を返すし、標準入力 (キーボード) から文字を全く読み取れなかった場合には EOF を返す。

(5) 値の比較による条件付き実行には if 文を使えばよい。

(6) プログラム全体の終了には関数 exit を呼び出せばよい。関数 exit を使うためにはプログラムの冒頭に次の 1 行を書いておく。

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

プログラムが正常に終了した場合は、exit(EXIT_SUCCESS); のように関数 exit の引数は EXIT_SUCCESS とし、何らかの異常終了なら、引数を EXIT_FAILURE として呼び出す[2]。どの関数が実行中でも exit が呼び出されるとプログラム全体が終了する。関数 main の中でプログラムを終了させる場合は return 文を使うこともできる。

(7) 繰り返しの処理には while 文を使う。while (1) { ... } のようにとりあえず無限ループにしておいて、適宜 break 文を使って繰り返しを終了させることもできるし、

```
while (scanf("%d", &...) == 1) { ... }
```

のようにすることもできる。

(8) 結果の値の出力には printf を使えばよい。

prog1-1.c のプログラム例

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

int main ()
{
    int max, new;

    printf("整数を入力してください => ");
    if (scanf ("%d", &max) != 1)
        exit(EXIT_SUCCESS);

    while (1) {
```

[2] stdlib.h の中で、EXIT_SUCCESS は 0 に、EXIT_FAILURE は 1 に、それぞれ #define によってマクロ定義されていることが多い。

```
        printf("整数を入力してください => ");
        if (scanf("%d", &new) != 1)
            break;
        if (new > max)
            max = new;
    }
    printf ("最大値 = %d\n", max);
    return 0;
}
```

1.2 もう 1 つの考え方

prog1-1.c と同じことをするプログラム prog1-2.c を、次のような実現方針とアルゴリズムにしたがって作ることもできます。

実現の方針 (1) 基本的には prog1-1.c のものと同じ。ただし、まだ 1 つもデータを読み込んでいないのか、既に (少なくとも 1 つは) データを読み込んだのか、そのどちらの状態にあるのかを保持しておく。

(2) 次々にデータを読み込んで、

(2-1) もしまだ 1 つもデータを読み込んでいなかったのならなら、読み込んだデータを「これまでの最大値」とする。

(2-2) そうでない場合は「これまでの最大値」と読み込んだデータを比較して、必要に応じて「これまでの最大値」を更新する。

ポイント (1) プログラムが、ある状態にあるのかないのかを記憶しておくための変数を「**フラグ** (旗)」と呼ぶ。フラグは、真か偽かいずれかの値 (真理値) を格納する論理型の変数と考えられる。フラグの値が真である (真にする) ことを「フラグが立っている (を立てる)」と言い表し、偽であることを「フラグが下りている (を下ろす)」と言い表す。

(2) C には真理値を表すデータ型は特に用意されていないので、int などの整数型で代用する。その際、 0 で偽を、 0 以外の値 (特に 1) で真を表現する。論理演算子として ! (否定)、&& (論理積)、|| (論理和) が用意されている。

アルゴリズム (1) 整数型変数 max と new を用意する。また、論理型変数 first を用意し、真 (1) に初期化しておく。

(2) 次の 3 つの作業を順に繰り返す。

(2-1) 入力された整数を new に読み込む。読み込めなければ (もう入力がなければ) 繰り返しを終了する。

(2-2) もし first が真であれば

(2-2-1) new の値を max へ代入する。

(2-2-2) first へ偽 (0) を代入する。

(2-3) そうでなくて、もし new > max が成り立っていれば、

(2-3-1) new の値を max へ代入する。

(3) first が真でなければ

(3-1) max の値を求める最大値として出力する。

prog1-2.c のプログラム例

```
#include <stdio.h>

#define TRUE    1
#define FALSE   0

int main ()
{
    int max, new, first;

    first = TRUE;
    while (TRUE) {
        printf("整数を入力してください => ");
        if (scanf("%d", &new) != 1)
            break;
        if (first) {
            max = new;
            first = FALSE;
        }
        else if (new > max)
            max = new;
    }
    if (!first)
        printf ("最大値 = %d\n", max);
    return 0;
}
```

1.3 実力チェック

下の実現方針にしたがって、標準入力 (キーボード) からいくつかの整数を順に入力すると、それらの最小値、最大値、平均値を次の実行例のように出力する C プログラム prog1-3.c 作りなさい。ただし、平均値は小数点以下 2 桁まで出力しなさい。また、整数が 1 つも入力されなかった場合には、プログラムは何も出力せずに終了するようにしなさい。

prog1-3.c の実行例

```
s1542h017% ./prog1-3
整数を入力してください = 456
整数を入力してください = 345
整数を入力してください = 87
整数を入力してください = 234
整数を入力してください = 652
整数を入力してください = 45
整数を入力してください = .
最小値 = 45, 平均値 = 303.17, 最大値 = 652
s1542h017% ./prog1-3
整数を入力してください = 45
整数を入力してください = .
```

```
最小値 = 45, 平均値 = 45.00, 最大値 = 45
s1542h017% ./prog1-3
整数を入力してください = .
s1542h017%
```

実現の方針 (1) 基本的には「1.2 もう 1 つの考え方」の節と同じ。

(2) 「これまでの最大値」の他に、プログラム中に「これまでの最小値」、「これまでのデータの総和」、「これまでのデータの個数」を保持しておく。

(3) 新しくデータを読み込む度に、必要に応じてこれら 4 つの値を更新していく。最初のデータを読み込んだときには、その値をそのまま「これまでの最小値」と「これまでの最大値」としなければならないことに注意。

(4) データが 1 つも入力されなかった場合には何も出力されないようにする。

アルゴリズム (1) 整数型変数 max、min、sum、new、num を用意しておく。

(2) sum と num を 0 で初期化する。

(3) 次の 5 つの作業を順に繰り返す。

(3-1) 入力された整数を new に読み込む。読み込めなければ (もう入力がなければ) 繰り返しを終了する。

(3-2) もし num が 0 であるか new < min であれば new の値を min へ代入する。

(3-3) もし num が 0 であるか new > max であれば new の値を max へ代入する。

(3-4) sum に new の値を足し込む。

(3-5) num の値を 1 増やす。

(4) num が 0 より大きければ、min の値、sum を num で割った値、max の値を、それぞれ求める最小値、平均値、最大値として出力する。

ポイント (1) 「これまでの最小値」、「これまでの最大値」、「これまでのデータの個数」、「これまでのデータの総和」は int 型の変数で扱えばよい。

(2) 平均値は整数型というわけには行かないが、例えば、int 型の変数 num と sum にそれぞれデータの個数と総和が格納されているのなら、

```
printf ("%.2f\n", (double)sum/num);
```

で、平均値を小数点以下 2 桁で出力することができる。

(3) この (double) は (その右に書かれた式の) 値のデータ型を変換する働きをする単項演算子で、(double)sum とすると、int 型の sum の値が double 型 (倍精度浮動小数点型) に変換される。このように型の変換を行うの演算子を**キャスト演算子**と呼び、一般に「(型指定)」の形式で書かれる。

(4) 先の例の (double)sum/num という式は、double 型の値を int 型の値で割った商を表すこととなるが、double 型と int 型の間での四則演算では、int 型の方の値が自動的に double 型に変換されるので、期待した通りの平均値 (double 型) を得ることできる。もちろん、(double)sum/num を (double)sum/(double)num や sum/(double)num と書いても結果は同じになる。

(5) printf の書式文字列中の "%.2f" は、double 型の値を小数点以下 2 桁までの書式で表示することを指示している。小数点以下 3 桁までの表示が必要なら "%.3f" とすればよい。

2 演習問題

実習室では、まず、次のように端末ウィンドウで /home/sample/mprog2/setup と rehash の 2 つのコマンドを実行して、「プログラミングおよび実習Ⅱ」実習環境の初期設定を行ってください。

実習環境の初期設定

```
s1542h017% /home/sample/mprog2/setup

プログラミングおよび実習IIの環境を設定しました。

s1542h017% rehash
s1542h017%
```

この初期設定で、各受講者のホームディレクトリに Prog2 というディレクトリが作られます。「プログラミングおよび実習 II」の演習問題のプログラムは、必ずこの Prog2 ディレクトリに作成するようにしてください。

初期設定は、次回以降は行う必要はありません。初期設定が終ったら、次の 2 つの課題に取り組みましょう。

2.1 prog1-3.c

「1.3 実力チェック」のプログラム prog1-3.c を実際に計算機上で作成、コンパイル、実行して、正しく動作することを確認しなさい。

プログラムが完成したら、次のように mprog2 コマンドを実行して、自分が作成したプログラムを提出してください。

prog1-3.c の提出方法

```
s1542h017% mprog2 prog1-3.c
prog1-3.c をチェックしています。 しばらくお待ちください ...

プログラム prog1-3.c を受理しました ... これからの課題も頑張ってください。

s1542h017%
```

「プログラム prog1-3.c を受理しました」が表示されれば ok です。もし、何らかの不具合を指摘された場合は、自分のプログラムを修正して、コンパイル、テストし、もう一度 mprog2 コマンドで提出してください。

2.2 prog-1.4c

prog1-3.c を prog1-4.c にコピーし、(整数だけでなく) 小数部のある数値を入力データとして扱えるように書き換え、コンパイル、実行して、正しく動作することを確認しなさい。ただし、入力を促すプロンプトは「数値を入力してください => 」と変更し、最小値、最大値、平均値はそれぞれ小数点以下 3 桁まで出力するようにしなさい。

ポイント (1) 小数部を持つ数値は double 型で扱うことができる。たとえば、

```
scanf("%lf", &new);
```

とすることで小数部を持つ数値 (もちろん整数も) を double 型の変数 new へ読み込むことができる。scanf の入力書式文字列は "%f" ではなくて "%lf" であることに注意。

プログラムが完成したら、prog1-3.c のときと同様に、mprog2 コマンドを実行して、自分が作成したプログラムを提出してください。

3 次回の予定

1. 標準入力 (キーボード) からいくつかの小数部付きの数値を読み込んで、それらの、平均値と標準偏差を出力するようなプログラムを作ります。入力されたデータが $x_1, x_2, x_3, \ldots, x_n$ で、それらの平均が m であった場合、これらのデータの標準偏差 σ は

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - m)^2}{n}}$$

で定義されます。

2. 選択ソートと呼ばれる整列アルゴリズムのプログラムを考えます。