プログラミング演習Ⅰ　レポート

該当する授業の数字を記入する

課題名：　課題１　階乗を求めるプログラム

課題名を記入

学籍番号	t032900
氏名	宇都宮　太郎
提出日	平成１６年４月１日
書式修正版提出日	平成　　年　月　日

学籍番号，氏名，提出年月日を記入する．学籍番号には小文字の半角英数字を使用し，チェックディジットはつけない．

書式修正の指示があった場合，修正版の提出日を記載する．

【チェック項目】 ――

以下の項目が正しく記載されているか確認し，○をつけること．※がついているものは必須項目ではない．

項目	自己チェック	担当記入欄	
		判定	コメント
表紙に必要事項を記入したか？	○	○	
課題の目的と概要を記述したか？	○	○	
データ構造について記述したか？	○	○	
アルゴリズムについて記述したか？	○	○	
ソースリストと説明を記述したか？	○	○	
ソースの書式は見やすいか？	○	×	リスト２はインデントがおかしい
実行結果とその説明を記述したか？	○	○	
考察を記述したか？	○	○	
※オプション課題を記述したか？			
※感想・意見を記述したか？	○	○	
参考文献の書式は正しいか？	○	○	
図表番号とタイトルをつけたか？	○	○	
ページ番号をつけたか？	○	○	
フローチャートの書き方は正しいか？	○	○	

記載漏れが無いか，提出前に自分で確認し，○を入れる．

担当記入欄に，判定と担当からのコメントが記入される．または，e-Learning システム上で指示される．

【教員記入欄】 ――

教員記入欄は，教員がコメントや評価を記入する際に使用するので，空欄にしておくこと

1. 課題の目的と概要

1.1 目的

階乗を求めるプログラムを作成することで，関数の使い方と再帰について理解する．

1.2 課題の概要

階乗とは，自然数 n に対し，1 から n までの自然数の総乗をいう．これを $n!$ と書く．

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 1 \qquad (1)$$

先の定義では $0!$ は定義されないが，便宜上 $0! = 1$ とされる．これは，$(n\text{-}1)! = n!/n$ であるので，$0! = 1!/1 = 1$ と考えられるためである．

(1)式は，(2)式のように表すことができる．

$$n! = \begin{cases} n \times (n-1)! & (n > 0) \\ 1 & (n = 0) \end{cases} \qquad (2)$$

本課題では，(2)式を再帰呼び出しを用いて実装し，n の階乗を計算するプログラムを作成する．入力は，定数宣言部で定義した回数分受付ける．

2. データ構造

本プログラムで使用したデータ構造を以下に述べる．

(1) 定数宣言

`MAX_INPUT` ：入力回数

(2) 型宣言

なし

(3) グローバル変数

なし

章の番号およびタイトルのフォントは MS ゴシック 12 ポイントを推奨

節の番号およびタイトルは，章タイトルよりやや小さく本文よりやや大きいものが読みやすい

本文は MS 明朝 10.5 ポイントを推奨

3. アルゴリズム

3.1 全体の概要

プログラム全体の動作のフローチャートを図 1 に示す．このプログラムでは，まずデータの入力を促し，整数型の変数 data に格納する．data が正の場合は，再帰呼び出しを用いて data の値の階乗を求め，結果を表示する．負の場合はエラーメッセージを出力する．階乗を計算する処理は，MAX_INPUT で定義された回数分繰り返す．

このように，図についての説明文を本文中に記載する．

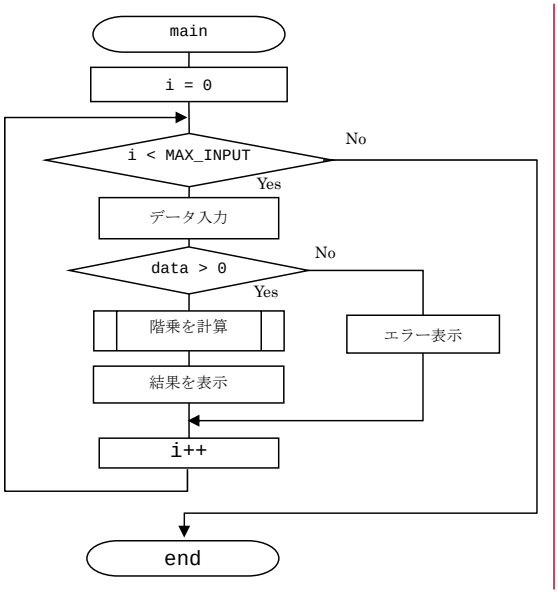

図 1 メイン部のフローチャート

図には図の下に番号とタイトルをつける

3.2 関数の説明

int factorial(int n)

引数 n： int 型．階乗を計算する値を表す．

戻り値 int 型．n の階乗を計算した結果．

関数 factorial は n の階乗を求める関数である．(2) 式の考え方に従って，再帰的に階乗の値を計算する．再帰的に階乗を計算するには…

4. ソースリスト

作成したプログラムのソースリストをリスト 1 に示す．プログラムの説明は，ソースリストの中にコメントとして記載した．

リスト１．階乗を求めるプログラム

```
 1: #include <stdio.h>
 2: /*** 定数宣言 ***/
 3: #define MAX_INPUT 3 // 入力回数
 4:
 5: /******************
 6: n の階乗を求める関数
 7: 入力 : 整数型
 8: 戻り値 : 整数型
 9: ***************** /
10: int factorial(int n)
11: {
12:     if (n > 0) {
13:         return n * factorial(n - 1);
14:     } else {
15:         return 1;
16:     }
17: }
18:
19: /******************
20: メインルーチン
21: ***************** /
22: int main(void)
23: {
24:     int data;   // 入力データ
25:     int i;      // ループカウンタ
26:
27:     for (i = 0; i < MAX_INPUT; i++) {
28:         // データの入力
29:         printf("[ %d 回目 ] 正の整数を入力して下さい ", i + 1);
30:         scanf("%d", &data);
31:
32:         // 結果の表示
33:         if (data > 0) {
34:             printf("計算結果 : %d! = %d¥n", data, factorial(data));
35:         } else {
36:             printf("error:入力値が不正です¥n");
37:         }
38:     }
39:     return 0;
40: }
```

リストの番号とタイトルをつける．

ソースリストのフォントは，等幅フォントを使用する．Courier New または MS ゴシックを推奨．フォントの大きさは 8 ポイント程度が読みやすい．行間は最小値，間隔は 0 pt が読みやすい．行間の設定はツールバーの「書式(O)」→「段落(P)」で行う．

Tabを使用してインデントし，見やすくする．タブ幅は 4 を推奨．きちんと Tabが入っているかどうかは，ツールバーの「ツール(T)」 →「オプション(O)」の「表示」にある編集記号の表示の“タブ”にチェックを入れれば表示され，確認できる．また，空行や空白を効果的に使用すると，処理の単位が分かり，読みやすくなる．

プログラムの説明を，コメントとして記入しておくと，後でソースを読み直したときに分かりやすい．整形やコメントの記入は，ソースコード自体に行っておくこと．レポートに貼ってから整形したのでは，読みやすいコードを書く習慣が身につかない．

（参考までに, 読みにくいソースリストの例をリスト 2 に示す. リスト 1 と同じプログラムでも, 書き方によって，分かりやすさが格段に異なっている.）

リスト 2. 読みにくいソースリストの例

```
#include <stdio.h>
#define bbbb 3
int factorial(int n)
{
if (n > 0) {
return n * factorial(n - 1);
} else {
return 1;
}
}
int main(void)
{
    int data;
    int i;
    for (i = 0; i < bbbb; i++) {
    printf("[ %d 回目 ] 正の整数を入力して下さい ", i + 1);
        scanf("%d", &data);
if (data > 0) {
    printf("計算結果：%d! = %d¥n", data, factorial(data));
        } else {
            printf("error:入力値が不正です¥n");
        }
    }
    return 0;
}
```

意味のある名前になっていないので分かりにくい

全くインデントされていないので読みにくい.

インデントがばらばらで読みにくい

5. 実行結果

実行結果を以下に示す．実行結果 1 では，MAX_INPUT 回(=3 回)データ入力を行い，それぞれ計算結果が出力されている．[1 回目]は 12 != 479,001,600 で，正しく計算されている．[2 回目]は，13 ! = 1,932,053,504 という結果が出たが，正しい計算結果は 6,227,020,800 であり，異なっている．この理由については，次の節で考察する．[3 回目]は入力が負の値であったので，正しくエラーメッセージが出力されている．

実行結果 2 は，実数値を入力した例である．この場合，入力値がこのプログラムの仕様と異なるので，正しく実行されない．

実行結果 3 は，アルファベットを入力した例である．この場合も，入力値がこのプログラムの仕様と異なるので，正しく実行されない．

実行結果 1

```
［ 1 回目 ］正の整数を入力して下さい 12
計算結果：12! = 479001600
［ 2 回目 ］正の整数を入力して下さい 13
計算結果：13! = 1932053504
［ 3 回目 ］正の整数を入力して下さい -3
error:入力値が不正です
```

実行結果 2

［ 1 回目 ］正の整数を入力して下さい 3.5
計算結果：3! = 6
［ 2 回目 ］正の整数を入力して下さい 計算結果：3! = 6
［ 3 回目 ］正の整数を入力して下さい 計算結果：3! = 6

実行結果 3

［ 1 回目 ］正の整数を入力して下さい a
error：入力値が不正です
［ 2 回目 ］正の整数を入力して下さい error：入力値が不正です
［ 3 回目 ］正の整数を入力して下さい error：入力値が不正です

6. 考察

このプログラムでは，13 以上の数値を入力すると正しい結果が得られなかった．これは，データを格納するために int 型を使用しているためであると考えられる．int 型で表現できる最大値は $2^{31}-1$（＝2,147,483,647）であり，13!(＝6,227,020,800）はこの最大値を超えたためオーバーフローを起こしたと考えられる．このプログラムで計算できる最大値は 12!であるので，入力時に，チェックを行う処理を加えれば，改善されると考えられる．あるいは，関数 factorial の引数や戻り値を実数型にすれば，扱える数値の範囲を広げることができる．

また，実数やアルファベットなど，想定外の入力に対するエラーチェック機能も改良の余地がある．

今回の課題では，再帰呼び出しを使うことで．単純なコードで階乗の計算を実現できた．ただし再帰呼び出しを行なうプログラムの実行時には，スタックの消費量に注意する必要がある[1]．階乗を計算する他の方法としては…

7. オプション課題

8. 感想

データ型と扱える数値の範囲について理解が深まった．再帰の考え方は理解するのが難しかったが，文献[2]が参考になった．

参考文献

[1] アスキーデジタル用語辞典，http://yougo.ascii24.com/gh/19/001982.html
[2] 柴田望洋，“定本 明解 C 言語〈第 1 巻〉入門編”，pp.194-197，ソフトバンクパブリッシング，2003.

問題点について考察している

改良の余地について検討している

参考にした文献の番号を挙げる．

他の手法について検討している

余力がある場合は，追加課題に取り組むと評価が高くなる．ただし，基本課題に不備がないよう気をつける．

参考文献の前には節番号(9．など)は付けない．
個々の文献の前には文献番号([1]など)を付ける．
演習課題や考察課題を解く際に参考にした書籍や URL を記載する．授業の URL は記載しないでよい．
書籍の場合，著者名，書名，ページ番号，出版社，発行年の順に記載するのが一般的である．