プログラミング演習I レポート

課題名：　課題 1 階乗を求めるプログラム

学籍番号	232100A
氏名	宇都宮 太郎
初回提出日	令和　7　年　4　月　1　日
書式修正版提出日	令和　　年　　月　　日

【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
(1) 表紙に必要事項を記入したか？	
(2) 目的・課題の概要	
(3) データ構造の説明	
(4) アルゴリズムの説明	
(6) ソースリストとその書式	
(7) 実行結果とその説明	
(8) 考察	
(9) 参考文献の書式は正しいか？	
(10) 図表番号とタイトルをつけたか？	
(11) ページ番号をつけたか？	
※　意見感想	

コメントの追加 [A1]: 各講義で配布されたレポートテンプレートを使用する．テンプレートが無い場合は，他の授業を参考に作成する．

コメントの追加 [A2]: 課題名を記入

コメントの追加 [A3]: 学籍番号，氏名，提出年月日を記入する．

コメントの追加 [A4]: 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．

コメントの追加 [A5]: 記載漏れが無いか，提出前に自分で確認し，○を入れる．

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．

2. 課題の概要

本課題では，nの階乗を再帰呼び出しで計算するプログラムを作成する．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い，式(1)で定義される．

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 1 \tag{1}$$

式(1)では0!は定義されないが，便宜上$0! = 1$とされる．これは，$(n-1)! = n!/n$であるので，$0! = 1!/1 = 1$と考えられるためである．したがって，式(1)を場合分けすると，式(2)のように表すことができる．

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

再帰呼び出しは，関数の中でその関数自身を呼ぶことを指し，そのような関数を再帰関数と言う．nの階乗を求める問題を，$(n-1)$の階乗を求めるという小さな問題に分割して計算する．

3. データ構造とアルゴリズム

3.1 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

3.2 アルゴリズム

プログラム全体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，結果を出力する．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．

コメントの追加 [A6]: 章立てのルールは各テンプレートに従うこと．オプション課題などの取り組み状況によっては，必要に応じて追加・削除する場合がある．

コメントの追加 [A7]: 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

コメントの追加 [A8]: 節の番号およびタイトルは，章タイトルよりやや小さく
本文と同じかやや大きいものが読みやすい

コメントの追加 [A9]: このように，図についての説明文を本文中に記載する．（本文で参照されない図や表を貼らないこと）

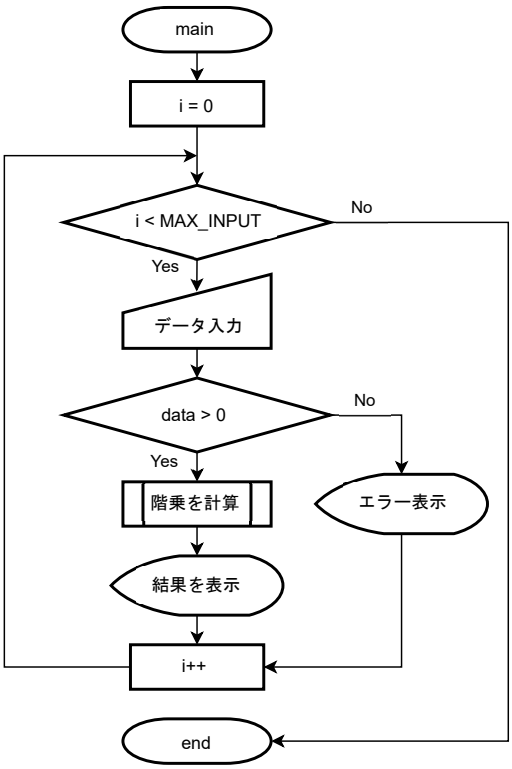

図 1 階乗を求めるプログラムのフローチャート

コメントの追加 [A10]: この図は draw.io で作成した．オフィスソフト以外で作成した図をレポートに載せる場合は，可能であれば，svg 形式などのベクタ画像で保存してから貼り付けると美しく表示できる．

図 1 中の階乗を計算する関数は次のように定義した．

`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…

コメントの追加 [A11]: 詳細については図なども用いて説明するとよい．

3.3　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:    }
```

コメントの追加 [A12]: リストの番号とキャプションを付ける．図や表と同様に本文中で参照すること．

コメントの追加 [A13]: ソースリストはスタイルの「ソースコード（行番号付）」を使って整える．セルが１つの表やテキストボックスで枠を付けるとよい．

コメントの追加 [A14]: レポートに貼る前に，開発環境上でソースコードを整形し，コメントを付けておく．

（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;
```

コメントの追加 [A15]: 意味のある名前になっていないのでわかりにくい．

コメントの追加 [A16]: 全くインデントされていない．

コメントの追加 [A17]: コメントが全くない

```
  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;
}
```

コメントの追加 [A18]: ソースコードがページをまたがっている．長いソースコードや大きな表は分割して載せる．

コメントの追加 [A19]: インデントが崩れている

4. 実行結果

実行結果を以下に示す．図 2 の実行結果 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（図 3）．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（図 4）．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．

```
tsuruta@DESKTOP-BFBVRFT:~/common$ ./factorial
[ 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
```

図 2 整数値を入力した場合の実行結果

コメントの追加 [A20]: 実行結果 1，出力画面 2 などと書かずに，わかりやすいキャプションにすること．

```
tsuruta@DESKTOP-BFBVRFT:~/common$ ./factorial
[ 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
```

図 3 実数値を入力した場合の実行結果

```
tsuruta@DESKTOP-BFBVRFT:~/common$ ./factorial
[ 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
```

図 4 文字を入力した場合の実行結果

5. 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 231－1（＝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…

6. オプション課題

7. 感想（任意）

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．

8. 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 (2025/4/1 参照).

[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.

コメントの追加 [A21]: 問題点について考察している．

コメントの追加 [A22]: 改良の余地について検討している．

コメントの追加 [A23]: 参考にした文献の番号を挙げる．

コメントの追加 [A24]: 他の手法について検討している．

コメントの追加 [A25]: 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．

コメントの追加 [A26]: 感想や意見は，考察とは分けて書くこと．

コメントの追加 [A27]: 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．URL の場合は参照した日付を記載する．