補助資料

■ 総論：始める前に

今回の「関数」のところは、この科目の最重要パートでもあります。対面授業では少しずつ、手を動かしながら、教室での受講生の様子（理解度・進み具合）を見ながら、適切なタイミングで白板の上に図を描く、デモをするなどして進みます。遠隔授業ではこれができないので、あなた方自身が、どれだけ焦らず、自分をコントロールして、じっくり理解を進めていけるか、が鍵です。こればかりは教員にはどうすることもできませんので、ホント、しっかり、手を抜かず進めて下さいね。

■ 作業の進め方

以下に教材の各段階ごとに、教室で説明する時に押さえているポイントなどについてコメントします。この補助資料と、元の教材を並列に見ながら少しずつ作業するようにしてください。

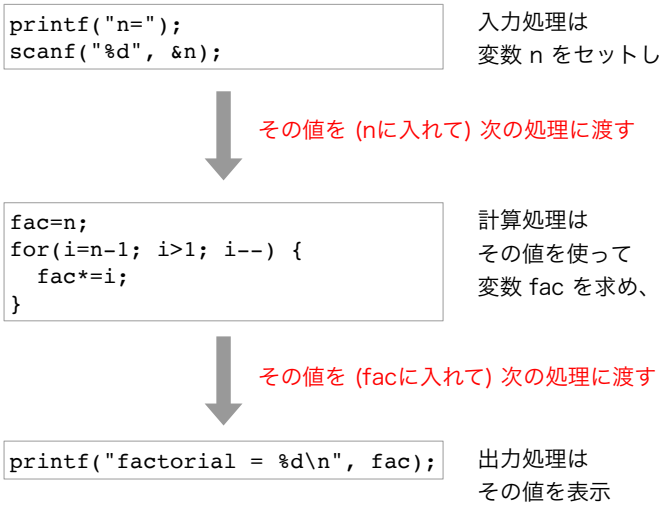
□ 簡単な関数の例 **1**

読むだけで無く、実際に動かして下さい。コードは教材ページの 08 セクションに置いてあります。
https://ylb.jp/2020a/kisop2/

元のコードをちゃんと読んで、何が起きているか把握してください。特に入力、計算、出力の 3 パートに分かれていて、それぞれがかなり独立していることを理解してください。

そして各パートが、右図のように
「値」を次の処理に受け渡している
ことを意識してください。
（ここでは受け渡すツール（媒介）として変数を使っています。）

この「ある処理を切り出して、独立した部品として定義」して、それを「呼び出す」ことで使っているのだ、と見立てて（全体構造を俯瞰的に見て把握して）ください。これが無いとこの先が全く進みません。



ここの部分がピンと来ない、あるいはこの先に進んで「どうも分からんなあ」と感じた人は、教科書に「少し異なるアプローチ」の説明がありますから、それを読んでみて下さい。「8.1 関数」p.216〜p.225 までです。掲載されているコードを動かしてみるのも良い方法です。

2 新しい関数を作ってみた例

ここでは余り説明なく、
・factorial(n) と書いて呼び出せること << このようにして値を「呼び出し**先**」に引き渡すこと
・return f と書いて処理を戻せること << このようにして値を「呼び出し**元**」に引き渡すこと
を出しています。このことは次のページで説明しています。（飛ばさず読んで理解して下さいね）

やはりここがピンとこない人は教科書「8.3 引数」p.226〜p.240 までを読んでみて下さい。

なお関数名に使っている factorial とは「階乗」のことです。変数名を fac としたのは、number を num とするのと同様、短くしてみた格好です。

□ 実際のプログラムの記述 3

この部分は、読み流さず、実際に手元の（上の作業のためにダウンロードしてきた）factrorial.c プログラムを図のような関数呼び出しの形に直して実行してみるのが良いです。（教室ではそのように指導しています。）

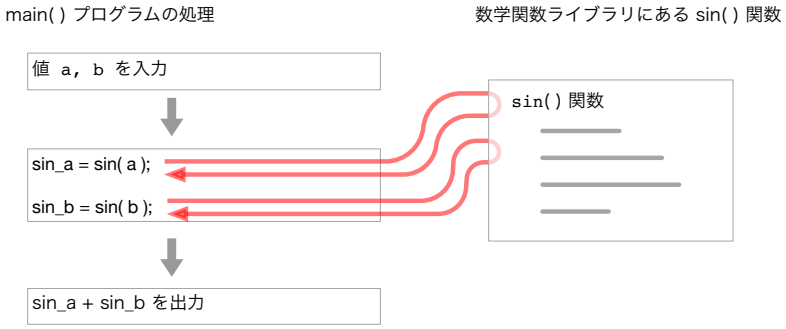
□ 課題 1. 4

上に書いたようにまず factorial.c プログラムを関数化したものを用意し、それを修正する形で進めて下さい。いきなり書き始めるのは無理があります。

このプログラムは「入力した二数の sin() の値の和を表示するプログラム」のつもりで作って下さい。

そのような場合、プログラムは恐らく次のような構造になると思います。
つまりどこかで誰かが作ってくれた sin() 関数を「二回」呼び出す処理を書く、と。

課題 1 では「誰かが作ってくれた sin()関数」の代わりに、「自分が作った factorial()関数を使うようにして作って下さい。



□ 課題 2. 5

重要なことは、「新しく提示された機能を満たす関数」を作ることです。
つまり factorial 関数は「引数（一つ）で渡された数の、階乗を return する」関数でしたが、
ここでは、「引数（二つ）で渡された数の、より大きな方を return する」関数を作って下さい。

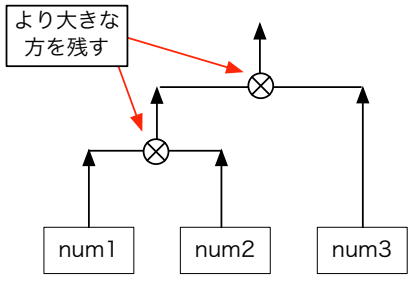
以下の点に注意して下さい。
・まさか関数名が factorial で良いはずはありません。適切な名前に変えて下さい。
・変数名 fac も同様です。

つまりこれは、関数の設計を行う、という課題です。関数名、引数の個数、名前、内部で使う変数の名前、戻り値の内容などについて適切に書けることを確認します。

□ 課題 3. 6

右図のように、課題 2. で作成した関数を二度使えば、つまり二度呼び出せば、三つのうち最大の数を得ることができることがわかりますか？

課題 3 は、新しい関数を作るのではなく、作った関数を複数回呼び出して新しい機能を作る体験をすることが目的です。
（main()関数側にその処理を書くのですよ。）



□ 課題 4. 7

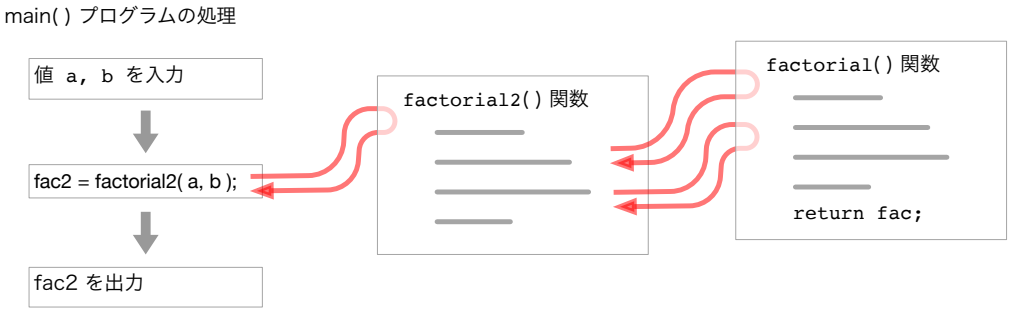
この課題は課題 2. に似ています。つまり「新しく提示された機能を満たす関数」を作ることが目的です。今度は引数の「型」に注意して下さい。

引数の型指定については、一つ前のページの後半「複数の引数・一般書式」に説明がありましたね。

□ 課題 5.

課題 1. 同様に「二数の何かの和」ではあるのですが、要求は「二数を引数として与える」ことである点に注意してください。ただし、単に「引数が二つ」のものを「新しく作る」のは課題 2, 4 でやっています。
ここで「新しい」ことは、過去に自分が使った関数を再利用する、ということです。
つまり
1. 「引数を二つとる関数」は作るが、
2. その内部処理で「階乗を二回求める」ためには、
3. 過去に作った factorial 関数を二回呼び出せば良い
と考えて下さい。

以下のような構造になるはずです。



■ おわりに

今後あなた方が作るプログラムは、どんどん「機能部品として関数を作り、それを再利用する」ことが増えます。そのつもりで取り組むと良いです。

提出課題は課題 1〜5 のすべてとします。教室ではおおよそこれらの課題を全部その場でやり、教員・TA・補助員でチェックして段階的に進んでもらっています。
提出されたすべての課題について細かくフィードバックすることはできないかも知れませんが、それでも必ず見ます。
（つまり受講生のなかには「提出」と設定しなかった課題については「やらない（スキップすべきだ）」と考える人が少なからず居るのです。今回の演習内容はとても重要で、それをしてしまうと次回どころか、この先ずっと、まったく手が付かなくなります。だからスキップを抑止するために「提出」することを「要求」しなければならないのです。提出作業だけでも手間が掛かって非効率なのですが、上のような副作用が大きすぎる状況ですのでご容赦ください。）

今回は課題 6, 7 までは進みません。これらは次回にやります。