C言語入学式

Part 1. はじめに

本クラスについて

本クラスの目的、目標、とりくみ方

- 目的
  - しっかりプログラミングの基礎をまなぶ
- 目標
  - **ほぼすべてのプログラミング言語に共通する部分をマスターしよう**
    - 変数
    - 条件
    - くりかえし
    - 配列
    - 関数 (できれば、ここまで)
  - **C言語特有の部分は必須としない。**例: ポインタ、構造体
- とりくみ方
  - **進捗は各自のペース**でOKです
    (営業時間は16:30までですが、切りの良いところで終わりにしてOK)

(脚注) 関数を使わなくても、根性で1万行のプログラムは書けます。 ただ、自分でも読みにくいし、デバッグし辛いし、他人が読めないので、 できれば関数はマスターしたいところ

とりくみ方(詳細)

- テキスト(紙)とワークシート(紙)を配ります
  - ワークシートはソースコードのテンプレートになっています
    - **論理の流れがコメントとして書いてあります**
  - テキストにも論理の流れと解説があります
- **紙に書く -> ソースコードを書く -> 実行して動作を確認**
  - いきなりPCでソースコードを書きはじめるのは禁止
  - そういうのは、もうすこし慣れてからやりましょう
- **典型的なミスやエラーは、なるたけ自力で**頑張って克服してください
  - つまづいたら、まずはテキスト、ELの教材を見直してください
  - エラーメッセージは、そのままGoogle先生につっこんでみましょう
- それでもわからないときは部屋の中をぐるぐる巡回している人に聞いてみましょう
  - そのために人的資源が投入されているので、じゃんじゃん使いましょう

こころがまえ

- 机の前で固まっていないように！なにかしら手を動かして何か試みましょう
- まずは教科書を読んでください
- ヒントのキーワードでGoolgeしてみましょう
- エラーメッセージは、**そのままGoogleにコピー＆ペースト**してみましょう
  (プロの一日もそんな感じだそうです)
- 数式は、簡単な例を実際に計算してみて納得しましょう。例: ジャンケンの判定
  - 3 + 2 - 1は4で3で割った余りが1だから負け(自分がパー、相手がチョキ)
  - 3 + 1 - 2は2だから勝ち(自分がチョキ、相手がパー)
  - **眺めていても先に進まないよ！やみくもに手を動かしてみることも大事** or *あがいてみよう*

(脚注) 数学者だって最初から難しい数式をこねくりまわしているわけではなく、 最初は具体的な数字をいじくってみると言ってましたよ (元ネタは Andrew Wiles, 350年解けなかった難問フェルマーの定理を8年かけて解いた)

評価

- **このクラスは別に評価をつけます**
  - 内容が異なるので、どうしても同一基準で成績はつけられません
  - こちらで真面目に取り組み、そこそこできてくれれば、なんとか...
  - オンライン組の普段のCBTも取り組まなくて大丈夫です(下記の夏休み話も参照)
- 一部、オンライン組と同じです
  - それもあって、**ZOOMの最初の連絡事項あたりは毎回きいておいてください**
  - 中間試験と期末試験は受けてください。
    - ただし、オンライン組と内容が異なるため、どうしても不利になります (下記の夏休み話も参照)
  - **最終課題も見ますが、オンライン組とは別枠で行います**
    - こちらのクラスは、このクラスで課題をみせてください
- 例年、夏休みの宿題としてリカバリのチャンスを用意しています
  - **普段のCBTが成績に占める割合はかなり大きい。** このCBTを夏休みに取り組めるようにします
  - 夏休みに取り組んでもらったCBTの結果は評価に反映させます
  - オンライン組と同じ範囲に追いついてもらうためですし、 オンライン組と同じ範囲なら、同じ基準が適用できるので、 評価も上げられます

(脚注) ただし進捗も異なる別コースなので、 リカバリしても同じ基準にはなりえないので、ご容赦ください

プログラミング演習全般について

この授業は第3言語(第2外国語)習得だと思ってください

- 第1言語(日本語)、第2言語(英語,第1外国語)ときて、 英語に似ているヨーロッパの言語どころか、 中国語あたりを習っている気分でいたほうがいいでしょう
- **英語っぽいけど、違います**
  - 人工の言語だから... それでも、まぁ英語っぽいといえば英語っぽい
- **数学の記号がでてきますが、いろいろ違います** (こちらのほうが罠)
  - キーボードにある文字だけで何とかしないといけない苦肉の策なのです;-)
  - `=` は意味が全く異なります
  - `>=` とか `<=` は数学にはないですよね
  - `>` とか `<` といった文字どおりの意味のものも一部あります

例：=（数学記号）の罠

- プログラムでは数学と同じ記号を使いますが、ここ**落とし穴**(なにしろ、キーボードにある文字は限られてますからね)
- プログラムは左上から右下へ順に読み進めます

```
変数 = 値;  // これは等式ではなく代入です

  x =  1;  // x と 1 が等しいという意味ではない
           // xという名前の箱を用意して1を入れる
```

  - **この行を読み始める時、まだxの中身はありません(未定)。** PCの処理が、**この行を通過すると、xの値が1**に変わります(そう読まないといけないのです。数学とは全然ちがう)

- もっと不思議な例

```
x = x + 1
```

これも不思議ですよね?でも数学では無いのでOKです。 これは「右側のxは上で代入した1で、それに1を足した」結果の2を左側のxに代入

- [コラム] 代入を`<-`とか`:=`とか違う書き方をする言語もあります。 **`<-`は`=`より気持ちが伝わりますが、打ちにくいんです…** 代入はプログラムで山ほど使うので打ちやすくしておきました! というC言語の`=`に慣れてください

```
[他の言語の例]
x <- 1
x := 1;
```

例：日本語の罠

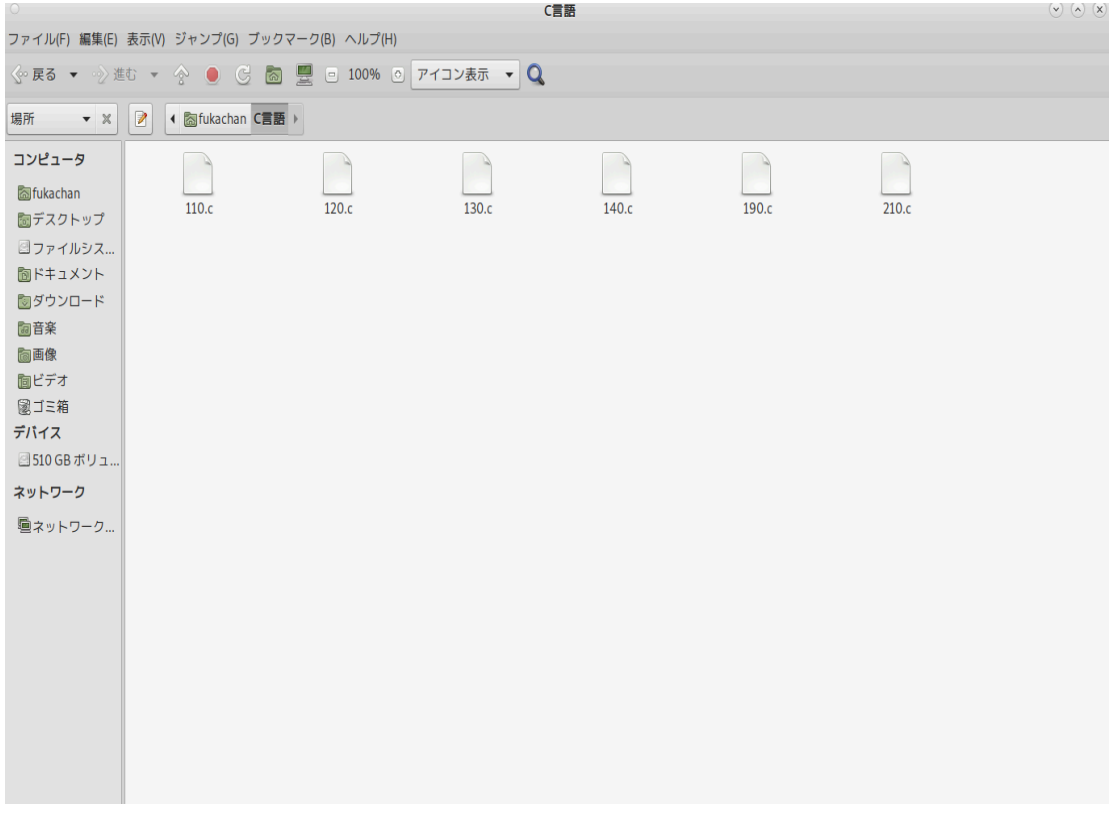
- みなさん日本語環境を使っているとおもいます
- そのため、たまに間違えて日本語を入力してしまうことがあります
  - 「　」(大文字の空白)は一番分かりにくいエラーです
  - エラーメッセージも共通していないので見つけにくい
  - どうみても間違ってない場合は、この大文字の空白の可能性があります。 ただ見つけにくいので、「あやしい行」(エラーメッセージで表示された行番号のところ)を消去して、 一から再入力する方が確実です

お作法を少々

- コンピュータの上も整理整頓が大事
- この授業専用のフォルダをつくろう
  - 環境依存なので、やりかたは、このあとで各自コンサルします
    (-> mkdirコマンドの使い方)
- ファイルには課題の番号をつけよう
  - たとえば課題110にはファイル名`110.c`をつける
- [処世術]課題120を始める時は課題110をコピーして編集する (**一から書く必要は無い**)
  - コピーではなく、`110.c`の編集に使っているアプリの「別名で保存」あたりを使い、`120.c`という名前で保存(`120.c`ファイルを新たに作成)して、次の課題を始めてもOK
  - 環境依存なので、やりかたは、このあとで各自コンサルします
    (-> cpコマンドの使い方)

例:［お作法］フォルダをつくり整理整頓

- 気分を伝える図です(右図)
  - 右図のOSはLinuxですが、気持ちはつたわると思います:-)
- 説明
  - OSにログインしたところに
    - そこをホームと呼びます
  - 「C言語」というフォルダを作成し
  - そのフォルダの中に`110.c`などとソースコードを作っている様子



先に進む前に環境の確認をしよう

少し確認させてください(人それぞれなので)

- C言語のコンパイルは、どうしていますか?
  - cc コマンドを使っている
    - Windows上のCygwin
    - Windows WSL2でcc
    - Unix/Linux環境
  - ブラウザ上のプログラミング環境を利用
    - ideon
    - paiza
    - その他
  - その他のやりかた(?)

- cc コマンドを使っている人へ質問です
  - 使っているOSは何ですか?
    - Windows11 or 10
    - Windows10以前(いないはず,もうEOS)
    - MacOSX (*)
    - Unix/Linux (*)
    - ChromebookのLinuxモード (*)
  - Windowsの方,ソースコードの編集方法は?
    - ブラウザ上のプログラミング環境を利用
    - Cygwin上でviを利用
      - vi難しくないですか?大丈夫?
    - その他

(脚注1) 用語が分からない/どれに該当するか分からなくても大丈夫。 百戦錬磨の我々に、あなたのPCを見せてごらん:-)

(脚注2) (*)のケースは、ほぼ Unix 準拠なので、そのまま御利用ください

(脚注3) [paiza.IO の使い方 (qiita)]

注意点(ideone,paizaなどブラウザ上の言語実行環境利用者向け)

- キーボードからの入力
  - `scanf()`を使う部分は一手間必要です
    - ブラウザでは、キーボードからの入力を直接実行できないようです。
    - [ideonの場合] 画面の下の方にある`stdin`というボタンをクリックすると、 その下に入力欄が開きます。 そこに、あらかじめキーボードから入れる値を入れておいてください。 入力後、runをクリックしてください

- ファイルの入出力
  (学期末の発展的内容で使いたい場合)
  - ideoneでは無理みたいです
  - paizaでは利用できます
    - (次回も使うために)ファイルを保存するには、 アカウントの登録が必要かもしれません。 その場合、**捨てアカの利用をおすすめ**します

(脚注)「無料の昼飯はありません」(月は無慈悲な夜の女王)。 無料のプログラミング環境 = 個人情報が欲しい = じゃんじゃん企業からのメールが来るらしいので、 大学のメールアドレスの登録は推奨しません (それがわかってて登録するならいいですけど)

C言語とは

C言語について

1. Unixオペレーティングシステムという**OSを書くために設計された言語**です
   - そのため非常にハードウエアを意識する必要があります。 ハードウエア制御をする仕事では長年C言語が使われてきました。 例: OS, ファームウエア(機械の中で動いているプログラム)
   - **ポインタ**というメモリ上のアドレス(住所)を直接あつかう機能が代表例
   - ここはメリットでもあり、かつデメリットです (書くのも難しいし、なにより**危ない**操作だから)
2. 言語自体の機能は厳選されています
   - C言語以降の言語(80年代ならPerl,Python)にくらべ、 言語自体が提供する機能は少ない
   - そのため、 **言語自体の学習コストは小さい**と言われますが、 **ハードウエアの意識**とライブラリ関数の使い方の勉強は必要です。 ラクチン言語満載の現代、逆に今の人には難しい？
   - C言語単体では`printf()`すら動きません。C言語処理系ではなく、 OS作成者が用意してくれた関数(**ライブラリ関数**)を使い、 プログラムを書いていきます -> `#include`文、ccコマンドの`-l`オプション

```
$ cc -o a.out 110.c -lc     # 実は cc 110.c を裏では、こう実行しています
```

(脚注1) 1970年代初頭の中規模コンピュータで動かすためのものでした。 今の価格で3000万円くらいのコンピュータが開発環境なのですが、 今のPCにくらべれば、ものすごく遅いものです。 おそらく、その遅さをイメージできない (脚注2) OSの新規追加部分は21世紀版C言語**Rust**で書こう！のが最近の風潮ですが、 半世紀におよぶ膨大なC言語の遺産があるのでCが読み書きできて損はありません

Hello World（基本の書き方、出力）

```c
// コメントです
/* これもコメント
   複数行のコメントを書くときに使います */

#include <stdio.h>              // 1.

int main () {                   // 2.

    printf("Hello World\n"); // 3.

    return 0;                   // 4.
                                // return(0);も可
}
```

1. (ライブラリ関数の)定義を読みこみます(**必須**)
2. 最後の `return 0;` と合わせるため、main関数の返り値は整数(int型)です。 また、main関数の引数は無いので `()` と書いています
3. ディスプレイへの出力には `printf()` 関数を使います。 C言語では**右端の;は必須**です。 `printf();` のような「 `名前();` 」形式の文字列を書くと**関数の呼び出し**になります。 関数の利用、実は最初からやっているのですね！
4. return(～を返す)の文字どおり、 プログラムは自分の処理結果について値を返せます
   - main関数のreturnは特別で「OSへ成功(=数字0)」を伝えるという意味です。 これが**[お作法]として正しい**(脚注も参照)

(脚注1) これをテンプレートにして、このあと改造していってください (脚注2) 最後の4.の部分は流派があります。 教科書によっては無かったり、別の書き方だったりしますが、 この科目ではreturnを使ったOSへの礼儀作法を書く流儀としましょう