C言語基礎講習会用資料　そのいち

釧路工業高等専門学校　プログラミング研究会　寺地海渡　2016/05/01　2016/08/30 第二版

NITKC KPC K.TERACHI(KMITL)

2016/05/01 初版
2016/08/30 第二章追加・改訂(KMITL)

著　寺地海渡／石川岳
監修　石川岳

前書き

今回、下級生にC言語を教えるにあたり、
「C言語は一番面倒な部分を他言語で使わないから糞」という意見をいただき、
「ほかの人間が教えるときに指針となるものがほしい」という意見もいただき、
「行き当たりばったりにC言語を教えるのは改めるべき」という意見もいただき、
「オープンキャンパス迄に形を披露できるようにすべき」という意見もいただき、
「計画的に教えないと時間がないから指導の手順を考えて」という意見もいただきました(吐血)

~~就活なんてどこ吹く風みたいな~~優秀な~~学蓄もとい~~人材を確保し、
教える順番を話し合い、
なるべく簡素な資料を作るように監修してもらい、
「図がないと解りにくい」「うるせぇそんなもの入れるスペースねぇ」と議論を交わし、
「教科書じゃないからネタに走ろう」「functionの意味違いますよねそれ」と編集者と確認しあい、
「あっ、書き換えやがったアイツ」「いや私やりませんよ!?」とお互いの信頼も厚く、
苦節二週間、貴重なGWを~~費やして~~大いに活用してこの資料を作成いたしました。
皆さんの勉学の友であり、励みになれば筆者一同幸いです。フィードバックお待ちしております。 2016.05.10 記

なんやねんこのイオ〇みたいなデザイン

- Officeのテンプレや。
- 基本的に発表用ではなく、各自で見るカンペみたいなスライド作ります。（高〇式メソッド）
  - Not Taka〇a〇a.
  - Not Akira.
- 教科書ではないので砕けた感じの文体で書いたりします。
- よい（就職希望と進学志望の）子は（このスライドの手法を）真似しないでね。
- 我々の創造を手助けしてくださることに敬意を込めて「コンピュータ様」と表記します。

おとこわり

- このスライドは、4月に加入した新人君たちに
- 7月の展示にむけてコンテンツを制作させるために
- 5月からの二ヶ月間で爆速でたたき込むための
- ちょうばうあとな資料です
- 過度な期待はしないでください。
  - これが伝わる世代だとは思えませんぞ

目次

第一章 プログラミングに必要な前提知識

- この「前提知識」とは、今回習得するCに限らず、多言語のプログラミングにおいても重要な知識です。
- 知識としては、プログラミングに直接かかわりがなさそうに見えますが、そうでもないんですよこれが。
- 覚える要点を４つに絞り込んだので、ゆっくりと覚えてください。
- でも、ちょうゆるふわなので後で正しく覚えなおしてください(半年後くらいに)。

ソースコードとは

- 料理のレシピみたいなもの
- コンピュータ様はソースコードを上から下へ読み進めて(処理して)いきます。
- また、人間共が自分たちで読めるよう(可読性を保つため)に命令の書き方を工夫していきます。

- コンピュータ様は「なんにも知らないけど言われた通りに頑張るお利口な子」です。
- ~~反抗期に入る前の無知な愛娘に対する愛情のようなものを持って接してあげましょう。~~
- つまり。
- 思った通りに動かなかったりエラーを返してくるのはコンピュータ様ではなく(大体)自分のせいです。

- 関連あるけど説明しない用語：コンパイラ / 機械言語 / 自然言語

ソースとは

平易なソース

```
#include<stdio.h>

int main(void){
        int i;

        for(i=0; i<20; i++){
                printf("%d¥n", i);
        }
        return 0;
}
```

平易なソース

```
#必要な道具<調度器具>

ソース　作り方{
        調味料 酒=１;
        調味料 みりん=２;
        調味料 砂糖=３;
        調味料 醤油=３;
        ソース すき焼き風味のタレ;

        すき焼き風味のタレ = 酒+みりん+砂糖+醤油;
        ボウルで混ぜる(すき焼き風味のタレ);
        完成 すき焼き風味のタレ;
}
```

変数とは

- $f(x) = Ax^n + C$
- この関数における x が「変数」である。
- 大文字は「定数」、つまり「決められた数」である。
- プログラミングにおける変数とは、**「実行中に値が変動/決定される要素」**を指し示す。
  - つまり、プログラムの実行中に作られる**変更可能なデータ(値)**のことを変数という。
- 一方、定数は**「ソースコードを書いている時点で既に値が決まっている要素」**のことである。
  - つまり、プログラムの実行前から作られている**変更不可能なデータ(値)**のことを定数という。

変数

```
int num=1;
num=10; //これはできる
```

定数

```
#define A 1
A=10; //これはできない
```

型とは

- 変数の形式を指し示す。
- 種類は様々あるが、ここではC言語における代表的な型を３つ紹介する。

整数型

- int型と呼ばれるもの。
- 整数とは…「１で割ったあまり(商)が0になる値」のことである。
- 環境によって扱える値の範囲は異なる（説明の放棄）

- int x = 100;
- int y = -1;
- int z = 10.0;　　※これはint型ではない (小数点ついてるから実数(小数点の付いた数)という扱い)

- 余計な話をすると、unsigned int型というものがある。
- 雑にせつめいすると「正規表現のビット」であり、要は自然数のみを扱う。

実数型

- double型と呼ばれるもの。
- int型の倍のメモリを使っているから「double型」
- 実数とは…「虚数を含まない値」のことである。
- 虚数とは…「$i = \sqrt{-1}$を含む値」のことである。
- 数学では「最も美しい方程式は $e^{\pi i} + 1 = 0$ である」と言われている。興味のある人は調べて、どうぞ。

- double x = 76.1;
- double y = -10.0;
- double z = π;　※$\pi = 3.14159265358979323384$ ... なので実数(無理数)ではあるが、実数値ではない。
  - なぜなら、コンピュータ様が扱える値には限りがあるから(無限は扱えない)
  - これらの無理数は、実際には四捨五入などした近似値(実数値)を代わりに用いることが多い
- double u = 3.1e-5　←あーあーあー浮動小数点数は有理数だけども(お話盛り)今の君たちにはまだ早いにょおおおお

文字型

- char型と呼ばれるもの。
- 「character」を扱うための型。
- これが説明一番面倒臭い。
- 「数値」と「(文字としての)数字」と「文字」と「文字列」の違いを理解してね。
  - 数値 … 何かしら意味のある（重みのある）「値」。 int と double はこれ。
  - (文字としての)数字 … 何の意味もない「文字」。
    - 人間には意味があっても、コンピュータ様にとっては無意味なもの。
  - 文字 … character（一文字だけ）
  - 文字列 … string（複数文字でもOK）
- char の中には数値が入る。ASCIIコードにそって文字→数値に変換されて代入される。
- char型変数一つには半角文字１文字しか入らない！！←２バイト文字使えないし一文字だけとか使えないじゃん…

文字列型

- char型変数一つには**半角文字１文字**しか入らない！！←２バイト文字使えないし一文字だけとか使えないじゃん…
  - そんな問題を解決するのが複数文字をいっぺんに扱える**文字列型**です。
    - 「紹介する型は３つ」って言ったじゃん嘘つき！！
      - いいえ、紹介する型は３つだけです。
  - なぜなら、**C言語には、文字列型は…ありませぇん！！！**(小保〇風)

- なので、文字列(複数文字でもOK)を扱うには**一工夫必要**です。(char型の配列をつかう)
- でも、char型の配列を扱うときは**ポインタの考え方が必要**。
- なので、char型は暫く講習会では扱わない予定である。

型についてまとめると

- int型とdouble型は「数値」である。
- char型は「文字を数値に変換したもの」である。

- int型 … 整数
- double型 … 実数
- char型 … 文字を数値に変換したもの

関数(FUNCTION)とは

- プログラミング的な関数の前に、数学的な関数のお話を。
- 関数とは、「ある一つの数が定まる時、もう一つの数も定まる関係にある方程式」のこと
- $y = Ax + b$:中学生の時に習ったこんな感じの方程式も、立派な関数。
- $f(x) = Ax^n + C$:数学における一般的な関数(方程式)
  - 簡単にいうと、**左辺(yとか $f(x)$)に右辺の式の結果を代入するもの**が関数

- プログラミングにおける関数も、意味合いとしては数学のそれと同義。
- int main(void){ の main(void) が $f(x)$ である。
  - …といっても何いってるかさっぱりだと思うので例をみて学ぼう

関数の一例(A と B の平均)

自作関数なし

```
#include<stdio.h>


int main(void){
        int result, a=10, b=20;
        result=(a+b)/2;
        a=100;
        result=(a+b)/2;
        return 0;
}
```

自作関数あり

```
#include<stdio.h>


int ave(int a, int b){
        return (a+b)/2;
}


int main(void){
        int result, a=10, b=20;
        result=ave(a,b);
        a=100;
        result=ave(a,b);
        return 0;
}
```

一見ただコードが
長くなっただけに見えるが…

関数の一例(A と B と C の平均)

自作関数なし

```
#include<stdio.h>


int main(void){
        int result, a=10, b=20, c=30;
        result=(a+b+c)/3;
        a=100;
        result=(a+b+c)/3;
        return 0;
}
```

自作関数あり

```
#include<stdio.h>


int ave(int a, int b, int c){
        return (a+b+c)/3;
}


int main(void){
        int result, a=10, b=20,c=30;
        result=ave(a,b,c);
        a=100;
        result=ave(a,b,c);
        return 0;
}
```

計算部分の修正箇所が少なくなる
main関数の中身がすっきりして人間が見やすいコードになる！
これが関数を使うメリットの一部です

第二章 C言語プログラミング
-これさえあればプログラミングはできないことはないこともない！ -

- さあ、いよいよプログラミングを始めましょう。
- この資料は、5月上旬から7月中旬までのおよそ10週間でプログラミングの基礎を固めるためのものです。
- 「10週間で学ぶC言語とプログラミングの基礎」というタイトルで本にして印税稼ぎたいところです。
- でもいまなら**あなたにだけ**、**無料**で**こっそり**教えちゃいます！！
- その方法とは…<続きは製品版で>

目次(第二章)

- 第一話『ハロワは楽しい!!』 hello, world.
- 第二話『もしかして、魔法使い!?』 #include<stdio.h>
- 第三話『私は魔法使い!!』 宣言と初期化
- 第四話『ふたりでなら…』 if, else
- 第五話『七転ころんでも、八回起きれば大丈夫!!』 for
- 第六話『メクルメク輪廻』 while
- 第七話『真と嘘』 true, false
- 第八話『魔法の呪文のその意味は』 #include
- 第九話『言いたいことと、受け入れたいもの。』 printf, scanf
- 第十話『本当は、知ってるんだ、あたし…』 しれっと使ってたやつ(i++; とか int x=0; とか)
- 第十一話『あなたって、最低…』 最低限のプログラミング作法(可読性のための最低限のスキル)
- 第十二話『総集編：寺地大先輩のQAコーナー』 ()と{}の使い分けとか

始めに諸々

- この章ではソースコードと解説を一対に記述して説明します。
- ほとんどのソースコードはコピペして実行することもできるようになっています。お試しあれ。
- ソースコードの文量の都合上、やむなく(というわけでもないけど)フォントサイズ維持のため略します。
- いきなり省略形の書き方で戸惑うと思うかもしれませんが、その点は第十話で補足説明します。
- ソースコードの重要な部分や説明で重要な部分は**赤文字**で表記しています。
- 色弱等でわかりにくい場合は著者・編集者へ申し出てください。表記方法を工夫したものを作ります。
  - というか色弱用の眼鏡が両目のレンズ合わせて30万円以下で買えるからそれをｋ（文章はここで途切れている）

第一話『ハロワは楽しい!!』

- (職業斡旋所のことじゃ)ないです。
- 「Hello, world!」は通過儀礼。
- わけもわからずやっておくもの。
- こんにちは、せかい！
  - 面白くないって？そうかい、こんにちは！！

- 事実上のハロワは赤字の部分のみ。
- 黒字の部分は直接的な働きはしません。

```
#include<stdio.h>

int main(void){
        int x;

        printf("Hello, world!¥n");

        scanf("%d", &x);

        return 0;
}
```

第二話『もしかして、魔法使い!?』

- いんくるーどぉ？ナニソレオイシイノ？
- これは魔法の呪文です、いいね？
- 「キュアップ・ラパ〇！」と同義です。
  - キュアップ・ラ〇パ！
  - ハローワールドといいなさい！
- これは魔法の呪文です。OK？
- わかった？よし、いい子だ。
  - 魔法の呪文がわからない人は土曜祝日だからといって昼過ぎまでぐうたら寝てる証拠。
  - または休日朝早くからバイトに勤しんでいる証拠。

```
#include<stdio.h>

int main(void){
        int x;

        printf("Hello, world!¥n");

        scanf("%d", &x);

        return 0;
}
```

第三話『私は魔法使い!!』

- 変数とは、**変更可能なデータ(値)**のこと。
- **変数**を使うには、その変数を使うことを
  コンピュータ様に教えてさしあげなければならない。
- コンピュータ様に変数を教えることを**宣言**という。

- int x;
  - int**型の変数**であることを教えて、(**宣言**)
  - **変数の名前**が x であることを教える。(**宣言**)

- /* **ここの地点で…** では、
  x にナニが入っているかわからない（0とは限らない）。

- 次行の x=0; で x に値を代入して、
  ようやく意味のある変数ができる。
  この作業を**初期化**という。
  - int x = 0; としても同じ。

```
#include<stdio.h>

int main(void){
        int x;

/* ここの地点で x の値はまだ不明(パルプンテ状態) */

        x=0;  // ここでようやくxの値が確定

        scanf("%d", &x);
        return 0;
}
```

第四話『ふたりでなら…』

- 赤字部分に着目して下さい。
- if … 条件分岐
  - 「もしpの値が1と等しければ…」
  - 「それ以外で、もしpの値が2と等しければ…」
  - 「それ以外なら…」
- 以上のように命令が記述されている。

- if(条件式) という定型文

- 命令自体は魔法の呪文で出力してもらってるだけ。

```
#include<stdio.h>
int main(void){
        int p;
        scanf("%d", &p);
        if(p == 1){
                printf("一人だと負けちゃうかも…¥n");
        } else if(p == 2){
                print("二人なら、負けない！¥n");
        } else {
                printf("みんなとなら、きっと勝てる！¥n");
        }
        scanf("%d", &p);
        return 0;
}
```

第五話『七転ころんでも、八回起きれば大丈夫!! 』

- for … 再帰(繰り返し)

```
int i;
for(i = 0;  i < 10; i++){
        命令;
}
```

- この部分は定型文(魔法の呪文と同じ様なもの)
- { } 以内の命令を指定された回数、再度処理する。

- 真ん中の数値を変えると繰り返す回数が変わる。
- for(i = 0; i < 10; i++){

- 左の数値を変えると数え始めの値が変わる。
- for(i = 0; i < 10; i++){

- 絶対に自分で変えてはいけない部分は以下の通り。
- for(i = 0; i < 10; i++){

```
#include<stdio.h>
int main(void){
        int Clara = 0, i;
        for(i = 0; i < 10;  i++){
                if(Clara < 8) printf("クラ〇が倒れた!¥n");
                else printf("ク〇ラが立った!¥n");
                Clara++;
        }
        scanf("%d", &i);
        return 0;
        }
}
```

第六話『メクルメク輪廻』

- while … 再帰(繰り返し)
- forとの違いは
  「繰り返す回数が一つに決まらない」時に使う。

- while(x != 0){
  - // ちなみに != はノットイコール(等しくない)
    - この部分でxの値が0かどうか判定、0でなければ{ }内の文を実行
- { }内は第四話でやった通り(if文)

- xが0になった (x == 0) 場合、
  while(x != 0){ の部分の条件を満たさなくなるため、
  while文は終わり、次の
  printf("そう、オンリーワンが大切よ。¥n");
  が実行される。

```
#include<stdio.h>
int main(void){
        int x=-1;// 初期値を一定に
        while(x != 0){ // 0でない場合
                printf("あなたは何番になりたいの？¥n");
                scanf("%d", &x);
                if(x == 1) printf("2番じゃダメなんですか？¥n");
                else if(x == 2) printf("1番になりたくないの？¥n");
                else if(x == 3) printf("表彰台で一番低い所なの？¥n");
                else if(x > 3) printf("表彰台にすら上らないの？¥n");
        }
        printf("そう、オンリーワンが大切よ。¥n");
        scanf("%d", &x);
        return 0;
}
```

第七話『真と嘘』

- ここで、 if, for, whileに共通する「条件式」について。

- 条件式 … 指定された要素が定められた範囲にあるか
- ( ) 内の条件を判定して、
  - 条件を満たしていれば(true) … { } 内の処理を実行
  - 条件を満たしていなければ(false) … { } を実行せずに下へ

- 変数の型に、本当(true)か嘘(false)かを、
  データ値として保有するbool型というのがある。
  - 当然、C言語にはありませぇん！！！(〇保方風)
    - stdbool.h か windows.h(窓のみ) を導入すると使えるよ。
      - でもC言語じゃ有用性低いから無くても平気。
        - (0, 1) の 整数値しか持たないくせにそれ以外の整数値代入できちゃうし。

↓条件式の例↓

```
if(x == 0){
} else if(x > 0){
} else {
}
```

```
for(i=0; i<10; i++){
}
```

```
while(x != 0){
}
```

第八話『魔法の呪文のその意味は』

- 魔法の呪文と言ったな。あれは嘘だ。
- include … 導入
  - 導入？何を？
  - stdio.h を導入しています。
    - stdio.h … standard in out header file のこと。
      - 入出力に関連する命令が記述されているファイルである。
      - 実は、printf も scanf もコンピュータ様は知らないのです。
- 何故、ヘッダファイル(.h)を導入するのか？
  - コンピュータ様は、本来は計算を行うものです。
    - 計算 … 四則演算のこと。(実は足し算のみでごり押し)
  - 計算以外の処理は .h にやり方が書いてあります。
- 「コンピュータ様、
  先ずはこのファイルをお読み下さい。」

```
#include<stdio.h>

int main(void){
        int x=0;

        printf("Hello, world!¥n");

        scanf("%d", &x);

        return 0;
}
```

第九話『言いたいことと、受け入れたいもの。』

- ヘッダファイルはたくさんあります。(一例)
  - 標準入出力用の stdio.h
  - 数学的記号や計算を行うための math.h
    - コンパイルする時にオプションで -lm を追加する
  - 時間を扱うための time.h
  - ランダムな数列を生成したりする stlib.h
- 用途に応じてヘッダファイルを導入しましょう。
  - 役割や仕様が不明な時はリファレンスを読むべし。

- stdio.h
  - printf (出力)
  - scanf (入力)
- math.h
  - M_PI (π)
  - sqrt(x) (平方根)
- time.h
  - time(NULL)
  - clock(CLOCKS_PER_SEC)
- stlib.h
  - srand() (seed値)
  - rand() (ランダム関数)

第十話『本当は、知ってるんだ、あたし…』

- 今までしれっと使ってたものを紹介。

- これはif文の省略した(一行にまとめた)書き方。
- 命令が一つだけなら{ }を省略できます。
- 二つ以上の命令を書くときは略せません。

```
if(Clara < 8) printf("クラ〇が倒れた!¥n");
else printf("ク〇ラが立った!¥n");
```

↓ ↓ ↓

```
if(Clara < 8){
        printf("クラ〇が倒れた!¥n");
} else {
        printf …
}
```

(第四話『七転ころんでも、八回起きれば大丈夫!! 』より)

第十話『本当は、知ってるんだ、あたし…』

- これは宣言を一行にまとめたもの。
- 変数の型が同一なら
  ,(カンマ) で区切って纏められる。

- しかし、このやり方は**要注意**。
- なぜなら、**変数の役割が異なるものを纏めてしまうと、可読性が損なわれてしまう**(つまり醜い)からである。
  - 見難い == 醜い
    - おっ、うまいじゃーーーん。

- ***役割の異なる変数は個別に宣言することを推奨***します。

int Clara = 0, i;　　//この書き方は正直見難い。

↓ ↓ ↓

```
int Clara;
int i;
Clara = 0;
```

※ int Clara = 0;　　//この書き方は見やすい。

(第四話『七転ころんでも、八回起きれば大丈夫!! 』より)

第十話『本当は、知ってるんだ、あたし…』

- return 0;
- 関数の解を返してします。
- どこに？
- int main(void){
- ↑ココ

- 関数の型に合わせた解を返しましょう。

- もし、 double main(void) だったら、
  returnする値は**実数**です。
- return 0.0;
- returnする値は関数内で処理した変数でも可。
  - むしろそれが本来の使い方。

```
#include<stdio.h>

int main(void){
        int x;

        printf("Hello, world!¥n");

        scanf("%d", &x);

        return 0;
}
```

(第一話『ハロワは楽しい!!』より)

第十話『本当は、知ってるんだ、あたし…』

- ¥n　(正確には半角の『＼(バックスラッシュ)n』)
- ナニコレ。
- 特殊記号。

- “ ” で囲まれている範囲は、「入出力する文字列」としてコンピュータ様に認識される。

- ところが、人間の見た目通りに改行しても
  コンピュータ様には「改行」がわからない。

- そこでこれ。改行の命令用の特殊記号。
- これでコンピュータ様が改行してくれるよ。

```
#include<stdio.h>

int main(void){
        int x;

        printf(“Hello, world!¥n”);

        scanf(“%d”, &x);

        return 0;
}
```

(第一話『ハロワは楽しい!!』より)

第十話『本当は、知ってるんだ、あたし…』

- %d
- これも特殊記号。
- 入出力時に変数を代入するための命令。
- printf(“… %d … ” , hoge);
  - 出力したい文章の合間に変数を入れることも可能。
- scanf(“%d”, &hoge);
  - 入力したい変数に&をつけるのを忘れずに。
    - この & にも意味はあるんだけど、ポインタの話が(ry

- %d は整数値専用。
- 実数値なら %f（scanfは %lf ）
- 文字なら %c
- 他にもいろいろあった気がする。

```
#include<stdio.h>

int main(void){
        int x;

        printf(“Hello, world!¥n”);

        scanf(“%d”, &x);

        return 0;
}
```

(第一話『ハロワは楽しい!!』より)

第十話『本当は、知ってるんだ、あたし…』

- (どこかのサイズでも何かしらの言語のことでも)ないです。
- インクリメントと言います。

- A--;
  - A = A -1;

- C++;
  - C = C + 1;

- それぞれ「１減らした値」「１増やした値」を
  自分に代入しています。

- 地味に複数の変数を一つの出力に詰め込んでいます。

- 第十話にページ割き過ぎぃ!!

```
#include<stdio.h>
int main(void){
        int A=0;
        int C=0;
        int i;

        for(i = 0; i < 10; i++){
                A--;
                C++;
                printf("A = %d¥nC = %d¥ni = %d¥n¥n", A, C, i);
        }
        scanf("%d", &i);
        return i;
}
```

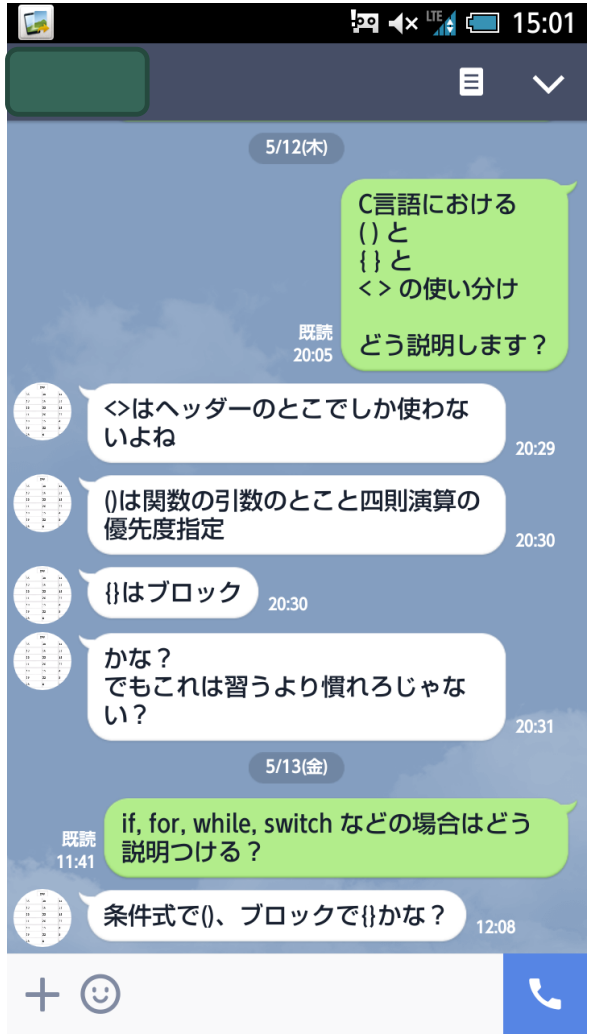
第十一話『あなたって、最低…』

- ここまでで第二章の基盤はおしまいです。
- ここで、**最低限のプログラミング作法**について。

1. インデント(行頭の空白)は { } 一組ごとに右へ。
   - IDEが勝手にずらしてくれることもよくある。
   - 目安はTabキー１個分。
2. main関数はint型、戻り値は正常終了なら0。
   - これなら開発環境にあまり左右されない。
3. カッコは開けたら先に閉じておく。
   - 閉じ忘れなどのケアレスミスを軽減できる。
4. エラーメッセージは読む。
   - 簡単な英語だから読めばミスした理由がわかる。

```
#include<stdio.h>
int main(void){
        int i;
        for(i=0; i<10; i++){
                printf("%d¥n", i);
        }
        scanf("%d", &i);
        return 0;
}
```

第十二話『総集編：寺地大先輩のQAコーナー』



第三章 C言語プログラミング
–人間の人間による人間のための可読性上のプログラミングスキル-

- この章では、「人間にとって見やすいソースコードを書くためのスキル」を紹介していきます。
- ソースコードを見やすくすると、後々変更があった時などの手直しする箇所が減って楽になったり。
- 少し複雑な使い方をするものもありますが、慣れると便利なものがたくさんあります。

- 今更だけど。金色のモザイクで目線を隠されたおっきい谷口君に「文字だらけで見辛い」って言われた。
- のをうすらぼんやりと思い出した。
- イラストかける人にイラスト書いてもらいたい。
  - 外注でも内製でもいい。
  - でも何をイラストにすればいいんだろうね？

目次(第三章)

第十三話『れっつくっきんぐ！』

- もう一度ハロワ。
- これはもう右のコードを見なくてもできるはず。
- （特に言うことは）ないです。

```
#include<stdio.h>

int main(void){
        int x;

        printf(“Hello, world!¥n”);

        scanf(“%d”, &x);

        return 0;
}
```

(第一話『ハロワは楽しい!!』より)

第十四話『料理のための下準備』

- ハロワを料理するよ。
  - ちょっとハロワは脇に置いといて
- 料理のための下準備だよ。

- 今までは、main関数のみを使っていた。
- ここで登場するのが「**自作関数**」というもの。
  - 実は第一章で登場済み。
- 今回の自作関数は「process()」という名前。
  - この名前は開発者自身で**任意に決定できる**。
    - 業務命令のある場合はこの限りではないかもしれない。

- なぜこれを使うのか？
  - 可読性のため
  - 変更があった時の作業量低減のため
  - 詳しくは後述。

```
int process(){
        int x = 5;
        return x;
}


int main(void) {
        int a;
        a = process();
        printf("%d", a);


        return(a);
}
```

第十五話『下準備に必要な材料』

- （新たなものは出てこ）ないです。
- 自作関数を作る方法は、実はmain()と変わらない。

- 第一章で自作関数の話をしたことは覚えてる？
  - たぶん「？？？」だったと思うんですよ。
- ここでも関数の話を詳しくするのは割愛します。

- 自作関数に必要な材料は、以下の二つだけ。
  - **変数**
  - **戻り値**
- main()との違いは、**戻り値が必須**なこと。
  - main()に戻り値が必要かは環境に依る。

```
int process(){
        int x = 5;    //変数
        return x;     //戻り値
}

int main(void) {
        int a;
        a = process();
        printf("%d", a);

        return(a);
}
```

第十六話『材料を要求する下準備』

- 自作関数process()に注目。
- 第十五話で出てきたprocess()との違いを比べてみて。
- process()の中で変数の宣言と初期化を行っていない！
- process()の中で使う変数を引数として要求しているから。

- 引数とは何か？　←ここでは int q
  - 材料（自作関数）が自分自身で宣誓したくない要素。
    - 鴨が「葱を寄越せ」と言っている感じ。
    - 第十五話の場合は「鴨が葱を背負って来た」感じ。

- なぜ引数を要求すると宣言と初期化をせずに済むのか？
  - main()で既に宣誓された要素を受け取るから。
    - 自作関数は鴨が貰った葱を自分で調理して返してくる感じ。

- 呼び出す関数側で宣誓されてない要素は渡せない。
  - とても説明が面倒なのでこの辺はマナーとして後述。

```
int process(int q){        //引数としてint型の値を一つ要求する
        q = q – q + 946 – 4649;
        return q;
}

int main(void){
        int ans = 0;
        int qwe = 10;

        printf(“%d”, ans);
        ans = process(qwe);      //引数として変数qweを与える
        printf(“%d”, ans);
        return 0;
}
```

第十七話『下準備に下準備したものを使う』

- main()から自作関数を呼び出せる。
- 自作関数から自作関数を呼び出すこともできる。
- 自作関数からmain()を呼び出すことはできない。

- 自作関数が自分自身を呼び出すこともできる。
  - 大抵プログラムが無限ループするから非推奨。
  - 無限ループ対策出来るなら有能。

- たけしし何かコメントは。

- 「TOKIOのみなさんで世界一うまいラーメンを作ってください」
  山口達也「それはどういうレベルでつくるの？ 小麦から？」

```
void go_around(){
        go_around();
}

void merry(){
        go_around();
}

int main(void){
        merry();
}
```

第十八話『れっとあすくっきんぐ！』

- ねぇ、知ってる？
  “Let’s”は“Let us”の省略形なんだよ。
  じゃあ”Let”の意味は知ってる？

- ここまでで自作関数については以上です。
- 第二章もここで半分。

- 自作関数の利点は簡単に前述（第十四話）した。
- くわしくはたけしし。

```
#include<stdio.h>
void Say_Hello(){
        printf(“Hello, world!”);
}

int main(void){
        int x;
        Say_Hello();

        scanf(“%d”, &x);
        return 0;
}
```

第十九話『既に下準備されているものを使う』

- 前回までは自作関数の呼び出しなどを学んだ。
- 今回はヘッダファイルの中にある関数を呼び出す。
- srand((unsigned int)time(NULL));
- srand() 関数：stlib.h
  - time() 関数：time.h　←srand()の引数として呼び出す
    - NULL：空。情報がない。　←time()の引数として渡す

- (unsinged int)
  - 単純な二進数(10進数でいうところの自然数)に変換
    - 第一章の整数型の所で雑にせつめい済。

- srand((unsigned int)time(NULL));

```
#include<stdio.h>
#include<time.h>
#include<stlib.h>
int main(){
        int z, i;
        srand((unsigned int)time(NULL));

        for(i=0; i<10; i++){
                z = rand();
                printf("%d", z);
        }
        return z;
}
```

第二十話『一人の可能性は一つじゃない・前篇』

- タイトルに統一性がない（諦観）
- 第一章の第四話、第六話辺りを見返してほしい。
- if(…){

  …

  }

  else if(…){

  …

  }

  else{

  …

  }

- 記述が面倒臭いと思った、そこのあなた！
- そんなあなたのためのswitch文。
- どうやるかは次ページ。

```
#include<stdio.h>
int main(void){
        int x=-1; // 初期値を一定に
        while(x != 0){ // 0でない場合
                printf("あなたは何番になりたいの？¥n");
                scanf("%d", &x);
                if(x == 1) printf("2番じゃダメなんですか？¥n");
                else if(x == 2) printf("1番になりたくないの？¥n");
                else if(x == 3) printf("表彰台で一番低い所なの？¥n");
                else if(x > 3) printf("表彰台にすら上らないの？¥n");
        }
        printf("そう、オンリーワンが大切よ。¥n");
        scanf("%d", &x);
        return 0;
}
```

第六話『メグルメク輪廻』より

第二十話『一人の可能性は一つじゃない・後篇』

- 行数は増えたものの、内容は簡潔になった。
  - でしょう？でしょう？でしょう。
- switch文を使える条件は以下の通り。
  - **参照する変数がひとつだけ。**（今回は x を使用）
  - その変数が実数ではない。（=double**型ではない**）
    - 要はint型かchar型で使える。
  - xの値が１**以下**の時、２**以下**の時…
    - ここ注意。break**を入れないと以降のケースを実行する。**
- 文字の場合はASCIIコードを参照する。
  - 要は「結局数値（コード番号）で判定できる」。

```
#include<stdio.h>
int main(void){
        int x=-1; // 初期値を一定に
        while(x != 0){ // 0でない場合
                printf("あなたは何番になりたいの？¥n");
                scanf("%d", &x);
                switch(x){
                        case 1: printf("2番じゃダメなんですか？¥n");
                                break;
                        case 2: printf("1番になりたくないの？¥n");
                                break;
                        case 3: printf("表彰台で一番低い所なの？¥n");
                                break;
                        case 4: printf("表彰台にすら上らないの？¥n");
                }
        }
        printf("そう、オンリーワンが大切よ。¥n");
        scanf("%d", &x);
        return 0;
}
```

第二十一話『四次元ポケット』

- 自作関数より先に説明すべきだったかも。
- 複数も同じような値を使うときにこれは面倒さ。
  - Ex: 座標の値、ステータス、もろもろ…
- てんでバラバラなところにある箱に値を入れるイメージ。
- じゃけん一元管理しましょうね～

```
#include<stdio.h>

void main(){
        int a1, a2, a3, a4;
        a1 = 32;
        a2 = 53;
        a3 = 67;
        a4 = 1005;

        printf("%d, %d, %d, %d,¥n", a1, a2, a3, a4);
}
```

第二十一話『四次元ポケット』

- はい。
- 配列です。
- 一列に並んだ箱に順繰りに値を入れていくイメージ。
- この辺の説明は図があるとわかりやすいので図ができてから。（説明の放棄）

- 本題はここから。
  - ページ分けするべきかもしれない
- a[][][][];
  - n次元まで配列化できます。（nは任意の自然数）
  - やりすぎると面倒なだけ。

```
#include<stdio.h>


void main(){
        int a[4] = {32, 53, 67, 1005};
        int i;
        for(i=0; i<4; i++){
                printf("%d", a[i]);
        }
        printf("¥n");    // puts(" "); でも許された気がする
}
```

第二十二話『料理のワンポイント』

第二十三話『ワンポイントポケット』

第二十四話『ぺんぱいなっぽーあっぽーぺんぽいんと』