Algo-semi
No2

本日の予定

- 前回の復習
- Github
  - 初めてのpull and push
  - Gitignoreについて
- C言語
  - Includeとは
  - Printfを使って出力しよう
  - 変数・型とは何か？
  - キーボード入力を読み取ろう
  - 配列を使ってみよう

・slackについて

前回の復習

- 前回はCがコンパイルできる環境を作った
- gcc filename.c –o outputName
- その後 ./outputName で実行できたね

- Githubではリポジトリをfork(複製)して、自分のローカルリポジトリにclone(コピー)した
- 変更が加えられたファイルをcommitしたね
- Commitにはsummary(題名),description(内容)を入力できたね
- 必ずsummaryだけは入力しよう

github~初めてのpull and push

Pull　ネットワークのリポジトリをローカルのリポジトリに引っ張る
　　簡単にいえばネットの変更をローカルに更新する
Push　ローカルのリポジトリをネットワークのリポジトリに押し付ける
　　簡単にいえばローカルの変更をネットに更新させる

githubではsyncボタンで一発だ!!

Github~gitignoreで除外設定をしよう~

この前exeファイルをcommitしたね
でもこれっていらないファイルなんだ!!
でもchangeに登録されちゃう
そんなときはgitignoreを編集しよう

編集する前に…

メモ帳やワードでプログラミングするのはダサいので、エディタをインストールしよう

Sublimetext

https://www.sublimetext.com/3

Atom

https://atom.io/

今回はこの中から選んでね

他にも色々あるので自分の好みのを見つけよう

Github~gitignoreで除外設定をしよう~

インストールしたエディタでgitignoreを開いてみよう
#*exeと書かれた行があるね
#はコメント行なので*.exeが無視されている
今回は説明のために#を付け足したので、これを消して保存してみよう
とりあえずcommit
これで、exeファイルはchangeに保存されないはず
hello以外のファイル名を指定してコンパイル
Githubを見てみるとchangeに表示されないね

C言語~Hello.cの中身~

#include <stdio.h>
このコードはStandard input output library . Header を
含むといった意味これを入れないとprintfが動かない

int main(){
C言語は最初にmain関数を読み込むように決められている
Intはint型という型の一種
C言語において関数は
型名 関数名(){}
と書き、()中は引数、{}中にメインの内容を書いていく

C言語~Hello.cの中身~

```
printf("Hello World¥n");
```
これは画面に出力する基本的な関数のひとつ
print(“hogehoge %*¥*”,変数名);
標準入力出力では%や¥(or バックスラッシュ)と合わせていくつかオプションを使用することができる
%は変数を扱うときに使用する
¥(or バックスラッシュ)は改行やタブなど… 今回は¥nで改行の意味

```
return 0;
```
これはなるべく入れておくと良い

```
}
```
関数は波括弧で閉めよう

C言語~¥オプション~

エスケープシーケンス	意味	ASCIIコード（16進）
¥n	復帰改行	0A
¥a	警報音	07
¥t	タブコード	09
¥b	バックスペース	08
¥¥	文字としての ¥	5C
¥'	文字としての '	2C
¥"	文字としての "	22
¥0	文字列終了コード	0

C言語~%オプション~

変換指定文字	意味	使われるデータ型
%c	1文字として出力する	文字型
%d	10進数で出力する	整数型
%x	16進数で出力する	
%o	8進数で出力する	
%f	[-]dddd.dddddの形式で出力する	浮動小数点型
%e	指数形式で出力する	
%s	文字列として出力する	文字列
変換指定文字	意味	使われるデータ型

C言語~Hello.cの中身~

色々いうよりかは動かしたほうが覚えると思うのでいくつか試してもらいたいと思います

1.出力　Hello World をHello Lelab に変更

2.出力を　　Hello World (改行)

　　Hello Lelab に変更

3.#include <stdio.h> を削除して実行

C言語~header fileの中身~

#include <stdio.h>
先ほども言ったとおりこれがなければprintfは動かない
同ディレクトリ上にあるstdio.hの中を見てみよう
これはコンパイラの中に入っているのをコピーしたもの
215行目にprintfについての記載がある
headerFileはライブラリではなく、ライブラリの場所を示している

C言語~header fileの中身~

まあ、headrFileをincludeしなくても、それについて記載すれば動く
helloIncStdio.cはhello.cの先頭にstdio.hの中身を追加したもの
当然コンパイルできる
基本的にはstdio.hやstdlib.h辺りが良く使うやつ
詳しいことは適時検索しよう

C言語~変数の型~

型指定	データ型	バイト幅	扱える数値の範囲
char	文字型	1	-128～127
int	整数型	2	-32768～32767
long	倍長整数型	4	-2147483648～2147483647
float	単精度浮動小数点型	4	3.4E-38～3.4E+38
double	倍精度浮動小数点型	8	1.7E-308～1.7E+308

基本的に使用するのは char,int,floatの3種類
char型は文字型でa-zなど文字を扱うことができる。
*日本語は1byteではないのでchar型に入れてもエラーが出る
int型は整数型、整数を扱うときに使う
*少数を入れると丸められるので気をつけよう
float型は少数型、精確性に欠けるのであまり使わないほうが良い

C言語~変数の型~

practice/0419/printf.cを開いてみよう
それぞれint,float,charについて例が用意されている
とりあえず実行してみよう

C言語~変数の型~

char型は1byteまでなので、一文字しか入れることができない

なので文字列を入れたいときは配列を使用しよう

配列の基本形　　型　変数名[*]

型はintやcharなど　変数名の後の[]には配列の個数を入れる(省略可)

個数に関しては多めに宣言しておくと楽

配列の一文字目は[0]から始まることに注意

C言語~scanfについて~

printfとは反対の入力、つまりキーボードの入力を読み取る関数
言うより試してみよう
scan_char.cをコンパイル、実行
入力された文字を返答してくれるコードになっている
scan_numをみてみよう
入力を2つ読み取る
2つ目の入力に移るときはenterを押す

C言語~scanfについて~

キーの入力ができるなら色々できそうな気はするが、実際の所あやふやでエラーばっかり吐くので、多用するとデバッグが辛いのが現状

scan_numに文字列を入れると…

詳しいことは後ほど説明するが、char型の文字列を読み取るときは

scanf(“%s”,varName);

int型の数字列などを読み取るときは

scanf(“%d”,&varName);

といったように&が必要になる

C言語~比較演算子について~

演算子	意味	使用例
>	より大きい	if (a > b)
>=	より大きいか、等しい （以上）	if (a >= b)
<	より小さい	if (a < b)
<=	より小さいか、等しい （以下）	if (a <= b)
==	等しい	if (a == b)
!=	等しくない	if (a != b)
&&	論理積（かつ）	if (a > 0 && b > 0)
||	論理和（または）	if (a > 0 || b > 0)
!	否定（でない）	if (!a)

C言語~条件分岐について~

if(a*b){~}else{~~}	aとbの関係*(比較演算子)が真ならば~を偽ならば~~を実行する
for(~;~~;~~~;){=}	初期条件~について~~ならば~~~を実行し、~を実行する
while(~){~~}	~が真ならば~~を実行する
do{~}while{~~}	~を実行した後、条件~~が真であれば再実行する
switch * case:~ case~~	変数*について、値が~であれば下記の内容を実行し、~~であれば…
break	ループを断ち切る

本日の課題

- task/0419をやってくること
- わからない所は調べる。調べてわからなければヒントを聞く。
- 重要なのはエラー吐かれても諦めない心、適切なワードでググる技能、キーボードを叩き続ける体力
- エラーの対象法としては、何行目に出ているかをチェックする
- 打ち間違えはないか、入力し忘れはないかをチェックする
- 特に”;)}などは忘れる
- 上記についてはインデントを整えると発見しやすい
- 最終的にはエラーをコピッてググる

本日の課題

- 質問するときはgitにpushしておき、URLを張ると良い
- slackに直接ファイルを張ることもできるが、乱雑にファイルが置かれるのはいやなので、できる限り避けること
- 最低一回は入門C的なサイトに目を通すこと