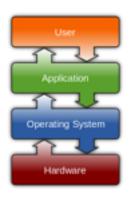
パソコンを自由自在に動かしたい。

パソコンがどう動いているかを知る必要がある。



結論 Windowsにおいては、exeファイル(実行ファイル)を作ればいい。

exeファイルの作り方

プログラミング言語で書かれたソースファイルを作成し、 コンパイラと呼ばれるアプリケーションでコンパイルする…だけじゃない 実はもう少しめんどくさい

これをまとめてやってくれるのがVisualStudioなどの統合開発環境(IDE)

結論 VisualStudioを使ってコード書いて、ビルドボタンを押すだけ。

なぜC#をやるのか

アルゴリズムとは何か?

「やりたいこと」を実現するための方法を、 「**すでに出来ること」**の「制御」で記述したマニュアル

例. 大学へ行くアルゴリズム

起きる

朝食をとる

時間に余裕がなくなるまでスマホをいじる

もし雨なら傘を持つ

家を出る

...

「すでにできること」とは何か?

言語によって違うけど、大体の高級言語ができることは

変数、配列を用意する

変数、配列に文字や数字を代入する

四則演算

数字の大小を比べる

文字が一致するか調べる

他人の作った関数を呼び出す ←結局コレ

今日使う関数

print()

read()

「制御」とはどんなものか

if もし○○なら~、そうでなければ~

while ○○となるまで~をくり返す

その他 非同期処理など

変数とは

```
データを持ち運ぶための名前の付いた箱
最後に代入されたデータとして使うことができる
```

```
例. a = 2;  //aに2を代入 b = 3;  //bに3を代入 a = a + b;  //aにa + b(2 + 3)を代入
```

配列とは

変数を一列に並べたようなもの 配列のn番目は変数のように使うことができる

```
例. A = {1,-2,5};
print(A[0]); //1が表示される
A[0] = A[1] + A[2] //3(-2 + 5)を代入
print(A[0]); // 3が表示される
```

制御文の使い方をもう少し詳しく

次のような擬似言語を定義します。

ただし、<*>は日本語の意味するところであり、[*]は置き換えて記述する。#*#は省略可能である。 [変数]が[整数]として使われる場合は最後に代入した値となる。

見やすいようにスペースや改行を入れて良い。

```
[複文]
```

```
{[文][文][文]...}
```

[文]

```
[命令文] [構文] [構文] if[命題]{[複文]}#else{[複文]}# print([整数]); while[命題]{[複文]} read([変数]);
```

[命題]

([整数][比較演算子][整数])

[整数]

<数字>

[変数]

([整数][四則演算子][整数])

[配列]の要素数

[変数]

<小文字のアルファベット>

[配列][[整数]]

[配列]

<大文字のアルファベット>

[演算子]

[四則演算子]	[比較演算子]	
+		>
-		<
*		>=
/		<=
%		==

例. 配列Aが与えられたとき、配列の要素を全て出力する複文

```
{
    i = 1;
    while(i == Aの要素数)
    {
        print(A[i]);
        i = (i+1);
    }
}
```



```
答え
問1
{
       c=a;
       a=b;
       b=c;
}
問2
{
       read(a);
if(Aの要素数>a)
              print(A[a]);
       }
}
問3
{
       a=A[0];
      i=1;
<mark>while</mark>(i==Aの要素数)
              if(A[i]>a)
                     a=A[i];
              }
```

print(a);

}