構造体と乱数

田中

構造体とは！？

- 複数の変数を一つにまとめたセットを作れる。これによってデータの管理が簡単になる！
- 例　人データの構造体

人データ

名前
性別
身長
体重

構造体の書き方

```
struct 構造体の名前{

        メンバー変数・・・・（いくつでもOK）

};
```

typedefとは・・typedefを使うことによって型の名前を決定できる！ 今回の場合はstructの名前を決定するのに必要な記述

例　人データの構造体

人データ
　名前
　性別
　身長
　体重

```
Struct Person {
        char* name;
        char* sex;
        double height;
        double weight;
};
```

赤文字で書かれたPersonとはこの構造体の型の名前です！

早速ここで作成した構造体を使ってみよう

```
struct  Person{
        char* name;
        char* sex;
        int height;
        int weight;
};

void main(){
        Person tarou;
        tarou.name=“太郎”;
        tarou.sex=“male”;
        tarou.height=170;
        tarou.weight=55;
}
```

先ほど名付けたPersonはあくまで型名なので、main関数のなかで構造体の名前を宣言しなければならない。

このソースコードの中では赤文字で書かれた**tarou**が構造体の名前となっています。構造体の中身の変数にアクセスするためにはドット演算子を使います。横の図では

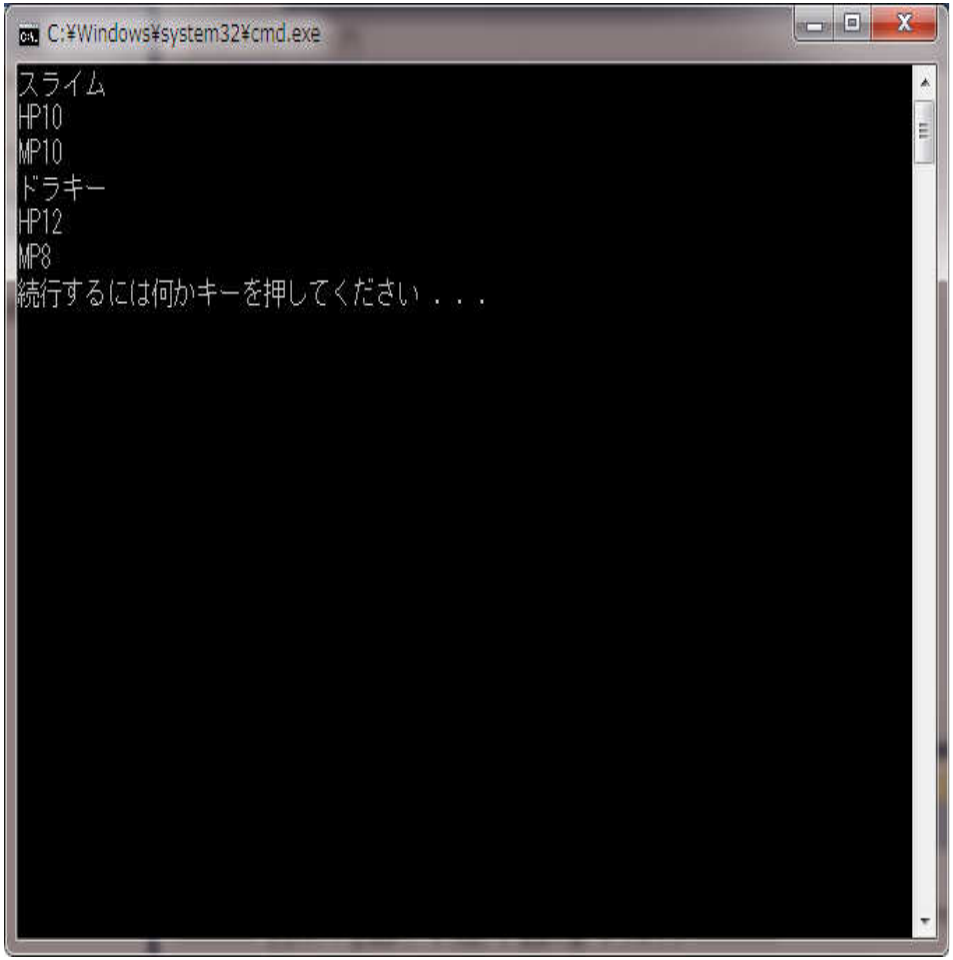
tarou.name=“**太郎**”

といった形で、構造体tarouに固有の値を与えています。

なお、同じ構造体の型でも、名前によって中身の値を区別することができます。

問題

char* name
int HP;
int MP;
をメンバーに持つ、一つの
構造体を使用して
名前”スライム”を持つslime、
名前”ドラキー”を持つdrakee
とそれぞれのパラメータを設
定して右記のように印字してみよう



```
#include <stdio.h>

Struct Status{                          //構造体宣言
char* name;
int HP;
int MP;
};

void main(){
Status slime;
slime.name = “スライム”;       //名前を設定
slime.HP = 10;                         //HPを設定
slime.MP = 10;                         //MPを設定

printf("%s¥nHP%d¥nMP%d¥n", slime.name,slime.HP,slime.MP);

Status drakee;
drakee.name = "ドラキー";
drakee.HP = 12;
drakee.MP = 8;

printf("%s¥nHP%d¥nMP%d¥n",drakee.name,drakee.HP,drakee.MP);
}
```

便利な構造体初期化方法

先ほどのプログラムを例にすると
Status slime と宣言した時に、配列の初期化と同じように

Status slime={“スライム”,10,10};

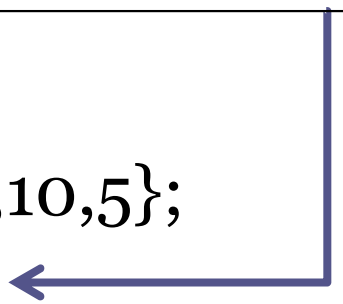
と一気に変数を初期化してあげても大丈夫です。

構造体の代入

```
Struct Status{
        char* name;
        int HP;
        int MP;
};

Void main(){
        Status slime1={“スライム”,10,5};
        Status slime2=monster1;
}
```

ここでmonster1の値がmonster2に代入される。
同じ構造体の型でないと代入不可能（ここでは構造体Status同士でないと代入不可能)



構造体の配列

```
struct Status{
        char* name;
        int HP;
        int MP;
};

void main(){
        Status monsterparty[3];
        monsterparty[0]={“スライムA”,10,10};
        monsterparty[1]={“スライムB”,10,10};
        monsterparty[2]={“ドラキー”,12,8};
}
```

乱数も作ってみよう

構造体はゲームなどの、キャラクター個体別にパラメータを設定するにはうってつけの機能でした。

次に、ゲームを作るには必須の乱数生成の方法を紹介します。

乱数を作る＜標準ライブラリ関数＞

今まで

#include<stdio.h>

と何もわからず書いていたと思いますが、これはprintf()やscanf()などの入出力用のライブラリです。

ちなみに#include<>とは<>内のライブラリを使用しますという宣言です。

乱数を使うためにincludeしなければいけないライブラリ

乱数を使うには stdlib.h ライブラリを#includeしてあげる必要があります。

プログラムの文頭に#include<stdlib.h>を付けることで乱数の機能、rand()関数を使用できるようになります。

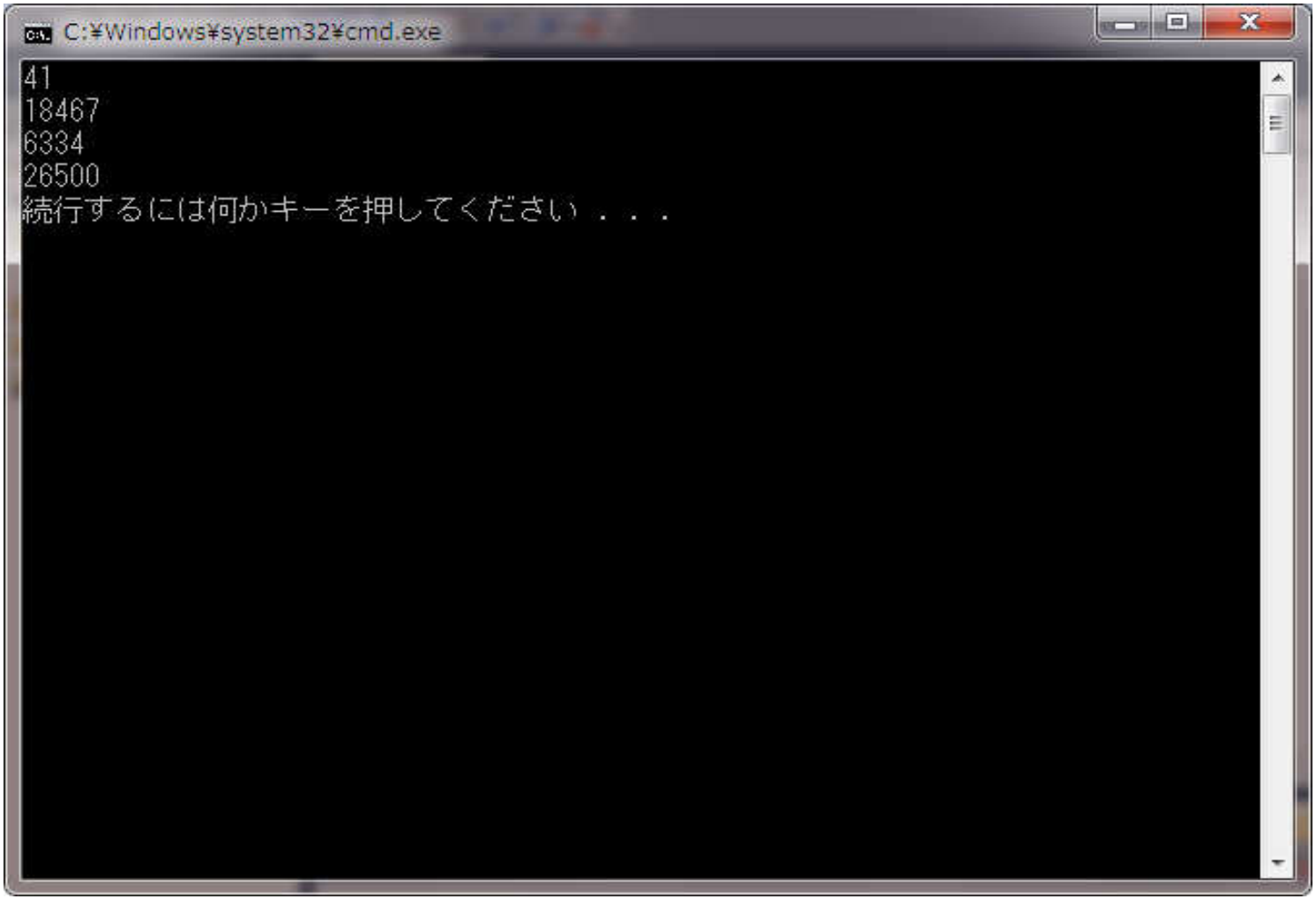
乱数を使ったプログラム

```
#include <stdio.h>
#include <stdlib.h>

void main(){
        int i;
        for(i=0;i<4;i++){
                printf("%d¥n",rand());
        }

}
```

実行結果（乱数の範囲指定なし）



乱数の範囲を指定するには

最小値 + (int)(rand() * (最大値 - 最小値 + 1.0) / (1.0 + RAND_MAX))

(int)とはランダムの結果で浮動小数点数が弾き出されても、少数点を切り捨てて整数に直すために使われています。これをキャストといいます。

RAND_MAXとは、ライブラリで定義してあるこの関数が出すことのできる乱数の最大値で、定数（変更できない数）です。

１～１０の乱数を印字

```
#include <stdio.h>
#include <stdlib.h>

int GetRandom(int max,int min);

void main(){
	int i;
	for(i=0;i<6;i++){
		printf("%d¥n",GetRandom(10,1));
	}

}


int GetRandom(int max,int min){  //整数のランダムの結果を返す関数
	return min + (int)(rand()*(max-min+1.0)/(1.0+RAND_MAX));
}
```

問題

構造体を使って

名前“スライム”
HP 20

のスライムを作り、更にそのスライムのHPを、
１～５の乱数を生成して徐々に減らしていき、スライムのHPが０になったら終わるプログラムを作ろう

答え

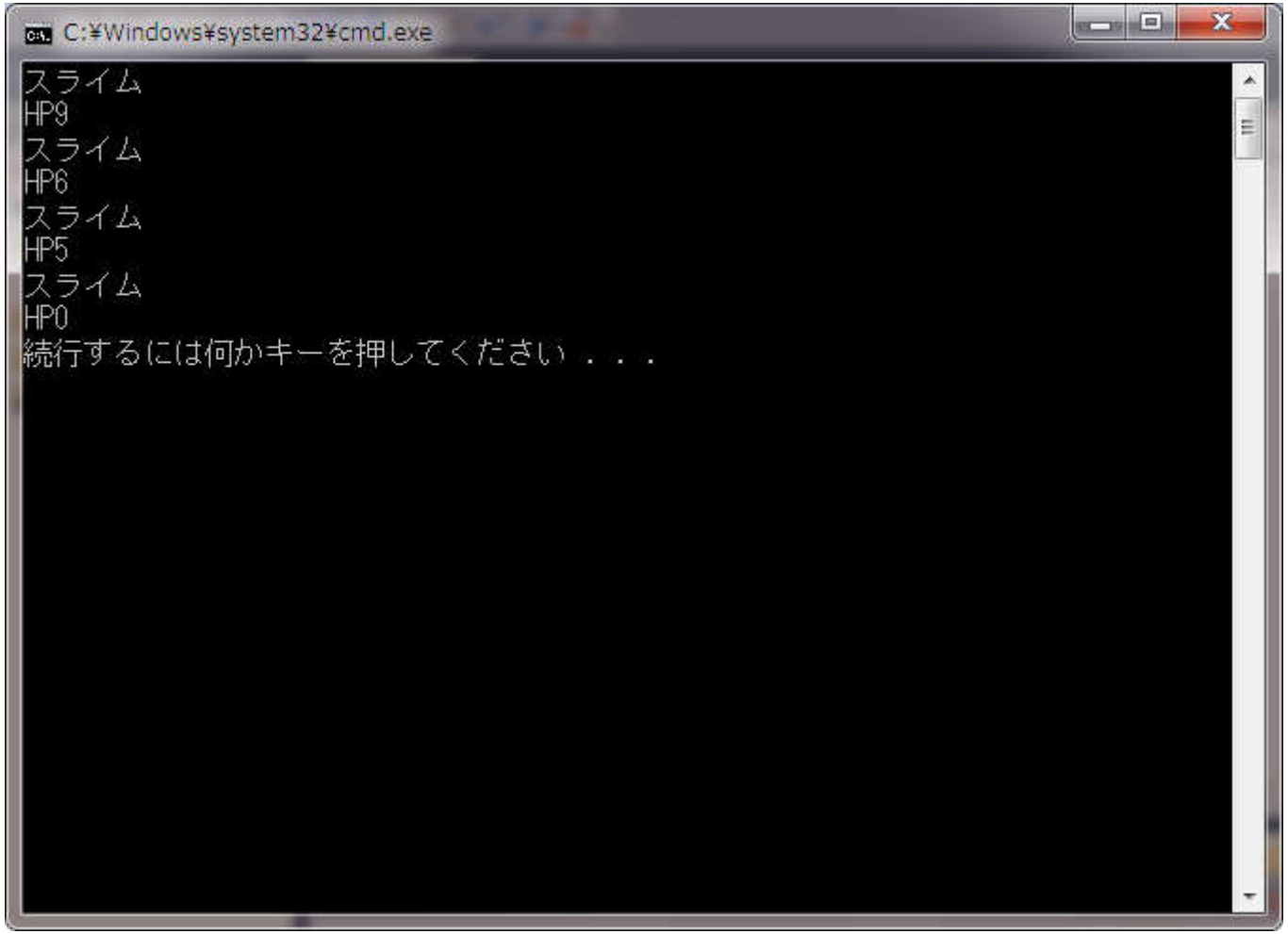
```
#include <stdio.h>
#include <stdlib.h>

int GetRandom(int max,int min);
struct Status{
        char* name;
        int HP;
        int MP;
};

void main(){
        Status slime={"スライム",10,10};
        while(slime.HP>0){
                slime.HP-=GetRandom(5,1);
                if(slime.HP<0){
                slime.HP=0;
                }
        printf("%s¥nHP%d¥n",slime.name,slime.HP);
        }
}

int GetRandom(int max,int min){  //整数のランダムの結果を返す関数
        return min + (int)(rand()*(max-min+1.0)/(1.0+RAND_MAX));
}
```

結果


C:¥Windows¥system32¥cmd.exe
スライム
HP9
スライム
HP6
スライム
HP5
スライム
HP0
続行するには何かキーを押してください . . .

しかし、これでは・・・・

何度も実行すればわかるが、このままだと何度実行しても値が同じになってしまう。

これは、乱数を計算するrand()関数の計算パターンが同じなため起こってしまっている。
それを解消するために、rand()が計算で使う元の値を変えなければならない。

srand関数(<stdlib.h>に入ってる)

srand関数はカッコ内の数字を乱数生成の元の数字に設定してくれる。
例えばsrand(5)とすると、5の数字を元にした乱数生成パターンを生み出してくれる。

しかし、さきほどと同じようにパターン固定現象は解決していない

時間に基づいた乱数生成式を作る

時間に基づいた処理をする場合は
<time.h>をインクルードします。
#include<time.h>をすれば、以下の魔法のコトバをsrandに入れるだけで時間に基づいた乱数（秒ごとに結果が変わる）が生成されます

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

先ほどのスライム討伐プログラムに
時間に基づく乱数を搭載してみよう

実行する度に値が変わるはず！！！

まとめ！

今回、stdio.hだけでなく、複数のライブラリを使いました。

このように、たくさん便利なライブラリがあるので、自分で調べて使ってみましょう！

例えば、ゲームを作る時に不可欠な

◯秒ウェイトする。といったような処理もライブラリを駆使すれば実現できます。

とにかく自主的にプログラムを書こう

- 本を買ったり、Webで見て学ぼう
- いきなり大きなプログラムに挑戦するのではなく、例えばRPGゲームを作りたいとしたら戦闘シーンから作るなど。
- アルゴリズムを学ぶ！　アルゴリズムの問題を解いたりしよう