第５回 Ｃ言語講座

１．構造体について
構造体とは、異なる型のデータをまとめて管理する事が出来るものです。
異なるデータを１つの型として定義し、扱うことができます。
サンプルコード

```c
#include<stdio.h>

#define STRINGLENGTH 64

typedef struct schoolMember
{
    char college_name[ STRINGLENGTH ];
    int grade;
    char belong[ STRINGLENGTH ];
    char name[ STRINGLENGTH ];
}smem;

int main()
{
    smem people1;

    printf( "１人分の情報を構造体に格納します¥n" );
    printf( "大学名を入力してください¥n" );
    scanf( "%s" , people1.college_name );

    printf( "学年を入力してください（整数値）¥n" );
    scanf( "%d" , &people1.grade );

    printf( "所属学科を入力してください¥n" );
    scanf( "%s" , people1.belong );

    printf( "名前を入力してください¥n" );
    scanf( "%*c%[^¥n]" , people1.name );

    printf( "¥n/*--- 入力された情報 ---*/¥n" );
    printf( "%s %s %d年 %s¥n" , people1.college_name , people1.belong , people1.grade , people1.name );

    return 0;
}
```

まず、構造体を宣言し、定義します。
それが、「typedef struct～～」の部分です。
こんなふうにすると使い易いです。
（実は構造体と typedef と呼ばれるものを組み合わせています）
こうすることで、構造体を普通の型と同じように扱うことが出来ます。
int [名前]のように、smem [名前]として変数を定義するように扱えます。
また、構造体内部の変数にアクセスしたい場合は、
[構造体で定義した名前].[構造体内部の名前]のようにします。

さらに、構造体は、変数と同じように代入することが可能です。
例

```
smem people1 = { … … … …  };
smem people2;

people2 = people1;
```

また、構造体も配列で用いることが出来ます。
ゲーム的要素を少しだけ加えたサンプルコード。
毎回、画面をクリアして描画しています。
esc キーで終了します。

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<Windows.h>

#define SHOTMAX 32

#define WIDTH 480
#define HEIGHT 320

typedef struct SH
{
    double x , y;
    double vx , vy;
    int power;
    int kind;
    int flag;
}shot_data;

int main()
{
    shot_data enemy_shot[ SHOTMAX ];
    int end_flag = 0;

    //初期化
    for( int i = 0 ; i < SHOTMAX ; i++ )
    {
        enemy_shot[ i ].flag = 0;
    }

    //敵弾を３個生成する
    for( int i = 0 ; i < 3 ; i++ )
    {
        int check = 0;
        for( int j = 0 ; j < SHOTMAX ; j++ )
        {
            if( enemy_shot[ j ].flag == 0 )
            {
                enemy_shot[ j ].x = (double)WIDTH / 2 - 20 + 50 * j;
                enemy_shot[ j ].y = (double)HEIGHT / 2;
                enemy_shot[ j ].vx = 0;
                enemy_shot[ j ].vy = 5;
                enemy_shot[ j ].flag = 1;

                check++;
                break;
            }
        }
    }

    //メインとするループ
    while( end_flag == 0 )
    {
        //画面初期化
        system( "cls" );
        //計算部分
        for( int i = 0 ; i < SHOTMAX ; i++ )
        {
            if( enemy_shot[ i ].flag == 1 )
            {
                enemy_shot[ i ].x += enemy_shot[ i ].vx;
                enemy_shot[ i ].y += enemy_shot[ i ].vy;
            }
        }
        //描画部分
        for( int i = 0 ; i < SHOTMAX ; i++ )
        {
            if( enemy_shot[ i ].flag == 1 )
            {
                printf( "%d番 x : %lf y : %lf¥n" , i , enemy_shot[ i ].x , enemy_shot[ i ].y );
            }
        }
        //待機
        Sleep( 30 );

        //esc入力で終了
        if( _kbhit() != 0 )
        {
            if( getch() == 0x1b )
            {
                end_flag = 1;
            }
        }
    }
    return 0;
}
```

２．動的確保について
配列やポインタ、定数などは定義したときにメモリを確保していきます。
特に配列は確保すると明示的に開放しないと、プログラムを終了しない限りメモリ領域を取り続けます。
例えば、弾幕ゲームで弾幕用の構造体を配列確保する場合があります。
これは、管理がとても簡単なのですが、起動時にメモリを確保してしまい、他のアプリの邪魔になる
といったことが起こりえます。
そこで、動的確保という必要なときに必要なだけメモリを確保する、という方法があります。
ただ、配列に比べ管理が難しく、基本的にメモリは潤沢にあるので
（ノートパソコンですら４ GB が普通だったり）
配列でいいのではないでしょうか。
C 言語では malloc()などを利用して確保します。
興味がある人は自分で調べてみるといいよ。

ヒント。
動的確保は、必要になるたびに malloc を使い確保するのだけれど、
前に確保したものと何らかのつながりが無いと上手く扱えません。
（用意だけして、何処にあるのかわからなくなったら困るよね）
基本的に、malloc で確保するのは構造体、構造体の中には次の構造体を示すアドレスを
代入出来るポインタ変数を入れておき、新しく確保するたびに次のアドレスを入れて行く、
という方法があります。
ぶっちゃけ厄介です。
が、アドレスという概念の勉強には最適なのかもしれません。
（キーワードとしては「連結リスト」とかかな）

＊オマケ
この動的確保ですが、弾幕ゲームで利用する場合、
静的確保するよりも少ない回数で描画チェックが出来ます。
弾幕ゲームにおいて、静的確保によって各弾幕（弾１つ）の情報を保持すると for 文で静的確保した
全ての構造体のフラグをチェックし、フラグのあるものだけを描画する、というふうに、
処理に手間が掛かる訳です。
対して、動的確保の場合、弾幕が生成されたときに情報がリストの末端に付加されるので、
リスト内にある弾幕は全て描画する、つまり if 文によるチェックがいらないわけです。
パソコンは、if 文などの条件分岐は単純計算やメモリ書き込みに比べ遅くなってしまいます。
（といっても人間にわかる程度ではありませんが）
ただ、CPU の速度は十分に早くなっているので、そんなに気にする必要は無いとは思いますよ？

3．アロー演算子

構造体をポインタで扱わなければ行けない場合というときが出てきます。
このときには、アロー演算子と呼ばれるものを利用していきます。
というわけでサンプルコード。

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

#define STRINGLENGTH 64

typedef struct schoolMember
{
    char college_name[ STRINGLENGTH ];
    int grade;
    char belong[ STRINGLENGTH ];
    char name[ STRINGLENGTH ];
}smem;

/*--- プロトタイプ宣言 ---*/
void inputState( smem* data );
void outputState( smem* data );

/*--- メイン関数 ---*/
int main()
{
    smem people1;

    inputState( &people1 );
    outputState( &people1 );

    return 0;
}

/*--- 関数本体 ---*/
void inputState( smem* data )
{
    printf( "学生データ保存します¥n" );
    printf( "１人分の情報を構造体に格納します¥n" );
    printf( "大学名を入力してください¥n" );
    scanf( "%s" , data -> college_name );

    printf( "学年を入力してください（整数値）¥n" );
    scanf( "%d" , &( data -> grade ) );

    printf( "所属学科を入力してください¥n" );
    scanf( "%s" , data -> belong );

    printf( "名前を入力してください¥n" );
    scanf( "%*c%[^¥n]" , data -> name );
}

void outputState( smem* data )
{
    printf( "¥n/*--- 入力された情報 ---*/¥n" );
    printf( "%s %s %d年 %s¥n" , data -> college_name , data -> belong , data -> grade , data -> name );
}
```

構造体配列を関数の引数にするときには必要になるので、使えるようになってくださいね。

練習問題

１．自機用の構造体（自機の座標と大きさ）と、
　　敵から射出される弾幕の構造体（弾の座標と大きさを格納できる構造体）を作成し、
　　当たり判定をする関数を作成しなさい。

２．学生の名前、学籍番号、５教科のテストの点数、合計点を保持する構造体を作り、５人分の名前、
　　学籍番号、５教科のテストの点数を入力させ、合計点の高い順に表示するプログラムを作成しなさい。

ヒント

１．それぞれの構造体は、今の所 x、y、r の３つの変数があればいいよね。
　　（いずれ弾幕ゲームを作成する場合は、さらに情報を付加することになりますよ）
　　また、自機と弾の当たり判定は、自機の大きさを示す円と、弾の大きさを示す円が
　　重なっているかどうかで判断出来るよね。
　　数学的に言えば、２つの円の交点が１つ以上あればいいよね。
　　２つの円の中心座標の距離が、２つの円の半径以下だと交点が存在しますよ？

２．関数を使って実装できるといいなぁ。