プログラミング演習 第11回
構造体

目 次

1 目標 **1**

2 関数とポインタに関する復習 **1**

3 構造体 **3**

3.1 構造体の値渡し 4

3.2 構造体のアドレス渡し 6

4 実習課題 **8**

1 目標

構造体の使い方と関数への渡し方を覚える．

2 関数とポインタに関する復習

C 言語は多数の小さな関数から構成されます．関数への値の渡し方には，値渡し (Call by Value) とアドレス渡し (Call by Reference) の 2 種類がありました．Call by Value では，引数の値が呼び出された関数にコピーされて渡されます．

```
/* プログラム 11-1.c */
#include <stdio.h>

void swap(int, int);

main()
{
    int a, b;

    a = 3;
    b = 5;
    swap(a, b);    /* 関数 main から関数 swap を呼び出す */

    printf("a の値は%d, b の値は%d\n", a, b);
    /* a, b の値は変わらない */
}

void swap(int a, int b)
{
    int c;

    c = a;
    a = b;
    b = c;

    return;
}
```

上の例では，引数a,b の値が関数 swap の中の変数a,b にそれぞれコピーされて渡されますが，main の中の変数a,b にはそのままです．従って，swap の後もa=3, b=5 のままです．

これに対し，Call by Reference では，引数にアドレスを用いますから，呼び出された関数にアドレスをコピーして渡します．アドレスの中身を呼び出された関数で書き換えることにより，呼び出し側の変数の値を変更出来ます．

```
/* プログラム 11-2.c */
#include <stdio.h>

void heisei(int *); /* 引数は int 型へのポインタです */

main()
{
    int a;

    a = 2000;

    heisei(&a);    /* &をつけてアドレスを渡す */

    printf("aの値は%d\n", a);  /* a の値が書き変わった！ */
}

void heisei(int *ptr_a)
{
    *ptr_a -= 1988;    /* アドレスの中身の参照は * をつける */
    return;
}
```

問 1 プログラム 11-2.c で出力されるa の値はいくつか

問 2 プログラム 11-1.c をアドレス渡しに書き替え，a とb の値が入れ代わって出力されるようにせよ．

また，配列を使ったときの配列名には配列の先頭番地のアドレスが格納されています．関数に配列そのものを渡すことは出来ませんが，配列の先頭番地 (すなわち配列名) を渡すことで，配列の中身を利用することが出来ます．

```
/* プログラム 11-3.c */
#include <stdio.h>

int strcpy(char s[], char t[]); /* ポインタを受け取ります */

main()
{
    char s[6], t[6] = "Hello";

    strcpy(s, t);

    printf("s = %s\n", s);      /* s を出力 */
}

int strcpy(char s[], char t[])
/* int strcpy(char *s, char *t) と書いてもよい */
{
    while((*s = *t) != '\0') {
        s++;   /* s のアドレスを 1 つずつ進める即ち
                  s, s+1, s+2, s+3 ... となる      */
        t++;   /* t のアドレスも 1 つずつすすめる   */
    }

    return;
}
```

問3 プログラム 11-3.c は何を行うプログラムか

3 構造体

これまで扱ってきた変数のデータ型には次のようなものがありました．

(unsigned) char (符号なし) 文字型，1 byte = 8 bit

(unsigned) int (符号なし) 整数型，4 byte = 32 bit

double 小数 8 byte = 64 bit

char 型は文字を代入することが多いというだけで，1 byte で表わされる範囲の整数 (0 ~ 255 または-128 ~ 127) を扱うことも出来ました．この他にポインタ型 (4 byte) もありました (配列はポインタ型の一種です .)

これらのデータ型をまとめて１つの型にしたものが構造体です．基本データをスカラーと考えると，構造体はベクトルのようなものです．ベクトルの成分に対応するものは，メンバと呼びます．

メンバーがベクトルである構造体を考えましょう．この構造体の定義は main の前で次のようにします．

```
typedef struct vec2d {
    double  x;
    double  y;
} VEC2D;
```

これで新しい型 VEC2D が定義されました．

- struct の直後の文字列は構造体タグと呼ばれるものです．これは省略することも出来ますが，複雑なプログラムでは，構造体タグが必要になります．
- struct+(構造体タグ名) の後ろの中括弧 {,} で囲まれた部分は，既存の型を並べたもので，１つ１つをメンバと呼びます．ベクトルの成分に対応するものです．

3.1 構造体の値渡し

基本データ型の変数と同じように，関数の引数に渡した構造体は，呼び出された関数にメンバの値がすべてコピーされます．また，関数の型を構造体としておけば，返り値を構造体とすることも出来ます．

次のプログラムは，2 次元のベクトルの足し算を構造体で記述したものです．

```
/* 2 次元ベクトルの和 ( 構造体の値渡し ) 11-4.c */
#include <stdio.h>

/* 構造体を使用する変数，関数の前で宣言します */
typedef struct vec2d {  /* 構造体タグ vec2d は省略出来ます */
    double  x;      /* x 成分 */
    double  y;      /* y 成分 */
} VEC2D;

VEC2D add2dvec(VEC2D, VEC2D);

main()
{
    VEC2D    vec1, vec2;    /* 2 つのベクトルを用意 */
    VEC2D    vec3;          /* vec1 + vec2 を計算 */

    /* 構造体の初期化 */
    vec1.x = 2.0;
    vec1.y = 3.0;
    vec2.x = 4.0;
    vec2.y = 5.0;

    /* vec1 vec2 を足し算する関数 */
    vec3 = add2dvec(vec1, vec2);

    printf("vec1 の x 成分は%f, y 成分は%f\n", vec1.x, vec1.y);
    printf("vec2 の x 成分は%f, y 成分は%f\n", vec2.x, vec2.y);
    printf("vec3 の x 成分は%f, y 成分は%f\n", vec3.x, vec3.y);
}

VEC2D add2dvec(VEC2D vec1, VEC2D vec2)
{
    vec1.x += vec2.x;    /* vec2 の x 成分を vec1 の x 成分に加える */
    vec1.y += vec2.y;    /* vec2 の y 成分を vec1 の y 成分に加える */

    return vec1;
}
```

基本データ型の場合と同様，main の中の変数 vec1,vec2 は，関数 add2dvec の中の変数

vec1,vec2 にコピーして渡されますから，関数 add2dvec の中で，vec1 が書き替えられても main の中の変数 vec1 には影響ありません．

構造体をベクトルと例えましたが，普通のベクトルの演算のように，構造体同士の足し算を行うことは出来ません．

3.2 構造体のアドレス渡し

関数の引数に構造体のアドレスを渡すと，呼び出された関数に構造体の番地がコピーされます．構造体へのポインタ (ポインタの指す中身が構造体という意味) は，基本データ型と同様に変数の前に * を付けて宣言します．

構造体へのポインタ型の変数が指す構造体のメンバを利用するときは，

```
(*hensu).member
```

と変数名に * をつけたものを括弧 () で括ってからピリオドでつなぎます．この部分は，次のように書くことが出来ます．

```
hensu->member
```

次のプログラムは，前のプログラムの関数をアドレス渡しに変えたものです．

```
/* ２次元ベクトルの和（構造体へのポインタ利用） 11-5.c */
#include <stdio.h>

typedef struct vec2d {  /* 構造体タグ vec2d は省略出来ます */
    double  x;      /* x 成分 */
    double  y;      /* y 成分 */
} VEC2D;

void add2dvec_p(VEC2D, VEC2D, VEC2D *);
/* 返り値はないので void です ３番目の引数はポインタです */

main()
{
    VEC2D    vec1, vec2;    /* ２つのベクトルを用意 */
    VEC2D    vec3;          /* vec1 + vec2 を計算 */

    /* 構造体の初期化 */
    vec1.x = 2.0;
    vec1.y = 3.0;
    vec2.x = 4.0;
    vec2.y = 5.0;

    /* vec1 vec2 を足し算して結果を vec3 に入れる関数 */
    add2dvec_p(vec1, vec2, &vec3);

    printf("vec1のx成分は%f, y成分は%f\n", vec1.x, vec1.y);
    printf("vec2のx成分は%f, y成分は%f\n", vec2.x, vec2.y);
    printf("vec3のx成分は%f, y成分は%f\n", vec3.x, vec3.y);
}

void add2dvec_p(VEC2D vec1, VEC2D vec2, VEC2D *vec3_p)
{
    vec3_p->x = vec1.x + vec2.x;    /* x成分の足し算 */
    /* (*vec3_p).x = vec1.x + vec2.x; と書いてもよい */
    vec3_p->y = vec1.y + vec2.y;    /* y成分の足し算 */
    /* (*vec3_p).y = vec1.y + vec2.y; と書いてもよい */

    return;
}
```

4 実習課題

- ３次元のベクトルの外積を求める関数を構造体の値渡しを用いて作成せよ (ヒント：プログラム 11-4.c 参考にせよ .)
- ３次元のベクトルの外積を求める関数を構造体のアド レス渡しを用いて作成せよ (ヒント：プログラム 11-5.c 参考にせよ .)

どちらのプログラムもベクトルの成分を要求するように書くこと. (例によって`scanf` を使用する) そして printf 文で値を出力してプログラムが正しいかチェックせよ .

レポート名 (Subject 名) を creportno.11-1, creportno.11-2 とし, 時間内にできなかった人は 6 月 28 日 (水) までに提出すること.