情報研究会CACTUS

第七回　C言語講習

講師　6/21・井上　6/22・谷口

今週の内容

- ポインタ

サンプルプログラム1

```c
#include <stdio.h>
#define N 4

int main(void){
    int array[N] = { 0, 1, 2, 3 }, i;
    int *p1, *p2;

    p1 = &array[0];
    printf("p1に格納されている値:%u\n", p1);

    p2 = array;
    printf("p2に格納されている値:%u\n", p2);

    for (i = 0; i < N; i++){
        printf("array[%d]のアドレス:%u\n", i, &array[i]);
    }
    printf("\n");

    for (i = 0; i < N; i++){
        printf("p2に格納されている値:%u\n", p2);
        printf("p2の指すアドレスの変数の中身:%d\n\n", *p2);

        p2++;
        printf("### p2++を実行 ###\n");
    }
    return 0;
}
```

実行結果

```
p1に格納されている値:2947408
p2に格納されている値:2947408
array[0]のアドレス:2947408
array[1]のアドレス:2947412
array[2]のアドレス:2947416
array[3]のアドレス:2947420

p2に格納されている値:2947408
p2の指すアドレスの変数の中身:0

### p2++を実行 ###
p2に格納されている値:2947412
p2の指すアドレスの変数の中身:1

### p2++を実行 ###
p2に格納されている値:2947416
p2の指すアドレスの変数の中身:2

### p2++を実行 ###
p2に格納されている値:2947420
p2の指すアドレスの変数の中身:3

### p2++を実行 ###
```

ポインタ

●ポインタと配列

配列を宣言するとメモリ上に連続に領域が確保される

・例　int array[ ]の場合

番地	変数
1000	arrary[0]
1001	
1002	
1003	
1004	array[1]
1005	
1006	
1007	

配列の添字を除いた部分だけを記述すると
その配列の先頭アドレスを指し示すことになる

・例　&array[0] と　array　は同じ値

ポインタ

ポインタに1を加えるということは
「ポインタに格納されているアドレス+1」ではなく
「ポインタに格納されているアドレス+型のサイズ」である
型のサイズはsizeof(型)で調べることが出来る

・例　int型配列　p = array;

番地	変数	
1000	arrary[0]	← *p
1001		
1002		
1003		
1004	array[1]	← *(p+1)
1005		
1006		
1007		
1008	array[2]	← *(p+2)

ポインタ

これらは多次元配列でも同様の考え方である（宣言の仕方などは違う）

・例　int型　配列[2][4]の場合　int *p　= array;

	番地	変数	
array = array[0] = &array[0][0] →	1000	arrary[0][0]	← *p
	1004	arrary[0][1]	← *(p + 1)
	1008	arrary[0][2]	← *(p + 2)
	1012	arrary[0][3]	← *(p + 3)
array[1] = &array[1][0] →	1016	arrary[1][0]	← *(p + 4)
	1020	arrary[1][1]	
	1024	arrary[1][2]	
	1030	arrary[1][3]	

確認問題1

配列とポインタを用いて、
フィボナッチ数列の第10項目を求めるプログラムを作成せよ

サンプルプログラム2

```c
#include <stdio.h>

void calculator(int x, int y, int *p1, int *p2, int *p3); // プロトタイプ宣言

int main(void){

    int x, y, wa ,sa ,seki;
    int *wa_p, *sa_p, *seki_p; // 和、差、積を格納する変数のアドレスを保存するポインタ

    wa_p = &wa; sa_p = &sa; seki_p = &seki;

    printf("整数を2つ入力してください\n");
    printf("x:");
    scanf("%d", &x);
    printf("y:");
    scanf("%d", &y);

    calculator(x, y, wa_p, sa_p, seki_p); // calculatorを呼び出す

    printf("%dと%dの和は%d,差は%d,積は%d\n", x, y, wa, sa, seki);

    return 0;
}

// xとyの和差積をポインタを経由して計算・格納する関数
void calculator(int x, int y, int *p1, int *p2, int *p3){
    *p1 = x + y;
    *p2 = x - y;
    *p3 = x * y;
}
```

実行結果

```
整数を2つ入力してください
x:3
y:2
3と2の和は5,差は1,積は6
```

ポインタ

●ポインタと関数

ポインタの引数の型にもポインタを指定することが出来る
引数の型に*をつけて指定する

・例　void test_function(int a, int *p)

いままでの関数では戻り値は1つだったが、
関数にポインタを渡すことで複数の戻り値を実現することができる

ポインタ

●サンプルプログラム2の流れ

8行目：wa_pにwaのアドレスが格納される

変数	値	アドレス
wa	1657946	1000
wa_p	1000	9999

18行目：calculatorにwa_pを渡しp1が受け取る

wa	1657946	1000
wa_p	1000	9999
p1	1000	5555

27行目：p1が指し示すアドレスに和を代入

wa	3+2	1000
wa_p	1000	9999
p1	1000	5555

確認問題2

サンプルプログラムを書き換え、入力した2つの値の
2乗をそれぞれ計算する関数を作成し、表示するプログラムを作成せよ

サンプルプログラム3

```c
#include <stdio.h>
#define N 25

void fibonacci(int *p); // プロトタイプ宣言

int main(void){
    int array[N] = { 1, 1 }, m = 1;

    fibonacci(array); // フィボナッチ数列を計算

    // 0が入力されるまで好きな項を表示
    while (m != 0){
        printf("表示したい項を入力してください（最大%d項目）\n", N);
        scanf("%d", &m);
        if (m == 0) break;

        printf("フィボナッチ数列の第%d項は%dです\n", m, array[m - 1]);
    }
    printf("終了します\n");
        return 0;
}

// 渡されたポインタが指している配列にフィボナッチ数列を格納する関数
void fibonacci(int *p){
    int n;
    for (n = 2; n<N; n++){
        p[n] = p[n - 1] + p[n - 2];
        //*(p + n) = *(p + (n - 1)) + *(p + (n - 2));
    }
}
```

実行結果

```
表示したい項を入力してください（最大25項目）
10
フィボナッチ数列の第10項は55です
表示したい項を入力してください（最大25項目）
25
フィボナッチ数列の第25項は75025です
表示したい項を入力してください（最大25項目）
0
終了します
```

ポインタ

●ポインタと配列と関数

配列の先頭アドレスが入ったポインタを関数に渡した場合
配列と同じ扱いができる

・例

```
void fibonacci(int *p){
    int n;
    for (n = 2; n<N; n++){
        p[n] = p[n - 1] + p[n - 2];
        //*(p + n) = *(p + (n - 1)) + *(p + (n - 2));
    }
```

2次元配列を引数にするときは何列なのかを教えなければいけない

・ 例 p = &array[0][0]
　　void test2_function(int a, int p[ ][3])

pは配列じゃなくてポインタなのになぜ？？

ポインタ

何度読んでも理解できない人は今はそれでいいかも

C言語は配列を実現するためにアドレスの足し算を行っている
ポインタ変数も同じことができるので同等の扱いができてしまう

宣言時の[ ]は要素数を示すが、使用時の[ ]はアドレスの足し算を行う

・例　int型配列 array[0] = array[1] + 2;　　int* p = array;

今までの解釈：配列arrayの0番目に1番目と2を足したものを代入

実は：array（先頭アドレス）から0個分番地を進めたものに
array(先頭アドレス)から1個分番地を進めたものと2を足したものを代入

後者は *p = *(p+1) + 2　と同じ解釈である

つまり、先頭アドレスを受け取れば[ ]を利用して配列と同じ扱いができる

練習問題1

要素数5の配列を2つ用意し配列Aにscanfを用いて任意の値を入力する
もう一つの配列Bに配列Aの要素を逆順に格納する関数を作成し、
表示するプログラムを作成せよ

実行例

```
配列の中身を入力してください
4 5 15 -6 0
配列aの中身は4 5 15 -6 0
配列bの中身は0 -6 15 5 4
```

練習問題2

横の列、縦の列を与えると100ます計算を行う関数を作成し表示せよ

実行例

```
横の列を入力してください
1 2 3 4 5 6 7 8 9 10
縦の列を入力してください
1 2 3 4 5 6 7 8 9 10
    1  2  3  4  5  6  7  8  9 10
 1  1  2  3  4  5  6  7  8  9 10
 2  2  4  6  8 10 12 14 16 18 20
 3  3  6  9 12 15 18 21 24 27 30
 4  4  8 12 16 20 24 28 32 36 40
 5  5 10 15 20 25 30 35 40 45 50
 6  6 12 18 24 30 36 42 48 54 60
 7  7 14 21 28 35 42 49 56 63 70
 8  8 16 24 32 40 48 56 64 72 80
 9  9 18 27 36 45 54 63 72 81 90
10 10 20 30 40 50 60 70 80 90 100
```

前回の練習問題1の解答例

2点間の距離を求めるプログラム

```c
#include <stdio.h>
#include <math.h>
typedef struct
{
    int x;
    int y;
    int z;
}point;

double distance(point,point);

int main(void)
{
    point p, q;

    printf("p(x,y,z): ");
    scanf("%d%d%d", &p.x, &p.y, &p.z);
    printf("q(x,y,z): ");
    scanf("%d%d%d", &q.x, &q.y, &q.z);

    printf("2点間の距離:%f\n",distance(p,q));

    return 0;
}

double distance(point p, point q)
{
    return sqrt(pow((p.x-q.x),2)+pow((p.y-q.y),2)+pow((p.z-q.z),2));
}
```

前回の練習問題2の解答例

本の購入巻を求めるプログラム

```c
#include <stdio.h>
#define N 15

typedef struct
{
    int series;
    int volume;
}books;

int main(void)
{
    int i, j, havenum, sellnum, flag=1;
    books have[N], sell[N];

    printf("持っている本の数:");
    scanf("%d", &havenum);
    for(i=0;i<havenum;i++){
        scanf("%d%d", &have[i].series, &have[i].volume);
    }

    printf("売っている本の数:");
    scanf("%d", &sellnum);
    for(i=0;i<sellnum;i++){
        scanf("%d%d", &sell[i].series, &sell[i].volume);
    }

    for(i=0;i<havenum;i++){
        for(j=0;j<sellnum;j++){
            if(have[i].series==sell[j].series&&have[i].volume==sell[j].volume){
                sell[j].series = 0;
            }
        }
    }

    printf("購入するべき本\n");
    for(i=0;i<sellnum;i++){
        if(sell[i].series){
            printf("%d %d\n", sell[i].series, sell[i].volume);
            flag = 0;
        }
    }

    if(flag) printf("None\n");

    return 0;
}
```