# Training 8
# データ構造とポインタ

株式会社イーシーエス　出版事業推進委員会

# Lesson1　一次元配列

**Point◆◇一次元配列を使えるようになろう!!**

一次元配列とは、複数の同じ型の変数を1つにまとめたものでデータが横（一次元）方向に広がっていく形のものです。
複数のデータを一つの配列としてまとめることができます。

## 【問題1】　次の配列 array に設定されているデータを答えなさい。

```
int array[ ] = { 85, 23, 46, 31, 97, 51, 10, 62 };
```

①array[0]　　②array[2]　　③array[3]　　④array[7]

## 【問題2】　次の配列 array に設定されているデータを答えなさい。

```
int array[7];
int x;

x = 10;

array[0] = x;
array[1] = x + 15;
array[2] = array[1];
array[3] = array[1] + 50;
array[4] = array[3] / array[2];
array[6] = array[5] = array[4];
```

① array[0]
② array[1]
③ array[2]
④ array[3]
⑤ array[4]
⑥ array[5]
⑦ array[6]

## 【問題3】　次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**
配列 array のデータを全て 0 に初期化する。

```
int i;
int array[100];

for( i = ① ; i < ② ; ③ )
{
    ④
}
```

## 【問題4】　次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**
変数 sum に配列 array のデータの合計を求める。

```
int i;
int sum = 0;
int array[ ] = { 10, 20, 30, 40, 50, 60, 70, 80, 90 };

for( i = ① ; i < ② ; ③ )
{
    ④
}
```

# Lesson2　多次元配列

**Point◆◇多次元配列を理解しよう!!**

多次元配列とは、一次元配列に対してデータが横以外（多次元）の方向にも広がっていく形のものです。
特に二次元配列は使用頻度も高いものなので覚えてください。

**【問題1】** 次の配列 **array** に設定されているデータを答えなさい。

```
int array[3][8] = { { 10, 11, 12, 13, 14, 15, 16, 17 },
                    { 30, 31, 32, 33, 34, 35, 36, 37 },
                    { 50, 51, 52, 53, 54, 55, 56, 57 }   };

①　array[0][0]　②　array[1][4]　③　array[2][7]　④　array[2][3]
```

**【問題2】** 次の配列 **array** に設定されているデータを答えなさい。

```
int array[3][4][6] = { { { 100, 101, 102, 103, 104, 105 },
                         { 110, 111, 112, 113, 114, 115 },
                         { 120, 121, 122, 123, 124, 125 },
                         { 130, 131, 132, 133, 134, 135 } },
                       { { 200, 201, 202, 203, 204, 205 },
                         { 210, 211, 212, 213, 214, 215 },
                         { 220, 221, 222, 223, 224, 225 },
                         { 230, 231, 232, 233, 234, 235 } },
                       { { 300, 301, 302, 303, 304, 305 },
                         { 310, 311, 312, 313, 314, 315 },
                         { 320, 321, 322, 323, 324, 325 },
                         { 330, 331, 332, 333, 334, 335 } } };

①　array[0][0][0]　　②　array[1][2][3]
③　array[2][3][5]　　④　array[0][3][2]
```

## 【問題3】 次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**

配列 array の最大値と最小値を表示する。

```
#include <stdio.h>

void main( void )
{
    /* 変数の宣言 */
    int i;                /* ループカウンタ          */
    int j;                /* ループカウンタ          */
    int max;              /* 最大値                  */
    int min;              /* 最小値                  */
    int max_x = 0;        /* 最大値の要素番号(x 軸) */
    int max_y = 0;        /* 最大値の要素番号(y 軸) */
    int min_x = 0;        /* 最小値の要素番号(x 軸) */
    int min_y = 0;        /* 最小値の要素番号(y 軸) */

    int array[3][4] = { { 500, 250, 350, 400 },
                        { 340, 405, 138, 245 },
                        { 175, 578, 762, 357 } };

    /* 初期値の設定 */
    max = array[0][0];
    min = array[0][0];

    /* 最大･最小値の判定 */
    for ( j = 0; j < 3; j++ )
    {
        for ( i = 0; i < 4; i++ )
        {
            if ( max < array[j][i] )
            {
                [ ① ] = i;
                max_y = j;
                [  ②  ];
            }
            if ( min > [  ③  ] )
            {
                 min_x = i;
                min_y = [④];

                [  ⑤  ];
            }
        }
    }

    /* 最大･最小値の出力 */
    printf ("最大値 array[%d][%d] = %d¥n", max_y, max_x, max );
    printf ("最小値 array[%d][%d] = %d¥n", min_y, min_x, min );
}
```

# Lesson3　文字列

**Point◆◇文字列の特性を理解しよう!!**

文字配列は、文字の集まりを配列に格納したものです。配列に格納する際１文字ずつ格納する場合と、
文字列を格納する場合で扱い方が違います。ここでは配列における文字と文字列使用時の違いについて練習しましょう。

**【問題1】** 次の配列を要素数を指定して宣言する場合、最低限必要な要素数を答えなさい。

① `char str1[] = { '1', '2', '3' };`

② `char str2[] = { "123456789" };`

**【問題2】** 次の配列 **str** に設定されているデータを答えなさい。

```
char str[] = { 'a', 'b', 'c', 'd' };
```

① str[0]　② str[2]　③ str[4]　④ str[1]

**【問題3】** 次の配列 **str** に設定されているデータを答えなさい。

```
char str[] = { "1234" };
```

① str[0]　② str[3]　③ str[1]　④ str[4]

**【問題4】** 次の配列 **str** に設定されているデータを答えなさい。

```
char str[ ][7] = { { 'a', 'b', 'c', 'd', 'e', 'f', 'g' },
                   { "ABCDEF" } };
```

① str[0][6]　② str[1][3]　③ str[0][2]　④ str[1][6]

**【問題5】** 次の配列 **str** に設定されているデータを答えなさい。

```
char org[ ] = { "Hello" };
char str[4];

str[0] = 'a';
str[1] = org[2];
str[2] = str[1];
str[3] = org[5];
```

① str[0]
② str[1]
③ str[2]
④ str[3]

## 【問題6】 次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**
2 つの文字列を結合する。

```
#include <stdio.h>

void main (void)
{
     char str1[256];
     char str2[256];
     char strd[512];

     int i;
     int j;

     /* 変数の初期化 */
     [ ① ];

    /* キー入力を文字列 1 として配列 1 に格納 */
     printf( "文字列 1 の入力を開始してください。¥n" );
     scanf( "%s", str1 );

     /* キー入力を文字列 2 として配列 2 に格納 */
     printf( "文字列 2 の入力を開始してください。¥n" );
     scanf( "%s", str2 );

     /*  文字列1を、結果の領域に入れる  */
     i = 0;
     while ( [ ② ] ) /* 文字列が終了するまでループ */
     {
          strd[j] = [ ③ ] ;
          i++;
          [ ④ ] ;
     }

     /*  文字列 2 を、結果の領域に入れる  */
     [ ⑤ ];

     [ ⑥ ] /* 文字列が終了するまでループ */
     {
          strd[j] = [ ⑦ ] ;
          [ ⑧ ] ;
          j++;
     }

     /*  文字列の終了  */
     strd[ [ ⑨ ] ] = [ ⑩ ] ;

     /* 出力 */
     printf( "文字列 1:%s¥n",str1 );
     printf( "文字列 2:%s¥n",str2 );
     printf( "結合後文字列:%s¥n",strd );
}
```

# Lesson4　配列のデータの大きさ

**Point◆◇配列データの大きさを把握しよう!!**

ここでは配列データの大きさをしっかりとおさえましょう。配列全体のサイズを把握していないと、予期せぬエラーが発生してしまいます。

**【問題1】** 下記配列の使用バイト数を答えなさい。**※int は 4 バイトとする**

① `unsigned char arr[10];`　　② `signed char arr[10];`

③ `float arr[10];`　　④ `unsigned int arr[10];`

⑤ `signed short int arr[10];`　　⑥ `double arr[10];`

**【問題2】** 下記配列の使用バイト数を答えなさい。

① `unsigned char arr[ ] = { 1, 2 ,3 };`

② `signed char arr[ ] = { 1, 2 };`

③ `float arr[ ] = { 1, 2, 3, 4 };`

④ `signed short int arr[ ] = { 1, 2, 3, 4, 5 };`

⑤ `unsigned short int arr[ ] = { 1, 2, 3 };`

⑥ `double arr[ ] = { 1, 2, 3, 4, 5, 6 };`

**【問題3】** 下記配列の使用バイト数を答えなさい。**※int は 4 バイトとする**

① `unsigned char arr[5][10];`　　② `signed char arr[10][20];`

③ `float arr[6][7];`　　④ `unsigned int arr[8][3];`

⑤ `signed short int arr[3][4];`　　⑥ `double arr[2][10];`

# Lesson5　配列のまとめ

**Point◆◇配列に入っている値を把握しよう!!**

ここは配列の総まとめです。配列を理解しつつ、プログラムも読めるように練習しましょう。

## 【問題1】 次のプログラムを読み以下の問いに答えなさい。(各ファイルは同じ階層にある)

**【処理内容】**

荷物の重さと 3 辺の長さを入力し、配達料金を配列を用いて求める。
ただし、料金表にない値が入力された場合、関数 hokan を用いて、料金を求める。
また、600 円未満と 2200 円を超える料金にはならない。

**main.h**

```
/* 基本型定義 */
typedef signed char        S1;
typedef unsigned char      U1;
typedef signed short       S2;
typedef unsigned short     U2;
typedef signed long        S4;
typedef unsigned long      U4;
typedef void               VD;

/* 関数プロトタイプ宣言 */
U2 hokan( double x, U2 x1, U2 x2, U2 y1, U2 y2);

/* 荷物の重さと長さの料金表配列 縦:荷物の重さ   横:荷物の 3 辺の長さ */
 const U2 map[6][6] ={
    {    0,   60,     80,     120,     140,     160 },
    {    2,  600,    800,    1000,    1200,    1400 },
    {    4,  800,   1000,    1200,    1400,    1600 },
    {    6, 1000,   1200,    1400,    1600,    1800 },
    {    8, 1200,   1400,    1600,    1800,    2000 },
    {   10, 1400,   1600,    1800,    2000,    2200 }
};
```

**main.c**

```
#include <stdio.h>
#include "main.h"

VD main (VD)
{
    double   in_weight;    /* 荷物の重さの入力変数 */
    double   in_length;    /* 荷物の長さ入力変数 */
    U2   fee1;             /* 補間用の料金を格納 */
    U2   fee2;             /* 補間用の料金を格納 */
    U2   weight_a;         /* 補間用の荷物の重さを格納 */
    U2   weight_b;         /* 補間用の荷物の重さを格納 */
    U2   length_a;         /* 補間用の荷物の長さを格納 */
    U2   length_b;         /* 補間用の荷物の長さを格納 */
    U2   j;                /* ループカウンタ変数 */
    U2   i;                /* ループカウンタ変数 */
    U2   ans_fee;          /* 料金(結果)を格納する変数 */

    printf("荷物の重さを入力してください(kg):");
    scanf( "%lf", &in_weight );
    fflush(stdin);
    printf("¥n¥n 荷物の3辺の長さの合計を入力してください:");
    scanf( "%lf", &in_length );
    printf("¥n¥n");
```

```
/*------------荷物の重さの補完値を求める-------*/
    /* 入力した重さが下限以下の時 */
    if( in_weight < 2 )
    {
        weight_a = 2;
        weight_b = 2;
    }
    /* 入力した重さが上限以上の時 */
    else if( in_weight >= 10 )
    {
        weight_a = 10;
        weight_b = 10;
    }
    else
    {
        i = 1;
        while( 1 )
        {
            /* 入力した重さが表にない時、その両端の数値を補完値とし変数に格納 */
            if( map[i][0] > in_weight )
            {
                weight_a = map[i - 1][0];
                weight_b = map[i][0];
                break;
            }
            /* 入力した重さが表にあった場合 */
            else if( map[i][0] == in_weight )
            {
                weight_a = map[i][0];
                weight_b = map[i][0];
                break;
            }
            i++;
        }
    }

    /*------------荷物の長さの補完値を求める-------*/
    /* 入力した長さが下限以下の時 */
    if( in_length < 60)
    {
        length_a = 60;
        length_b = 60;
    }
    /* 入力した長さが上限以上の時 */
    else if( in_length >= 160 )
    {
        length_a = 160;
        length_b = 160;
    }
    else
    {
        j = 1;
        while( 1 )
        {
            /* 入力した長さが表にない場合その両端の数値を補完値として変数に格納 */
            if( map[0][j] > in_length )
            {
                length_a = map[0][j - 1];
                length_b = map[0][j];
                break;
            }
            /* 入力した長さが表にあった場合 */
            else if( map[0][j] == in_length )
            {
                length_a = map[0][j];
                length_b = map[0][j];
                break;
            }
            j++;
        }
    }
```

```
    /* 長さ、重さがともに表にあった場合 */
    if( ( weight_a == weight_b ) && ( length_a == length_b ) ) ·····(1)
    {
        ans_fee = map[i][j];  ·····A
    }
    /* 重さが表にあり、長さが表にない場合 */
    else if( ( weight_a == weight_b ) && ( length_a != length_b ) )
    {
        ans_fee = hokan( in_length, length_a, length_b, map[i][j - 1], map[i][j]);·····B
    }
    /* 重さが表になく、長さが表にあった場合 */
    else if( ( weight_a != weight_b ) && ( length_a == length_b ) )
    {
        ans_fee = hokan( in_weight, weight_a, weight_b, map[i - 1][j], map[i][j]);····C
    }
    /* 長さ、重さがともに表にない場合 */
    else
    {
        fee1 = hokan( in_weight,weight_a,weight_b,map[i - 1][j - 1], map[i][j - 1] );
        fee2 = hokan( in_weight,weight_a,weight_b,map[i - 1][j], map[i][j] );

        ans_fee = hokan( in_length, length_a, length_b, fee1, fee2 );·····D
    }
    printf( "お預かりした荷物の重さは%.2lfkg、長さは%.2lf cmです。¥n",in_weight, in_length ); ·····(2)
    printf("料金は%d 円です。¥n",ans_fee); ·····(2)

}

U2 hokan( double X, U2 X1, U2 X2, U2 X3, U2 X4)
{
    double ryoukin;

    ryoukin = ( ( x2 - X ) * y1 + (X - x1 ) * y2 ) / ( x2 - x1 );

    return (U2)(ryoukin);
}
```

① in_weight = 4.5, in_length = 138 したとき、(1)の判定時 i と j の値を答えなさい。

② in_weight = 3, in_length = 115 のとき、上記のプログラムの A,B,C,D のどれが実行されるか答えなさい。

③ in_weight = 3, in_length = 140 のとき、上記のプログラムの A,B,C,D のどれが実行されるか答えなさい。

④ in_weight = 3, in_length = 115 のときの(2)の printf 実行結果を書きなさい。

# Lesson6　ポインタ変数

**Point◆◇ポインタの特性を理解しよう!!**

ポインタは文字通り、ある場所を示します。ポインタを使用して、メモリのアドレスの先に値を格納したり、
**間接参照**といってアドレスを介してその先に格納されている値を参照します。

## 【問題1】　次のプログラムの出力結果を答えなさい。

```
void main(void)
{
    int i;
   int *p;

    i = 20;
    p = &i;

    i = 30;
    printf("%d¥n",*p);
}
```

## 【問題2】　次のプログラムの出力結果を答えなさい。

```
void main(void)
{

    int a;
    int *p;
    int *q;

    a = 50;
    p = &a;
    q = p;

    *p += 20;
    printf("%d¥n",*q);

}
```

## 【問題3】　次のプログラムの出力結果を答えなさい。

```
void main(void)
{

    int i;
    int *p;
    int **pp;

    i = 5;
    p = &i;
    pp = &p;

    *p += 10;
    **pp -= 5;
    printf("%d¥n",**pp);

}
```

## 【問題4】 次のプログラムの出力結果を答えなさい。

```
void main(void)
{
    int arr[] = { 0, 5, 10, 15, 20};

    printf("%d¥n",*(arr + 1) + 3);
}
```

## 【問題5】 次のプログラムの出力結果を答えなさい。

```
void main(void)
{
    char str[] = "test";
    char *p;

    p = str;
    printf("%c¥n",*(p + 2));
}
```

## 【問題6】 次のプログラムの出力結果を答えなさい。

```
void main(void)
{
    int x[2][3] = { { 2,  4,  6},
                    { 8, 10, 12} };
    int *p;

    p = x[0];
    printf("%d %d¥n",*(x[1] + 1),*(p + 4));
}
```

# Lesson7　ポインタ配列

**Point◆◇ポインタ配列の特性を理解しよう!!**

ポインタ配列は、ポインタを要素として持つ配列のことです。
ポインタ配列を使用することによって、無駄なメモリ領域を作らないメリットがあります。
文字列の先頭アドレスをポインタに格納できるので複数の文字列を宣言するときに便利です。

## 【問題1】　次のプログラムの出力結果を答えなさい。

```
void main(void)
{
    char *str[] = {
            "Sunday", "Monday" , "Tuesday" , "Wednesday" ,
            "Tursday" , "Friday" , "Saturday"
            };

    printf("%s¥n",str[2]);
}
```

## 【問題2】　次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**

red , green , blue の文字列をポインタ配列に格納し、表示する。

※ただし、ポインタ配列名は color とし、配列要素数は指定しない。

```
void main(void)
{
    int i;
    char  [          ①          ] ;
    for(i = 0; i < 3; i++)
    {
        printf("%s¥n",[  ②  ]);
    }
}
```

出力結果

```
red
green
blue
```

## 【問題3】　次のポインタ配列で書かれたプログラムを、2 次元配列を使ったプログラムで書き直しなさい。

```
void main(void)
{
    char *name[] = { "taro" , "hanako" , "kouji" , "hiroyuki" };

    printf("%s¥n", name[1]);

}
```

# Lesson8　ポインタのデータの大きさ

**Point◆◇ポインタのデータの大きさに注意しよう!!**

ポインタを使用したデータの大きさはアドレスの大きさとそこに格納されているデータの大きさで違います。
この違いを理解していないと、メモリ容量を無駄にしてしまうことがあります。しっかりと押さえましょう。

## 【問題1】 sizeof 関数で求められるバイト数を答えなさい。※int は 4 バイトとする

① 
```
unsigned char *p1;
sizeof(*p1);
```

② 
```
signed char *p2;
sizeof(p2);
```

③ 
```
float *p3;
sizeof(p3);
```

④ 
```
sizeof(unsigned int *);
```

⑤ 
```
signed short int *p5;
sizeof(*p5);
```

⑥ 
```
sizeof(double *);
```

## 【問題2】 sizeof 関数で求められるバイト数を答えなさい。

① 
```
unsigned char arr[10];
unsigned char *ptr = arr;
sizeof(ptr);
```

② 
```
signed char arr[10];
signed char *ptr = arr;
sizeof(*ptr);
```

③ 
```
float arr[10];
float *ptr = arr;
sizeof(*ptr);
```

④ 
```
unsigned int arr[10];
unsigned int *ptr = arr;
sizeof(ptr);
```

⑤ 
```
signed short int arr[10];
signed short int *ptr = arr;
sizeof(*ptr);
```

⑥ 
```
double arr[10];
double *ptr = arr;
sizeof(ptr);
```

## 【問題3】 次のプログラムを読み、以下の問いに答えなさい。

```
#include <stdio.h>
void point(char *);

void point(char *p2)
{
    *p2 *= 5;
}

void main(void)
{
    char *p1;
    char val_a;
    val_a = 12;
    p1 = &val_a;
    point( p1 );

    printf("val_a の値は%d です。", val_a );       ・・・・・・・・・・・・・A
    printf("p1 のサイズは%d バイトです。", sizeof( p1 ) ); ・・・・・B
    printf("*p1 のサイズは%d バイトです。", sizeof( *p1 ) );    ・・・・・C
}
```

① A で出力される値を答えなさい。
② B で出力される値を答えなさい。
③ C で出力される値を答えなさい。

# Lesson9　ポインタと関数

**Point◆◇関数でのポインタの特性を理解しよう!!**

関数の引数には、値を渡す方法の他にアドレス値を渡す方法があります。
アドレス値を渡した場合、関数内で値を変更すると、間接的に変数の値も変更されるので戻り値は必要なくなります。
ポインタはアドレス参照のため、別のところで同じアドレスを参照していた場合そちらも変化しますので注意が必要です。

## 【問題1】　次のプログラムの出力結果を答えなさい。

```
#include <stdio.h>

void function(int x,int y,int *z)
{
    x = x + 6;
    y = y + 7;
    *z = x + y;
}

void main(void)
{
    int x;
    int y;
    int z;

    x = 3;
    y = 4;
    z = 5;

    function( x, y, &z);

    printf("x = %d , y = %d , z = %d¥n", x, y, z);
}
```

## 【問題2】　次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**
2つの配列の値を入れ替える。

```
#include <stdio.h>

void swap( [　　①　　] )
{
    int tmp;
    tmp = *x;
    [　②　]

    *y = tmp;
}

void main(void)
{
    int data1[] = {  1,   2,  3};
    int data2[] = { 11, 12, 13};
    int i;

    printf("data1[] = { %d, %d, %d }¥n",data1[0],data1[1],data1[2]);
    printf("data2[] = { %d, %d, %d }¥n",data2[0],data2[1],data2[2]);

    for(i = 0;i < 3; i++)
    {
        swap( [　　③　　] );
    }
    printf("data1[] = { %d, %d, %d }¥n",data1[0],data1[1],data1[2]);
    printf("data2[] = { %d, %d, %d }¥n",data2[0],data2[1],data2[2]);
}
```

## 【問題3】 次の□部分を埋め、左のプログラムと同じ動作をするプログラムを、ポインタを使い完成させなさい。

**【処理内容】**
入力値 x と入力値 y の積を求める。

```
#include <stdio.h>

int mult(int x, int y)
{
     int mult;

     mult = x * y;

     return mult;
}

void main(void)
{
     int x;
     int y;
     int z;

     printf("x : ");
     scanf("%d",&x);
     printf("y : ");
     scanf("%d",&y);

     z = mult( x , y );

     printf("z = %d¥n",z);
}
```

```
#include <stdio.h>

void mult( [ ① ] )
{
     [ ② ];
}

void main(void)
{
     int x;
     int y;
     int z;

     printf("x : ");
     scanf("%d",&x);

     printf("y : ");
     scanf("%d",&y);

     mult( [ ③ ] );

     printf("z = %d¥n",z);
}
```

## 【問題4】 次のプログラムを読み、以下の問に答えなさい。

**【処理内容】**
10 進数の整数を、引数の値で基数を変更できる関数 convert を作成した。
この関数は 10 進数の整数を受け取り、指定された基数に変換する際に、数字を文字として格納する。

```
#include <stdio.h>

#difine KISUU 16
void convert16(unsigned int number , [ A ] str);

void main(void)
{
     unsigned int number;
     char string[8];

     number = [ Z ] ;

     convert( number , string , 16 );

     printf("%s¥n",string);
}

void convert16(unsigned int number , [ A ] str)
{
     int i;
     int k;
     int s;
     unsigned int val_kisuu;
     int weight_table[8];

     char num[17] = "0123456789ABCDEF";
```

```
    if( number == 0 )
    {
        *str = '0';
        str++;
    }
    else
    {
        val_kisuu = 1;
        k = 0;

        /* 桁数を求め、weight_table に各重みの数値を格納する処理 */
        while( ( number >=  val_kisuu ) && ( k < 8 ) )
        {
            weight_table[k] = val_kisuu;
            val_kisuu *= KISUU;
            k++;
        }
    }

    /* str に文字として数字を格納 */
    for( i = 0 ; i < k ; i++ )    ……(1)
    {
        s = number / weight_table[ k - i – 1 ];
        [ B ] = num[ s ];

        /* 次にずらす */
        [ C ] ;

        number -= s * weight_table[ k - i - 1];
    }

    /* 終端文字を入れる */
    [      D      ] ;
}
```

① 空欄 A から D を埋めてプログラムを完成させなさい。

② 空欄 Z に 15、256、65535 を代入したとき、プログラム中の(1)の時点での k の値を答えなさい。

# Lesson10　構造体データへのアクセス方法

**Point◆◇構造体を理解しよう!!**

構造体データへのアクセス方法はバブル（直接参照）、アロー（間接参照）の 2 種類があります。
構造体データへのアクセスはプログラムに欠かすことのできない内容の為、必ずマスターしてください。
構造体の最大の魅力はデータ型の違うものをまとめて扱えるということにあります。違う型のデータを一枚のカードとして扱えるのです。

## 【問題1】　次の□部分を埋め、プログラムを完成させなさい。

**【処理内容】**
社員データを入力して表示する。

```
#include <stdio.h>

struct SYAIN{
     int number;       /* 社員番号    */
     char name[20];    /* 氏名        */
     double height;    /* 身長        */
};

void main(void)
{
     struct SYAIN syain1 = { 2200, "三河 太郎", 170.0 };
     printf("社員番号 %d¥n", [ ① ] );

     printf("氏名     %s¥n", [ ② ] );

     printf("身長    %lf¥n", [ ③ ] );
}
```

## 【問題2】　次の□部分を埋め、以下のプログラムを完成させなさい。

**【処理内容】**
社員データを入力して表示する。

```
#include <stdio.h>

struct SYAIN{
     int number;           /* 社員番号    */
     char name[20];        /* 氏名        */
     double height;        /* 身長        */
     double weight;        /* 体重        */
};

void output(struct SYAIN *);

void main(void)
{
     struct SYAIN syain1 = { 2200, "三河 太郎", 170.0 , 60.0 };

     output( &syain1 );
}

void output(struct SYAIN *fw)
{
     printf("社員番号 %d¥n", [ ① ] );

     printf("氏名     %s¥n", [ ② ] );

     printf("身長    %lf¥n", [ ③ ] );

     printf("体重    %lf¥n", [ ④ ] );
}
```

## 【問題3】 次の□部分を埋め、以下のプログラムを完成させなさい。

**【処理内容】**
社員データを入力して表示する。

```
#include <stdio.h>

struct SYAIN{
    int number;       /* 社員番号    */
    char pname[20];   /* 氏名        */
    double height;    /* 身長        */
    double weight;    /* 体重        */
};

void input(struct SYAIN *, int , char *, double, double );

void main(void)
{
    struct SYAIN syain1;

    input( &syain1, 2200, "三河 太郎", 170.0, 60.0 );
    printf("社員番号   %d¥n", [   ①   ] );

    printf("氏名       %s¥n", [   ②   ] );

    printf("身長     %lf¥n", [   ③   ] );

    printf("体重     %lf¥n", [   ④   ] );
}

void input(struct SYAIN *fw, int num, char *name, double ht, double wt )
{
    int i = 0;

    [   ⑤   ] = num;

    while (*name != '¥0')
    {
        [   ⑥   ] = *name++;
    }

    [   ⑦   ] = '¥0';

    [   ⑧   ] = ht;

    [   ⑨   ] = wt;
}
```

# Lesson11　構造体変数

**Point◆◇構造体を理解しよう!!**

構造体のメンバを参照する為の実態定義されたものを構造体変数といいます。
尚、テンプレート宣言の後に変数名を記述することで構造体変数の宣言も可能です。

**【問題1】**　以下のプログラムを実行した場合、最終的に①～③にどのような値が代入されてるか答えなさい。

```
#include <stdio.h>

void main(void)
{
    struct  XYZ{
        int      x;
        long     y;
        double   z;
    };

    struct XYZ  sa = { 5, 12345678L, 3.1415926535 };
}
```

① sa.x
② sa.y
③ sa.z

**【問題2】**　以下のプログラムを実行した場合、最終的に①～④にどのような値が代入されるか答えなさい。

```
#include <stdio.h>

struct WXYZ{
    char     w[20];
    int      x;
    long     y;
    double   z;
};

void input(struct WXYZ *);

void main(void)
{
    struct WXYZ sa;

    input( &sa );
}

void input(struct WXYZ *fw )
{
    int i = 0;
    char *name = "dog";

    while ( *name != '¥0' )
    {
        fw->w[i++] = *name ++;
    }

    fw->w[i] ='¥0';
    fw->x = 2;
    fw->y = 15L;
    fw->z = 2.8;
}
```

① sa.w
② sa.x
③ sa.y
④ sa.z

# Lesson12　構造体配列

**Point◆◇構造体の配列を理解しよう!!**

構造体定義にて宣言した配列を構造体配列といいます。
同じ構造体変数を複数使用したい時などに使用しますのでマスターしてください。
構造体配列は多人数の共通するデータ項目を管理するのに便利です。

**【問題1】** 以下のプログラムを実行した場合、最終的に①～⑧にどのような値が代入されるか答えなさい。

```
#include <stdio.h>

struct WXYZ {
    int      x;
    double   y;
    double   z;
};

void main(void)
{
    struct WXYZ sa[10];

    int i;

    sa[0].x = 0;
    sa[0].y = 0.0;
    sa[0].z = 0.0;

    for (i = 1; i < 10; i++ )
    {
        sa[i] = sa[0];
    }

    sa[5].x = 2;
    sa[5].y = 15.0;
    sa[5].z = 2.8;
    sa[7].x = 4;
    sa[7].y = 10.0;
    sa[7].z = 1.3;
}
```

① sa[4].x
② sa[2].y
③ sa[3].z
④ sa[7].x
⑤ sa[6].y
⑥ sa[5].z

【問題2】 以下のプログラムは商品金額の大きい順にソートするプログラム **rank** である。
この関数は構造体配列の中身を入れ替えるのではなく、金額の大きい構造体配列の要素番号を、配列 **str** 内で並び替えている。
空欄①～④に当てはまるものを解答群より選びなさい。(各ファイルは同じ階層にある)

**main.h**

```
/* 構造体テンプレート宣言 */
struct MANAGE{
    unsigned char no;                   /* 購入 NO */
    unsigned char goods_name[50];       /* 商品名 */
    unsigned int goods_price;           /* 商品金額 */
};

/* プロトタイプ宣言 */
void rank( [ ① ] , int str[] );
```

**main.c**

```
#include "main.h"

void main(void)
{
    /* 構造体配列の宣言 */
    struct MANAGE data_no[5] = {
            { 1, "椅子",       140000},
            { 2, "パソコン",    200000},
            { 3, "冷蔵庫",     70000 },
            { 4, "箪笥",       50000 },
            { 5, "机",         20000 }
    };

    int rank_table[5];          /* 構造体配列の要素番号を格納する配列 */

    rank( data_no , rank_table);
}

void rank( [ ① ] , int str[] )
{
    int i;              /* ループカウンタ */
    int j;              /* ループカウンタ */
    int tmp;            /* 並び替えの時に一時保管する変数 */

    /* 並び替えのため配列を用意する */
    for( i = 0; i < 5 ; i++ )
    {
        str[i] = i;
    }

    for( i = 0; i < [ ② ]; i++ )
    {
        for( j = i + 1; j < 5; j++ )
        {
            if(data_no[ [ ③ ] ].goods_price < data_no[ [ ④ ] ].goods_price )
            {
                tmp = str[j];
                str[j] = str[i];
                str[i] = tmp;
            }
        }
    }
}
```

解答群

| | |
|---|---|
| ア. str[j] | イ. str[i] |
| ウ. 6 | エ. data_no[] |
| オ. struct MANAGE data_no[] | カ. 4 |

# Lesson13　構造体のデータの大きさ

**Point◆◇構造体のデータの大きさに注意しよう!!**

構造体のデータの大きさは構造体のメンバの大きさをすべて足したものになります。
そのため、構造体を使用する際は非常に多くの容量を取る場合があります。
構造体の容量を意識し、コーディングしましょう。

**【問題1】**　下記構造体変数のバイト数を答えなさい。
ただし構造体を作成するときに必要な記憶領域の中に空き領域が発生しないものとする。

構造体の定義

```
struct STUDENT{
    int number;
    char name[20];
    double height;
} student1;
```

①　sizeof(student1.number)
②　sizeof(student1.name)
③　sizeof(student1.height)
④　sizeof(student1)

**【問題2】**　下記構造体変数のバイト数を答えなさい。
ただし構造体を作成するときに必要な記憶領域の中に空き領域が発生しないものとする。

構造体の定義

```
struct STUDENT {
    int number;
    char name[20];
    double height;
};

struct STUDENT student1[30];
```

①　sizeof(student1[0].number)
②　sizeof(student1[5].name)
③　sizeof(student1[10].height)
④　sizeof(student1)

**【問題3】**　下記構造体変数のバイト数を答えなさい。
ただし構造体を作成するときに必要な記憶領域の中に空き領域が発生しないものとする。

構造体の定義

```
struct STUDENT {
    int number;
    char name[20];
    double height;
};

struct STUDENT student1[ ] = {
    2006001, "東京太郎"  , 165.2 ,
    2006002, "名古屋太郎", 170.5 ,
    2006003, "大阪太郎"  , 178.4
};
```

①　sizeof(student1[0].number)
②　sizeof(student1[2].name)
③　sizeof(student1[1].height)
④　sizeof(student1)

# Lesson14　ポインタを使用した構造体

**Point◆◇ポインタと構造体の組み合わせを使えるようにしよう!!**

構造体もポインタを使用してメンバの参照をすることが可能です。
引数が構造体ポインタの場合、メンバの値を直接操作しますので注意してください。

**【問題1】**　以下のプログラムを実行した場合、最終的に①～④にどのような値が代入されるか答えなさい。

```
#include <stdio.h>

struct WXYZ{
     int      x;
     long     y;
     double   z;
};

void main(void)
{
     struct WXYZ  sa;
     struct WXYZ  *sb;

     sb = &sa;
     sb->x = 2;
     sb->y = 4;
     sb->z = 0.0;
}
```

①　sa.x
②　sa.y
③　sa.z

**【問題2】**　以下のプログラムを実行した場合、A 時点での①～②、B 時点での③～④にどのような値が代入されるか答えなさい。

```
#include <stdio.h>

struct TESTRES{
     int no;
     int score;
};

void setdata(struct TESTRES *);

void main(void)
{
     struct TESTRES  sa;

     sa.no = 1;
     sa.score = 100;

     setdata( &sa );
} ·····························A

void setdata(struct TESTRES *sb)
{
     sb->no = 5;
     sb->score = 50;
} ·····························B
```

①　sa.no
②　sa.score
③　sb->no
④　sb->score

# Lesson15　typedef を使用した構造体

**Point◆◇typedef を使いこなそう!!**

構造体のテンプレート宣言は typedef を使用して宣言すると有効です。
それにより可読性の高いプログラムを作成することができますので、この機会にマスターしてください。
typedef を使用すれば見やすくなるだけでなく、プログラムのコーディングも楽になります。

## 【問題1】　次の□部分を埋め、以下のプログラムを完成させなさい。

**【処理内容】**
社員データを入力して表示する。

```
#include <stdio.h>

typedef struct{
    int number;           /* 社員番号    */
    char name[20];        /* 氏名        */
    double height;        /* 身長        */
    double weight;        /* 体重        */
} SYAIN;

void output( [ ① ] *);

void main(void)
{
    [ ② ] syain1 = { 2200, "三河 太郎", 170.0 , 60.0 };

    output( &syain1 );
}

void output( [ ③ ] *fw)
{
    printf("社員番号 %d¥n", fw->number );

    printf("氏名     %s¥n", fw->name );

    printf("身長     %4.1lf¥n", fw->height );

    printf("体重     %4.1lf¥n", fw->weight );
}
```

## 【問題2】 次の□部分を埋め、以下のプログラムを完成させなさい。

**【処理内容】**
社員データを入力して表示する。

```
#include <stdio.h>

typedef struct{
    int  number;      /* 社員番号    */
    char name[20];    /* 氏名        */
    char shozoku[20]; /* 所属部署    */
} LIST1;

typedef struct{
    double   height;  /* 身長 */
    double   weight;  /* 体重 */
} LIST2;

typedef struct{
    LIST1    list1;
    ①       list2;
    int      yy;  /* 生年月日(年) */
    int      mm;  /* 生年月日(月) */
    int      dd;  /* 生年月日(日) */
} SYAIN;

void output(SYAIN *);

void main(void)
{
    SYAIN syain1 = { 2200, "三河 太郎", "技術部", 170.0, 60.0, 1985, 10, 8 };

    output( &syain1 );
}

void output(SYAIN *fw)
{
    printf("社員番号 %d¥n", ②          );

    printf("氏名     %s¥n", ③          );

    printf("所属部署 %s¥n", ④          );

    printf("身長     %4.1lf¥n", ⑤     .height );

    printf("体重     %4.1lf¥n", ⑥     .weight );

    printf("生年月日 %d.%d.%d¥n", fw->yy, fw->mm, fw->dd );
}
```

# 解答

## Training8 データ構造とポインタ

### Lesson1 一次元配列

| | |
|---|---|
| 問題 1 | ①85　②46　③31　④62 |
| 問題 2 | ①10　②25　③25　④75<br>⑤3　⑥3　⑦3 |
| 問題 3 | ①0　②100　③i++<br>④array[i] = 0; |
| 問題 4 | ①0　②9　③i++<br>④Sum += array[i]; or sum = sum + array[i]; |

### Lesson2 多次元配列

| | |
|---|---|
| 問題 1 | ①10　②34　③57　④53 |
| 問題 2 | ①100　②223　③335　④132 |
| 問題 3 | ①max_x　②max = array[j][i]<br>③array[j][i]　④J<br>⑤min = array[j][i] |

### Lesson3 文字列

| | |
|---|---|
| 問題 1 | ①3　②10 |
| 問題 2 | ①'a'　②'c'　③不定値　④'b' |
| 問題 3 | ①'1'　②'4'　③'2'<br>④'¥0' (NULL) or 0x00 |
| 問題 4 | ①'g'　②'D'　③'c'<br>④'¥0' (NULL) or 0x00 |
| 問題 5 | ①'a'　②'l'　③'l'<br>④'¥0' (NULL) or 0x00 |
| 問題 6 | ①j = 0　②str1[i] != '¥0'<br>③str1[i]　④j++<br>⑤i = 0<br>⑥While ( str2[i] != '¥0')<br>⑦str2[i]　⑧i++　⑨j　⑩'¥0' |

### Lesson4 配列のデータの大きさ

| | |
|---|---|
| 問題 1 | ①10　②10　③40　④40<br>⑤20　⑥80 |
| 問題 2 | ①3　②2　③16　④10<br>⑤6　⑥48 |
| 問題 3 | ①50　②200　③168　④96<br>⑤24　⑥160 |

### Lesson5 配列のまとめ

| | |
|---|---|
| 問題 1 | ①i = 3,<br>j = 4　②D　③C<br>④お預かりした荷物の重さは 3.00kg、長さは<br>115.00 cmです。<br>料金は 1075 円です。 |

### Lesson6 ポインタ変数

| | |
|---|---|
| 問題 1 | 30 |
| 問題 2 | 70 |
| 問題 3 | 10 |
| 問題 4 | 8 |
| 問題 5 | s |
| 問題 6 | 10　10 |

### Lesson7 ポインタ配列

| | |
|---|---|
| 問題 1 | Tuesday |
| 問題 2 | ①*color[ ] = {"red" , "green" , "blue" }<br>②color[i] |
| 問題 3 | void main(void)<br>{<br>char name[4][10]= {<br>{"taro"},{"hanako"},{"kouji"},<br>{"hiroyuki"}<br>};<br>printf("%s¥n", name[1]);<br>}<br>※ 配列要素数は、文字列が格納できる大きさが確保されれば問題無しとする。 |

### Lesson8 ポインタのデータの大きさ

| | |
|---|---|
| 問題 1 | ①1　②4　③4　④4<br>⑤2　⑥4 |
| 問題 2 | ①4　②1　③4　④4<br>⑤2　⑥4 |
| 問題 3 | ①60　②4　③1 |

**【解説】問題 3**

val_a のアドレスをポインタ変数 p1 に格納し、p1 の値であるアドレスの先の値を変更することで、val_a の値も変化します。よって上記のプログラムを実行した結果、val_a の値は「12×5 =60」となります。

sizeof(p1)とした時のサイズはアドレスデータの大きさなので**データ型に関係なく 4 バイト**です。しかし、**sizeof(*p1)**とした場合はアドレスの先の値のデータの大きさなので 1 バイトとなるのです。

## Lesson9 ポインタと関数

| | |
|---|---|
| 問題１ | x = 3 , y = 4 , z = 20 |

| | |
|---|---|
| 問題２ | ①int *x , int *y　②*x = *y;<br>③&data1[i],&data2[i] |

| | |
|---|---|
| 問題３ | ①int x , int y , int *mult<br>②*mult = x * y　③x , y , &z |

| | |
|---|---|
| 問題４ | ①　A: char *　B: *str<br>C: str++　D: *str = '¥0' (0x00)<br>number = 15:　k = 1<br>②　number = 256:　k = 3<br>number = 65535:　k = 4 |

**【解説】問題２**
まず main 関数から配列 data1,data2 に格納されている値を先頭から順に入れ替えるため二つの配列の i 番目のアドレスを swap 関数に送ります(i は 0～2 まで変化する)。
次に main から送られてきた2つ配列のアドレスを swap 関数でポインタ変数として受け取り、入れ替えを行います。アドレスが持つ値を変化させているため、そのまま 2 つの配列を出力すれば中身が入れ替わっています。

**【解説】問題３**
この問いの mult 関数には戻り値がないため、x と y の積を最終的に main の z で出力しなければなりません。
そのため、z のアドレスを渡し、アドレスの中の値を変化させます。

## Lesson10 構造体データへのアクセス方法

| | |
|---|---|
| 問題１ | ①syain1.number　②syain1.name<br>③syain1.height |

| | |
|---|---|
| 問題２ | ①fw->number　②fw->name<br>③fw->height　④fw->weight |

| | |
|---|---|
| 問題３ | ①syain1.number　②syain1.pname<br>③syain1.height　④syain1.weight<br>⑤fw->number　⑥fw->pname[i++]<br>⑦fw->pname[i]　⑧fw->height<br>⑨fw->weight |

**【解説】問題１**
**構造体のテンプレート宣言について**
```
struct SYAIN{
   int number;        /* 社員番号    */
   char name[20];     /* 氏名        */
   double height;     /* 身長        */
};
```
この宣言により、struct SYAIN という 4(int) + 1×20(char[20])+ 8(double) = 32 バイトの大きさの型を作成したことになります。

**構造体の宣言、初期化について**
```
struct SYAIN syain1 = { 2200, "三河　太郎", 170.0 };
```
32 バイト型の syain1 という変数を宣言し、変数宣言の中で初期化を行っています。(2200 が number 、"三河　太郎"が name[20]、170.0 が height に対応しています。)

**社員番号、氏名、身長の出力について**
```
printf("社員番号 %d¥n", syain1.number );
printf("氏名     %s¥n", syain1.name );
printf("身長     %lf¥n", syain1.height );
```

ここでは、構造体の中のそれぞれの項目を出力しています。

配列ではそれぞれに対応した要素番号を指定して値を取得していましたが、構造体では　変数名.メンバ名　とすることで、値を取得することができます。

**【解説】問題２**
input 関数では構造体をポインタ変数の fw ( *fw )で受け取っている。よって input 関数の中では構造体参照する際にはアロー演算子を用いて参照します。

## Lesson11 構造体変数

| | |
|---|---|
| 問題１ | ①5　②12345678　③3.1415926535<br>※②は型が long 型であることに注意。 |

| | |
|---|---|
| 問題２ | ①dog　②2　③15　④2.8 |

## Lesson12 構造体配列

| | |
|---|---|
| 問題１ | ①2　②0.0　③0.0　④4<br>⑤0.0　⑥2.8 |

| | |
|---|---|
| 問題２ | ①オ　②カ　③イ　④ア |

**【解説】問題１**
この問いでは一時的にすべての構造体配列に sa[0]をコピーしています。そして後で構造体配列を指定し、値を代入しています。

**【解説】問題２**
この問いは商品の金額の大きい順に並び変えるソートの問題です。
並び変えの方法ですが、構造体の中身を入れ替えるのではなく、配列の要素番号だけを並びかえるプログラムが問題のプログラムになります。

① 構造体の先頭アドレスの受け取り方の問題です。
② バブルソートの問題です。「並び変える値-1」が実際に入ります。
③ 左の数を右と比べ、右が大きかったら入れ替えるので str[i]となる
④　③が str[i]なので、str[j]となる。

## Lesson13 構造体のデータの大きさ

| | |
|---|---|
| 問題１ | ①4　②20　③8　④32 |

| | |
|---|---|
| 問題２ | ①4　②20　③8　④960 |

| | |
|---|---|
| 問題３ | ①4　②20　③8　④　96 |

**【解説】問題２**
構造体配列数が 30 のため、一つの構造体サイズ 32 に 30 を掛けた数がこの構造体配列のサイズです。

## Lesson14 ポインタを使用した構造体

| | |
|---|---|
| 問題１ | ①2　②4<br>③0.0 |

| | |
|---|---|
| 問題２ | ①5　②50　③5　④50 |

**【解説】**
sa のアドレスを sb が受け取っているため、sb の値を変更すると sa の値も変化します。

## Lesson15 typedef を使用した構造体

| | |
|---|---|
| 問題１ | ①SYAIN　②SYAIN　③SYAIN |

| | |
|---|---|
| 問題２ | ①LIST2　②fw->list1.number<br>③fw->list1.name<br>④fw->list1.shozoku<br>⑤　fw->list2.height　⑥fw->list2.weight |