2006 年 1 月 23 日（火）実施

構造体と typedef

typedef 宣言によって，struct 構造体タグ名という表記を再定義し，データ型名のように扱うことができる。構文は typedef struct 構造体タグ名 {・・・} 再定義名; となり，この場合の構造体変数の宣言は，再定義名を用いて行うことができる。なお，ここでは構造体タグ名は省略可能である。

例題 1

次のプログラムを入力し，翻訳・編集して実行形式のファイルを作成し，実行せよ。ここで，ソースプログラム名は prog13-1.c とする。

```
/* prog13-1.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{
    char   n[7];      /* name */
    double    s;      /* side */
    double    h;      /* height */
    double    a;      /* area */
} TRIRECT;            /* triangle/rectangle */

TRIRECT getArea(TRIRECT, int);

int main(void)
{

    TRIRECT figure = {"", 0.0, 0.0, 0.0};
    int type;

    printf("図形が三角形であれば 0, 長方形であれば 1, "
     "正方形であれば 2 を入力してください：");
    scanf(" %d", &type);

    if (type == 0)
    {
        printf("三角形の底辺の長さを入力してください：");
        scanf(" %lf", &figure.s);
        printf("三角形の高さを入力してください：");
        scanf(" %lf", &figure.h);
    }
    else if (type == 1)
    {
        printf("長方形の一辺 X の長さを入力してください：");
        scanf(" %lf", &figure.s);
        printf("長方形の一辺 Y の長さを入力してください：");
        scanf(" %lf", &figure.h);
    }
    else if (type == 2)
    {
```

```
        printf("正方形の一辺の長さを入力してください：");
        scanf(" %lf", &figure.s);
        figure.h = figure.s;
    }
    else
    {
        printf("入力する数値は 0, 1, 2 のいずれかです。");
        exit(1);
    }

    figure = getArea(figure, type);

    printf("%s の面積は%f です。¥n", figure.n, figure.a);

    return 0;
}

TRIRECT getArea(TRIRECT fig, int t)
{
    if (t == 0)
    {
        strncpy(fig.n, "三角形", 6);
        fig.a = fig.s * fig.h / 2;
    }
    else
    {
        fig.a = fig.s * fig.h;

        if (t == 1)
            strncpy(fig.n, "長方形", 6);
        else
            strncpy(fig.n, "正方形", 6);
    }

    return fig;
}
```

【解説】

TRIRECT figure は，宣言の際に初期化を行っている。

構造体を指すポインタ

ポインタ変数の解説の際（第 9 回教材）に，構造体を指すポインタ変数が利用できることを述べた。これは，struct 構造体タグ名 *ポインタ変数名; または 再定義名 *ポインタ変数名; という宣言によって利用可能である。

ポインタ変数が構造体を指すには,例えばポインタ変数名を sp,構造体変数名を str とすると，sp = &str; という代入を行えばよい。このとき，str にはメンバが存在するが，例えばメンバの一つが x であるとすると，str.x をポインタ変数で表すには，(*sp).x となる。これを*sp.x と書くと，*(sp.x)を意味することとなり，誤りである。(*sp).x は簡潔に sp->x と表記することができる。

例題 2

次のプログラムを入力し，翻訳・編集して実行形式のファイルを作成し，実行せよ。ここで，ソースプログラム名は prog13-2.c とする。

```
/* prog13-2.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 5
#define MAXBUF 256

typedef struct
{
    char         id[8];
    char      name[21];
    char      class[9];
    char      point[4];
    char       eval[2];
} STUDENT;

void dispdata(STUDENT *, int);

int main(void)
{

    STUDENT prog[MAX];

    int i;
    char temp[MAXBUF];
    char headln[MAXBUF];
    FILE *fpr;

    if (NULL == (fpr = fopen("stprog.csv","r")))
    {
        printf("stprog データファイルが開けません。¥n");
        exit(1);
    }

    fgets(headln, sizeof headln, fpr);

    printf("   %s", headln);

    for (i=0; i<MAX; i++)
    {
        if (fgets(temp, sizeof temp, fpr) == NULL) break;

        strcpy(prog[i].id, strtok(temp,","));
        strcpy(prog[i].name, strtok(NULL,","));
        strcpy(prog[i].class, strtok(NULL,","));
        strcpy(prog[i].point, strtok(NULL,","));
        strcpy(prog[i].eval, strtok(NULL,"¥n"));

        dispdata(&prog[i], i);
    }
```

```
    fclose(fpr);

    return 0;
}

void dispdata(STUDENT *data, int i)
{
        printf("%d) %s,%s,%s,%s,%s¥n", i+1,
          data->id,data->name,data->class,data->point,data->eval);
}
```

【解説】

prog12-2.c では，strtok で指定した文字列を",¥n"として，各項目がカンマで区切られている場合と項目ごとに改行されている場合，1 レコードの最後の項目の後ろにもカンマが付いている場合に対応していたが，このプログラムでは prog12-1.exe を正しく実行して作成されたファイル stprog.csv の存在を前提としている。