

構造体

三池 克明

処理をブロックとしてまとめるが関数ですが、複数の変数をひとまとめにするのが本章で解説する構造体です。

—目 次—

1.	構造体を使ってみる.....	1
1.1.	構造体を使ったプログラム.....	1
1.2.	構造体とポインタ.....	5
2.	typedef を使う	7
3.	構造体を使った集計処理.....	10
3.1.	クラスの合計点を集計をする.....	10
3.2.	三科目（国語、数学、英語）の集計をする.....	13
4.	演習問題.....	16

1. 構造体を使ってみる

1.1. 構造体を使ったプログラム

以下のプログラムを基に構造体を使ったプログラムを作ってみましょう。

syukei7.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: main()
7: {
8:     FILE    *in, *out;
9:     char    str[LINESIZE];
10:    int      kokugo, sugaku;
11:    double   heikin;
12:    int      n;
13:
14:    if((in = fopen("score.csv", "r")) == NULL) {
15:        printf("score.csv のオープンに失敗しました。¥n");
16:        exit(1);
17:    }
18:    if((out = fopen("syukei.csv", "w")) == NULL) {
19:        printf("syukei.csv のオープンに失敗しました。¥n");
20:        fclose(in);
21:        exit(1);
22:    }
23:
24:    fprintf(out, "番号, 国語, 数学, 平均¥n");
25:    for(n = 1; fgets(str, LINESIZE, in) != NULL; n++) {
26:        sscanf(str, "%d,%d", &kokugo, &sugaku);
27:        heikin = (kokugo + sugaku) / 2.0;
28:        fprintf(out, "%d,%d,%d,%g¥n", n, kokugo, sugaku, heikin);
29:    }
30:
31:    fclose(in);
32:    fclose(out);
33: }
```

こちらが構造体を使ったプログラムです。

ソースを入力し、コンパイル・実行して結果を確かめてみましょう。

struct1.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: struct  score{
7:     int    kokugo;
8:     int    sugaku;
9:     double heikin;
10: };
11:
12: main()
13: {
14:     FILE      *in, *out;
15:     char      str[LINESIZE];
16:     struct score sco;
17:     int       n;
18:
19:     if((in = fopen("score.csv", "r")) == NULL){
20:         printf("score.csv のオープンに失敗しました。¥n");
21:         exit(1);
22:     }
23:     if((out = fopen("syukei.csv", "w")) == NULL){
24:         printf("syukei.csv のオープンに失敗しました。¥n");
25:         fclose(in);
26:         exit(1);
27:     }
28:
29:     fprintf(out, "番号, 国語, 数学, 平均¥n");
30:     for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
31:         sscanf(str, "%d,%d", &sco.kokugo, &sco.sugaku);
32:         sco.heikin = (sco.kokugo + sco.sugaku) / 2.0;
33:         fprintf(out, "%d,%d,%d,%g¥n", n, sco.kokugo, sco.sugaku, sco.heikin);
34:     }
35:
36:     fclose(in);
37:     fclose(out);
```

なお、実行する前に「syukei.csv」を削除しておく、

- 実行後「syukei.csv」が作成された
出力はされている
- 実行しても「syukei.csv」が見つからない
出力の段階で何かしらの問題がある。

となるので、きちんと実行されたかどうかを確認する目安になるでしょう。

	A	B	C	D
1	番号	国語	数学	平均
2	1	100	80	90
3	2	60	50	55
4	3	70	80	75
5	4	90	60	75
6	5	80	80	80
7	6	50	70	60
8	7	90	100	95
9	8	80	90	85
10	9	70	90	80
11	10	60	70	65

作成された「syukei.csv」を Microsoft Excel で開き、左図のようになっていれば問題はありません。

それではソースプログラムをたどってみましょう。

```

6: struct score{
7:     int    kokugo;
8:     int    sugaku;
9:     double heikin;
10: };

```

これは構造体の宣言をしています。

基本的な文型は、

```
struct タグ名 {
    型    要素名;
    ...  ...
};
```

となります。

```
15:    ...
16:    struct score    sco;
17:    ...
```

16 行目では、その構造体を変数として使うことを宣言しています。
基本的な文型は、

```
struct タグ名    変数名, ...;
```

です。

この様に構造体を使用するには、

- a. 構造体の中身を宣言する
- b. その構造体を変数として使用することを宣言する

の 2 ステップが必要です。

```
31:    sscanf(str, "%d,%d", &sco.kokugo, &sco.sugaku);
32:    sco.heikin = (sco.kokugo + sco.sugaku) / 2.0;
33:    fprintf(out, "%d,%d,%d,%g\n", n, sco.kokugo, sco.sugaku, sco.heikin);
```

また構造体の各要素はこのように、

```
変数名. 要素名
```

とします。

1.2. 構造体とポインタ

構造体をポインタとして宣言することもできます。

構造体内部に関する処理を関数にまとめました。

こうすることで長いソースコードをある程度読みやすくすることができます。

struct2.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: struct  score{
7:     int    kokugo;
8:     int    sugaku;
9:     double heikin;
10: };
11:
12: void readscore(struct score *, char *);
13: void setaverage(struct score *);
14:
15: main()
16: {
17:     FILE          *in, *out;
18:     char          str[LINESIZE];
19:     struct score  sco;
20:     int           n;
21:
22:     if((in = fopen("score.csv", "r")) == NULL){
23:         printf("score.csv のオープンに失敗しました。¥n");
24:         exit(1);
25:     }
26:     if((out = fopen("syukei.csv", "w")) == NULL){
27:         printf("syukei.csv のオープンに失敗しました。¥n");
28:         fclose(in);
29:         exit(1);
```

```

30:     }
31:
32:     fprintf(out, "番号, 国語, 数学, 平均¥n");
33:     for(n = 1; fgets(str, LINESIZE, in) != NULL; n++) {
34:         readscore(&sco, str);
35:         setaverage(&sco);
36:         fprintf(out, "%d,%d,%d,%g¥n", n, sco.kokugo, sco.sugaku, sco.heikin);
37:     }
38:
39:     fclose(in);
40:     fclose(out);
41: }
42:
43: void readscore(struct score *sco, char *str)
44: {
45:     int          kokugo, sugaku;
46:
47:     sscanf(str, "%d,%d", &kokugo, &sugaku);
48:     sco->kokugo = kokugo;
49:     sco->sugaku = sugaku;
50: }
51:
52: void setaverage(struct score *sco)
53: {
54:     sco->heikin = (sco->kokugo + sco->sugaku) / 2.0;
55: }

```

それでは重要な部分の解説をしましょう。

```

48:     sco->kokugo = kokugo;
49:     sco->sugaku = sugaku;
...     .....
54:     sco->heikin = (sco->kokugo + sco->sugaku) / 2.0;

```

このように構造体へのポインタの各要素にアクセスするには、

ポインタ名->要素名

とします。ピリオド“.”ではないことに注意してください。

2. typedef を使う

「struct score」を typedef を使って「SCORE」に定義しておきました。
それが以下のソースプログラムです。
宣言の語が短くなったのがわかります。

struct3.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: typedef struct {
7:     int    kokugo;
8:     int    sugaku;
9:     double heikin;
10: } SCORE;
11:
12: void readscore(SCORE *, char *);
13: void setaverage(SCORE *);
14:
15: main()
16: {
17:     FILE    *in, *out;
18:     char    str[LINESIZE];
19:     SCORE   sco;
20:     int     n;
21:
22:     if((in = fopen("score.csv", "r")) == NULL){
23:         printf("score.csv のオープンに失敗しました。¥n");
24:         exit(1);
25:     }
26:     if((out = fopen("syukei.csv", "w")) == NULL){
27:         printf("syukei.csv のオープンに失敗しました。¥n");
28:         fclose(in);
29:         exit(1);
30:     }
31:
32:     fprintf(out, "番号, 国語, 数学, 平均¥n");
33:     for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
```

```

34:         readscore(&sco, str);
35:         setaverage(&sco);
36:         fprintf(out, "%d,%d,%d,%g\n", n, sco.kokugo, sco.sugaku, sco.heikin);
37:     }
38:
39:     fclose(in);
40:     fclose(out);
41: }
42:
43: void readscore(SCORE *sco, char *str)
44: {
45:     int    kokugo, sugaku;
46:
47:     sscanf(str, "%d,%d", &kokugo, &sugaku);
48:     sco->kokugo = kokugo;
49:     sco->sugaku = sugaku;
50: }
51:
52: void setaverage(SCORE *sco)
53: {
54:     sco->heikin = (sco->kokugo + sco->sugaku) / 2.0;
55: }

```

typedef とは型名の同義語を作ります。#define に似ていますね。
 ですが typedef は型名の同義語に限定されます。
 基本的な文法は、

```
typedef 既存の型名    同義語
```

となります。例えば、

```
typedef  char    BYTE;
BYTE    data;
```

とした場合、変数 data は char 型となります。
 また構造体の場合は

```
typedef struct {  
    型      要素名;  
    ...    ...  
} 構造体の同義語;
```

となります。

—FILE 型—

前章からファイル入出力で利用しているファイルポインタはFILE型のポインタのように利用しています。

実はこのFILE型はtypedefで定義された構造体です。

まったく意識してなかったと思います。

このように構造体を利用すれば細かいデータの集まりをまとめて、プログラム全体像を把握しやすくなります。

3. 構造体を使った集計処理

前章で作成した成績集計処理プログラムを、構造体を使って改良していきます。

3.1. クラスの合計点を集計をする

各科目や平均点の合計点を最後の行に書き込ませましょう。

struct4.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: typedef struct {
7:     int    kokugo;
8:     int    sugaku;
9:     double heikin;
10: } SCORE;
11:
12: void clearscore(SCORE *);
13: void readscore(SCORE *, char *);
14: void addscore(SCORE *, SCORE *);
15: void setaverage(SCORE *);
16:
17: main()
18: {
19:     FILE    *in, *out;
20:     char    str[LINESIZE];
21:     SCORE   sco, goukei;
22:     int     n;
23:
24:     if((in = fopen("score.csv", "r")) == NULL){
25:         printf("score.csv のオープンに失敗しました。¥n");
26:         exit(1);
27:     }
28:     if((out = fopen("syukei.csv", "w")) == NULL){
```

```

29:             printf("syukei.csv のオープンに失敗しました。¥n");
30:             fclose(in);
31:             exit(1);
32:         }
33:
34:         clearscore(&goukei);
35:         fprintf(out, "番号, 国語, 数学, 平均¥n");
36:         for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
37:             readscore(&sco, str);
38:             setaverage(&sco);
39:             addscore(&goukei, &sco);
40:             fprintf(out, "%d,%d,%d,%g¥n", n, sco.kokugo, sco.sugaku, sco.heikin);
41:         }
42:         fprintf(out, "合計,%d,%d,%g¥n", goukei.kokugo, goukei.sugaku, goukei.heikin);
43:
44:         fclose(in);
45:         fclose(out);
46:     }
47:
48: void clearscore(SCORE *sco)
49: {
50:     sco->kokugo = 0;
51:     sco->sugaku = 0;
52:     sco->heikin = 0.0;
53: }
54:
55: void readscore(SCORE *sco, char *str)
56: {
57:     int    kokugo, sugaku;
58:
59:     sscanf(str, "%d,%d", &kokugo, &sugaku);
60:     sco->kokugo = kokugo;
61:     sco->sugaku = sugaku;
62: }
63:
64: void setaverage(SCORE *sco)
65: {
66:     sco->heikin = (sco->kokugo + sco->sugaku) / 2.0;
67: }
68:
69: void addscore(SCORE *gou, SCORE *sco)
70: {

```

```

71:      gou->kokugo += sco->kokugo;
72:      gou->sugaku += sco->sugaku;
73:      gou->heikin += sco->heikin;
74:  }

```

	A	B	C	D
1	番号	国語	数学	平均
2	1	100	80	90
3	2	60	50	55
4	3	70	80	75
5	4	90	60	75
6	5	80	80	80
7	6	50	70	60
8	7	90	100	95
9	8	80	90	85
10	9	70	90	80
11	10	60	70	65
12	合計	750	770	760

このように最後の行に国語・数学・平均のクラス合計点が書き込まれました。

念のため SUM 関数を使って検算をしておいたほうが良いでしょう。

3.2. 三科目（国語、数学、英語）の集計をする

	A	B	C
1	100	80	50
2	60	50	90
3	70	80	80
4	90	60	70
5	80	80	60
6	50	70	80
7	90	100	50
8	80	90	80
9	70	90	60
10	60	70	80
11	80	100	70
12	50	60	100
13	80	70	90
14	60	90	90
15	80	80	70
16	70	50	100
17	100	90	60
18	90	80	70
19	90	70	90
20	70	60	80

「score.csv」を左図のようにデータを追加します。

なお3列目のデータは今回追加する英語の点数です。

それではこのデータの集計をするプログラムを作りましょう。

struct5.cpp

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: #define LINESIZE      128
5:
6: typedef struct {
7:     int    kokugo;
8:     int    sugaku;
9:     int    eigo;
10:    double heikin;
11: } SCORE;
12:
13: void clearscore(SCORE *);
14: void readscore(SCORE *, char *);
15: void addscore(SCORE *, SCORE *);
16: void setaverage(SCORE *);
17:
18: main()
19: {
```

```

20: FILE *in, *out;
21: char str[LINESIZE];
22: SCORE sco, goukei;
23: int n;
24:
25: if((in = fopen("score.csv", "r")) == NULL){
26:     printf("score.csv のオープンに失敗しました。¥n");
27:     exit(1);
28: }
29: if((out = fopen("syukei.csv", "w")) == NULL){
30:     printf("syukei.csv のオープンに失敗しました。¥n");
31:     fclose(in);
32:     exit(1);
33: }
34:
35: clearscore(&goukei);
36: fprintf(out, "番号, 国語, 数学, 英語, 平均¥n");
37: for(n = 1; fgets(str, LINESIZE, in) != NULL; n++){
38:     readscore(&sco, str);
39:     setaverage(&sco);
40:     addscore(&goukei, &sco);
41:     fprintf(out, "%d,%d,%d,%d,%g¥n", n, sco.kokugo, sco.sugaku, sco.eigo, sco.heikin);
42: }
43: fprintf(out, "合計,%d,%d,%d,%g¥n", goukei.kokugo, goukei.sugaku, goukei.eigo, goukei.heikin);
44:
45: fclose(in);
46: fclose(out);
47: }
48:
49: void clearscore(SCORE *sco)
50: {
51:     sco->kokugo = 0;
52:     sco->sugaku = 0;
53:     sco->eigo = 0;
54:     sco->heikin = 0.0;
55: }
56:
57: void readscore(SCORE *sco, char *str)
58: {
59:     int kokugo, sugaku, eigo;
60:
61:     sscanf(str, "%d,%d,%d", &kokugo, &sugaku, &eigo);

```

```

62:         sco->kokugo = kokugo;
63:         sco->sugaku = sugaku;
64:         sco->eigo = eigo;
65:     }
66:
67: void setaverage(SCORE *sco)
68: {
69:     sco->heikin = (sco->kokugo + sco->sugaku + sco->eigo) / 3.0;
70: }
71:
72: void addscore(SCORE *gou, SCORE *sco)
73: {
74:     gou->kokugo += sco->kokugo;
75:     gou->sugaku += sco->sugaku;
76:     gou->eigo += sco->eigo;
77:     gou->heikin += sco->heikin;
78: }

```

	A	B	C	D	E
1	番号	国語	数学	英語	平均
2	1	100	80	50	76.6667
3	2	60	50	90	66.6667
4	3	70	80	80	76.6667
5	4	90	60	70	73.3333
6	5	80	80	60	73.3333
7	6	50	70	80	66.6667
8	7	90	100	50	80
9	8	80	90	80	83.3333
10	9	70	90	60	73.3333
11	10	60	70	80	70
12	11	80	100	70	83.3333
13	12	50	60	100	70
14	13	80	70	90	80
15	14	60	90	90	80
16	15	80	80	70	76.6667
17	16	70	50	100	73.3333
18	17	100	90	60	83.3333
19	18	90	80	70	80
20	19	90	70	90	83.3333
21	20	70	60	80	70
22	合計	1520	1520	1520	1520

このように3科目の生徒別平均点と科目別合計点が集計されるようになりました。

4. 演習問題

- 15-1. 「score.csv」に理科、社会の点数データを追加し、それを集計するプログラム「ensyu15-1.cpp」を作りなさい。なお追加する点数データはテストデータなので各自で適当に入力して構わない。
- 15-2. 「ensyu15-1.cpp」では科目ごと（平均点含む）の合計点を最後の行に算出しているが、これを平均点にするプログラム「ensyu15-2.cpp」を作りなさい。
- 15-3. 「ensyu15-2.cpp」では科目ごと（平均含む）の平均点を最後の行に算出しているが、更に科目ごと（平均点含む）の最高点を算出するプログラム「ensyu15-3.cpp」を作りなさい。