情報基礎 BII　課題の解説

柳本 豪一

ソースコード

課題 1

```
#include <stdio.h>

#define MAX_NUM 1000

int main(void){
  int i, j, sum;

  for(i = 2; i <= MAX_NUM; i++){
    sum = 0;
    for(j = 1; j <= i/2; j++){
      if((i % j) == 0){
        sum += j;
      }
    }
    if(i == sum){
      printf("%d\n", i);
    }
  }

  return(0);
}
```

課題 2

```
#include <stdio.h>

#define NAME_BUF 256
#define MAX_PERSON 5

typedef struct{
  char name[NAME_BUF];
  int math;
  int english;
  double ave;
} person;

int main(void){
  int i, j;
  person student[MAX_PERSON], tmp;

  for(i = 0; i < MAX_PERSON; i++){
    printf("Name: ");
    scanf("%s", student[i].name);
    printf("Score(Math): ");
    scanf("%d", &student[i].math);
    printf("Score(English): ");
    scanf("%d", &student[i].english);
    student[i].ave =
        (student[i].math + student[i].english) / 2.0;
  }

  for(i = 0; i < MAX_PERSON-1; i++){
    for(j = i+1; j < MAX_PERSON; j++){
      if(student[i].ave < student[j].ave){
        tmp = student[i];
        student[i] = student[j];
        student[j] = tmp;
      }
    }
  }

  for(i = 0; i < MAX_PERSON; i++){
    printf("%s %lf\n", student[i].name,
                                  student[i].ave);
  }

  return(0);
}
```

課題 1

1000 までの数字の中で完全数を求めるためには、各数字の約数をもれなく探すことが必要である。一番簡単な方法としては、1 から自分自身未満の数字で割って、割り切れたものの和を求めるのが基本である。しかし、自分自身を除いた約数の最大数は必ず自分自身の半分以下の和であるはずなので、約数を調べる計算は上記の和まで調べれば良いこととなる。このアイデアをもとにループ数を制限したものが回答例のプログラムである。とくに難しいところはなかったと思われる。

課題 2

授業のサンプルを見れば問題なくできたと思われる。気を付けなくてはいけないところは、まず平均値を保存する構造体のメンバ変数を必ず float 型または double 型にしておくことである。平均を計算する際には、必ず除算が含まれるので小数が出てくる可能性が高い。したがって、小数を保存できるようにしておく必要がある。そして、平均値を計算するときにも、数学と英語の点数を int 型として保存している人は、キャスト変換をすることを忘れないようにしなくてはいけない。上記のプログラムでは、2.0 で割ることによって暗黙のうちにキャスト変換を行なっている。

最後に先週の授業で使った資料中のサンプルプログラムの間違いであるが、構造体の配列に名前を登録するとき、プログラムのように書く場合には&は不要である。理由は、scanf 関数では記憶する変数のアドレスを必要としているため、配列の場合には配列名だけで、配列の先頭のアドレスを表わすので、&は不要となる。これは構造体のメンバ変数でも同じであり、配列 name の配列名だけのときには&は不要となる。一方、メンバ変数が配列ではなくて int 型の変数の場合には、従来通り&が必要となる。

typedef で構造体の型を定義する場合には、構造体タグはあってもなくても良いので、プログラムのように書くことも可能である。