情報研究会CACTUS

第三回　C言語講習

先週の復習

- switch文
- while・for文
- 数学関数

```
switch(式){
        case 1:
        case 2:
        default:
}
```

```
while(式){
        処理
        i++;
}
```

```
for(初期化;条件式;変化式){
        処理
}
```

今週の内容

- 精度・フィールド幅
- 配列
- 関数

精度・フィールド幅

●精度

小数点以下の桁数を少数第何位まで表示するか

%.数字f

ルート2を少数で表示する例

```c
#include <stdio.h>

int main(void){

    double two = 1.41421356237;

    printf("ルート2は%.4fです\n", two);

    return 0;
}
```

実行結果

```
ルート2は1.4142です
続行するには何かキーを押してください
```

精度・フィールド幅

- フィールド幅

  最低で何桁表示するかを決める。足りない桁は空白スペースで埋まる

%数字d

6桁に揃えて数字を表示する例

```
3   int main(void){
4   
5       int n = 364;
6       int m = 500089;
7       double l = 3.1415;
8   
9       printf("n=%6d\n", n);
10      printf("m=%6d\n", m);
11      printf("l=%6.2f\n", l);
12  
13      return 0;
14  }
```

実行結果

```
n=   364
m=500089
l=  3.14
```

サンプルプログラム1

4人の数学のテストの合計点を表示するプログラム

```c
#include <stdio.h>

int main(void){

    int i, sum = 0;
    int math[4] = { 44, 68, 87, 51 };

    for (i = 0; i < 4; i++){

        printf("math[%d]の中身は%dです\n", i, math[i]);
        sum += math[i];

    }

    printf("4人の数学のテストの合計は%d点です\n", sum);

    return 0;
}
```

```
math[0]の中身は44です
math[1]の中身は68です
math[2]の中身は87です
math[3]の中身は51です
4人の数学のテストの合計は250点です
```

実行結果

配列

●配列とは

同じ型・同じ名前の変数を複数同時に宣言する

型　名前[要素数];

・例　実数型配列を5つ宣言

```
double array[5];
```

・配列名[要素番号(添え字)]で中身のデータにアクセス。添え字は0から始まる

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

・添え字に変数・定数を使用することも可能

```
n=3;
printf("%d",array[n]);
```

配列

- 配列を表示してみる

```c
#include <stdio.h>

int main()
{
    int array[3];

    array[0] = 10;
    array[1] = 100;
    array[2] = 1000;

    printf("%d\n%d\n%d\n", array[0], array[1], array[2]);

    return 0;
}
```

```
10
100
1000
```

・注意しておくこと （要素数4の配列の場合）

一番最後は[要素数-1]

配列

●配列の初期化

・{}でくくってコンマで区切ることで値を代入することが出来る

① `int array[3] = {6, 8, 15};`　（a[0]に6、a[1]に8、a[2]に15を代入）

② `int array[3] = {0};`　（a[0]～a[2]全てに0を代入）

③ `int array[3] = {6};`　（a[0]に6、初期値がないところには全て0を代入）

・また、①で初期化するときに要素数を省略することが出来る

`int array[] = {6, 8, 15};`　（①と同義）

確認問題1

サンプルプログラムを書き換え、以下の5人の生徒の
英語・国語の点数それぞれの合計と平均を求める
プログラムを作成してください

	生徒1	生徒2	生徒3	生徒4	生徒5
英語	57	80	61	77	45
国語	83	53	45	31	58

実行結果

```
英語のテストの合計は320点，平均点は64です
国語のテストの合計は270点，平均点は54です
```

サンプルプログラムには改善すべき点が1つある。考えてみよう

配列

●2次元配列

配列の要素数をもう一つ指定する

型　名前[要素数1] [要素数2] ;

例　int ary[2][4]

ary[0][0]	ary[0][1]	ary[0][2]	ary[0][3]
ary[1][0]	ary[1][1]	ary[1][2]	ary[1][3]

●多次元配列

要素数を増やすことで配列の次元を増やすことが出来る。
[ ]の数に応じてn次元配列と呼び2次元配列以上のものを多次元配列と呼ぶ

例　5次元配列　　int ary[4][4][4][4][4];

配列

●2重ループ

for文の中にfor文が入っている構造

```
for (i=0;i<5;i++) {

    for (j=0; j<3; j++) {
        処理;
    }
}
```

・forループ1が1回処理されるのに対して、forループ2は最後まで回りきる

・for文に限った話ではなくwhile文でも同じ

確認問題2

確認問題1の内容を2次元配列を使った形で作成してください

サンプルプログラム2

1から100までの総和を表示するプログラム

```c
#include <stdio.h>
void sum();
int main(void){
    sum();
    return 0;
}
void sum(){
    printf("1から100までの総和は%dです\n", (1 + 100) * 100 / 2);
}
```

実行結果

```
1から100までの総和は5050です
```

関数

●関数とは

C言語は関数と呼ばれる小規模の処理が集まって構成されている
mainやprintfやscanfも関数である
main関数はプログラムが実行されたときに一番最初に実行される関数

・関数の宣言　宣言はmain関数外で行う

戻り値の型　関数名（引数1,引数2･･･）{処理}

・戻り値と引数は必要なければvoidを書いておく（引数は何も書かなくてもよい）

例　戻り値・引数無しでfunction関数を宣言

```
void function(void){
    printf("関数1内でprintfを実行\n");
}
```

関数

●関数の実行

main関数や他の関数などから別の関数を実行する

戻り値 ＝ 関数名（引数1,引数2…）;

・戻り値や引数がない関数はその部分を省略

例　main関数からfunction関数を実行

```c
int main(void){

    function();

    return 0;

}
```

```c
void function(void){

    printf("関数1内でprintfを実行\n");

}
```

関数

●関数のメリット・デメリット

メリット

・機能・処理ごとに分けることができるのでソースコードが簡潔になる

・新しい機能を追加するときなどにバグが発見しやすくなる

・コードを読まなくても機能を把握するだけで他人の関数を使用できる

デメリット

・関数を呼び出すのに時間がかかる

・他の関数のローカル変数を参照できない

関数

●プロトタイプ宣言

main関数の後に関数を宣言するとどうなるか

・さっきのfunctionの例でやってみよう

→ビルドが通らない！

・プログラムは上から順に処理されるのでmain関数はfunction関数の存在を知らない

```
1  #include <stdio.h>
2
3  int main(void){
4
5      function();
6
7      return 0;
8
9  }
```

解決法はmain関数より先に関数を書くこと
もしくはプロトタイプ宣言をすること

関数

●プロトタイプ宣言とは

関数の中身を書かずにどんな関数がこのプログラムに存在するかを
main関数の前に書いておくこと

例　戻り値・引数の無いfunction関数をプロトタイプ宣言

```c
#include <stdio.h>
void function(void);
int main(void){
    function();
    return 0;
}
void function(void){
    printf("関数1内でprintfを実行\n");
}
```

宣言が一致

関数

●stdio.hの正体

printfやscanfも関数ならばどこで宣言しているか

・stdio.hの中にプロトタイプ宣言が書かれている

・.hの拡張子で保存されているファイルをヘッダファイルという

・ヘッダファイルの中には様々な関数がプロトタイプ宣言してある

・ソースファイル内でヘッダファイルを#includeすることでヘッダファイル内の宣言などを読み込むことができる

・大規模なプログラムでは、関数のプロトタイプ宣言などをヘッダファイルに記述しておくことが多い

確認問題3

サンプルプログラムに、次の処理が書かれた
関数triangleを新たに作成し実行してください
ただし、プロトタイプ宣言を使用し、main関数の中身は書き換えないこと

```
printf("三角形の面積は[底辺*高さ/2]です\n");
```

実行結果

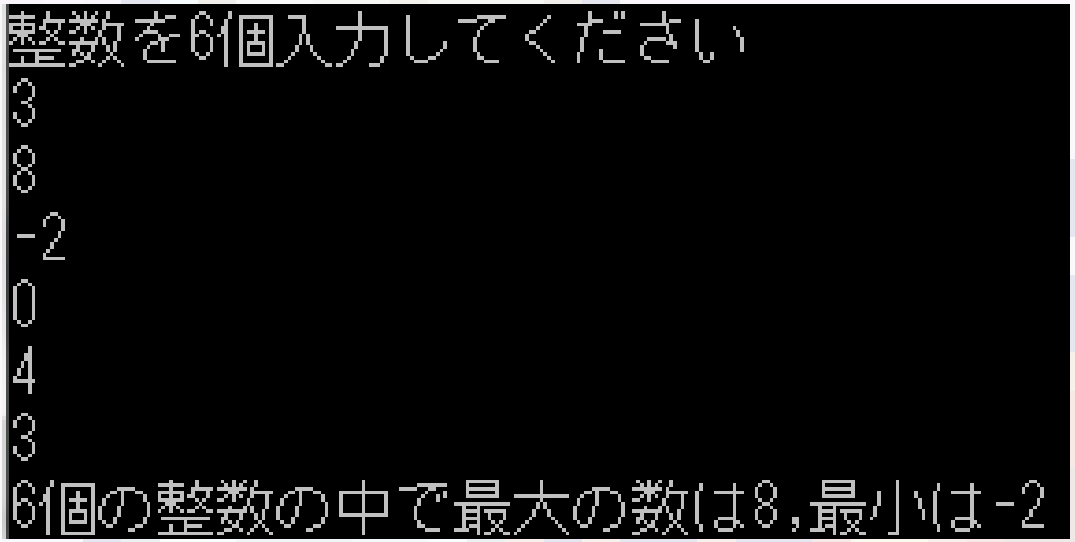
```
1から100までの総和は5050です
三角形の面積は[底辺*高さ/2]です
```

そういえばライブラリって聞いたことあるけど一体何？調べてみよう

練習問題1

任意の整数を6つ入力し、一番大きかった数と一番小さかった数を表示するプログラムを作成せよ。ただし配列を用いること。

例　3, 8, −2, 0, 4, 3を入力した場合



練習問題2

2次元配列を用いて次の行列Aと行列Bの和を求め表示せよ。

$$A = \begin{vmatrix} 3 & -1 & 2 \\ 1 & 0 & 2 \\ 1 & -2 & 3 \end{vmatrix} \qquad B = \begin{vmatrix} 1 & -2 & 0 \\ -3 & 2 & -1 \\ 2 & 1 & 0 \end{vmatrix}$$

実行例

```
 4 -3  2
-2  2  1
 3 -1  3
```

前回の練習問題1の解答例

月の日数を表示するプログラム

```c
#include <stdio.h>

int main(void)
{
        int month;

        printf("月を入力してください:");
        scanf("%d", &month);

        switch(month)
        {
                case 1:
                case 3:
                case 5:
                case 7:
                case 8:
                case 10:
                case 12: printf("%d月の日数は31です\n", month); break;
                case 4:
                case 6:
                case 9:
                case 11: printf("%d月の日数は30です\n", month); break;
                case 2:  printf("2月の日数は28です\n");
        }

        return 0;
}
```

前回の練習問題2の解答例

3の倍数はFizz、5の倍数はBuzz、15の倍数はFizzBuzzを表示するプログラム

```c
#include <stdio.h>

int main(void)
{
        int i;

        for(i=1;i<=100;i++)
        {
                if(!(i%15))     printf("FizzBuzz ");

                else if(!(i%3)) printf("Fizz ");

                else if(!(i%5)) printf("Buzz ");

                else    printf("%d ", i);

                if(!(i%10))     printf("\n");
        }

        return 0;
}
```

前回の練習問題3の解答例

n桁整数の各桁の数字の和を求めるプログラム

```c
#include <stdio.h>
#include <math.h>
#define A 10

int main(void)
{
        int n, ketasu, i=0, sum=0;

        printf("桁数を入力してください");
        scanf("%d", &ketasu);

        printf("整数を入力してください:");
        scanf("%d", &n);

        while(ketasu >= 0)
        {
                i = pow(A,ketasu);

                while(n>=i)
                {
                        sum++;
                        n -= i;
                }

                ketasu--;
        }

        printf("各桁の数字の和は%d\n", sum);

        return 0;
}
```