

情報研究会CACTUS

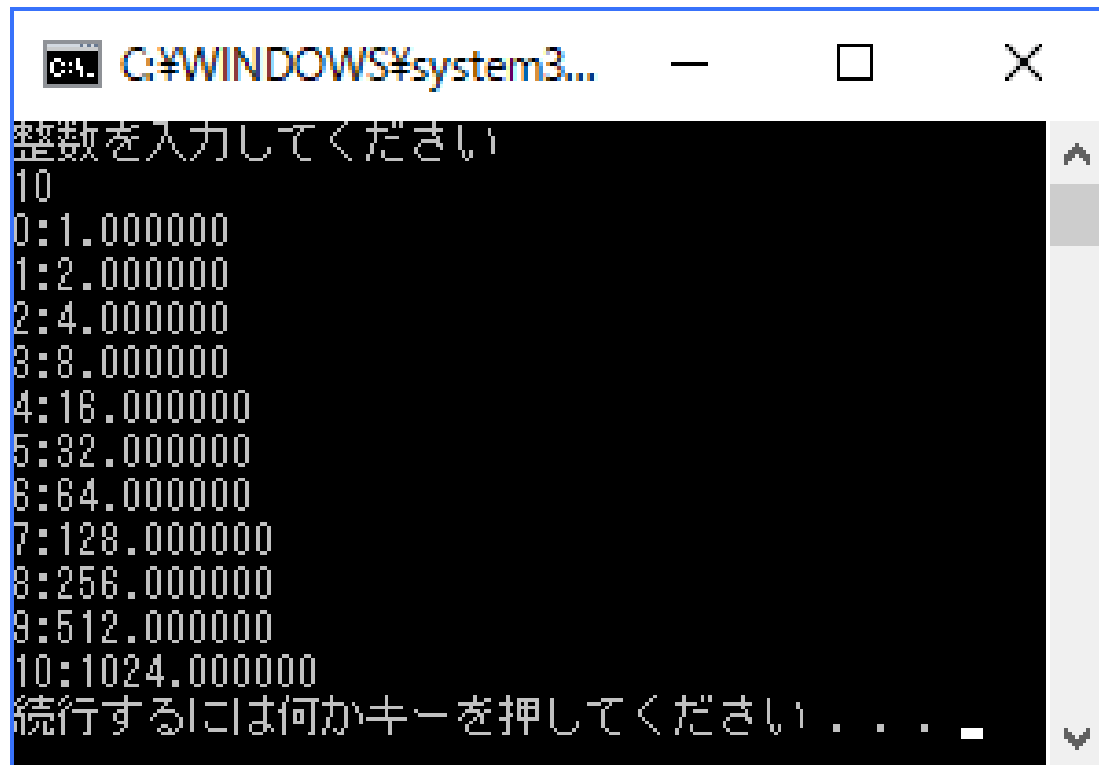
第3回 講義資料

今週の内容

- ・ フィールド幅、精度
- ・ 配列
- ・ 関数

先週のおさらい問題

2の0乗から2のn乗(入力された整数)までを表示させるプログラム。ただし $n > 0$ でOK



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt displays the prompt '整数を入力してください' (Please enter an integer). The user has entered '10'. The program then outputs the powers of 2 from 2^0 to 2^10, each on a new line, formatted as 'index: value'. The values are: 1.000000, 2.000000, 4.000000, 8.000000, 16.000000, 32.000000, 64.000000, 128.000000, 256.000000, 512.000000, and 1024.000000. At the bottom, the program prompts the user to press a key to continue: '続行するには何かキーを押してください . . . '.

```
C:\WINDOWS\system32\cmd.exe
整数を入力してください
10
0:1.000000
1:2.000000
2:4.000000
3:8.000000
4:16.000000
5:32.000000
6:64.000000
7:128.000000
8:256.000000
9:512.000000
10:1024.000000
続行するには何かキーを押してください . . .
```

回答例

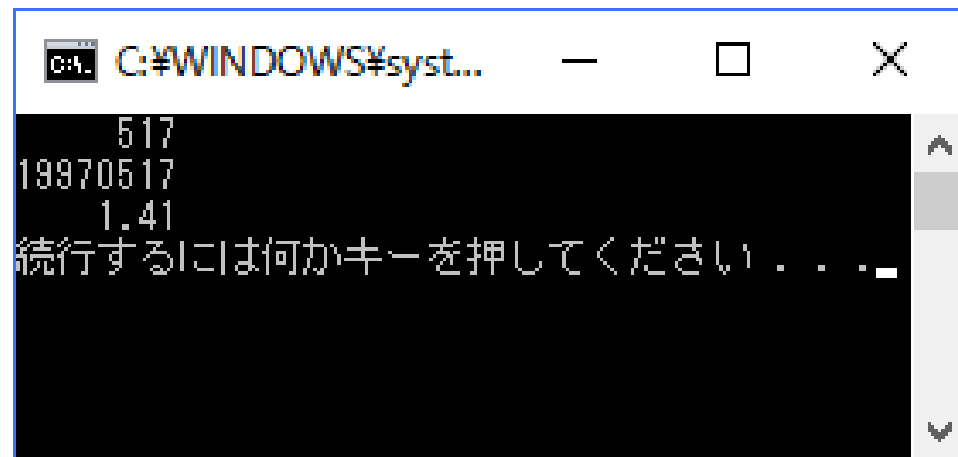
```
1  #include<stdio.h>
2  #include<math.h>
3
4  int main(void) {
5      int n;//入力用の変数
6      int i;//カウンタ変数
7
8      printf("整数を入力してください\n");
9      scanf_s("%d", &n);
10
11     for (i = 0; i <= n; i++) {
12         printf("%d:%lf\n", i, pow(2, i));
13     }
14     return 0;
15 }
```

質問があればどうぞ

フィールド幅・制度

- ・フィールド幅: 何桁表示するかを決める。足りない分は空白スペースで埋める。
- ・精度: 小数点以下の桁数を小数第何位まで表示するか決める。

```
1  #include<stdio.h>
2  #include<math.h>
3
4  int main(void) {
5      int n = 517;
6      int m = 19970517;
7      double l = 1.4142;
8
9      printf("%8d\n", n);
10     printf("%8d\n", m);
11     printf("%8.2f\n", l);
12     return 0;
13 }
```



```
C:\WINDOWS\system32\cmd.exe
517
19970517
1.41
続行するには何かキーを押してください . . .
```

サンプルプログラム1

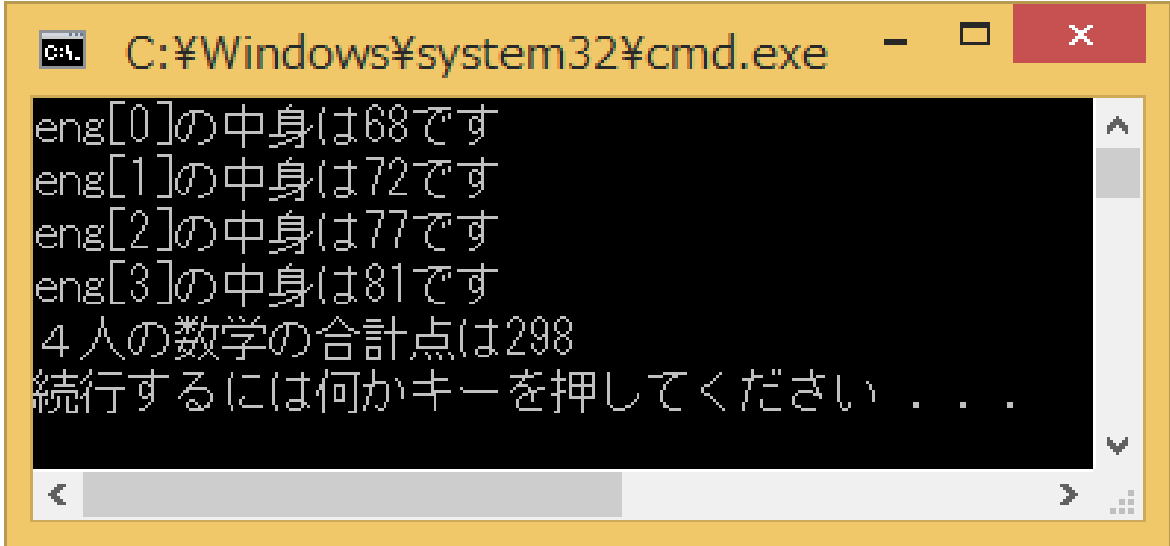
4人の数学のテストの合計点を求めるプログラム

```
#include<stdio.h>

int main(void){
    int eng[4] = { 68, 72, 77, 81 };
    int i;
    int sum=0;

    for (i = 0; i < 4; i++){
        printf("eng[%d]の中身は%dです\n", i, eng[i]);
        sum += eng[i];
    }
    printf("4人の数学の合計点は%d", sum);
    return 0;
}
```

実行結果



```
C:\Windows\system32\cmd.exe

eng[0]の中身は68です
eng[1]の中身は72です
eng[2]の中身は77です
eng[3]の中身は81です
4人の数学の合計点は298
続行するには何かキーを押してください . . .
```

配列

- ・配列とは

同型・同名の変数を複数同時に宣言すること

型 名前[要素数]

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

eng[0]

eng[1]

eng[2]

eng[3]

- ・添え字に変数を使うこともできる

配列

・配列の初期化

- ・ {} でくくってコンマで区切ることで、値を代入できる。

1 `int a[3]={5,12,37};`

`a[0]`に 5、`a[1]`に12、`a[2]`に37を代入

2 `int a[3]={0};`

すべての要素に 0 を代入

3 `int a[3]={5};`

`a[0]`に 5、その他の初期値がない要素すべてに 0 を代入

4 `int a[]={5,12,37};`

要素数を省略しても初期化できる。（結果は 1 と一緒）

配列

・2次元配列

配列の要素数をもう1つ指定する

型 名前[要素数1][要素数2]

例) int b[2][3]

b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]

・多次元配列

要素数の数を増やすことで配列の次元を増やすことができる。

□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) int array[3][3][3][3] 4次元配列

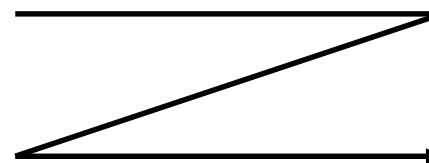
配列

- 2次元配列の初期化

{ } の中でさらに { } でくくって次元を分ける。

例) `array[2][2]={{5,6},{1,2}};`

<code>array[0][0]=5</code>	<code>array[0][1]=6</code>
<code>array[1][0]=1</code>	<code>array[1][1]=2</code>



左上から右下に向けて代入されていく。

配列

・2重ループ

for文のなかにfor文が入っている構造。

1つ目のforループが1回まわるうちに、2つ目のforループはm回まわる。

```
for(i=0;i<n;i++){  
    for(j=0;j<m;j++){  
        処理  
    }  
}
```

質問があればどうぞ

配列

確認問題

1から20までの数字を5つごとに改行して表示するプログラム。(キレイにそろえて表示しよう！)

```
1  2  3  4  5
```

```
6  7  8  9 10
```

```
11 12 13 14 15
```

```
16 17 18 19 20
```

```
続行するには何かキーを押してください . . .
```

余
白

サンプルプログラム2

入力された2つの値の合計値を求めて表示する関数を作る。

```
#include<stdio.h>

void sum(int, int);

int main(){
    int n1,n2;
    printf("n1の値を入力してください:");
    scanf_s("%d", &n1);
    printf("n2の値を入力してください:");
    scanf_s("%d", &n2);

    sum(n1, n2);

    return 0;
}

void sum(int a, int b){
    printf("%d\n", a + b);
}
```

```
n1の値を入力してください:17
n2の値を入力してください:94
111
続行するには何かキーを押してください . . .
```

関数

- ・関数とは

- ・C言語は関数と呼ばれる小規模な処理が集まって構成されている。printfやscanfなども関数である。main関数はプログラムが実行されたときに一番初めに実行される関数。

- ・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)

関数

- ・関数の実行

- ・main関数やほかの関数などから関数を実行する。

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して書く

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

どんな関数が存在するかをあらかじめmain関数の前に宣言しておくこと。

```
#include<stdio.h>

void sum(int, int);

int main(){
    int n1,n2;
    printf("n1の値を入力してください:");
    scanf_s("%d", &n1);
    printf("n2の値を入力してください:");
    scanf_s("%d", &n2);

    sum(n1, n2);

    return 0;
}

void sum(int a, int b){
    printf("%d\n", a + b);
}
```

宣言が一致

関数

- `stdio.h`

`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

質問があればどうぞ

練習問題1

5人分の得点を入力して、階数別に分けてそれぞれの階数が何人いるのか表示するプログラム。

0～100点までの5人分の点数を入力してください

23

38

67

69

92

0点以上10点未満は0人

10点以上20点未満は0人

20点以上30点未満は1人

30点以上40点未満は1人

40点以上50点未満は0人

50点以上60点未満は0人

60点以上70点未満は2人

70点以上80点未満は0人

80点以上90点未満は0人

90点以上100点未満は1人

100点満点は0人

続行するには何かキーを押してください . . . ■

練習問題2

次の行列の和を求めるのです

$$A = \begin{pmatrix} 1 & -2 & 0 \\ 4 & 6 & 5 \\ 3 & 0 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 7 & 3 & -4 \\ 9 & 1 & 0 \\ 6 & 8 & 3 \end{pmatrix}$$