注意点

printf()やscanf()を使う際は、#include<stdio.h>が必要である。
最初のプログラム以降、特別な場合を除き書かないので適宜読み替えてください。

C言語の基本

画面への出力

printf()関数を使う。

例

```
#include<stdio.h>

int main(void){
    printf("%c\n", 'A'); //一文字の文字
    printf("%d\n", 123); //整数
    printf("%f\n", 1.5); ///小数

    return 0;
}
```

フォーマット指定子	型	備考
%c	char	一文字の文字
%s	char	文字列
%d	int	整数
%f	float/double	小数

エスケープシーケンス	動作
\n	改行

文字コード

文字コード上では、Aの次はBであるが、それを次のプログラムで表すことができる。

例

```
int main(void){
    printf("%c", 'A'); //-> A
    printf("%c", 'A'+1); //-> B
}
```

基準の文字列をAとしたとき、+2に置き換えたら、Cが出力される。

Fの一文字前はEであるが、同様に表すことができる。

例

```c
int main(void){
    printf("%c", 'F'); //-> F
    printf("%c", 'F'-1); //-> E

    return 0;
}
```

変数

メモリの一部に名前をつけ、データを記憶できるようにしたもの（変数）。
変数に記録できるデータの種類を型という。

型	(type)	フォーマット指定子
文字列型	char	%c / %s
整数型	int	%d
浮動小数点型	float/double	%f

例：画面への出力を書き換え
文字列の場合

```c
int main(void){
    char mojiA = 'A'; //代入
    printf("%c", mojiA); //-> A

    char mojiB = 'A'+1;
    printf("%c", mojiB); //-> B

    return 0;
}
```

整数の場合

```c
int main(void){
    int one = 1; //代入
    int five = 5;

    int ans = one + five;

    printf("%d", ans); //-> 6

    return 0;
}
```

小数の場合

```
int main(void){
    float g = 1.4; // gramme
    float kg = 2.5; // kilogramme

    float kg_tmp = kg*1000; // x1000をして単位を揃える

    float sum = g + kg_tmp;

    printf("sum amount = %fg", sum); //-> 2501.399902g
    printf("sum amount = %.1fg", sum); //-> 2501.4g

    return 0;
}
```

floatとdoubleでは精度が違うことを次のプログラムで示す。

```
int main(void){
    float tmp_fl = 1.4;
    double tmp_db = 1.4;

    printf("float = %.21f\n", tmp_fl); //->1.399999976158142089844
    printf("double = %.21f\n", tmp_db); //->1.399999999999999911182
}
```

キーボードからの入力

scanf()関数を使う。

配列の章で、書かれているが
文字列を受け取る場合は、&をつけないので注意

-> 文字列以外（int型やfloat型）は&をつける

例

```
int main(void){
    int input;

    printf("input integer number: ");
    scanf("%d", &input); // 100を入力
    printf("%d\n", input); // 100が出力

    float input_fl;
    printf("input decimal(float) number: ");
    scanf("%f", &input_fl); // 1.4を入力
    printf("%.21f\n", input_fl); //->1.399999976158142089844

    double input_db;
    printf("input decimal(double) number: ");
    scanf("%lf", &input_db); // 1.4を入力
    printf("%.21f\n", input_db); //->1.399999999999999911182
}
```

資料より、

> キーボードから入力した値を変数へ 代入して計算に使う
> 例

```
int main(void){
    float g; // gramme
    float kg; // kilogramme

    printf("input gramme[g]");
    scanf("%f", &g); //1.4を入力

    printf("input kilogramme[kg]");
    scanf("%f", &kg); //2.5を入力

    float kg_tmp = kg*1000; // x1000をして単位を揃える

    float sum = g + kg_tmp;

    printf("sum amount = %fg\n", sum); //-> 2501.399902g
    printf("sum amount = %.1fg\n", sum); //-> 2501.4g

    return 0;
}
```

文字列以外の場合、
scanf("フォーマット指定子", &変数);

型	フォーマット指定子
int	%d
float	%f
double	%lf

演算子の種類

算術演算子	用途
+	足し算
-	引き算
*	掛け算
/	割り算
%	割った余り

ビット演算子	用途
&	AND
|	OR
^	XOR
~	NOT

インクリメント・デクリメント

用語	動作
インクリメント	値に1を増やす
デクリメント	値から1を減らす

例

```
int main(void){
    //13分0秒
    int minute = 13;
    int min1 = 0;

    int second = 0;
    int sec1 = 0;

    printf("%d:%d\n", minute, second); //-> 13:0

    sec1 = second++; //後置きインクリメント
    printf("second = %d\n", second); //-> 1
    printf("sec1 = %d\n", sec1); //-> 0

    printf("minute = %d\n", minute); //-> 13
    min1 = ++minute; //前置きインクリメント
    printf("%d\n", minute); //-> 14
    printf("%d\n", min1); //-> 14

    return 0;
}
```

デクリメントも同様の仕組みである。

場合に応じた処理（条件分岐）

if-else文

条件分岐（場合分け）したいとき、if-else文を使う。

例

```
int main(void){
    int integer;
    printf("input integer number: ");
    scanf("%d", &integer);

    if(integer < 200){//入力された数が200未満のとき
        printf("number was inputted is less than 200\n");
    }else if(integer > 200){//入力されたが200以上のとき
        printf("number was inputted is greater than 200");
    }else{//入力された数が上の2つ以外だったとき
        printf("number was inputted is 200");
    }
}
```

if-else文：関係演算子と論理演算子

例

```
int main(void){
    int temp;//temprature
    printf("input temprature: ");
    scanf("%d", &temp);

    if(temp > 0){
        if(0 <= temp && temp < 25){
            printf("normal temprature\n");//普通の気温
        }else if(25 <= temp && temp <= 30){
            printf("Natubi\n");//夏日
        }else if(30 <= temp && temp <= 35){
            printf("Manatubi\n");//真夏日
        }else if(35 <= temp){
            printf("Moushobi\n");//猛暑日
        }
    }else{
        printf("Fuyubi\n");//冬日
    }
}
```

0度以上25度未満を表すには0<= temp && temp < 25のように
&&を用いて連結させる必要がある。

演算子	用途
&&	両方の値が満たすのとき真
	どちらか片方の値が満たすとき真

switch文

例

```
int main(void){
    int integer;
    printf("input number which you like : ");
    scanf("%d", &integer);

    switch (integer % 2){
        case 0:
            printf("Even\n");
            break;
        case 1:
            printf("Odd\n");
            break;
        default:
            printf("Neither Even nor Odd");
            break;
    }
}
```

switch文を書くときは、break;を書き忘れないように注意。
書き忘れてしまうと、case 0の場合だけ実行したかったのに、case 1の処理も実行されてしまうことが起きる。

繰り返し処理

for文

1から10までの値を出力し、総和を求めたいとする。
一回一回、printf()で出力し、足していくのは現実的でない。

何か処理を繰り返したい場合に使用する。

```
int main(void){
    int i;
    int sum = 0;//総和を代入する変数・初期化を忘れない
    for(i = 1; i <= 10; i++){
        printf("%d\n", i);
        sum += i;
    }

    printf("sum = %d\n", sum);
}
```

ここでsumという変数の初期化をすることに注意しよう。
なぜなら、なにかしらの型で宣言した場合、変数sumには適当な数字が入ってしまっているからだ。

while文

繰り返す回数が決まっていない場合に使う。
例えば、1000という数は、1から順番に何回足せば超えるのか知りたいとする。

例

```
int main(void){
    int i = 1;
    int sum = 0; //総和を代入する変数
    int cnt = 1; //回数を数える変数
    while(1){
        printf("%d: %d\n", i, sum);
        sum += i;

        i++;
        cnt++;
        if(sum > 1000){
            printf("----\n");
            printf("%d: %d\n", i, sum);
            break;
        }
    }
    printf("%dtimes", cnt);
}
```

while-do文

とりあえず一回実行してから繰り返すかどうか判断する。

例

```
int main(void){
    int integer;
    do{
        printf("input positive integer: ");
        scanf("%d", &integer);
    }while(integer < 0);//入力された数が0未満なら
    printf("do....\n");
}
```

for文のネスト

長方形があったとする。その面積は、縦a[m] x 横b[m]で与えられるときのあり得る全ての面積を出力するプログラムを作る。
aの範囲: 0 < a < 5
bの範囲: 0 < b < 5

例

```
int main(void){
    int a, b;
    for(a = 0; a < 6; a++){
        for(b = 0; b < 6; b++){
            printf("a x b = %d [a: %d, b: %d]\n", a*b, a, b);
        }
    }
}
```

配列

int number[3];と宣言したとき、アクセスする番号は0番目からであることに注意する。
つまり、number[0]、number[1]、number[2]である。

例

```
int main(void){
    int i;
    int number[5] = {0, 13, 4, 23, 95};
    /* number[0] = 0
       number[1] = 13
       number[2] = 4
       number[3] = 23
       number[4] = 95 */
    for(i = 0; i < 5; i++){// i = 0, 1, 2,.., 4
        printf("number[%d]: %d\n", i, number[i]);
    }
}
```

sort（ソート）

配列arrayに0から100までの数が10要素あったとする。
昇順ソートした結果を表示するプログラムを作る。

```
int main(void){
    int i, j, tmp;
    int array[10] = {0, 13, 43, 65, 31, 34, 44, 23, 14, 96};

    for(i = 0; i < 10; i++){//i = 0, 1,.., 9
        for(j = i+1; j < 10; j++){//j = 1, 2,.., 9
            if(array[i] > array[j]){
                //ex. i=3, j=4 -> 65 > 31 より入れ替え
                tmp = array[i];
                array[i] = array[j];
                array[j] = tmp;
            }
        }
    }
    printf("after sort\n");
    for(i = 0; i < 10; i++){
        printf("array[%d] = %d", i, array[i]);
    }
}
```

文字列

文字列は、char型の配列として扱われる。

文字列について、キーボードからの入力をおこなう。

NULL文字が入るため、入力文字+1の領域が必要である。
また、%sを使う際は、%9sのように入力できる文字数を制限する必要がある。（オーバーフローが起こるから）

```
int main(void){
    char urname[10];//9文字文の配列

    printf("input your name: ");
    scanf("%9s", urname);//9文字までの入力を受けつける
    printf("Hello, %s\n", urname);
}
```

文字の二次元配列

文字自体、char型の配列として扱っているので、複数の文字が要素の配列は二次元配列を用いる。

```c
int main(void){
    int i;

    // [要素数][最大の文字数]
    char str[3][7] = {"apple", "banana", "orange"};//NULL文字が入るので要素+1の配列の個数

    for(i = 0; i < 3; i++){
        printf("str[%d]: %s\n", i, str[i]);
    }
}
```