STAGE 1-6

入力を扱う

インタラクティブなプログラミングへ

🚩 この STAGE の目標

- 入出力の概念を知る
- ユーザーの入力に応じた出力を行えるようになる

にゅう-りょく

入力

input (in + put)

プログラムに値や文字列など
データを与えること、または与えるデータそのもの

しゅつ-りょく

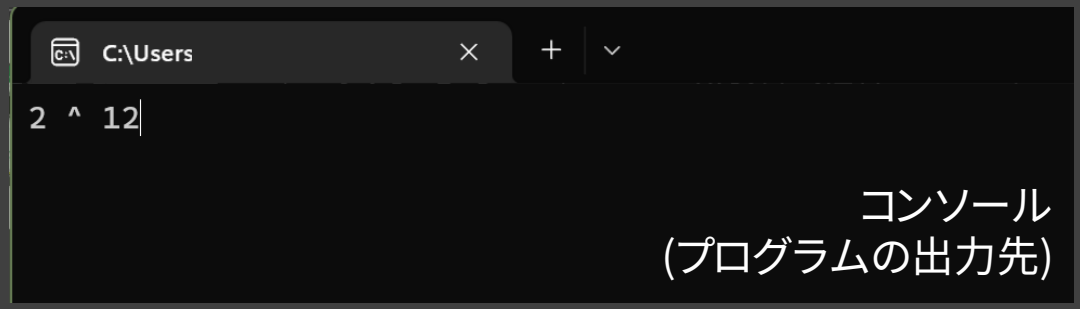
出力

output (out + put)

プログラムから与えられるデータのこと
関数の戻り値や、プログラムを実行したときの結果、表示される文字列などをさす

```
printf(“Hello World!¥n”);
```

printf 関数は
引数1 をコンソールへ出力する



しゅつ-りょく

出力

output (out + put)

プログラムから与えられるデータのこと
関数の戻り値や、プログラムを実行したときの結果、表示される文字列などをさす

にゅう-りょく

入力

input (in + put)

プログラムに値や文字列など
データを与えること、または与えるデータそのもの

しゅつ-りょく

出力

output (out + put)

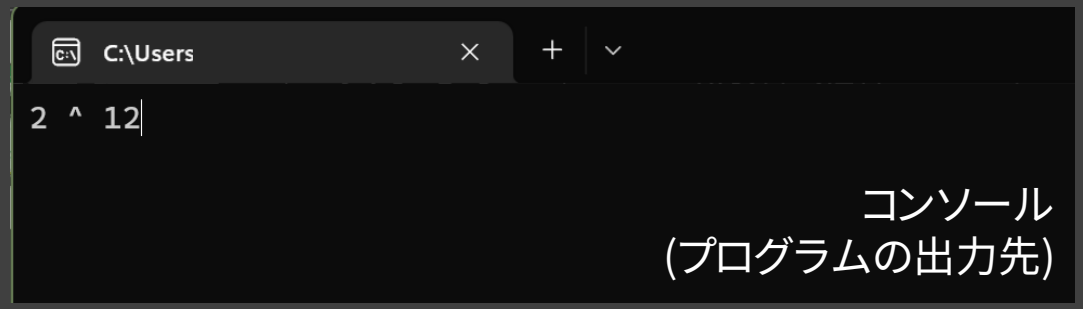
プログラムから与えられるデータのこと
関数の戻り値や、プログラムを実行したときの結果、表示される文字列などをさす

にゅう-りょく

入力

input (in + put)

プログラムに値や文字列など
データを与えること、または与えるデータそのもの



コンソールに文字を打つと、その文字がプログラムに「入力」される
コンソールに入力された文字を受け取る関数がある

にゅう-りょく

入力

input (in + put)

プログラムに値や文字列など
データを与えること、または与えるデータそのもの

コンソールに打った文字を読み取るには scanf_s 関数を使うとよい

コード

```
int a = 0;
printf( “aの値は？” );
scanf_s( “%d”, &a ); // 整数をひとつ読み取る
printf( “a=%d”, a );
```

★ scanf_s 関数の使い方

戻り値 `int` 読み取った値の数 (EOF 文字なら読み取れなかったことを表す)

引数1 `string` 読み取りたいデータの形式を、% のあとに アルファベット の形式で指定する
%d (整数) や %lf (小数)
複数のデータを入力するには、半角スペースで区切る
(この場合、キーボードで入力してもらうときも半角スペースで区切らなければならない)

引数n 読み取ったデータを代入する変数のポインタを、順番に指定する
(変数名の前に & をつけると覚えればよい)

コード

```
int a = 0;
printf( “aの値は？” );
scanf_s( “%d”, &a ); // 整数をひとつ読み取る
printf( “a=%d”, a );
```

コンソール

```
aの値は？
> 1        ← 黄色い字は
a=1        ユーザーが打つ内容
           入力ともいう
```

練習

与えられた整数を2乗して返す
プログラムを作ろう (main 関数に直接書いてOK)

標準入力 | 整数1つ

出力 | 上の整数を2乗した数 (わかりやすいよう文章を加えてよい)

例

コンソール

```
2乗したい値は？
> 22
22の2乗は 484 です
```

入力 22 の 2 乗 (22 × 22) は 484 なので、printf 関数を使って 484 を出力します
わかりやすいように説明を加えてもかまいません

ただし、あまりにも大きい数を2乗しようとすると「桁あふれ」が起こって正しく計算できなくなるので、
その場合は計算中に扱う型を変えてもOKです。scanf_s 関数の入力の型を変えるのをお忘れなく

練習

与えられた整数を2乗して返す
プログラムを作ろう (main 関数に直接書いてOK)

標準入力 | 整数1つ

出力 | 上の整数を2乗した数 (わかりやすいよう文章を加えてよい)

コード

```
int main(void) { // main 関数に直接書く
  int original;
  printf( “2乗したい値は？” );
  scanf_s( “%d”, &original );
  int result;
  result = original * original;
  printf( “%dの2乗は %d です” , original, result);
}
```

コンソール

```
2乗したい値は？
> 22
22の2乗は 484 です
```

これで
コンピュータと対話できるようになった

💭 このSTAGEのまとめ

コンソールに対して

出力→
```
printf(“文字列”[, …]);
```

```
printf(“int型の値を表示する: %d”, 3)
int型の値を表示する: 3
```

入力→
```
scanf_s(“型指定”, &入力値を代入する変数, …);
```

🚩 この STAGE の目標

- ✅ 入出力の概念を知る
- ✅ ユーザーの入力に応じた出力を行えるようになる

できるようになったこと

ユーザーの入力する値によって
結果が変わるプログラムが書ける

練習

与えられた整数を2乗して返す
プログラムを作ろう (main 関数に直接書いてOK)

標準入力	整数1つ
出力	上の整数を2乗した数 (わかりやすいよう文章を加えてよい)

コード

```c
int main(void) { // main 関数に直接書く
  double original;
  printf( “2乗したい値は？” );
  scanf_s( “%lf”, &original );
  double result;
  result = original * original;
  printf( “%lfの2乗は %lf です” , original, result);
}
```

コンソール

```
2乗したい値は？
> 22
22の2乗は 484 です
```

桁あふれ対策には
long long int か double を使うと便利
(注意: printf 関数が非常に小さい小数や大きな数を
表示するには桁数の指定が必要)

★ scanf_s 関数の使い方

戻り値 `int` 読み取った値の数 (EOF 文字なら読み取れなかったことを表す)

引数1 `string` 読み取りたいデータの形式を、% のあとに アルファベット の形式で指定する
%d (整数) や %lf (小数) 、%c (1文字) %s (文字列) …
複数のデータを入力するには、半角スペースで区切る
(この場合、キーボードで入力してもらうときも半角スペースで区切らなければならない)

引数n 読み取ったデータを代入する変数のポインタを、順番に指定する
(変数名の前に & をつけると覚えればよい)
ただし、文字列 (%s, %S) や1文字(%c, %C) の場合、読み取る最大の文字数を、変数のポインタの引数の次に指定する
※ n: 2以上の整数. 定数を書いてもいいです

コード

```c
int a = 0;
printf( “aの値は？” );
scanf_s( “%d”, &a ); // 整数をひとつ読み取る
printf( “a=%d”, a );
```

コンソール

```
aの値は？
> 1          ← 黄色い字は
a=1          ユーザーが打つ内容
             入力ともいう
```

★ scanf_s 関数の使い方

戻り値 `int` 読み取った値の数 (EOF 文字なら読み取れなかったことを表す)

1文字を読み取るには…

コード

```
char ch;
printf( “好きな文字は？” );
scanf_s( “%c”, &ch , 1); // 文字を1文字まで読み取る
printf( “文字は %c”, ch );
```

コンソール

```
好きな文字は？
> B
文字は B
```

文字列を読み取るには…

コード

```
string st;
printf( “好きな言葉は？(50字まで)” );
scanf_s( “%s”, &st , sizeof(st) ); // 最大まで読み取る
printf( “言葉は %s”, st );
```

コンソール

```
好きな言葉は？
> 単位
言葉は 単位
```

※ 50文字目より先は読まれず、無視される