

---

## 03 繰り返し

---

### 目標

- 繰り返し処理のイメージを掴む.
- シンプルな for 文が使えるようになる.
- 変数の演算や代入に関する表現のバリエーションを増やす.

### 例題

#### 【Homework】

出典：AtCoder Beginner Contest 163 B

#### 問題文

高橋君の夏休みは  $N$  日間です.

夏休みの宿題が  $M$  個出されており,  $i$  番目の宿題をやるには  $A_i$  日間かかります.

複数の宿題を同じ日にやることはできず, また, 宿題をやる日には遊ぶことができません.

夏休み中に全ての宿題を終わらせるとき, 最大何日間遊ぶことができますか?

ただし, 夏休み中に全ての宿題を終わらせることができないときは, かわりに  $-1$  を出力してください.

#### 制約

- $1 \leq N \leq 10^6$
- $1 \leq M \leq 10^4$
- $1 \leq A_i \leq 10^4$

#### 入力

入力は以下の形式で標準入力から与えられる.

```
N M
A_1 \cdots A_N
```

#### 出力

高橋君が遊ぶことのできる日数, または,  $-1$  を出力せよ.

#### 入力例 1

```
41 2
5 6
```

#### 出力例 1

```
30
```

例えば, 最初の 5 日間で 1 番目の宿題をやり, その後 30 日間遊んで, 最後の 6 日間で 2 番目の宿題をやることで, 30 日間遊ぶことができます.

## 入力例 2

```
10 2
5 6
```

## 出力例 2

```
-1
```

宿題を終わらせることができません。

### 考察

- 宿題をいつ、どのような順番で取り組んでも、遊ぶことのできる日数に影響は無い。
- 問題を言い換えると、宿題を終わらせるのに合計何日かかるかを求めればよい。
  - $N$  日より多くかかる場合、答えは  $-1$ 。
  - $N$  日以内に終わるのであれば、 $N$  から引いた分だけ遊べる。
- もし宿題がちょうど  $M = 3$  個出ているなら、

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int N, M, Ai, sum;
    cin >> N >> M;
    sum = 0;

    // 1 kome
    cin >> Ai;
    sum = sum + Ai;

    // 2 kome
    cin >> Ai;
    sum = sum + Ai;

    // 3 kome
    cin >> Ai;
    sum = sum + Ai;

    if(sum > N){
        cout << -1 << endl;
    }
    else{
        cout << N - sum << endl;
    }
}
```

のように求められる。

- 宿題が  $M$  個の場合も、`cin` と `sum` の計算部分を  $M$  回繰り返せばよい。

## 解答例

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int N, M, Ai;
    cin >> N >> M;
    long long sum = 0;
    for(int i=0; i<M; i++){
        cin >> Ai;
        sum += Ai;
    }
    if(sum > N){
        cout << -1 << endl;
    }
    else{
        cout << N - sum << endl;
    }
}
```

## 解説

### ① long long 型

変数には箱の中に入れる値によって種類が決まっており、この変数の種類を **型** と呼ぶ。これまでは int 型と呼ばれる、整数を入れるための型を使ってきた。

今回の解答例では **long long 型** と呼ばれる新たな型を用いている。long long 型も整数を入れるための型であるが、入れることのできる値の範囲に差がある。

型	値の範囲
int	-2,147,483,648 ~ 2,147,483,647
long long	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

上の表のように、int 型は  $\pm 2 \times 10^9$  程度までの値しか扱えないのに対し、long long 型は  $\pm 9 \times 10^{18}$  程度の値まで扱うことができる。

ただし、その分 long long 型は多くの記憶容量を必要とするため、必ずしも long long 型を使う方がよいとは限らない。int 型では値が収まらないと想定される場合に限り、long long 型を使うようにしよう。

今回の例題の場合、 $M \leq 10^4$ 、 $A_i \leq 10^4$  より、宿題を終わらせるの必要な日数は最大で  $10^8$  程度になると見積もることができるため、int 型でも収まると計算できる。

### ② 定義と同時に代入

変数は定義と同時に初期値を代入することもできる。

```
long long sum = 0;
```

この場合、long long 型の変数 sum を定義し、初期値として 0 を代入している。

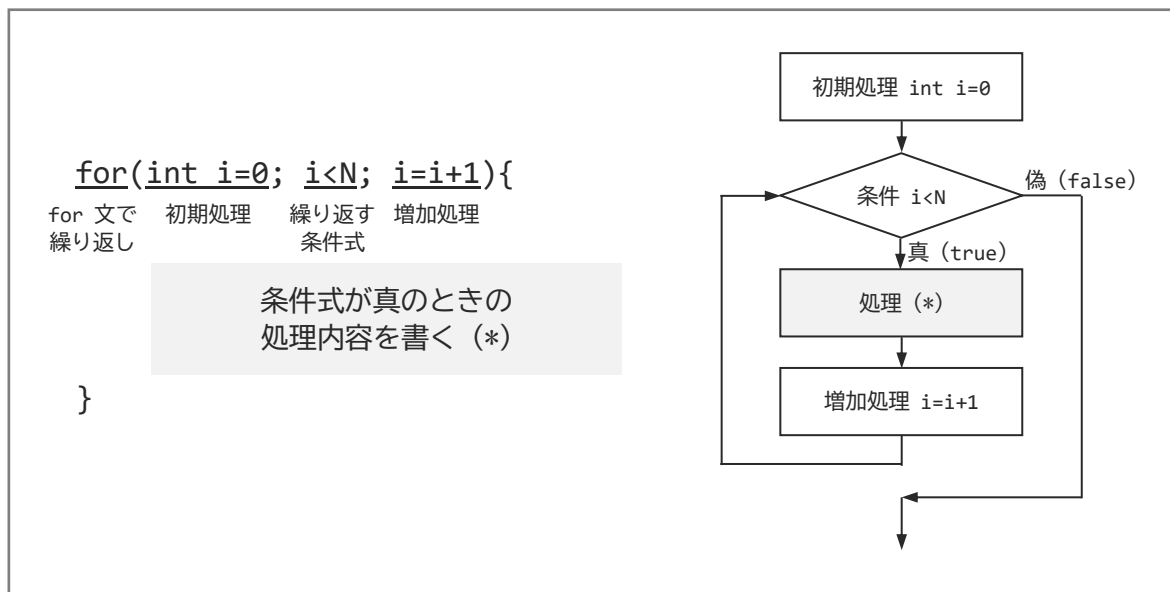
### ③ for 文

繰り返し処理を実装するには **for 文** を用いる。if(初期処理; 条件式; 増加処理){ 処理 } の形で記述することで、条件式が満たされる場合のみ { } 内の処理が繰り返し実行される。

```
for(int i=0; i<N; i=i+1){  
    cout << i << endl;  
}
```

例として、上のような for 文を考えてみよう。

- まず、初期処理として、int i=0 が実行される。つまり、int 型整数 i を定義して、0 を代入する。
- 条件式 i<N が真である間、{ } 内の処理を繰り返す。
- { } 内の処理を 1 回行ったら、増加処理 i=i+1 を実行して、また条件式の判定に戻る。



なお、初期処理や { } 内で定義した変数は { } 内でしか使うことができないので注意する。特に、初期処理で定義された変数はカウンタと呼ばれ、i と名前が付けられることが多い。

### ④ 繰り返し処理の考え方

繰り返し処理を実装するときは、

- どのような処理を繰り返したいか
- 何回（いつまで）繰り返したいか
- どんな変数が必要か

といった点に注意すると良い。

なお、for 文では 3 つの式を操ることで様々な処理を書けるようになるが、今のレベルでは繰り返す条件式の N の部分を書き換えるだけで十分である。このとき、i は 0, 1, ..., N-1 と変化し、単純に「N 回同じ処理を繰り返す」という意味で捉えられる。

## ⑤ インクリメント演算子

ある変数に 1 加算したい時、インクリメント演算子を使うことができる。例えば、変数 `i` に 1 加算したい場合、次のようにする。

```
i++;
```

また、ある変数から 1 減算したい時、デクリメント演算子を使うことができる。例えば、変数 `i` から 1 減算したい場合、次のようにする。

```
i--;
```

これらの表現は `for` 文の増加処理で用いられることが多い。

## ⑥ 複合代入演算子

四則演算と代入処理を同時に行う演算子を複合代入演算子と呼び、次のような種類がある。

	記号	例 1	例 2
加算代入演算子	<code>+=</code>	<code>a = a + 5</code>	<code>a += 5</code>
減算代入演算子	<code>-=</code>	<code>b = b - c</code>	<code>b -= c</code>
乗算代入演算子	<code>*=</code>	<code>d = d * d</code>	<code>d *= d</code>
除算代入演算子	<code>/=</code>	<code>e = e / 2</code>	<code>e /= 2</code>
剰余算代入演算子	<code>%=</code>	<code>f = f % 3</code>	<code>f %= 3</code>

## 表現

(1) 標準入力から自然数  $N$  と  $N$  個の整数  $A_i$  を順に受け取り,  $A_i \geq 0$  を満たすものの個数を出力する

```
int N, Ai, ans = 0;
cin >> N;
for(int i=0; i<N; i++){
    cin >> Ai;
    if(Ai >= 0){
        ans++;
    }
}
cout << ans << endl;
```

(2) 標準入力から自然数  $N$  と  $N$  個の自然数  $A_i$  を順に受け取り,  $A_i$  の最大値を出力する

```
int N, Ai, maximum = 0;
cin >> N;
for(int i=0; i<N; i++){
    cin >> Ai;
    if(Ai > maximum){
        maximum = Ai;
    }
}
cout << maximum << endl;
```

(3) 標準入力から自然数  $N$  を受け取り, 1 以上  $N$  以下の自然数のうち 3 の倍数または 5 の倍数であるものの個数を出力する

```
long long N, ans = 0;
cin >> N;
for(long long i=0; i<N; i++){
    if((i+1)%3 == 0 || (i+1)%5 == 0){
        ans++;
    }
}
cout << ans << endl;
```

## 類題

- ☐ Roller Coaster (AtCoder Beginner Contest 142 B)
- ☐ Achieve the Goal (AtCoder Beginner Contest 151 B)
- ☐ FizzBuzz Sum (AtCoder Beginner Contest 162 B)
- ☐ Great Ocean View (AtCoder Beginner Contest 124 B)
- ☐ 双六 (JOI 2017/2018 予選 B)