

# Introduction to Programming

情報知識ネットワーク研究室 B4 谷 陽太

# 目次

- おまじない
- 変数
- 四則演算
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- 配列

**簡単なプログラムの  
書き方を紹介するよ！**

**使用言語：C++**

# 目次

- **おまじない**
- 変数
- 四則演算
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- 配列

おまじない

```
int main(void){
```

```
    return 0;
```

```
}
```

おまじない

```
int main(void){
```



プログラムのコードはこの間に書くよ！

```
return 0;
```

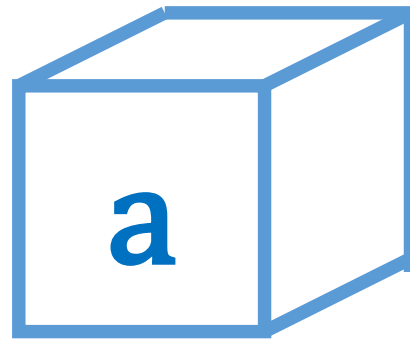
```
}
```

# 目次

- おまじない
- **変数**
- 四則演算
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- 配列

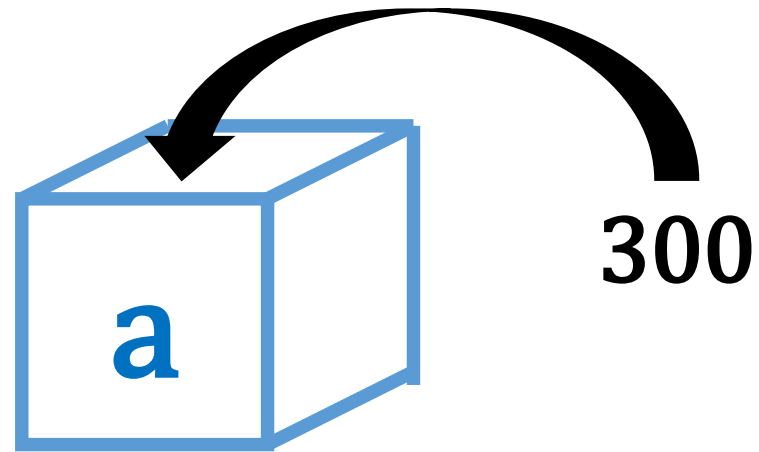
# 変数とは

- 変数とは値を格納する箱である！



# 変数とは

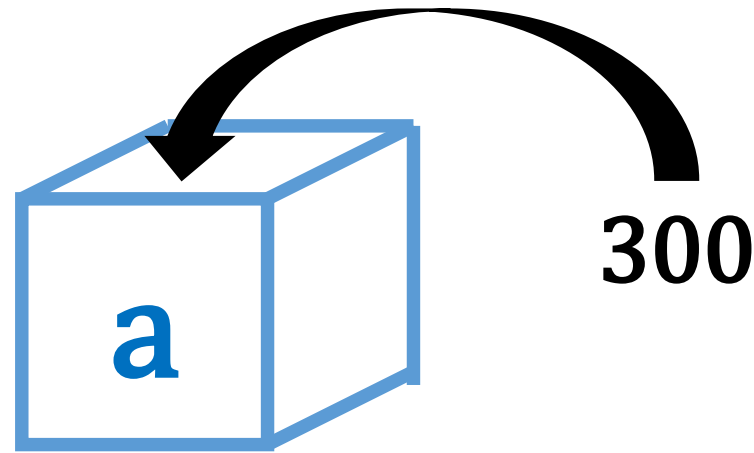
- 変数とは値を格納する箱である！





# 変数とは

- 変数とは値を格納する箱である！



- ただし、どんな値でも格納できるわけではない**  
格納できる値のタイプによって「型」が分かれている

# 変数の型

- 型いろいろ  
整数型、小数(実数)型、文字型、非負の整数型……

# 変数の型

- 型いろいろ  
整数型、小数(実数)型、文字型、非負の整数型……
- もっとも基礎となる型: **int型**

integerの略で、整数しか扱えない  
小数点以下は切り捨てになる

扱える範囲

-2,147,483,648 ~ 2,147,483,647

# 変数の宣言と代入

- int型の「a」って名前の変数を作りたい！（変数宣言）

**int a;**

- 「a」に「3」という値を格納したい！（代入）

**a = 3;**

- 変数宣言と代入は同時に行うこともできる

**int a = 3;**

# 目次

- おまじない
- 変数
- **四則演算**
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- 配列

# 四則演算

- プログラムでは四則演算ができる！

加算: +    減算: -    乗算: \*    除算: /    剰余算: %

# 四則演算

- プログラムでは四則演算ができる！

加算: +    減算: -    乗算: \*    除算: /    剰余算: %

- 例: int型の変数「a」を宣言し、 $4 \times 3$ の結果を代入する

```
int main(void){  
    int a;  
    a = 4 * 3;  
    return 0;  
}
```

# 四則演算

- 四則演算の実行順序は数学と同じ

乗算・除算・剰余算が優先、加算・減算は後回し  
優先順位が同じ場合は、左から順に処理する

- 略記法が存在する！

例: a自身を3倍する（乗算以外も3以外でも同様に表記してOK）

**$a = a * 3;$  →  $a *= 3;$**

例: a自身を1増やす（加減算の場合かつ差分が1の場合のみ）

**$a = a + 1;$  →  $a++;$**



# 目次

- おまじない
- 変数
- 四則演算
- **入出力**
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- 配列

# 入出力

- 変数に入力を受け取ろう！

**cin >> (変数名);**

例：cin >> a;

- 変数の中身や値を出力しよう！

**cout << (変数名);**

**cout << (値);**

例：cout << a;

例：cout << 3 << endl;  
cout << “Hello World”;

改行



# 入出力

- cin, cout したくなったら  
プログラムの最初に、こう書こう！

```
#include <iostream>  
using namespace std;
```

- **int main(void)よりも前！**
- とりあえず「おまじない」だと思っておこう！

# 問題1 Hello world

- 「Hello World」と出力せよ  
(出力の最後に改行するのを忘れないように)

```
#include <iostream>
using namespace std;
```

```
int main(void){
    cout << "Hello World" << endl;
    return 0;
}
```

## 問題2 X Cubic

- 入力 $x$ を受け取り $x^3$ を出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int x;
    cin >> x;
    cout << x * x * x << endl;
    return 0;
}
```

# 問題3 Rectangle

- 長方形の2辺の長さを受け取り  
面積と外周を半角空白で区切って出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int h, w;
    cin >> h >> w;
    cout << h * w << " " << h * 2 + w * 2 << endl;
    return 0;
}
```

# 目次

- おまじない
- 変数
- 四則演算
- 入出力
- **浮動小数点型**
- 条件式, 条件分岐
- ループ
- 配列

# 浮動小数点型

- int型は整数値しか扱えない

例: `int d = 3.14;`

`cout << d << endl;` → 「3」と出力される

**欲求：小数点以下の値を含んだ値を扱いたい**



# 浮動小数点型

- int型は整数値しか扱えない

例: `int d = 3.14;`

`cout << d << endl;` → 「3」と出力される

**欲求：小数点以下の値を含んだ値を扱いたい**

**→ double型を使おう！**

# double型とは

- 倍精度浮動小数点型  
**Double** precision floating point number

# double型とは

- ~~倍精度浮動小数点型~~  
~~Double precision floating point number~~

例:

```
double d = 3.14;  
cout << d << endl;
```

→ 「3.14000」と出力される！

# double型の注意点

- 剰余演算子「%」は使えない
- うまくいかない例1:  
**int a = 3, b = 2;**  
**double c = a / b;**
- うまくいかない例2:  
**double c = 3 / 2;**

# なぜうまくいかないのか

```
int a = 3, b = 2;  
double c = a / b;
```



# なぜうまくいかないのか

```
int a = 3, b = 2;  
double c = a / b;
```



int型同士の計算 → 結果はint型「1」

# なぜうまくいかないのか

```
int a = 3, b = 2;  
double c = a / b;
```



int型同士の計算 → 結果はint型「1」

1.00000



# うまくいく例1

```
double a = 3.66, b = 2.44;
```

```
double c = a / b;
```





# うまくいく例1

```
double a = 3.66, b = 2.44;  
double c = a / b;
```



double型同士の計算 → 結果はdouble型 「1.50000」

1.50000




代入

## うまくいく例2

```
double a = 3.66;
```

```
int b = 2;
```

```
double c = a / b;
```



## うまくいく例2

```
double a = 3.66;  
int b = 2;  
double c = a / b;
```

高精度な方に統一される

double型とint型では  
double型のほうが高精度

double型とint型 → 結果はdouble型 「1.83000」

1.83000

代入

# どうすればうまくいくのか

```
int a = 3, b = 2;  
double c = a / b;
```

- 分かったこと: 少なくとも一方がdouble型なら良い

# どうすればうまくいくのか

```
int a = 3, b = 2;  
double c = a / b;
```

- 分かったこと: 少なくとも一方がdouble型なら良い

コンピュータを騙して  
片方をdouble型だと思い込ませれば良い！

# キャスト(強制型変換)

```
int a = 3, b = 2;  
double c = (double)a / b;
```

# キャスト(強制型変換)

```
int a = 3, b = 2;  
double c = (double)a / b;
```



a 「俺double型っスウッスウッス」

# キャスト(強制型変換)

```
int a = 3, b = 2;  
double c = (double)a / b;
```



a 「俺double型っスウッスウッス」

→ うまくいく！



# 問題4 A/B Problem

- 2つの整数a, bを受け取って、以下の値を計算し半角スペースで区切って出力せよ

$a \div b : d$  (整数)

$a \div b$  の余り :  $r$  (整数)

$a \div b : f$  (浮動小数点数)

- **Constraints**

$$1 \leq a, b \leq 10^9$$

- **Output**

$d, r, f$  を1つの空白で区切って1行に出力

$f$ については、0.00001以下の誤差があってもよい

## 問題4 A/B Problem

```
#include <iostream>
using namespace std;

int main(void){
    int a, b;

    cin >> a >> b;
    int d = a / b;
    int r = a % b;
    double f = (double)a / b;
    cout << d << " " << r << " " << f << endl;
    return 0;
}
```

# 問題4 A/B Problem

- 実行例1

入力: 3 2

出力: 1 1 1.5

- 実行例2

入力: 12300 99

出力: 124 24 124.242

# 問題4 A/B Problem

- 実行例1

入力: 3 2

出力: 1 1 1.5

- 実行例2

入力: 12300 99

出力: 124 24 124.242 ← 実はよくない

**表示桁数が足りていない！**

coutは、気を利かせて適当なところで表示をやめてしまう

# 小数点以下n桁を出力させたい！

## printfを使おう！

- 例:

```
int a = 3, b = 2;
```

```
printf("アイ%dウエ%d才¥n", a, d);
```

# 小数点以下n桁を出力させたい！

## printfを使おう！

- 例:

```
int a = 3, b = 2;  
printf("アイ%dウエ%dオ¥n", a, d);
```

改行  
└┘

→ 出力: アイ3ウエ2オ

%dのところに対応するint型の値が入る

# printfの注意

- printf したくなったら  
プログラムの最初に、こう書こう！

```
#include <cstdio>  
using namespace std;
```

- int型の場合は**%d**だけど  
double型の場合は**%f**になる！

**型によって異なるので注意！**

# printfだと何が嬉しいのか

- 表示桁数を簡単に指定することができる！

```
printf("%.6f", a);
```

→ 小数点以下を6桁表示する！



# 問題4 A/B Problem

```
#include <iostream>
#include <cstdio>
using namespace std;

int main(void){
    int a, b;
    cin >> a >> b;
    int d = a / b;
    int r = a % b;
    double f = (double)a / b;
    printf("%d %d %.6f\n", d, r, f);
    return 0;
}
```

# 問題4 A/B Problem

- 実行例1

入力: 3 2

出力: 1 1 1.500000

- 実行例2

入力: 12300 99

出力: 124 24 124.242424

# 目次

- おまじない
- 変数
- 四則演算
- 入出力
- 浮動小数点型
- **条件式, 条件分岐**
- ループ
- 配列

# 条件分岐

欲求: 状況に応じて処理を変更したい

→ if文を使おう！

# if文とは

- 条件式が成立しているかどうかで処理が変わる！

```
if ( 条件式 ) {  
    条件が成立してた場合の処理  
}  
else {  
    条件が成立していなかった場合の処理  
}
```

# if文とは 例

- 入力xが0より大きければ「Hello World」と出力し  
それ以外だったら「Bye World」と出力

```
cin >> x;  
if ( x > 0 ) {  
    cout << "Hello World" << endl;  
}  
else {  
    cout << "Bye World" << endl;  
}
```

# 3分岐以上したい場合は「else if」

```
if ( 条件式 A ) {  
    条件Aが成立していた場合の処理  
}  
else if ( 条件式B ) {  
    条件Aは成立せず、Bは成立していた場合の処理  
}  
else if ( 条件式C ) {  
    条件AとBは成立せず、Cは成立していた場合の処理  
}  
else {  
    条件AもBもCも成立していなかった場合の処理  
}
```

# 条件式いろいろ

- aはb以上  $a \geq b$
- aはbより大きい  $a > b$
- aはb以下  $a \leq b$
- aはbより小さい  $a < b$
- aとbは等しい  $a == b$
- aとbは異なる  $a != b$

成立する場合 : true (1)

成立しない場合: false (0)

- 条件Aかつ条件B  
**条件式A && 条件B**
- 条件Aか条件B  
(少なくともどちらか一方)  
**条件式A || 条件式B**
- 条件Aじゃない  
**!(条件式A)**



## 問題5 Range

- 3つの整数 $a, b, c$ を受け取って  
 $a < b < c$  なら「Yes」と出力し  
それ以外であれば「No」と出力せよ

## 問題5 Range

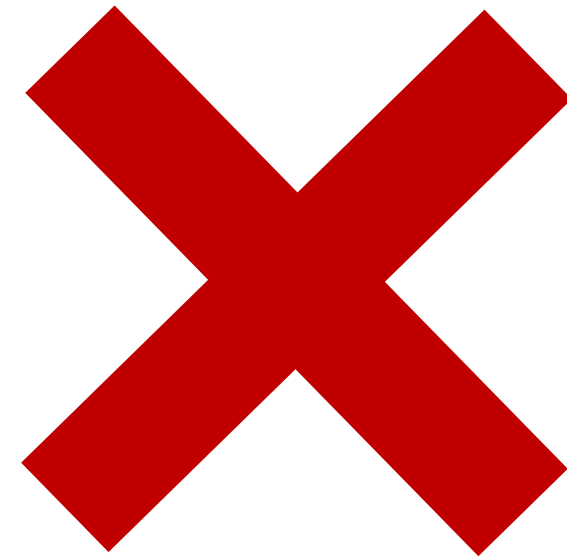
- 3つの整数a, b, cを受け取って  
a < b < c なら「Yes」と出力し  
それ以外であれば「No」と出力せよ

```
cin >> a >> b >> c;  
if ( a < b < c ) {  
    cout << "Yes" << endl;  
}  
else{  
    cout << "No" << endl;  
}
```

## 問題5 Range

- 3つの整数a, b, cを受け取って  
a < b < c なら「Yes」と出力し  
それ以外であれば「No」と出力せよ

```
cin >> a >> b >> c;  
if ( a < b < c ) {  
    cout << "Yes" << endl;  
}  
else{  
    cout << "No" << endl;  
}
```




# なぜ $a < b < c$ はダメなのか

- 条件式も数学の計算順序に則る

$$a < b < c$$

# なぜ $a < b < c$ はダメなのか


- 条件式も数学の計算順序に則る

$$a < b < c$$


ここが先に評価される

# なぜ $a < b < c$ はダメなのか

- 条件式も数学の計算順序に則る

$$a < b < c$$


ここが先に評価される

$a < b$  が成立していたとすると……

$$1 < c$$

やりたいことと違う！

$$a < b \ \&\& \ b < c$$

をしよう！

# 問題5 Range

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int a, b, c;
    cin >> a >> b >> c;
    if (a < b && b < c) {
        cout << "Yes" << endl;
    }
    else {
        cout << "No" << endl;
    }
    return 0;
}
```

# 目次

- おまじない
- 変数
- 四則演算
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- **ループ**
- 配列



# ループ

欲求: 同じ処理を何度もやりたい！

→ ループを使おう！

- ループいろいろ

**while文**

**for文**

do while文 (←今日はやらない)

# while文とは

- 条件式が成立している限りループ！

```
while ( 継続条件式 ){  
    繰り返したい処理  
}
```

# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int i = 0;
    while ( i < 1000 ){
        cout << "Hello World" << endl;
        i++;
    }
    return 0;
}
```

# for文とは

- カウントアップに便利なループ！

```
for (処理A; 継続条件式; 処理B){  
    繰り返したい処理  
}
```

処理A: ループ突入前にやりたい処理

処理B: ループ一周終わるごとにやりたい処理

# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int i = 0;
    while ( i < 1000 ){
        cout << "Hello World" << endl;
        i++;
    }
    return 0;
}
```

```
for (処理A; 継続条件式; 処理B){
    繰り返したい処理
}
```

# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
```

```
    int i = 0;
```

```
    while ( i < 1000 ){
```

```
        cout << "Hello World" << endl;
```

```
        i++;
```

```
    }
```

```
    return 0;
```

```
}
```

```
for (処理A; 継続条件式; 処理B){
    繰り返したい処理
}
```

# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int i = 0;
    while ( i < 1000 ){
        cout << "Hello World" << endl;
        i++;
    }
    return 0;
}
```

```
for (処理A; 継続条件式; 処理B){
    繰り返したい処理
}
```

# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int i = 0;
    while ( i < 1000 ){
        cout << "Hello World" << endl;
        i++;
    }
    return 0;
}
```

```
for (処理A; 継続条件式; 処理B){
    繰り返したい処理
}
```



# 問題6 Print Many Hello World

- 「Hello World」と1000回出力せよ

```
#include <iostream>
using namespace std;
```

```
int main(void){
    for (int i = 0; i < 1000; i++) {
        cout << "Hello World" << endl;
    }
    return 0;
}
```

# 問題7 Min, Max and Sum

- $n$  個の整数  $a_i$  ( $i=1,2, \dots, n$ ) を入力し  
それらの最小値、最大値、合計値を出力せよ
- **Input**
  - 1行目に整数の数  $n$  が与えられる
  - 2行目に  $n$  個の整数  $a_i$  が空白区切りで与えられる
- **Constraints**
  - $0 < n \leq 10000$
  - $-1000000 \leq a_i \leq 1000000$

```
#include <iostream>
using namespace std;

int main(void){
    int n, a;
    int min = 2000000, max = -2000000;
    int sum = 0;

    cin >> n;
    for(int i = 0; i < n; i++){
        cin >> a;
        if(a > max){ max = a; }
        if(a < min){ min = a; }
        sum += a;
    }
    cout << min << " " << max << " " << sum << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int n, a;
    int min = 2000000, max = -2000000;
```

```
int sum = 0; ←実はよくない
```

```
    cin >> n;
    for(int i = 0; i < n; i++){
        cin >> a;
        if(a > max){ max = a; }
        if(a < min){ min = a; }
        sum += a;
```

```
    }
    cout << min << " " << max << " " << sum << endl;
    return 0;
```

```
}
```

# 問題の制約をよく見てみる

- **Constraints**

$$0 < n \leq 10000$$

$$-1000000 \leq a_i \leq 1000000$$

# 問題の制約をよく見てみる

- Constraints

$$0 < n \leq 10000$$

$$-1000000 \leq a_i \leq 1000000$$

- 仮に10000個の入力がすべて1000000だったら

**合計は10000000000000 → int型に収まらない！**

(int型の範囲は正負それぞれ約20億)

# 問題の制約をよく見てみる

- **Constraints**

$$0 < n \leq 10000$$

$$-1000000 \leq a_i \leq 1000000$$

- 仮に10000個の入力がすべて1000000だったら

**合計は1000000000000 → int型に収まらない！**

(int型の範囲は正負それぞれ約20億)

- **long long int型を使おう！**

正負それぞれ約900京くらいまで対応した整数型

```
#include <iostream>
using namespace std;

int main(void){
    int n, a;
    int min = 2000000, max = -2000000;
    long long int sum = 0;

    cin >> n;
    for(int i = 0; i < n; i++){
        cin >> a;
        if(a > max){ max = a; }
        if(a < min){ min = a; }
        sum += a;
    }
    cout << min << " " << max << " " << sum << endl;
    return 0;
}
```



# 問題8 Print a Rectangle

- 入力H, Wを受け取り、 $H \times W$ の長方形を # で出力せよ

- Sample Input

3 4

- Sample Output

####

####

####

※ 実際の問題は複数データセットですが、ここではスルー

## 問題8 Print a Rectangle

```
#include <iostream>
using namespace std;

int main(void){
    int h, w;
    cin >> h >> w;
    for(int i = 0; i < h; i++){
        for (int j = 0; j < w; j++) {
            cout << "#";
        }
        cout << endl;
    }
    return 0;
}
```

# 無限ループ

- やめられない止まらない！

```
while ( true ){  
    繰り返したい処理  
}
```

または

```
for ( ;; ){  
    繰り返したい処理  
}
```

# 無限ループ 例

- 無限に「Hello World」と出力せよ

```
while ( true ) {  
    cout << "Hello World" << endl;  
}
```

# ループからの脱出

- 「やめられない止まらない」はやばい

→ ループからの脱出手段が必要

`break;`

# 問題9 Grading

- 複数の学生のテストの点数を、1人につき3つ読み込む

中間試験の点数  $m$ 、期末試験の点数  $f$ 、再試験の点数  $r$   
中間試験と期末試験は 50 点満点、再試験は 100 点満点  
試験を受けていない場合は点数を -1 とする

後述するルールに従って成績をつけて出力せよ

- **Input**

複数のデータセットが入力として与えられる  
各データセットでは、 $m$ 、 $f$ 、 $r$ が1行に与えられる  
 $m$ 、 $f$ 、 $r$ がすべて -1 のとき入力の終わりとする

# 問題9 Grading

- 以下の手順で成績が付けられる:
  - 中間試験、期末試験のいずれかを欠席した場合成績は F
  - 中間試験と期末試験の合計点数が 80 以上ならば成績は A
  - 中間試験と期末試験の合計点数が 65 以上 80 未満ならば成績は B
  - 中間試験と期末試験の合計点数が 50 以上 65 未満ならば成績は C
  - 中間試験と期末試験の合計点数が 30 以上 50 未満ならば成績は D  
ただし、再試験の点数が 50 以上ならば成績は C
  - 中間試験と期末試験の合計点数が 30 未満ならば成績は F

```

#include <iostream>
using namespace std;

int main(void){
    int m, f, r;
    while(true){
        cin >> m >> f >> r;
        if(m == -1 && f == -1 && r == -1){    break;    }
        if(m == -1 || f == -1){                cout << "F" << endl;        }
        else if(m + f >= 80){                    cout << "A" << endl;        }
        else if(m + f >= 65){                    cout << "B" << endl;        }
        else if(m + f >= 50){                    cout << "C" << endl;        }
        else if(m + f >= 30){
            if(r >= 50){    cout << "C" << endl;        }
            else{    cout << "D" << endl;        }
        }
        else{    cout << "F" << endl;        }
    }
    return 0;
}

```



```
#include <iostream>
using namespace std;
```

```
int main(void){
    int m, f, r;
    while(true){
        cin >> m >> f >> r;
        if(m == -1 && f == -1 && r == -1){    break;    }
        if(m == -1 || f == -1){                cout << "F" << endl;    }
        else if(m + f >= 80){                    cout << "A" << endl;    }
        else if(m + f >= 65){                    cout << "B" << endl;    }
        else if(m + f >= 50){                    cout << "C" << endl;    }
        else if(m + f >= 30){
            if(r >= 50){    cout << "C" << endl;    }
            else{    cout << "D" << endl;    }
        }
        else{    cout << "F" << endl;    }
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
```

```
int main(void){
    int m, f, r;
    while(true){
        cin >> m >> f >> r;
        if(m == -1 && f == -1 && r == -1){ break; }
        if(m == -1 || f == -1){ cout << "F" << endl; }
        else if(m + f >= 80){ cout << "A" << endl; }
        else if(m + f >= 65){ cout << "B" << endl; }
        else if(m + f >= 50){ cout << "C" << endl; }
        else if(m + f >= 30){
            if(r >= 50){ cout << "C" << endl; }
            else{ cout << "D" << endl; }
        }
        else{ cout << "F" << endl; }
    }
    return 0;
}
```

# 目次

- おまじない
- 変数
- 四則演算
- 入出力
- 浮動小数点型
- 条件式, 条件分岐
- ループ
- **配列**

# 問題10 Reversing Numbers

- 与えられた数列を逆順に出力せよ

- **Input**

$n$

$a_1 a_2 \dots a_n$

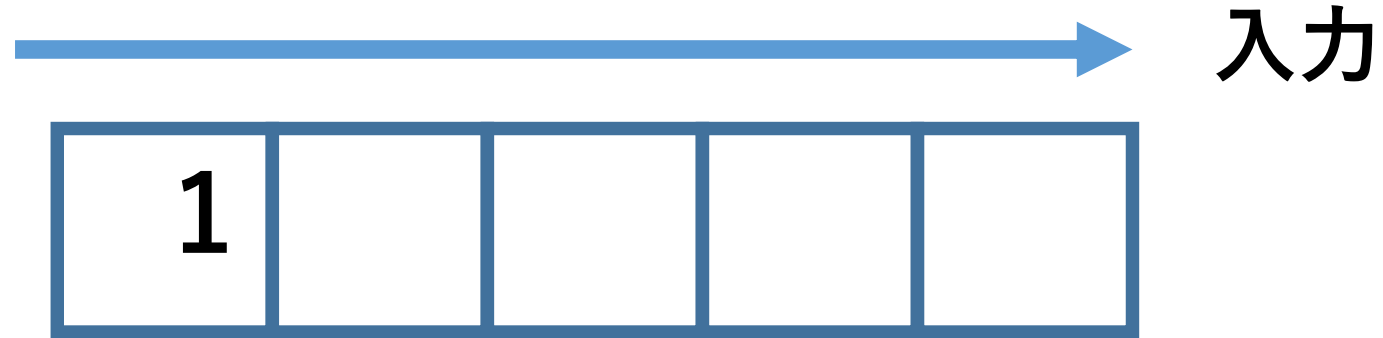
- **Constraints**

$n \leq 100$

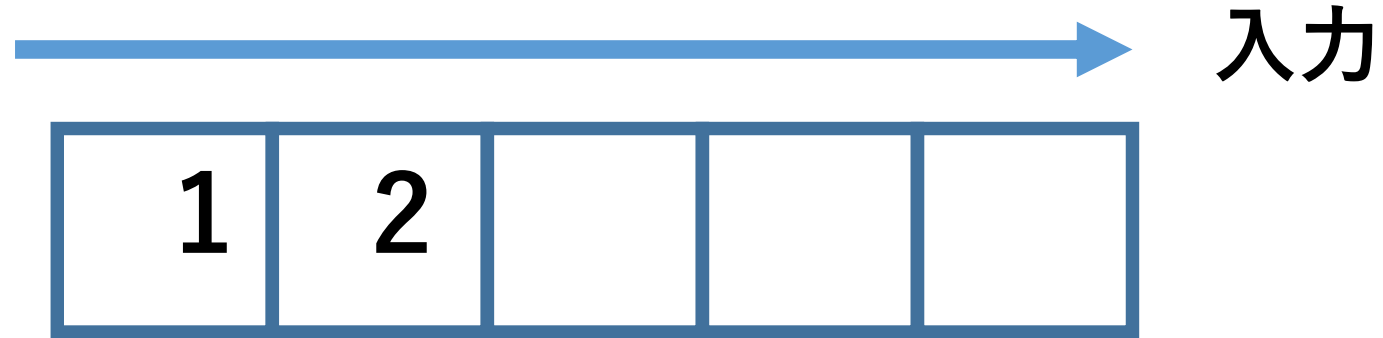
# 問題10の解き方を考える

--	--	--	--	--

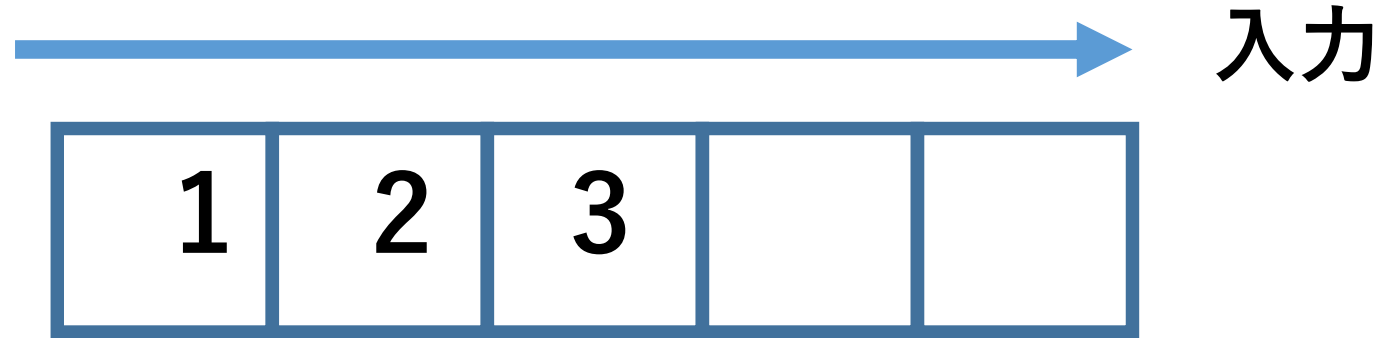
# 問題10の解き方を考える



# 問題10の解き方を考える

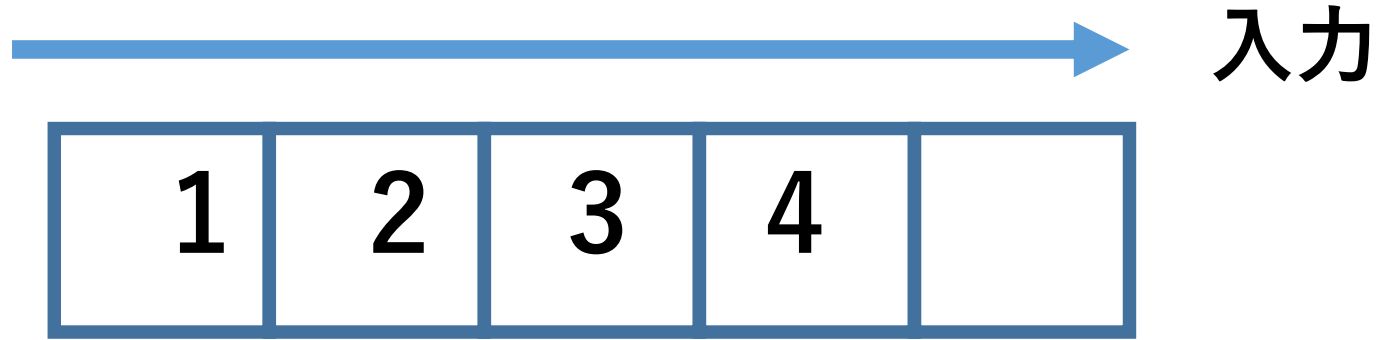


# 問題10の解き方を考える

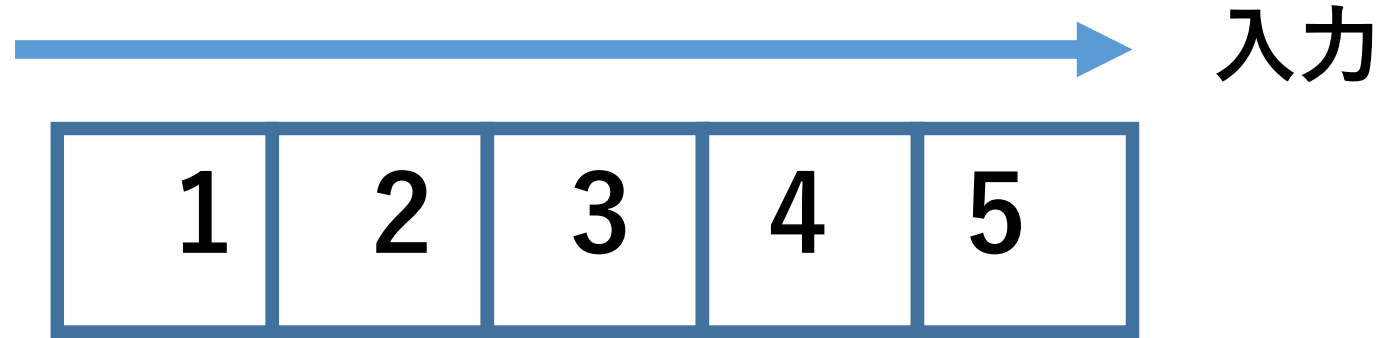




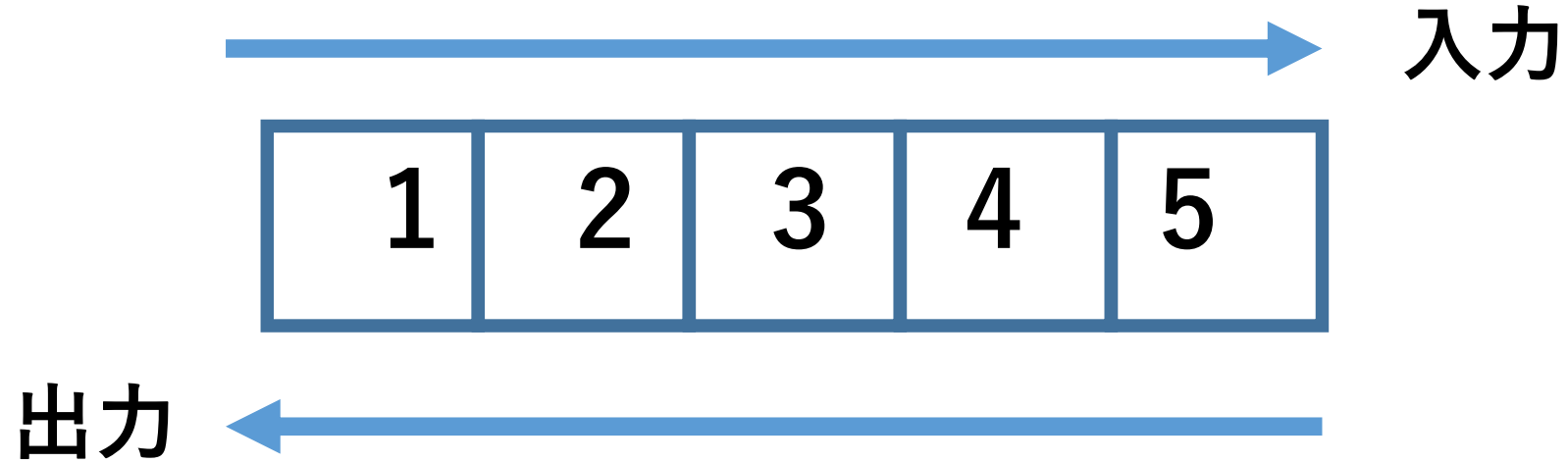
# 問題10の解き方を考える



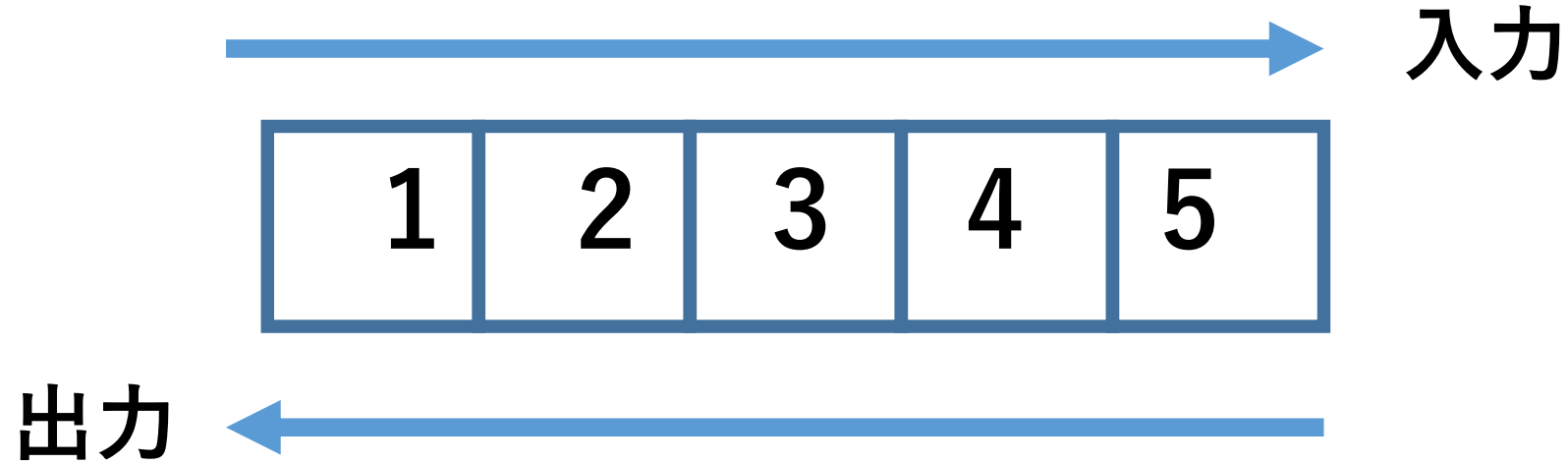
# 問題10の解き方を考える



# 問題10の解き方を考える

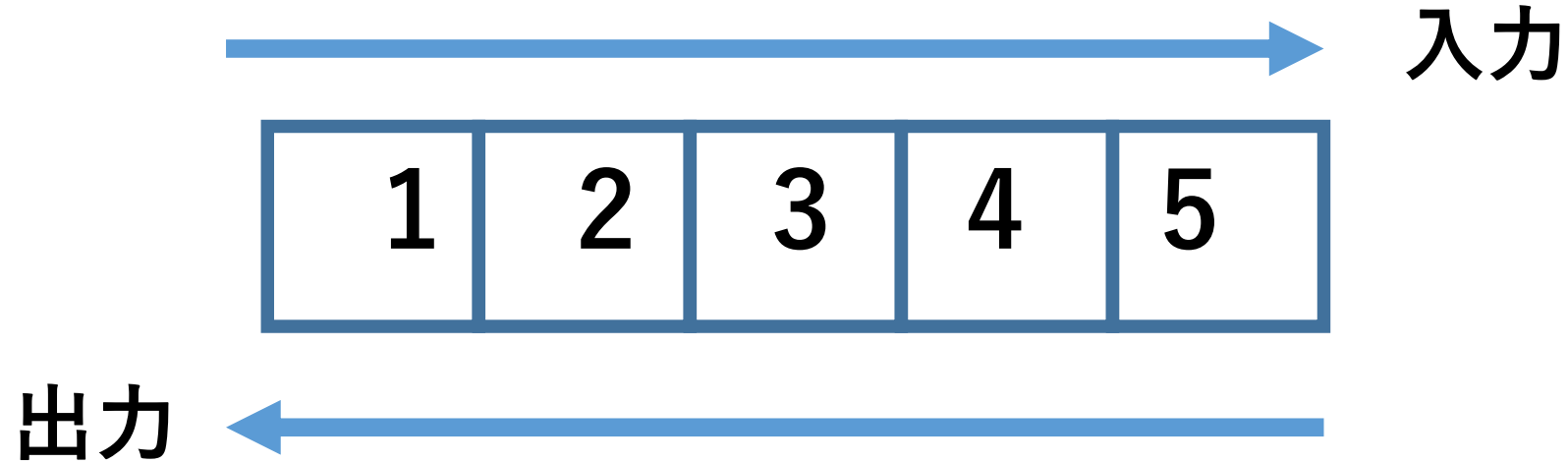


# 問題10の解き方を考える



**欲求: 列になった変数みたいなのが欲しい！**

# 問題10の解き方を考える

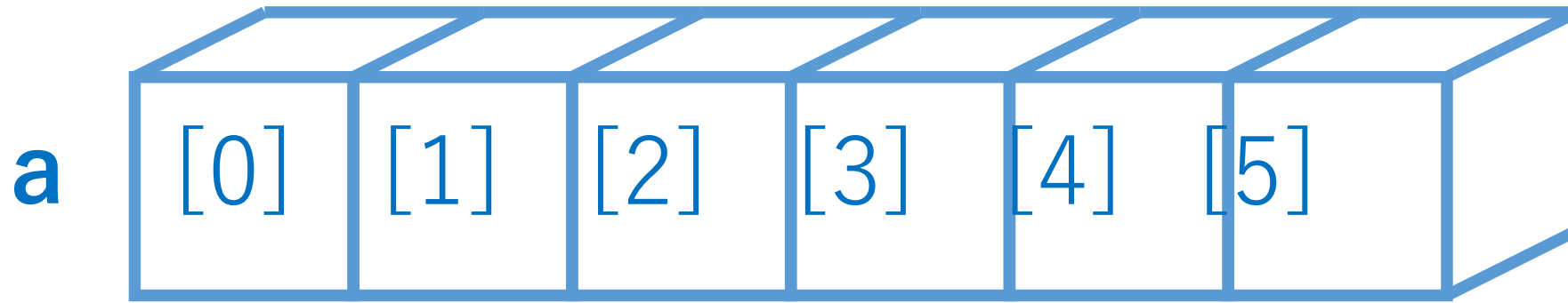


欲求: 列になった変数みたいなのが欲しい！

→ 配列を使おう！

# 配列とは

- 変数の列である



- 使うときは基本的に  
「配列aのなんちゃら番目」と指定して使う

# 配列とは

- 宣言方法

```
int a[10];
```

# 配列とは

- 宣言方法

int a[10];  
型



# 配列とは

- 宣言方法

```
int a[10];
```

名前

# 配列とは

- 宣言方法

```
int a[10];
```

長さ

# 配列とは

- 宣言方法

```
int a[10];
```

- 使い方の例

```
a[3] = 9;
```

```
cin >> a[5];
```

```
cout << a[7] << endl;
```

# 配列とは

- 宣言方法

```
int a[10];
```

- 使い方の例

```
a[3] = 9;
```

**何番目の要素をいじりたいか**

# 配列とは

- 初期化方法その1

```
int a[10] = {2, 3, 5, 8, 6, 0, 7, 9, 1, 4};
```

全要素を列挙する

**この方法は、宣言と同時にしか使えない**

# 配列とは

- 初期化方法その1

```
int a[10] = {2, 3, 5, 8, 6, 0, 7, 9, 1, 4};
```

全要素を列挙する

この方法は、宣言と同時にしか使えない

- 初期化方法その2

```
for(int i = 0; i < 10; i++){  
    a[i] = 3;  
}
```

# 問題10 Reversing Numbers

- 与えられた数列を逆順に出力せよ

- **Input**

$n$

$a_1 a_2 \dots a_n$

- **Constraints**

$n \leq 100$

# 問題10 Reversing Numbers

```
#include <iostream>
using namespace std;

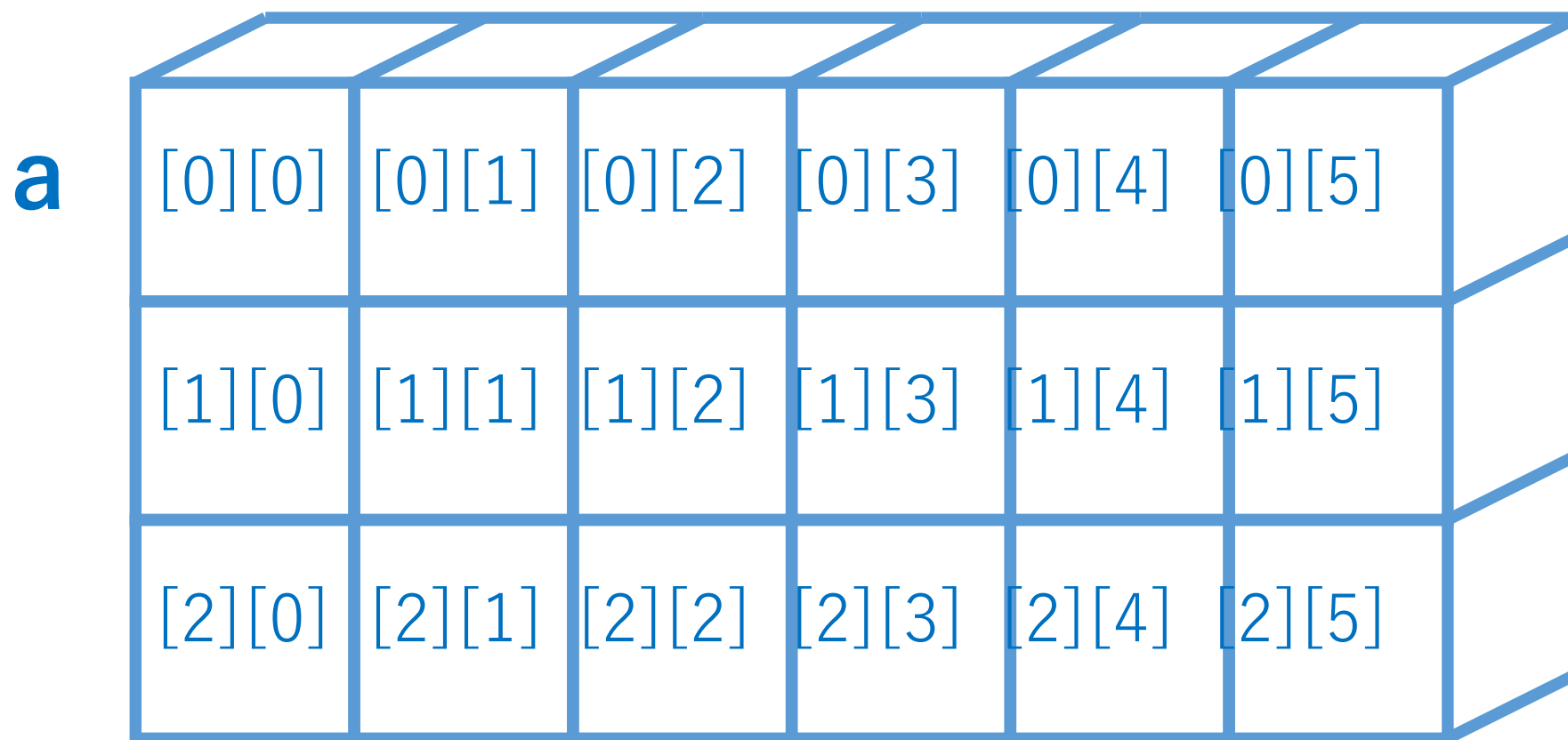
int main(void){
    int n;
    int a[200];

    cin >> n;
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    for(int i = n - 1; i >= 0; i--){
        cout << a[i];
        if(i != 0){ cout << " "; }
    }
    cout << endl;
    return 0;
}
```



# 多次元配列

**int a[3][6];**    平面的や立体的な配列もできる！



# 問題11 Matrix Multiplication

- $n \times m$  の行列Aと  $m \times l$  の行列Bを入力し  
それらの積である  $n \times l$  の行列Cを出力せよ
- **Input**  
1行目に  $n$ 、 $m$ 、 $l$  が空白区切りで与えられる  
続く行に  $n \times m$  の行列A と  $m \times l$  の行列 B が与えられる
- **Constraints**  
 $1 \leq n, m, l \leq 100$   
 $0 \leq a_{ij}, b_{ij} \leq 10000$

# 問題11 Matrix Multiplication

- **Sample Input**

```
3 2 3
1 2
0 3
4 5
1 2 1
0 3 2
```

- **Sample Output**

```
1 8 5
0 9 6
4 23 14
```

# 問題11 Matrix Multiplication

- **Sample Input**

```
3 2 3
1 2
0 3
4 5
1 2 1
0 3 2
```

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 8 & 5 \\ 0 & 9 & 6 \\ 4 & 23 & 14 \end{bmatrix}$$

- **Sample Output**

```
1 8 5
0 9 6
4 23 14
```

# 問題11 Matrix Multiplication

```
1 #include <iostream>
2 using namespace std;
3
4 int main(void){
5     int n, m, l;
6     long long int a[200][200], b[200][200], c[200][200];
7
8     // 初期化
9     for (int i = 0; i < 200; i++) {
10         for (int j = 0; j < 200; j++) {
11             c[i][j] = 0;
12         }
13     }
14
15     // 行列のサイズを受け取る
16     cin >> n >> m >> l;
17
18     // 行列Aを受け取る
19     for (int i = 0; i < n; i++) {
20         for (int j = 0; j < m; j++) {
21             cin >> a[i][j];
22         }
23     }
24
```

```
25     // 行列Bを受け取る
26     for (int i = 0; i < m; i++) {
27         for (int j = 0; j < l; j++) {
28             cin >> b[i][j];
29         }
30     }
31
32     // 行列Cを計算する
33     for (int i = 0; i < n; i++) {
34         for (int j = 0; j < l; j++) {
35             for (int k = 0; k < m; k++) {
36                 c[i][j] += a[i][k] * b[k][j];
37             }
38         }
39     }
40
41     // 出力(計算のところにまとめてもOK)
42     for (int i = 0; i < n; i++) {
43         for (int j = 0; j < l; j++) {
44             cout << c[i][j];
45             if(j != l - 1){ cout << " "; }
46         }
47         cout << endl;
48     }
49
50     return 0;
51 }
```

# まとめ

- 条件分岐、ループ、配列をマスターしたぞ！

**今日からキミも立派なプログラマだ！**

おわり