STAGE 1-7

# 真理値と条件分岐

「もしそうなら」を定義する

## 🚩 このSTAGEの目標

- 新たな型「真理値」の演算を知る
- 条件分岐の代表例を知る

しん-り-ち

# 真理値

boolean

2種類の値 `true` か `false` しかとらない型

`bool` 型と呼ばれる

コード

```
int main(void) {
  bool yes = true;

  bool no = false;

}
```

しん-り-ち

# 真理値

boolean

2種類の値 `true` か `false` しかとらない型

`bool` 型と呼ばれる

コード

```
int main(void) {
  bool yes = true;
  printf("true = %d", yes);
  bool no = false;
  printf("false = %d", no);
}
```

出力

```
true = 1
false = 0
```

printf で `"%d"` で表示できるが
数字になってしまう

`true` は `1` 、`false` は `0` に変換される

# 真理値

`bool`

しん-り-ち　　2種類の値 `true` か `false` しかとらない型

**論理演算**（ろんりえんざん）

NOT や AND に代表される、条件が正しいか判断する演算
プログラミングでは「〜と等しい」「〜より大きい」も含む

基本的に2つの数を
**論理演算子** (論理演算に使う記号) の前後に書く

コード

```
int main(void) {
  printf( “true ANÐ false = %d”, true & false );
  return 0;                                  ↑ 論理演算子
}
```

## 論理演算（ろんりえんざん）

NOT や AND に代表される、条件が正しいか判断する演算
プログラミングでは「〜と等しい」「〜より大きい」も含む

比較演算子

| コード内の書き方 | 説明 | 例 |
| --- | --- | --- |
| == | 値が等しい | `1 == 1` = true<br>`1 == 2` = false |
| != | 値が異なる | `6 != 9` = true<br>`1 != 1` = false |
| > | 前の値が後の値よりも大きい | `12 > 7` = true<br>`1 > 2` = false |
| < | 前の値が後の値よりも小さい | `3 < 8` = true<br>`1 < 1` = false |
| >= | 前の値が後の値以上である | `1 >= 1` = true<br>`1 >= 2` = false |
| <= | 前の値が後の値以下である | `3.1 <= 3.1` = true<br>`9 <= -2` = false |

## 論理演算（ろんりえんざん）

NOT や AND に代表される、条件が正しいか判断する演算
プログラミングでは「〜と等しい」「〜より大きい」も含む

### 真理値どうしで演算することもできる

論理演算子

| 名称 | コード内の書き方 | 説明 | 例 |
|---|---|---|---|
| AND | & | どちらも true のとき true を返す | true & true = true　　true & false = false |
| AND | && | アンパサンド2つ | AND の代わり（少し仕様が異なる） |
| OR | \| | 1つ以上 true のとき true を返す | true \| false = true　　false \| false = false |
| OR | \|\| | 絶対値2つ | OR の代わり（少し仕様が異なる） |
| NOT | ! | 直後の真理値を反転する | !false = true　　!true = false |

Python などで使える AND や OR などのキーワードは C++ では使えないので注意

## 論理演算（ろんりえんざん）

NOT や AND に代表される、条件が正しいか判断する演算
プログラミングでは「〜と等しい」「〜より大きい」も含む

### 真理値どうしで演算することもできる

論理演算子

| 名称 | コード内の書き方 | 説明 | 例 |
|---|---|---|---|
| AND | & | どちらも true のとき true を返す | true & true = true　　true & false = false |
| AND | && | アンパサンド2つ | AND の代わり（少し仕様が異なる） |
| OR | \| | 1つ以上 true のとき true を返す | true \| false = true　　false \| false = false |
| OR | \|\| | 絶対値2つ | OR の代わり（少し仕様が異なる） |
| NOT | ! | 直後の真理値を反転する | !false = true　　!true = false |

Python などで使える AND や OR などのキーワードは C++ では使えないので注意

# if 文を使うと、値に応じてプログラムの流れを変更できる

コード

```
if ( bool型/論理式 )
{ // 波かっこで開始
  /*
  bool型/論理式が
  true のときに
  実行される
  コードを書く
  */
} // 波かっこで終了
```

コード

```
int main(void) {
  int valueX = 5;
  int valueY = 1;
  if ( valueX > valueY ) { // valueX > valueY が true なら実行
    printf("X=%dはY=%dより大きいです", valueX, valueY );
  }
  if ( ! ( valueX > valueY ) ) { // ! (NOT) で、↑がfalseなら実行
    printf("X=%dはY=%dより大きくありません", valueX, valueY );
  }
  return 0;
}
```

出力

```
X=5はY=1より大きいです
```

# if 文を使うと、値に応じてプログラムの流れを変更できる

コード

```
if ( bool型/論理式 )
{ // 波かっこで開始
  /*
  bool型/論理式が
  true のときに
  実行される
  コードを書く
  */
} // 波かっこで終了
```

コード

```
int main(void) {
  int valueX = -100; // 変更
  int valueY = 3;    // 変更
  if ( valueX > valueY ) { // valueX > valueY が true なら実行
    printf("X=%dはY=%dより大きいです", valueX, valueY );
  }
  if ( ! ( valueX > valueY ) ) { //!(NOT)を使い、↑がfalseなら実行
    printf("X=%dはY=%dより大きくありません", valueX, valueY );
  }
  return 0;
}
```

出力

```
X=-100はY=3より大きくありません
```

# if 文の閉じかっこの直後に else を入れると否定の条件を書かなくていい

コード

```
if ( bool型/論理式 )
{
 // bool型/論理式が
 // true のときに
 // 実行される
} else {
 // bool型/論理式が
 // false のときに
 // 実行される
}
```

コード

```
int main(void) {
  int valueX = -100;
  int valueY = 3;
  if ( valueX > valueY ) { // valueX > valueY が true なら実行
    printf("X=%dはY=%dより大きいです", valueX, valueY );
  } else { // ( ! ( valueX > valueY ) ) と同等
    printf("X=%dはY=%dより大きくありません", valueX, valueY );
  }
  return 0;
}
```

出力

```
X=-100はY=3より大きくありません
```

# if 文の閉じかっこの直後に else if を入れると、「そうでないとき」が書ける

コード

```
if ( bool型/論理式① ) {
  // bool型/論理式① が
  // true のとき実行される
} else if ( bool型/論理式② ) {
  // bool型/論理式①がfalseで
  // bool型/論理式② が
  // true のとき実行される
} else {
  // ここまですべて(①と②) が
  // false のとき
  // ここが実行される
}
```

コード

```
int main(void) {
  int valueX = -100;
  int valueY = 3;
  if ( valueX > valueY ) {
    printf("X=%dはY=%dより大きいです", valueX, valueY );
  } else if ( valueX == valueY ) {
    printf("X=%dはY=%dと等しいです", valueX, valueY );
  } else {  // valueX > valueY でなく valueX == valueY でない
    printf("X=%dはY=%dより小さいです", valueX, valueY );
  }
  return 0;
}
```

出力

```
X=-100はY=3より大きくありません
```

## 💭 このSTAGEのまとめ

`bool`型は`true`か`false`だけ格納できる

論理演算・比較の結果は`bool`型

```
( (1 > 3) == true ) // 結果は true
```

`if ~ else`で分岐できる

コード

```
if ( bool型/論理式 ) {
 // bool型/論理式が true のとき実行されるコード
} else { // else 以降は省略可能
 // bool型/論理式が false のとき実行されるコード
}
```

## 🚩 この STAGE の目標

- ✅ 新たな型「真理値」の演算を知る
- ✅ 条件分岐の代表例を知る

## できるようになったこと

値の内容によって
結果が変わるプログラムが書ける

論理演算ができる

# 真理値 `bool`

しん-り-ち　　2種類の値 `true` か `false` しかとらない型

コード

```
int main(void) {
  bool yes = 1;
  printf("true = %d", yes);
  bool no = "no";
  printf("false = %s", no);
}
```

コンソール

```
Code.cpp(5,12): warning C4305: '初期化中':
'int' から 'bool' へ切り詰めます。
```

`true` `false` 以外も代入できるが
「切り詰め」の**警告**が表示される

ゼロ以外の値は、すべて `true`
厳密にゼロなら、 `false`