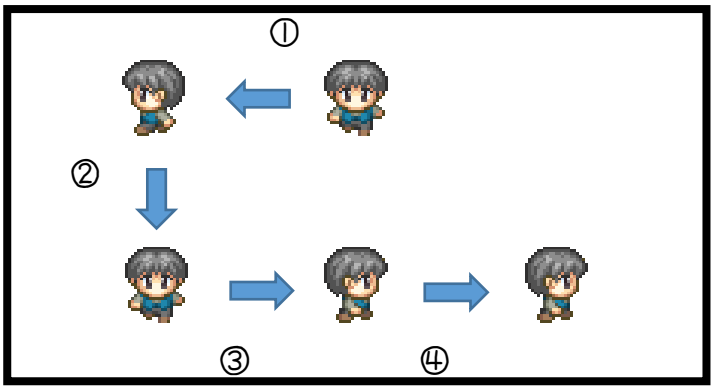
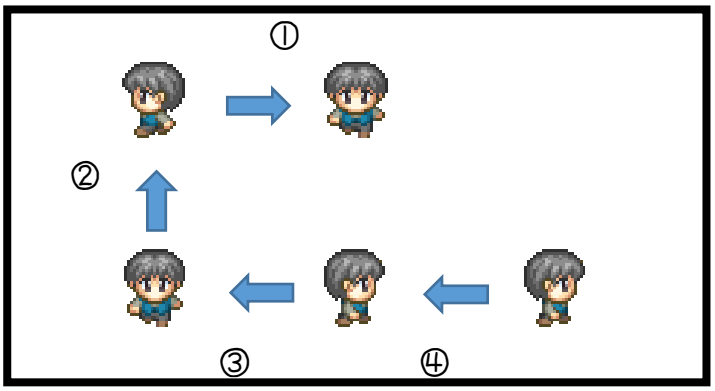
巻き戻し機能

プレイヤーの操作を記録し、【最後に操作した順】に再生することで、
巻き戻し機能を実装することができます。



① 左に移動
② 下に移動
③ 右に移動
④ 右に移動

巻き戻し！



④ 左に移動
③ 左に移動
② 上に移動
① 右に移動

移動方向は反対に。

実装の手引き

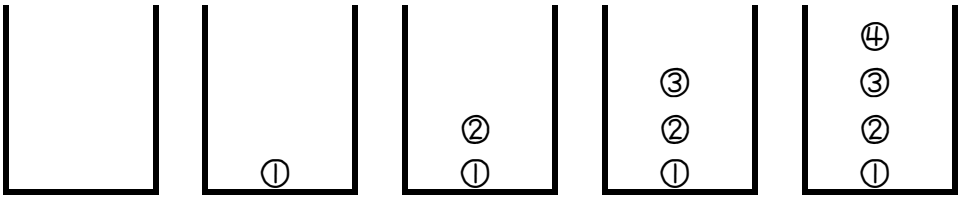
- キャラクターが操作されたことを記録しておくこと
- 記録しておく情報を作りながらで良いので、意識しておくこと
- 再生する時は、最後に記録された順から処理を行うこと

順番を逆にする

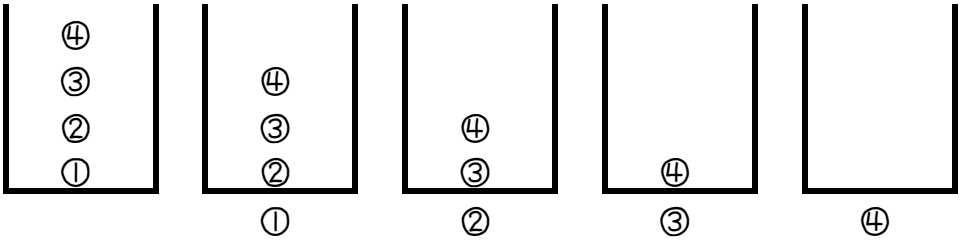
普段の生活の中では、順番は早い順ということが多いですが、
プログラミングの世界では、逆の順番で行っていった方が、
都合が良いことがあります。

普通は、①②③の順で清算して貰えるが、逆だと③②①

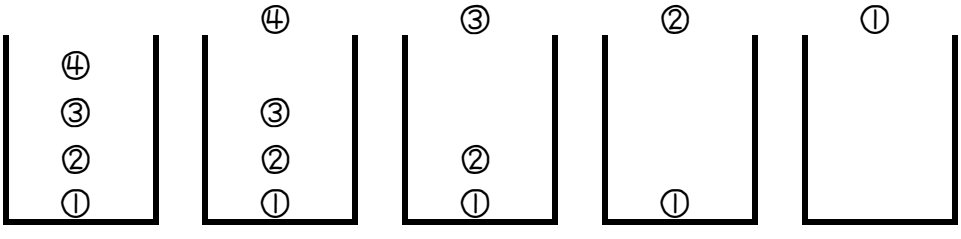
順番



FIFO(先入れ先出し)



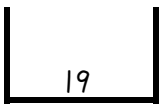
LIFO(後入れ先出し)



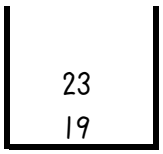
このLIFOを、スタック(stack)と呼び、
プログラムの多くは、標準でstack機能が実装されています。

```
std::stack<int> st;
st.push(1);                 要素の追加
int stTop = st.top();       次の要素の取得
st.pop();                   次の要素を削除する
```

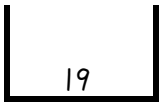
```
std::stack<int> st;
st.push(19);
```



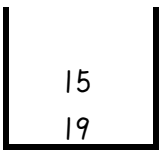

```
st.push(23);
```



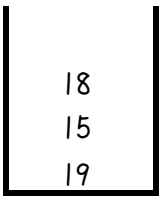

```
st.pop();
```



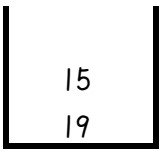

```
st.push(15);
```



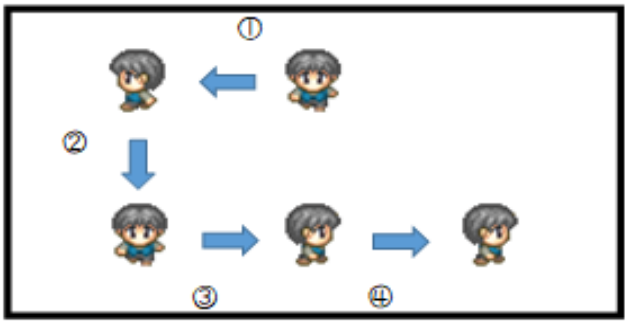

```
st.push(18);
```




```
st.pop();
```

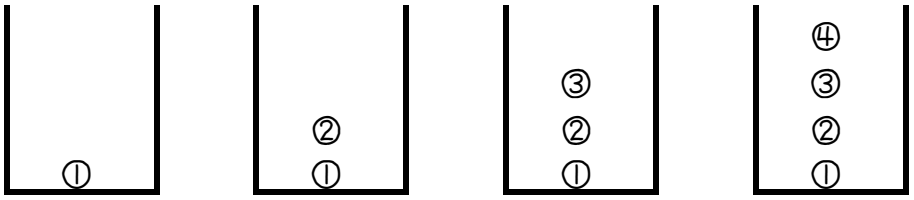


stackの仕組みを使って、まずは操作履歴を格納していきます。



① 左に移動
② 下に移動
③ 右に移動
④ 右に移動

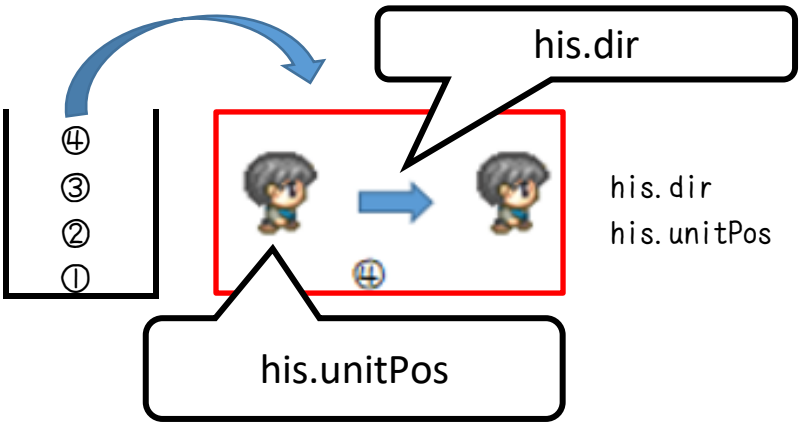
```
struct History {
    DIR dir;                移動方向
    Vector2 unitPos;        ユニットの移動前座標
    Box* box;               押し出した荷物
    Vector2 boxPos;         荷物の移動前座標
};
```




```
mHistoryBack.push(History{左方向….});
mHistoryBack.push(History{下方向….});
mHistoryBack.push(History{右方向….});
mHistoryBack.push(History{右方向….});
```

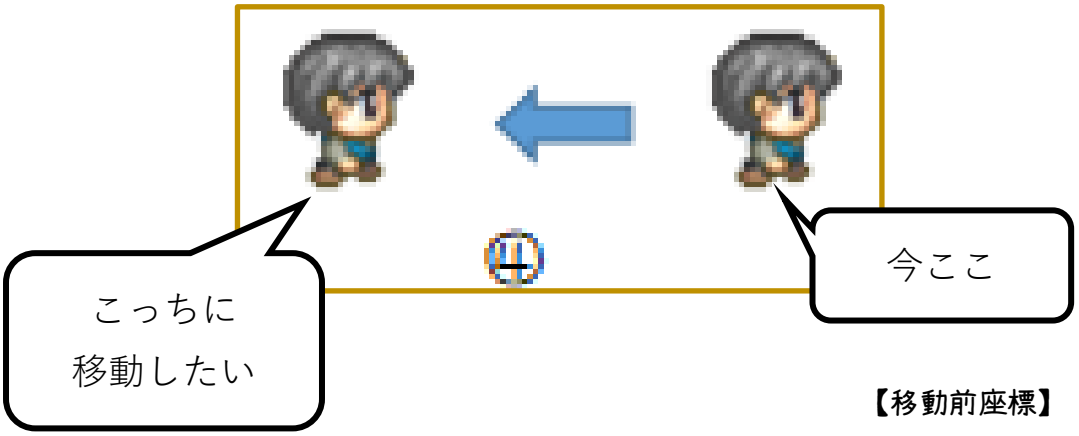
操作の巻き戻し

```
History his = mHistoryBack.top();
```



his.dir　　右方向
his.unitPos　　移動前座標

Unitの移動に使用している線形補間の移動は、
【移動前座標】と【移動先座標】が決まれば、勝手に移動してくれる。



【移動前座標】

【移動先座標】

巻き戻し後に、stackから削除

```
mHistoryBack.pop();
```