# **１．スタック（Stack）**

スタックは、最初に格納したデータが最後に取り出されるというデータ構造です（Last In, First Out、略してLIFO）。データの一時退避などで使われます。例えばJavaであるメソッドを呼び出すとそのメソッドの処理が終わったら呼び出し元へ戻ってきますが、この戻ってくるアドレスを覚えておくような場合にスタックが使われます。

データ格納（プッシュ）

データ取り出し（ポップ）

3

2

6

1

47

0

23

①

②

6

0

3

2

47

23

1

sp

②

①

sp

データ取り出し時は

①spを-1してから

②データを取り出す。

データ格納時は

①スタックポインタ（sp）の指す場所にデータを格納し、

②spを+1する。

# **２．キュー（Queue、データシフト方式）**

キューは、最初に格納したデータが最初に取り出されるというデータ構造です（First In, First Out、略してFIFO）。非同期処理をするプログラム間でデータを受け渡す場合などで使われます。代表的なものにWindowsのメッセージキューがあります。ゲームではゲームプログラムとサウンドプログラムでデータのやり取りをする場合などに使われます。待ち行列やリングバッファとも呼ばれます。

データ格納（プッシュ）

データ読み出し（ポップ）

3

2

6

1

47

0

23

①

0

3

2

1

sp

①

②

sp

6

47

①

23

データ読み出し時は

①0番からデータを取り出し、

②spを-1すると同時に残っているデータをずらす。

データ格納時は

①spの指す場所にデータを格納し、

②spを+1する。

# **３．キュー（リングバッファ）**

配列の先頭データを取り出して残りのデータをひとつずつシフトさせると、かなりのオーバーヘッドが発生します。配列の最後と先頭がつながっているように扱うリングバッファを使えば、この問題が解決します。

**リングバッファの実装①**

データ書き込み場所を表すwriteとデータ読み出し場所を表すreadを使って（readがwriteを追いかけるように）作成します。データの書き込みや読み出しを行うと1進み、配列の最後まで進むと先頭へ戻ります。また、writeがreadを追い越さないように間をひとつ空けるため、格納するデータ数＋1のバッファサイズが必要となります。

データ格納

データ読み出し

3

2

6

1

47

0

23

①

0

3

2

47

1

②

6

write

②

read

①

23

データ読み出し時は

①読み出し位置（read）の指す場所から読み出し、

②readを+1する。readがバッファサイズを越えたときは0にする。

データ格納時は

①書き込み位置（write）の指す場所に格納し、

②writeを+1する。writeがバッファサイズを越えたときは0にする。

**リングバッファの実装②**

データの先頭位置を表すheadとデータ数を表すcountを使って作成します（headは**実装①**のreadに該当）。データの書き込みはhead＋countの場所に行います。書き込みを行うとcountを1増やします。読み出しはheadの場所を読み出し、headを1進めcountを1減らします。先頭のデータ（head）と最後のデータ（head＋count）の間に空席を設けなくても動作します。