プログラミング基礎

http://bit.ly/prog2d

前期の復習（スタックの実装）

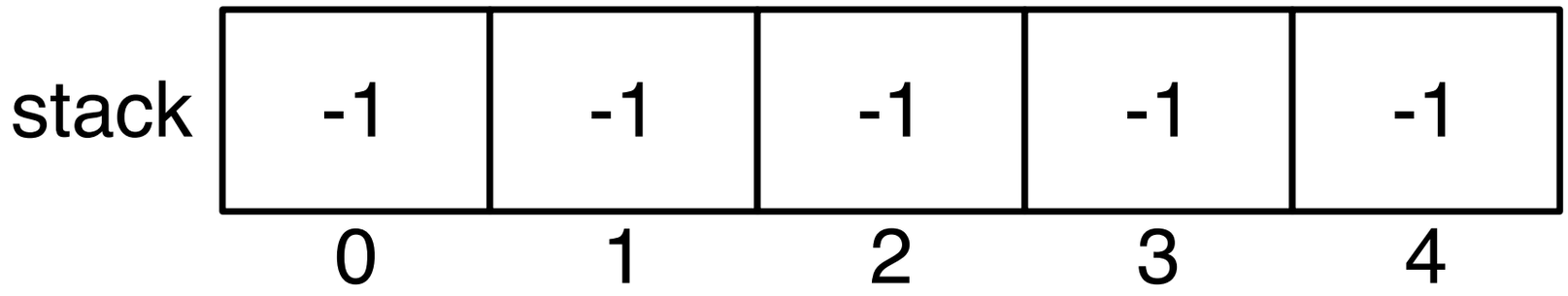
後期 第1週

2017/9/25

スタックとは…

データを一次元で格納する際に使われるデータの管理方法です。コンピュータでは、データを一時的に退避したり復元したりする時に使われます。
今回は、配列を使って実現します。

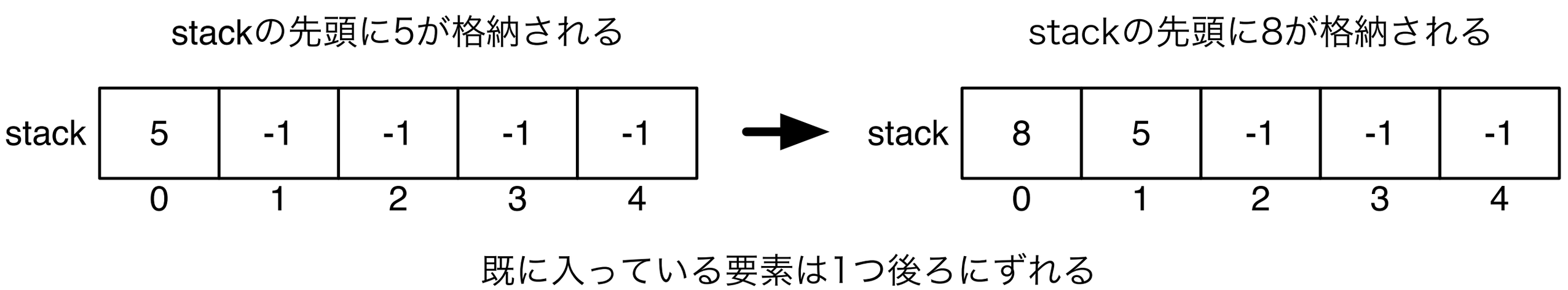
配列stackに5個分のデータを格納できる



「何もデータが格納されていない」は-1で表すことにする

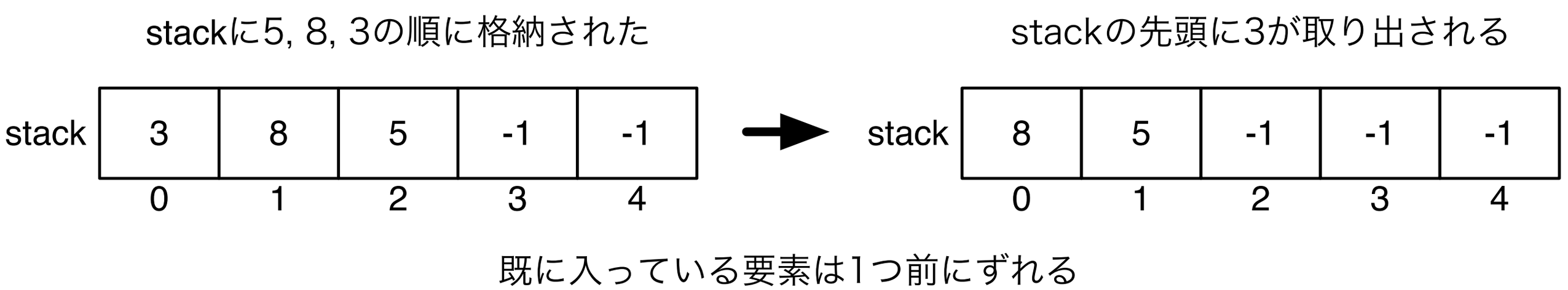
スタックへの格納

データを格納する操作は「push」と呼び、配列の全ての要素を後ろに1つずらした後に、配列の0番目に格納します。



スタックからの取り出し

データを取り出す操作は「pop」と呼び、配列の0番目の要素を取り出し、1番目以降の全ての要素を前に1つずらします。



つまり、スタックは「1番最後に格納したデータから取り出される」データ管理方法

【課題1-1】

スタックを表すint型の配列stack（要素は5個）をグローバル変数として宣言し、スタックを空の状態にする関数clear()とスタックを出力する関数show()を作成しましょう。

[この関数のプロトタイプ宣言]

```
void clear();
 /* stackの全ての要素に-1を代入する */


void show();
 /* stackの全ての要素を出力する */
```

【課題1-1】

[mainでの処理]

```
  clear();
  show();
```

[実行結果]

```
  -1  -1  -1  -1  -1
```

（←全て空の要素が出力される）

【課題1-2】

スタックにデータを格納する関数push()を作成しましょう。

[この関数のプロトタイプ宣言]

```
 void push(int x);

/* stackのi-1番目の要素をi番目にずらす
    (配列の後ろから処理する繰り返しにした方が作りやすい) */
/* ずらした結果、最後の要素の整数値は消える */
/* stackの0番目にxを代入する */
```

【課題1-2】

[mainでの処理（前の課題のmainに続けて書く）]

```
  push(5);
  show();
  push(8);
  push(3);
  show();
```

[実行結果]

```
  5  -1  -1  -1  -1        （←  5をpushした後の出力）
  3   8   5  -1  -1        （←  続けて8と3をpushした後の出力）
```

【課題1-3】

スタックからデータを取り出す関数pop()を作成しましょう。

[この関数のプロトタイプ宣言]

```
int pop();

/* stackの0番目の値を、戻り値とする変数に代入しておく */
/* stackのi+1番目の要素をi番目にずらす
   (配列の先頭から処理する繰り返しで作れる) */
/* ずらした後に、stackの最後の要素に-1を代入する */
/* 戻り値とした変数を「スタックから取り出した値」として
    returnする */
```

【課題1-3】

[mainでの処理（前の課題のmainに続けて書く）]

```
  printf("pop: %d\n", pop());
  show();
  printf("pop: %d\n", pop());
  show();
```

[実行結果]

```
  pop: 3
  8   5 -1 -1 -1          （← popで3が取り出された後の出力）

  pop: 8
  5 -1 -1 -1 -1          （← popで8が取り出された後の出力）
```

まだ余裕のある人は…

【課題1-4】

課題1-3までのスタックでは…

- 既に5個データが格納されている場合でもpushできてしまう
  → 最初に格納したデータが消えるてしまう

- スタックが空である場合でもpopできてしまう
  → 空の場合、ずらす必要がない

まだ余裕のある人は…

【課題1-4】

スタックに何個のデータが格納されているのかカウントする変数（例えばcount）をグローバル宣言して、作成した関数を以下のように改良してみましょう。

- **関数clear()では、countを0にリセットする。**
- **関数push()では、countが5未満の場合はstackに格納する処理をして、そうでない場合は「スタックが一杯である」メッセージを出力する。また、データを格納したらcountを1つ増やす。**
- **関数pop()では、countが0より大きい場合はstackから取り出す処理をして、そうでない場合は「スタックが空である」メッセージを出力する。また、データを取り出したらcountを1つ減らす。**

【課題1-4】

[mainでの処理]

```
  clear();
  push(5); push(8); push(3); push(7); push(9);
  show();
  push(4);  /* ←スタックが一杯の場合にpushをする */
  show();

  printf("pop: %d\n", pop()); printf("pop: %d\n", pop());
  printf("pop: %d\n", pop()); printf("pop: %d\n", pop());
  printf("pop: %d\n", pop());
  show();

  /* ↓スタックが空の場合にpopをする */
  printf("pop: %d\n", pop());
  show();
```

【課題1-4】

[実行結果]

```
  9   7   3   8   5     (← pushを5回以上した場合)
  Stack is full.
  9   7   3   8   5
  pop: 9                (← popを5回以上した場合)
  pop: 7
  pop: 3
  pop: 8
  pop: 5
  -1 -1 -1 -1 -1
  Stack is empty.
  pop: -1
  -1 -1 -1 -1 -1
```