Stack & Queue

TA 葉宥辰

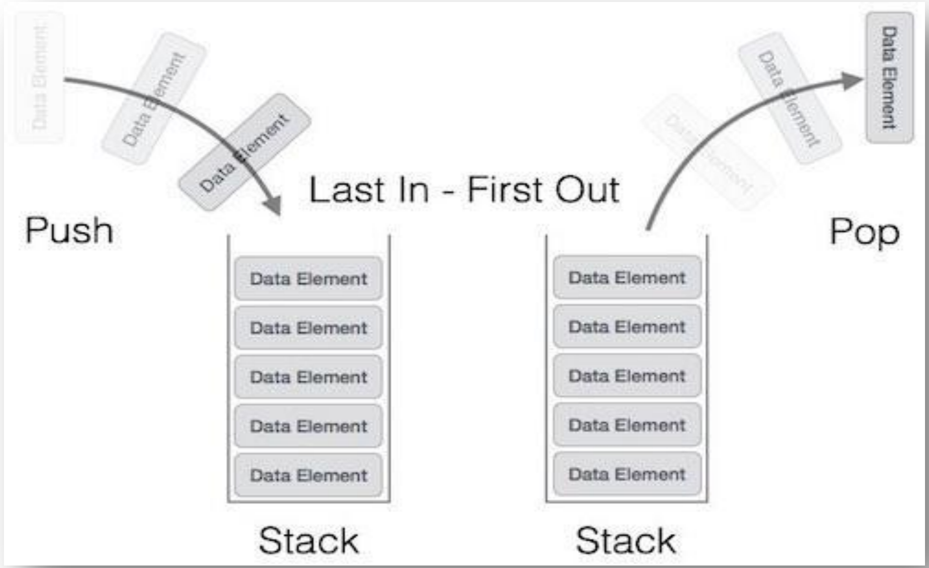
Outline

- Stack
  - Operation of stack
  - Implementation
  - Application of stack
- Queue
  - Operation of queue
  - Implementation
  - Application of queue

Stack

Stack

- It is named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc.



Operation of stack

- Push
- Pop
- Top
- Empty

```
stack<int> stk;
for (int i = 0; i < 5; i++)
  stk.push(i);
// stk = {0, 1, 2, 3, 4}

stk.pop();
// stk = {0, 1, 2, 3}

cout << stk.top();
// output: 3
```

Actually you can just use vector

Implementation

- Array implementation

```
struct Stack {
  int st[MAXSIZE];
  int tp = 0;
  void push(int x);
  void pop();
  int top();
  bool empty();
};
```

```
void push(int x) {
  st[++tp] = x;
}
```

```
void pop() {
  if (empty()) return;
  tp--;
}
```

```
int top() {
  return st[tp];
}
```

```
void empty() {
return tp == 0;
}
```

Application of stack

- Recursion function call
- Undo sequence of text editor
- Component of other data structures
- std::stack<int, std::vector<int>>

Balanced Brackets in an expression

- Given an expression string exp, write a program to examine whether the pairs and the orders of “{“, “}”, “(“, “)”, “[“, “]” are correct in the given expression.

- Ex:
- Input = “[()]{}{[()()]()}”   output = Balanced
- Input = “[(])”   output = Not Balanced

Queue

Queue

- One end is always used to insert data (enqueue) and the other is used to remove data (dequeue).

LAST IN
LAST OUT
ONE WAY
FIRST IN
FIRST OUT

Operation of queue

- Push
- Pop
- Front
- Empty

```
queue<int> q;
for (int i = 0; i < 5; i++)
  q.push(i);

// q = {0, 1, 2, 3, 4}
//        ^           ^
//      head         tail

q.pop();
// q = {1, 2, 3, 4}

cout << q.front();
// output: 1
```

Implementation

- Array implementation

```
struct Queue {
  const static int MAXSIZE = 100000;
  int q[MAXSIZE];
  int head = 0, tail = 0;
  void push(int x);
  void pop();
  int front();
  void empty();
};
```

Implementation

```
void push(int x) {
  q[tail] = x;
  tail = (tail + 1) % MAXSIZE;
}
```

```
void pop() {
  if (empty()) return;
  head = (head + 1) % MAXSIZE;
}
```

```
int front() {
  return q[head];
}
```

```
void empty() {
  return head == tail;
}
```

Application of queue

- Waiting list
- Component of other data structures (E.g. BFS)

Class Implementation

Bracket Matching: https://ideone.com/HHghQ4