

# Algorytmy i struktury danych - Stosy i kolejki

Marcin Żurowski

21 grudnia 2020

```
STRUCT-STACK S
INIT(S) /*inicjuje zmienne*/
PUSH(S,1)
PUSH(S,2)
WRITE(S) //1 2
PUSH(S,3)
WRITE(S) //1 2 3
WRITE(STACK-EMPTY(S)) //false
WRITE(POP(S)) //3
WRITE(POP(S)) //2
WRITE(S) //1
WRITE(POP(S)) //1
WRITE(POP(S)) //niedomiar
WRITE(S) //
WRITE(STACK-EMPTY(S)) //true
```

```
STRUCT-QUEUE Q
INIT(Q) /*inicjuje zmienne*/
ENQUEUE(Q,1)
ENQUEUE(Q,2)
WRITE(Q) //1 2
ENQUEUE(Q,3)
WRITE(Q) //1 2 3
WRITE(QUEUE-EMPTY(Q)) //false
WRITE(DEQUEUE(Q)) //1
WRITE(DEQUEUE(Q)) //2
WRITE(Q) //3
WRITE(DEQUEUE(Q)) //3
WRITE(DEQUEUE(Q)) //niedomiar
WRITE(Q) //
WRITE(QUEUE-EMPTY(Q)) //true
```

# Stos - funkcje i procedury

```
STRUCT-STACK  
  data[1..10]  
  top = 0  
STACK-EMPTY(S)  
PUSH(S,k)  
POP(S)
```

## Kolejka - funkcje i procedury

```
STRUCT-QUEUE  
  data[1..10]  
  length = 10  
  head = 1  
  tail = 1  
QUEUE-EMPTY(Q)  
ENQUEUE(Q, k)  
DEQUEUE(Q)
```

Napisz algorytm który:

- 1 Scalanie dwóch stosów
- 2 Scalanie dwóch kolejek
- 3 Rozmiar kolejki
- 4 Implementacja stosu przy pomocy dwóch kolejek (0.5 pkt. lub zad.5)
- 5 Implementacja kolejki przy pomocy dwóch stosów (0.5 pkt. lub zad.4)