

# Zaawansowane algorytmy - Listy z dowiązaniem

Marcin Żurowski

6 marca 2024

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
- Oceny:

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
- Oceny:

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 100 punktów za wypracowanie
  - 100 punktów za egzamin
  - 100 punktów za seminarium
- Oceny:
  - 100 punktów za wypracowanie
  - 100 punktów za egzamin
  - 100 punktów za seminarium

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5



# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5

# Zaliczenie

- Obecność:
  - minimalnie 8 obecności
  - 3 zajęcia nieusprawiedliwione
  - tydzień na przyniesienie usprawiedliwienia
- Punkty:
  - 2 – 5 zadań programistycznych po 1 – 5 punktów (po terminie połowa punktów)
- Oceny:
  - 50 + % punktów - 3
  - 60 + % punktów - 3.5
  - 70 + % punktów - 4
  - 80 + % punktów - 4.5
  - 90 + % punktów - 5

# Listy z dowiązaniem

```
STRUCT-LIST L
INIT(L) /*inicjuje zmienne*/
WRITE(L) //
LIST-INSERT(L,1)
LIST-INSERT(L,2)
WRITE(L) //2 1
LIST-INSERT(L,3)
LIST-SEARCH(L,2) //2
LIST-SEARCH(L,4) //-1
LIST-INSERT-AFTER(L,4,2)
WRITE(L) //3 2 4 1
LIST-DELETE(L,2)
WRITE(L) //3 4 1
LIST-DELETE(L,5)
WRITE(L) //3 4 1
LIST-INSERT-BEFORE(L,5,3)
```

## Listy z dowiązaniem

```
WRITE(L) //5 3 4 1  
CLEAR(L) /*zwalnia pamięć*/  
WRITE(L) //
```

# Listy z dowiązaniemami - funkcje i procedury

```
STRUCT-LIST
```

```
  head = NIL
```

```
  tail = NIL
```

```
STRUCT-NODE
```

```
  key = NIL
```

```
  next = NIL
```

```
  prev = NIL
```

```
LIST-INSERT(L, k)
```

```
LIST-SEARCH(L,k)
```

```
LIST-INSERT-AFTER(L, j, k)
```

```
LIST-INSERT-BEFORE(L, j, k)
```

```
LIST-DELETE(L,k)
```



# Zadania

Napisz algorytm który:

- 1 Stos za pomocą listy
- 2 Kolejka za pomocą listy

## Stos

```
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
```

# Kolejka

```
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
```

```
  List data
```

```
  top = 0
```

```
STACK-EMPTY(S)
```

```
PUSH(S,k)
```

```
POP(S)
```

# Kolejka - funkcje i procedury

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