

# Algotmy i struktury danych - Haszowanie

Marcin Żurowski

1 lutego 2021

# Adresowanie bezpośrednie

```
D-A-SEARCH(T, k)
```

```
  return T[k]
```

```
D-A-INSERT(T, x)
```

```
  T[x.key] = x
```

```
D-A-DELETE(T, x)
```

```
  T[x.key] = NIL
```

# Tablice z haszowaniem

U - zbiór kluczy

m - wielkość tablicy T

$T[0, \dots, m - 1]$

$T[i] = \text{NIL}$

$h: U \rightarrow 0, m - 1$

H-SEARCH(T, k)

    return  $T[h(k)]$

H-INSERT(T, x)

$T[h(x.\text{key})] = x$

H-DELETE(T, x)

$T[h(x.\text{key})] = \text{NIL}$

# Haszowanie modularne

```
h(p)  
  return p MOD m
```

# Haszowanie przez mnożenie

$0 < A < 1$

$A = (\text{sqrt}(5) - 1) / 2$

$h(k)$

return  $\lfloor m(kA \text{ MOD } 1) \rfloor$

# Metoda łańcuchowa

```
T[0..9]
INIT(T)
CH-H-INSERT(T,206)
CH-H-INSERT(T,142)
CH-H-INSERT(T,303)
WRITE(T) //0 0 142 303 0 0 206 0 0 0
CH-H-INSERT(T,213)
WRITE(T) //0 0 142 213->303 0 0 206 0 0 0
WRITE(CH-H-SEARCH(T,303)) //303
CH-H-DELETE(T,303)
WRITE(T) //0 0 142 213 0 0 206 0 0 0
WRITE(CH-H-SEARCH(T,213)) //213
CLEAR(T) /*zwalnia pamięć*/
WRITE(T) //0 0 0 0 0 0 0 0 0 0
```

# Metoda łańcuchowa - funkcje i procedury

CH-H-SEARCH(T, k)

CH-H-INSERT(T, k)

CH-H-DELETE(T, k)

## Adresowanie otwarte

```
T[0..9]
OP-H-INSERT(T,206)
OP-H-INSERT(T,142)
OP-H-INSERT(T,303)
WRITE(T) //0 0 142 303 0 0 206 0 0 0
OP-H-INSERT(T,213)
WRITE(T) //0 0 142 303 213 0 206 0 0 0
WRITE(OP-H-SEARCH(T,303)) //303
OP-H-DELETE(T,303)
WRITE(T) //0 0 142 DEL 213 0 206 0 0 0
WRITE(OP-H-SEARCH(T,213)) //213
```



# Adresowanie otwarte - funkcje i procedury

OP-H-SEARCH(T, k)

OP-H-INSERT(T, k)

OP-H-DELETE(T, k)

```
h(k, i)
  return (h'(k) + i) MOD m
```

## Adresowanie kwadratowe

```
h(k, i)
  return (h'(k) + c1i + c2i2) MOD m
```

# Adresowanie dwukrotne

```
h(k,i)
  return (h1(k) + ih2(k)) MOD m
```