

AuD Übung 03

[PDF](#)

5

1

a)

Algorithm delete Array

Input: Sequence S , index p

Output: Sequence S

```
function DELETE( $S, p$ )  
    while  $p < \text{len}(S) - 1$  do  
         $\text{setContent}(\alpha_S(p), \text{content}(\alpha_S(p) + \text{size}(T)))$   
         $p \leftarrow p + 1$   
    end while  
     $\text{len}_S \leftarrow \text{len}_S - 1$   
    return  $S$   
end function
```

Algorithm get Array

Input: Sequence S , index p

Output: Element e

```
function GET( $S, p$ )  
    return  $\text{content}(\alpha_S(p))$   
end function
```

Algorithm concat Array

Input: Sequence S , Sequence L

Output: Sequence res

```
function CONCAT( $S, L$ )  
     $a \leftarrow \text{eine geeignete Adresse}$   
     $res \leftarrow \text{empty}(a)$   
     $\text{len}_{res} \leftarrow \text{len}_S + \text{len}_L$   
     $c \leftarrow 0$   
    for all  $j$  in  $S$  do  
         $\text{setContent}(\alpha_S(c), j)$   
         $c \leftarrow c + 1$   
    end for  
    for all  $j$  in  $L$  do  
         $\text{setContent}(\alpha_L(c), j)$   
         $c \leftarrow c + 1$   
    end for  
    return  $res$   
end function
```

2

```
type D-listElement =  
    sorts:  
        T, p, le  
  
    functions:  
        new: T -> le  
        getValue: le -> T  
        setValue: le x T -> le  
        getNext: le -> p  
        setNext: le x p -> le  
        getPrev: le -> p  
        setPrev: le x p -> le  
  
end.
```

T: I(T) Datentyp des Grundtypen
I(p) = {x|x ∈ allokiertes Speicher}
I(le) = I(T) × I(p) × I(p)

3

insert

Algorithm Insert Element

```
function INSERT(list, element, position)  
    newNode ← new(element)  
    if position = 0 then  
        newNode.setNext(list.head)  
        if list.head ≠ null then  
            list.head.setPrev(newNode)  
        end if  
        list.head ← newNode  
    else  
        current ← list.head  
        for i ← 1 to position − 1 do  
            current ← current.getNext()  
        end for  
        newNode.setNext(current.getNext())  
        newNode.setPrev(current)  
        if current.getNext() ≠ null then  
            current.getNext().setPrev(newNode)  
        end if  
        current.setNext(newNode)  
    end if
```

end function

delete

Algorithm Delete Element

Input: nicht leere DLL *list*, index *position*

Output: DLL *list* without elm at position

```
function DELETE(list, position)
  if position = 0 then
    temp  $\leftarrow$  list.head
    list.head  $\leftarrow$  list.head.getNext()
    if list.head  $\neq$  null then
      list.head.setPrev(null)
    end if
  else
    current  $\leftarrow$  list.head
    for i  $\leftarrow$  1 to position do
      current  $\leftarrow$  current.getNext()
    end for
    current.getPrev().setNext(current.getNext())
    if current.getNext()  $\neq$  null then
      current.getNext().setPrev(current.getPrev())
    end if
  end if
end function
```

concat

Algorithm Concat Lists

Input: DLL *list1*, DLL *list2*

Output: DLL *list* concatenated *list1* and *list2*

```
function CONCAT(list1, list2)
  if list1.tail  $\neq$  null then
    list1.tail.setNext(list2.head)
  end if
  if list2.head  $\neq$  null then
    list2.head.setPrev(list1.tail)
  end if
  list1.tail  $\leftarrow$  list2.tail
  return list1
end function
```
