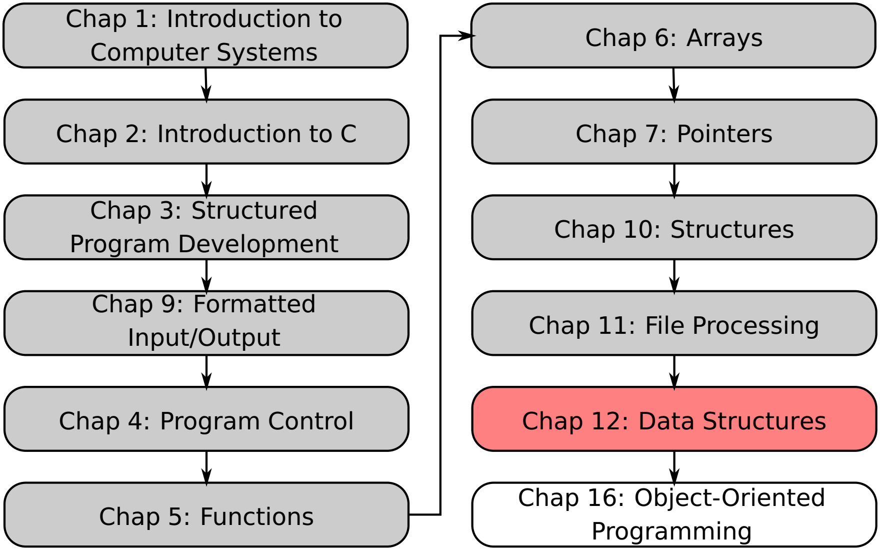
Computer Programming 143 – Lecture 29
Dynamic Data Structures II

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Module Overview


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

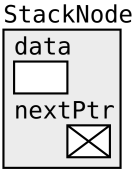
Lecture Overview

12.5 Stacks

Stacks

- Constrained version of a linked list
  - Is a row/line of self-referential structures
- Elements may only be inserted or deleted from the top (front) of a stack
  - A stack is therefore a last-in-first-out structure (LIFO)

```
struct stackNode {
    int data;
    struct stackNode *nextPtr;
};
typedef struct stackNode StackNode;
```

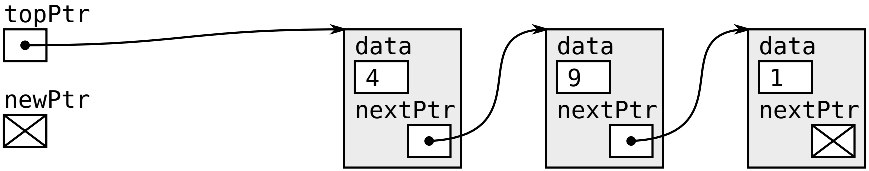


Use of stacks

- Function calls – store of return address and automatic local variables
- Used by compilers when evaluating expressions

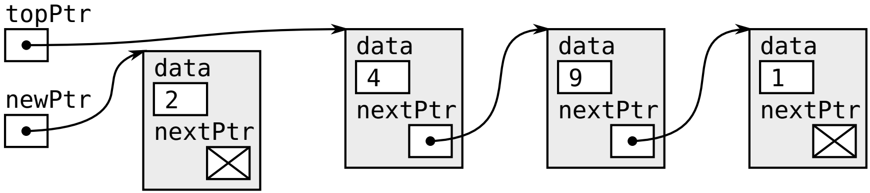
12.5 Insertion in Stacks (“Push”)

```
newPtr = malloc( sizeof( StackNode ) );
newPtr->data = value;
newPtr->nextPtr = topPtr;
topPtr = newPtr;
```



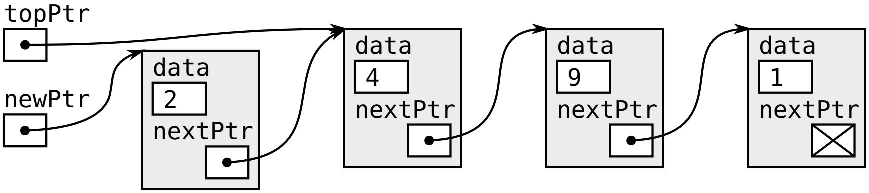
12.5 Insertion in Stacks (“Push”)

```
newPtr = malloc( sizeof( StackNode ) );
newPtr->data = value;
newPtr->nextPtr = topPtr;
topPtr = newPtr;
```



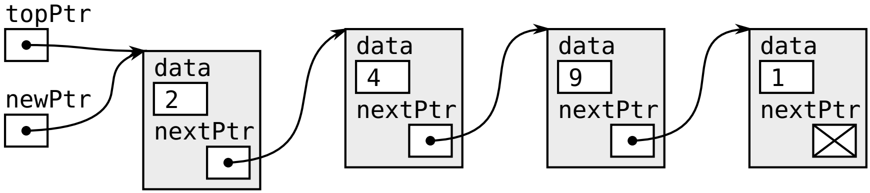
12.5 Insertion in Stacks (“Push”)

```
newPtr = malloc( sizeof( StackNode ) );
newPtr->data = value;
newPtr->nextPtr = topPtr;
topPtr = newPtr;
```



12.5 Insertion in Stacks (“Push”)

```
newPtr = malloc( sizeof( StackNode ) );
newPtr->data = value;
newPtr->nextPtr = topPtr;
topPtr = newPtr;
```



12.5 Deletion in Stacks (“Pop”)

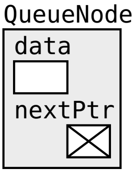
```
\\ Class exercise / Klasoefening















\\
```

12.6 Queues

Queues

- Constrained version of a linked list
  - Is a row/line of self-referential structures
- Elements may only be deleted from the front (head) of a stack and inserted at the back (tail)
  - A queue is therefore a first-in-first-out structure (FIFO)

```
struct queueNode {
    int data;
    struct queueNode *nextPtr;
};
typedef struct queueNode QueueNode;
```



Use of Queues

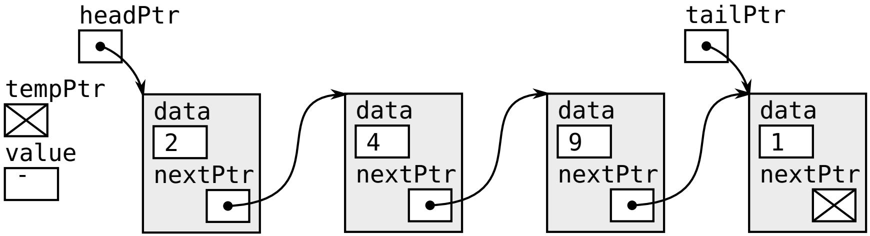
- Supporting print spooling
- Routing of data packets in computer systems
- Buffers

12.6 Insertion in Queues (“Enqueue”)

```
\\ Class exercise / Klasoefening











\\
```

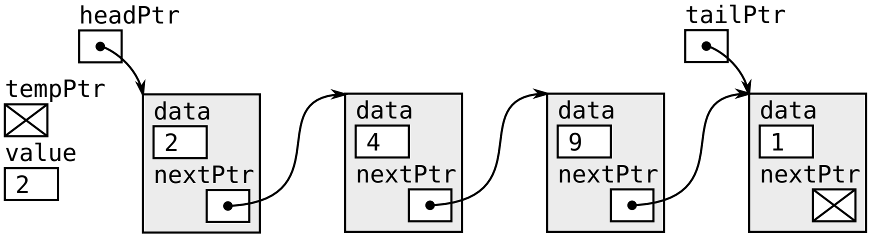
12.6 Deletion in Queues (“Dequeue”)

```
value = headPtr->data;
tempPtr = headPtr;
headPtr = headPtr->nextPtr;
free( tempPtr );
```



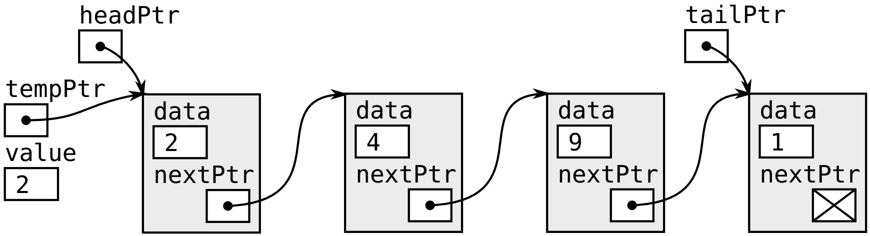
12.6 Deletion in Queues ("Dequeue")

```
value = headPtr->data;
tempPtr = headPtr;
headPtr = headPtr->nextPtr;
free( tempPtr );
```



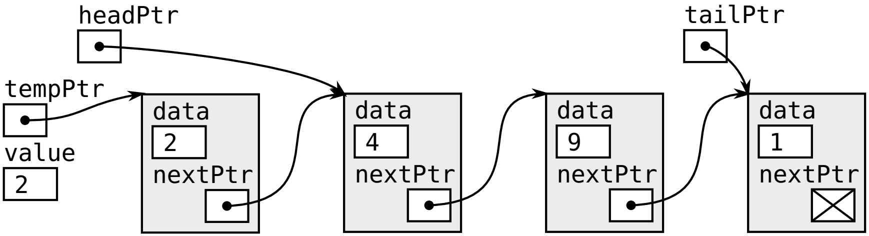
12.6 Deletion in Queues ("Dequeue")

```
value = headPtr->data;
tempPtr = headPtr;
headPtr = headPtr->nextPtr;
free( tempPtr );
```



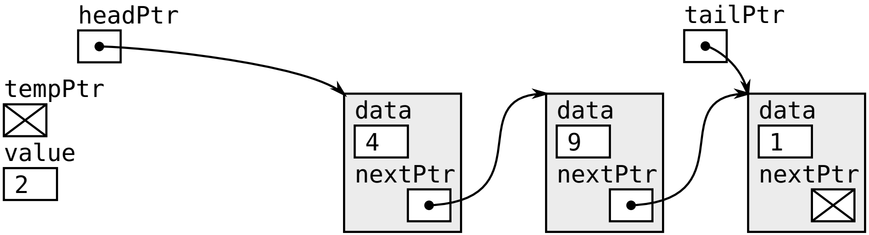
12.6 Deletion in Queues (“Dequeue”)

```
value = headPtr->data;
tempPtr = headPtr;
headPtr = headPtr->nextPtr;
free( tempPtr );
```



12.6 Deletion in Queues (“Dequeue”)

```
value = headPtr->data;
tempPtr = headPtr;
headPtr = headPtr->nextPtr;
free( tempPtr );
```



Perspective

Today

Dynamic Data Structures II

- Stacks
- Queues

Next lecture

Dynamic Data Structures III

- Trees

Homework

1. Study Sections 12.5-12.6 in Deitel & Deitel
2. Do Self Review Exercises 12.2, 12.3 in Deitel & Deitel
3. Do Exercise 12.11 in Deitel & Deitel