## INFORMAL SPEC

### Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**



## INFORMAL SPEC

### Integer Queue with Capacity = 12

**(1) SYNTAX**

OUTPUT VARIABLES

| Variable Name | Type |
|---|---|
| front | <integer> |

ACCESS PROGRAMS

| Program Name | Arg#1 | Value |
|---|---|---|
| Q12INIT | | |
| ADD | <integer>:V | |
| REMOVE | | |
| FRONT | | <integer> |

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**

## INFORMAL SPEC

### Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

canonical(T) $\leftrightarrow$ T is Q12INIT followed by a string of zero to 12 ADD operations

**(3) EQUIVALENCES**

**(4) VALUES**



## INFORMAL SPEC

### Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

T.Q12INIT $\equiv$ an initialized queue.

T.ADD(a) $\equiv$

| conditions | equivalences |
|---|---|
| 12 elements in queue | %full% |
| Queue not initialized | %NoInit% |
| Normal queue | Add one element to trace |

T.REMOVE $\equiv$

| conditions | equivalences |
|---|---|
| uninitialized | %NoInit% |
| empty | %NoElement% |
| non-empty | remove left most ADD |

T.FRONT $\equiv$

| conditions | equivalences |
|---|---|
| uninitialized queue | %NoInit% |
| empty but initialized | %NoElement% |
| non-empty | unchanged |

**(4) VALUES**

## INFORMAL SPEC

### Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**

OUTPUT VALUES
V[front](T) =

| conditions | values |
|---|---|
| uninitialized | %NoInit% |
| empty but initialized | %NoElement% |
| non-empty | first element in queue |

RETURN VALUES

| Program Name | Argument No | Values |
|---|---|---|
| FRONT | Value | front |



## INFORMAL SPEC

### Integer Queue with Capacity = 12

TYPE IMPLEMENTED: <queue12>

**(1) SYNTAX**

OUTPUT VARIABLES

| Variable Name | Type |
|---|---|
| front | <integer> |

ACCESS PROGRAMS

| Program Name | Arg#1 | Value |
|---|---|---|
| Q12INIT | | |
| ADD | <integer>:V | |
| REMOVE | | |
| FRONT | | <integer> |

**(2) CANONICAL TRACES**

canonical(T) $\leftrightarrow$ T is Q12INIT followed by a string of zero to 12 ADD operations

**(3) EQUIVALENCES**

T.Q12INIT $\equiv$ an initialized queue.
T.ADD(a) $\equiv$

| conditions | equivalences |
|---|---|
| 12 elements in queue | %full% |
| Queue not initialized | %NoInit% |
| Normal queue | Add one element to trace |

T.REMOVE $\equiv$

| conditions | equivalences |
|---|---|
| uninitialized | %NoInit% |
| empty | %NoElement% |
| non-empty | remove leftmost ADD |

T.FRONT ≡

| conditions | equivalences |
|---|---|
| uninitialized queue | %NoInit% |
| empty but initialized | %NoElement% |
| non-empty | unchanged |

**(4) VALUES**

OUTPUT VALUES

V[front](T) =

| conditions | values |
|---|---|
| uninitialized | %NoInit% |
| empty but initialized | %NoElement% |
| non-empty | first element in queue |

RETURN VALUES

| Program Name | Argument No | Values |
|---|---|---|
| FRONT | Value | front |



## Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**

## Integer Queue with Capacity = 12

**(1) SYNTAX**

OUTPUT VARIABLES

| Variable Name | Type |
|---|---|
| front | <integer> |

ACCESS PROGRAMS

| Program Name | Arg#1 | Value |
|---|---|---|
| Q12INIT | | |
| ADD | <integer>:V | |
| REMOVE | | |
| FRONT | | <integer> |

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**



## Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

(canonical(T) $\leftrightarrow$ ( T = Q12INIT() . $[ADD(a_i)]_{i=1}^{n}$ ) $\wedge$ $(0 \le n \le 12)$ ) $\vee$ T = _

**(3) EQUIVALENCES**

**(4) VALUES**

## Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

T.Q12INIT ≡ Q12INIT

T.ADD(a) ≡

| conditions | equivalences |
|---|---|
| count(T1, ADD) = 12 where T = Q12INIT . T1 | %full% |
| T = _ | %NoInit% |
| count(T1, ADD) < 12 where T = Q12INIT . T1 | T.ADD(a) |

T.REMOVE ≡

| conditions | equivalences |
|---|---|
| T = _ | %NoInit% |
| T = Q12INIT | %NoElement% |
| T ≠ _ ∧ T ≠ Q12INIT | Q12INIT . S1 where T = Q12INIT . ADD(a) . S1 |

T.FRONT ≡

| conditions | equivalences |
|---|---|
| T = _ | %NoInit% |
| T = Q12INIT | %NoElement% |
| T ≠ _ ∧ T ≠ Q12INIT | T |

**(4) VALUES**



## Integer Queue with Capacity = 12

**(1) SYNTAX**

**(2) CANONICAL TRACES**

**(3) EQUIVALENCES**

**(4) VALUES**

OUTPUT VALUES

V[front](T) =

| conditions | values |
|---|---|
| T = _ | a where *true*, %NoInit% |
| T = Q12INIT | a where *true*, %NoElement% |
| (T ≠ _) ∧ (T ≠ Q12INIT) | a where T = Q12INIT.ADD(a).S1 |

RETURN VALUES

| Program Name | Argument No | Values |
|---|---|---|
| FRONT | Value | front |

## Integer Queue with Capacity = 12

TYPE IMPLEMENTED: <queue12>

**(1) SYNTAX**

OUTPUT VARIABLES

| Variable Name | Type |
|---|---|
| front | <integer> |

ACCESS PROGRAMS

| Program Name | Arg#1 | Value |
|---|---|---|
| Q12INIT | | |
| ADD | <integer>:V | |
| REMOVE | | |
| FRONT | | <integer> |

**(2) CANONICAL TRACES**

$$\text{canonical}(T) \leftrightarrow (\ T = \text{Q12INIT}\ .\ [ADD(a_i)]_{i=1}^{n}\ ) \wedge\ (0 \le n \le 12)$$

**(3) EQUIVALENCES**

T.Q12INIT ≡ Q12INIT

T.ADD(a) ≡

| conditions | equivalences |
|---|---|
| count(T1, ADD) = 12 where<br>T = Q12INIT . T1 | %full% |
| T = _ | %NoInit% |
| count(T1, ADD) < 12 where<br>T = Q12INIT . T1 | T.ADD(a) |

T.REMOVE ≡

| conditions | equivalences |
|---|---|
| T = _ | %NoInit% |
| T = Q12INIT | %NoElement% |
| T ≠ _ ∧ T ≠ Q12INIT | Q12INIT . S1 where<br>T = Q12INIT . ADD(a) . S1 |

T.FRONT ≡

| conditions | equivalences |
|---|---|
| T = _ | %NoInit% |
| T = Q12INIT | %NoElement% |
| T ≠ _ ∧ T ≠ Q12INIT | T |



**(4) VALUES**

OUTPUT VALUES

V[front](T) =

| conditions | values |
|---|---|
| T = _ | a where *true*, %NoInit% |
| T = Q12INIT | a where *true*, %NoElement% |
| (T ≠ _) ∧ (T ≠ Q12INIT) | a where T = Q12INIT.ADD(a).S1 |

RETURN VALUES

| Program Name | Argument No | Values |
|---|---|---|
| FRONT | Value | front |

## Informal Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**



## Informal Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of integer |

VARIABLES

| Type Definition/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | "Don't Care" |
| 0..QSIZE-1 | F, R | "Don't Care" |
| <boolean> | FULL | "Don't Care" |
| <boolean> | old | false |

Abbreviation:

**<qs>** the set of possible values of the data structure.

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

## Informal Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

af: maps from the data structures, qs, to abstract traces.
af(DATA,F,R,FULL) $\stackrel{df}{=}$

| | |
|---|---|
| initialised, not empty and no wraparound | Queue data stored between F and R in DATA |
| initialised, not empty with wraparound | Queue data stored from F to 0 and then from QSIZE-1 to R |
| initialised, empty | No data in queue |
| not initialised | Not yet a queue |

**(3) PROGRAM FUNCTIONS**



## Informal Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

| pf_Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | changes the data structure |
| gpf_ADD | <integer> | changes the data structures based on an integer and old value |
| pf_REMOVE | | changes the data structure |
| pf_FRONT | | computes integer and changes the data structure |

pf_Q12INIT $\stackrel{df}{=}$

| | |
|---|---|
| F' = | 0 |
| R' = | 1 |
| FULL' = | false |
| DATA' \| | true |
| old = | true |

gpf_ADD(a) $\stackrel{df}{=}$ F won't change and only DATA at new R can change <u>and</u>

| | Must wraparound and initialised | | | Not a wraparound case and initialised | | | Not initialised |
|---|---|---|---|---|---|---|---|
| | edge case | | not edge case | edge case | | not edge case | |
| | ‘FULL | ¬ ‘FULL | | ‘FULL | ¬ ‘FULL | | |
| DATA'[R'] = | ‘DATA[‘R] | a | a | ‘DATA[‘R] | a | a | ‘DATA[‘R] |
| R' = | ‘R | QSIZE-1 | QSIZE-1 | ‘R | ‘R – 1 | ‘R – 1 | ‘R |
| FULL' = | ‘FULL | false | ‘F = QSIZE-2 | ‘FULL | false | *edge*' | ‘FULL |

pf_REMOVE $\stackrel{df}{=}$ No change to DATA, old, or R <u>and</u>

| | initialised and not empty | | empty or not initialised |
|---|---|---|---|
| | (‘F = 0) | (‘F > 0) | |
| F' = | QSIZE-1 | ‘F – 1 | ‘F |
| FULL' = | false | false | ‘FULL |

pf_FRONT $\stackrel{df}{=}$ NC(R,FULL, DATA, F) <u>and</u>

| | not empty and initialised | empty or not initialised |
|---|---|---|
| return | ‘DATA[‘F] | |

## Informal Design Document for Queue12: Implementation 1 - Pascal

### (1) DATA STRUCTURE

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of integer |

VARIABLES

| Type Definition/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | “Don’t Care” |
| 0..QSIZE-1 | F, R | “Don’t Care” |
| <boolean> | FULL | “Don’t Care” |
| <boolean> | old | false |

Abbreviation:

a predicate edge is true if the rear is just behind the front or front is at the top of the array and the rear is at the first array element.

**<qs>** the set of possible values of the data structure.

### (2) ABSTRACTION FUNCTION

af: maps from the data structures, qs, to abstract traces.
af(DATA,F,R,FULL) $\stackrel{df}{=}$

| | |
|---|---|
| initialised, not empty and no wraparound | Queue data stored between F and R in DATA |
| initialised, not empty with wraparound | Queue data stored from F to 0 and then from QSIZE-1 to R |
| initialised, empty | No data in queue |
| not initialised | Not yet a queue |



### (3) PROGRAM FUNCTIONS

| pf_Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | changes the data structure |
| gpf_ADD | <integer> | changes the data structures based on an integer and old value |
| pf_REMOVE | | changes the data structure |
| pf_FRONT | | computes integer and changes the data structure |

pf_Q12INIT d

| | |
|---|---|
| F’ = | 0 |
| R’ = | 1 |
| FULL’ = | false |
| DATA’ \| | true |
| old = | true |

gpf_ADD(a) $\stackrel{df}{=}$ F won’t change and only DATA at new R can change

| | Must wraparound, initialised | | | Not a wraparound case, initialised | | | Not initialised |
|---|---|---|---|---|---|---|---|
| | edge case | | not edge case | edge case | | not edge case | |
| | ‘FULL | ¬ ‘FULL | | ‘FULL | ¬ ‘FULL | | |
| DATA’[R’] = | ‘DATA[‘R] | a | a | ‘DATA[‘R] | a | a | ‘DATA[‘R] |
| R’ = | ‘R | QSIZE-1 | QSIZE-1 | ‘R | ‘R – 1 | ‘R – 1 | ‘R |
| FULL’ = | ‘FULL | false | ‘F = QSIZE-2 | ‘FULL | false | *edge*’ | ‘FULL |

pf_REMOVE $\stackrel{df}{=}$ No change to DATA, old, or R and

| | initialised and not empty | | empty or not initialised |
|---|---|---|---|
| | (‘F = 0) | (‘F > 0) | |
| F’ = | QSIZE-1 | ‘F – 1 | ‘F |
| FULL’ = | false | false | ‘FULL |

pf_FRONT $\stackrel{df}{=}$ NC(R,FULL) ∧

| | not empty and initialised | empty or not initialised |
|---|---|---|
| return | ‘DATA[‘F] | |

## Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**



## Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of integer |

VARIABLES

| Type Definition/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | “Don’t Care” |
| 0..QSIZE-1 | F, R | “Don’t Care” |
| <boolean> | FULL | “Don’t Care” |
| <boolean> | old | false |

Abbreviation:

$edge \stackrel{df}{=} (R = F + 1) \vee (F = QSIZE\text{-}1) \wedge (R = 0)$
$`edge \stackrel{df}{=} (`R = `F + 1) \vee (`F = QSIZE\text{-}1) \wedge (`R = 0)$
$edge' \stackrel{df}{=} (R' = F' + 1) \vee (F' = QSIZE\text{-}1) \wedge (R' = 0)$

**<qs>** $\stackrel{df}{=}$ qds $\times$ 0..QSIZE-1 $\times$ 0..QSIZE-1 $\times$ boolean $\times$ boolean

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

## Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

af: <qs> → <queue12>
af(DATA,F,R,FULL) ≝

| | |
|---|---|
| (¬ *edge* ∨ FULL) ∧ (F ≥ R) ∧ old | Q12INIT.ADD(DATA[F]).ADD(DATA[F–1]). … .ADD(DATA[R]) |
| (¬ *edge* ∨ FULL) ∧ (F < R) ∧ old | Q12INIT . ADD(DATA[F]). … .ADD(DATA[0]).ADD(DATA[QSIZE-1]). … .ADD(DATA[R]) |
| *edge* ∧ ¬ FULL ∧ old | Q12 INIT. |
| ¬ old | — |

**(3) PROGRAM FUNCTIONS**



## Design Document for Queue12: Implementation 1 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

| pf_Name | Arg#1 | Value | |
|---|---|---|---|
| pf_Q12INIT | | <qs> | → <qs> |
| gpf_ADD | <integer> | <qs> × <integer> | → <qs> |
| pf_REMOVE | | <qs> | → <qs> |
| pf_FRONT | | <qs> | → <qs> × <integer> |

pf_Q12INIT ≝

| | |
|---|---|
| F’ = | 0 |
| R’ = | 1 |
| FULL’ = | false |
| DATA’ \| | true |
| old = | true |

gpf_ADD(a) ≝ NC(F) ∧ ∀j (j ≠ R’) [NC(DATA[j])] ∧ NC(a) ∧

| | (‘R = 0) ∧ old ∧ | | | (‘R ≠ 0) ∧ old | | | ¬ old |
|---|---|---|---|---|---|---|---|
| | ‘*edge* ∧ | | ¬ ‘*edge* | ‘*edge* ∧ | | ¬ ‘*edge* | |
| | ‘FULL | ¬ ‘FULL | | ‘FULL | ¬ ‘FULL | | |
| DATA’[R’] = | ‘DATA[‘R] | a | a | ‘DATA[‘R] | a | a | ‘DATA[‘R] |
| R’ = | ‘R | QSIZE-1 | QSIZE-1 | ‘R | ‘R – 1 | ‘R – 1 | ‘R |
| FULL’ = | ‘FULL | QSIZE=1 | ‘F = QSIZE-2 | ‘FULL | QSIZE=1 | *edge*’ | ‘FULL |

pf_REMOVE ≝ NC(DATA,R) ∧

| | (¬ ‘*edge* ∨ ‘FULL) ∧ old ∧ | | (‘*edge* ∧ ¬‘FULL) ∨ ¬ old |
|---|---|---|---|
| | (‘F = 0) | (‘F > 0) | |
| F’ = | QSIZE-1 | ‘F – 1 | ‘F |
| FULL’ = | false | false | ‘FULL |

pf_FRONT ≝ NC(R,FULL, DATA, F) ∧

| | (¬ ‘*edge* ∨ ‘FULL) ∧ old | (‘*edge* ∧ ¬‘FULL) ∨ ¬ old |
|---|---|---|
| DATA’ = | ‘DATA | ‘DATA |
| F’ = | ‘F | ‘F |
| return | ‘DATA[‘F] | |

## Design Document for Queue12: Implementation 1 - Pascal

### (1) DATA STRUCTURE

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of integer |

VARIABLES

| Type Definition/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | "Don't Care" |
| 0..QSIZE-1 | F, R | "Don't Care" |
| <boolean> | FULL | "Don't Care" |
| <boolean> | old | false |

Abbreviation:

*edge* $\stackrel{df}{=}$ (R = F + 1) ∨ (F = QSIZE-1) ∧ (R = 0)
'*edge* $\stackrel{df}{=}$ ('R = 'F + 1) ∨ ('F = QSIZE-1) ∧ ('R = 0)
*edge'* $\stackrel{df}{=}$ (R' = F' + 1) ∨ (F' = QSIZE-1) ∧ (R' = 0)
**<qs>** $\stackrel{df}{=}$ qds × 0..QSIZE-1 × 0..QSIZE-1 × boolean

### (2) ABSTRACTION FUNCTION

af: <qs> → <queue12>

af(DATA,F,R,FULL,old) $\stackrel{df}{=}$

| | |
|---|---|
| (¬ *edge* ∨ FULL) ∧ (F ≥ R) ∧ old | Q12INIT.ADD(DATA[F]).ADD(DATA[F–1]). … .ADD(DATA[R]) |
| (¬ *edge* ∨ FULL) ∧ (F < R) ∧ old | Q12INIT . ADD(DATA[F]). … .ADD(DATA[0]).ADD(DATA[QSIZE-1]). … .ADD(DATA[R]) |
| *edge* ∧ ¬ FULL ∧ old | Q12INIT |
| ¬ old | |



### (3) PROGRAM FUNCTIONS

| pf_Name | Arg#1 | Value |
|---|---|---|
| pf_Q12INIT | | <qs> → <qs> |
| gpf_ADD | <integer> | <qs> × <integer> → <qs> |
| pf_REMOVE | | <qs> → <qs> |
| pf_FRONT | | <qs> → <qs> × <integer> |

pf_Q12INIT $\stackrel{df}{=}$

| | |
|---|---|
| F' = | 0 |
| R' = | 1 |
| FULL' = | false |
| DATA' \| | true |
| old = | true |

gpf_ADD(a) $\stackrel{df}{=}$ NC(F) ∧ ∀j (j ≠ R') [NC(DATA[j])] ∧ NC(a) ∧

| | ('R = 0) ∧ old ∧ | | | ('R ≠ 0) ∧ old ∧ | | | ¬ old |
|---|---|---|---|---|---|---|---|
| | '*edge* ∧ | | ¬ '*edge* | '*edge* ∧ | | ¬ '*edge* | |
| | 'FULL | ¬ 'FULL | | 'FULL | ¬ 'FULL | | |
| DATA'[R'] = | 'DATA['R] | a | a | 'DATA['R] | a | a | 'DATA['R] |
| R' = | 'R | QSIZE-1 | QSIZE-1 | 'R | 'R – 1 | 'R – 1 | 'R |
| FULL' = | 'FULL | false | 'F = QSIZE-2 | 'FULL | false | *edge*' | 'FULL |

pf_REMOVE $\stackrel{df}{=}$ NC(DATA,R) ∧

| | (¬ '*edge* ∨ 'FULL) ∧ old ∧ | | ('*edge* ∧ ¬'FULL) ∨¬ old |
|---|---|---|---|
| | ('F = 0) | ('F > 0) | |
| F' = | QSIZE-1 | 'F – 1 | 'F |
| FULL' = | false | false | 'FULL |

pf_FRONT $\stackrel{df}{=}$ NC(R,FULL, DATA, F) ∧

| | ¬ '*edge* ∨ 'FULL old ∧ | ('*edge* ∧ ¬'FULL) ∨ ¬ old |
|---|---|---|
| return value = | 'DATA['F] | |

## Informal Design Document for Queue12: Implementation 2 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**



## Informal Design Document for Queue12: Implementation 2 - Pascal

**(1) DATA STRUCTURE**

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of <integer> |

VARIABLES

| Type Definitions/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | “Don’t Care” |
| -1 .. QSIZE | Size | -1 |

Abbreviation

**<qs>** $\stackrel{\text{df}}{=}$ qds $\times$ 0..QSIZE-1

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

### (1) DATA STRUCTURE

### (2) ABSTRACTION FUNCTION

af:<qs> → <queue12>
af(DATA,Size) $\stackrel{df}{=}$

| | |
|---|---|
| Size > 0 | initialised queue with newest element in DATA[0] |
| Size = 0 | Q12INIT |
| Size< 0 | — |

### (3) PROGRAM FUNCTIONS





### (1) DATA STRUCTURE

### (2) ABSTRACTION FUNCTION

### (3) PROGRAM FUNCTIONS

| Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | changes the data structure |
| gpf_ADD (a) | <integer> | changes data structure depending on integer |
| pf_REMOVE | | changes data structure |
| pf_FRONT | | changes data structure and returns integer |

pf_Q12INIT $\stackrel{df}{=}$ Set Size to zero.

gpf_ADD(a) $\stackrel{df}{=}$ NC(a) ∧

| | Queue full | Queue not full |
|---|---|---|
| DATA' \| | NC(DATA) | shift data up and insert a at DATA[0]. |
| Size' = | ‘Size | ‘Size + 1 |

pf_REMOVE $\stackrel{df}{=}$ NC(DATA) ∧

| | Queue empty | Queue not empty |
|---|---|---|
| Size' = | ‘Size | ‘Size - 1 |

pf_FRONT $\stackrel{df}{=}$ NC(DATA,Size) ∧ return = ‘DATA[Size-1]

## Informal Design Document for Queue12: Implementation 2 - Pascal

### (1) DATA STRUCTURE

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of <integer> |

VARIABLES

| Type Definitions/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | “Don’t Care” |
| -1 .. QSIZE | Size | -1 |

Abbreviation

**<qs>** $\overset{df}{=}$ qds $\times$ 0..QSIZE-1

### (2) ABSTRACTION FUNCTION

af:<qs> $\rightarrow$ <queue12>
af(DATA,Size) $\overset{df}{=}$

| | |
|---|---|
| Size > 0 | initialised queue with newest element in DATA[0] |
| Size = 0 | Q12INIT |
| Size< 0 | |



### (3) PROGRAM FUNCTIONS

| Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | changes the data structure |
| gpf_ADD (a) | <integer> | changes data structure depending on integer |
| pf_REMOVE | | changes data structure |
| pf_FRONT | | changes data structure and returns integer |

pf_Q12INIT $\overset{df}{=}$ Set Size to zero.
gpf_ADD(a) $\overset{df}{=}$ NC(a) $\wedge$

| | Queue full | Queue not full |
|---|---|---|
| DATA’ \| | NC(DATA) | shift data up and insert a at DATA[0]. |
| Size’ = | ‘Size | ‘Size + 1 |

pf_REMOVE $\overset{df}{=}$ NC(DATA) $\wedge$

| | Queue empty | Queue not empty |
|---|---|---|
| Size’ = | ‘Size | ‘Size - 1 |

pf_FRONT $\overset{df}{=}$ NC(DATA,Size) $\wedge$ return = ‘DATA[Size-1]

## Design Document for Queue12: Implementation 2 - Pascal

**(1) DATA STRUCTURE**

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**



## Design Document for Queue12: Implementation 2 - Pascal

**(1) DATA STRUCTURE**

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of <integer> |

VARIABLES

| Type Definitions/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | “Don’t Care” |
| -1 .. QSIZE | Size | -1 |

Abbreviation

**<qs>** $\stackrel{df}{=}$ qds $\times$ 0..QSIZE-1

**(2) ABSTRACTION FUNCTION**

**(3) PROGRAM FUNCTIONS**

## Design Document for Queue12: Implementation 2 - Pascal

### (1) DATA STRUCTURE

### (2) ABSTRACTION FUNCTION

af:<qs> → <queue12>
af(DATA,Size) $\stackrel{df}{=}$

| | |
|---|---|
| Size > 0 | Q12INIT . ADD(DATA[Size-1]). … .ADD(DATA[0]) |
| Size = 0 | Q12INIT |
| Size< 0 | — |

### (3) PROGRAM FUNCTIONS



## Design Document for Queue12: Implementation 2 - Pascal

### (1) DATA STRUCTURE

### (2) ABSTRACTION FUNCTION

### (3) PROGRAM FUNCTIONS

| Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | <qs> → <qs> |
| gpf_ADD (a) | <integer> | <qs> × <integer> → <qs> |
| pf_REMOVE | | <qs> → <qs> |
| pf_FRONT | | <qs> → <qs> × <integer> |

pf_Q12INIT $\stackrel{df}{=}$ Size' = 0

gpf_ADD(a) $\stackrel{df}{=}$ NC(a) ∧

| | 'Size = QSIZE ∨ 'Size = 1 | 'Size ≠ QSIZE ∧ Size ≥ 0 |
|---|---|---|
| DATA' | NC(DATA) | ∀j(1 ≤ j ≤ 'Size)[ DATA'[j] = 'DATA[j-1] ] ∧<br>∀j(('Size+1) ≤ j ≤ QSIZE-1) [ NC(DATA[j]) ] ∧<br>DATA[0]' = a |
| Size' = | 'Size | 'Size + 1 |

pf_REMOVE $\stackrel{df}{=}$ NC(DATA) ∧

| | 'Size ≤ 0 | 'Size > 0 |
|---|---|---|
| Size' = | 'Size | 'Size - 1 |

pf_FRONT $\stackrel{df}{=}$ NC(DATA,Size) ∧ return = 'DATA[Size-1]

## Design Document for Queue12: Implementation 2 - Pascal

**(1) DATA STRUCTURE**

CONSTANTS

| Constant Name | Definition |
|---|---|
| QSIZE | 12 |

TYPES

| Type Name | Definition |
|---|---|
| <qds> | array[0..QSIZE-1] of <integer> |

VARIABLES

| Type Definitions/Name | Variables | Initial Values |
|---|---|---|
| <qds> | DATA | "Don't Care" |
| -1 .. QSIZE | Size | -1 |

Abbreviation

**<qs>** $\stackrel{df}{=}$ qds × 0..QSIZE-1

**(2) ABSTRACTION FUNCTION**

af:<qs> → <queue12>
af(DATA,Size) $\stackrel{df}{=}$

| | |
|---|---|
| Size > 0 | Q12INIT . ADD(DATA[Size-1]). … .ADD(DATA[0]) |
| Size = 0 | Q12INIT |
| Size< 0 | |



**(3) PROGRAM FUNCTIONS**

| Name | Arg#1 | Signature |
|---|---|---|
| pf_Q12INIT | | <qs> → <qs> |
| gpf_ADD (a) | <integer> | <qs> × <integer> → <qs> |
| pf_REMOVE | | <qs> → <qs> |
| pf_FRONT | | <qs> → <qs> × <integer> |

pf_Q12INIT $\stackrel{df}{=}$ Size' = 0
gpf_ADD(a) $\stackrel{df}{=}$ NC(a) ∧

| | 'Size = QSIZE | 'Size ≠ QSIZE |
|---|---|---|
| DATA' \| | NC(DATA) | ∀j(1 ≤ j ≤ 'Size)[ DATA'[j] = 'DATA[j-1] ] ∧ ∀j(('Size+1) ≤ j ≤ QSIZE-1) [ NC(DATA[j]) ] ∧ DATA[0]' = a |
| Size' = | 'Size | 'Size + 1 |

pf_REMOVE $\stackrel{df}{=}$ NC(DATA) ∧

| | 'Size ≤ 0 | 'Size > 0 |
|---|---|---|
| Size' = | 'Size | 'Size - 1 |

pf_FRONT $\stackrel{df}{=}$ NC(DATA,Size) ∧ return = 'DATA[Size-1]