Kolejka

Autor: Marlena Hącel
Pod patronatem mgr Radosława Grzesiaka

```
TYPE  EP=^ELEMENT;
  KP=^KOLEJKA;

  ELEMENT=REKORD
       WARTOSC:BYTE;
       NASTEPNY:EP;
  END;

  KOLEJKA=REKORD
       POCZATEK:EP;
       KONIEC:EP;
  END;
```

Stworzenie typu :

EP wskazującego na rekord ELEMENT

KP wskazującego na rekord KOLEJKA

Rekord ELEMENT zawierający WARTOSC i NASTEPNY potrzebnie do dodawania kolejnych elementów do kolejki

Rekord KOLEJKA zawierający zmienne POCZATEK i KONIEC potrzebnie do tworzenia kolejki

```
PROCEDURE ENQUEUE(K
   :KP; E:EP);
BEGIN
   IF K^.POCZATEK <> NIL
   THEN
   BEGIN

         K^.KONIEC^.NASTEP
   NY := E;
         K^.KONIEC := E;
   END
   ELSE
   BEGIN
         K^.POCZATEK := E;
         K^.KONIEC := E;
   END;
END;
```

Procedura ENQUEUE (dodawanie elementów do kolejki)

K: kolejka, E: element

Jeśli początek kolejki jest różny od NIL (zera) wtedy

Końcowi kolejki wskazujący na NASTEPNY przypisujemy E ;

Koniec kolejki przyjmuje wartość zmiennej E

W przeciwnym razie

Kolejka wskazująca na początek przyjmuje wartość zmiennej E oraz Kolejka wskazująca na kolec przyjmuje wartość zmiennej E

```
FUNCTION DEQUEUE(K
  :KP):EP;
VAR TEMP:EP;
  IF K^.POCZATEK <>
  NIL THEN
  BEGIN

      TEMP:=K^.POCZAT
  EK;
      K^.POCZATEK :=
  TEMP^.NASTEPNY;
      TEMP^.NASTEPNY
  := NIL;
      DEQUEUE := TEMP
  END;
  ELSE
      DEQUEUE := NIL;
END;
```

Funkcja DEQUEUE (zdejmowanie elementu z kolejki)

Zmienna pomocnicza TEMP Typu EP

Jeśli Początek kolejki jest różny od NIL (zera) wtedy

TEMP staje się początkiem kolejki ;

Początek kolejki staje się Następy TEMP

Następny TEMP staje się NILEM

Zdejmowanie TEMP z kolejki

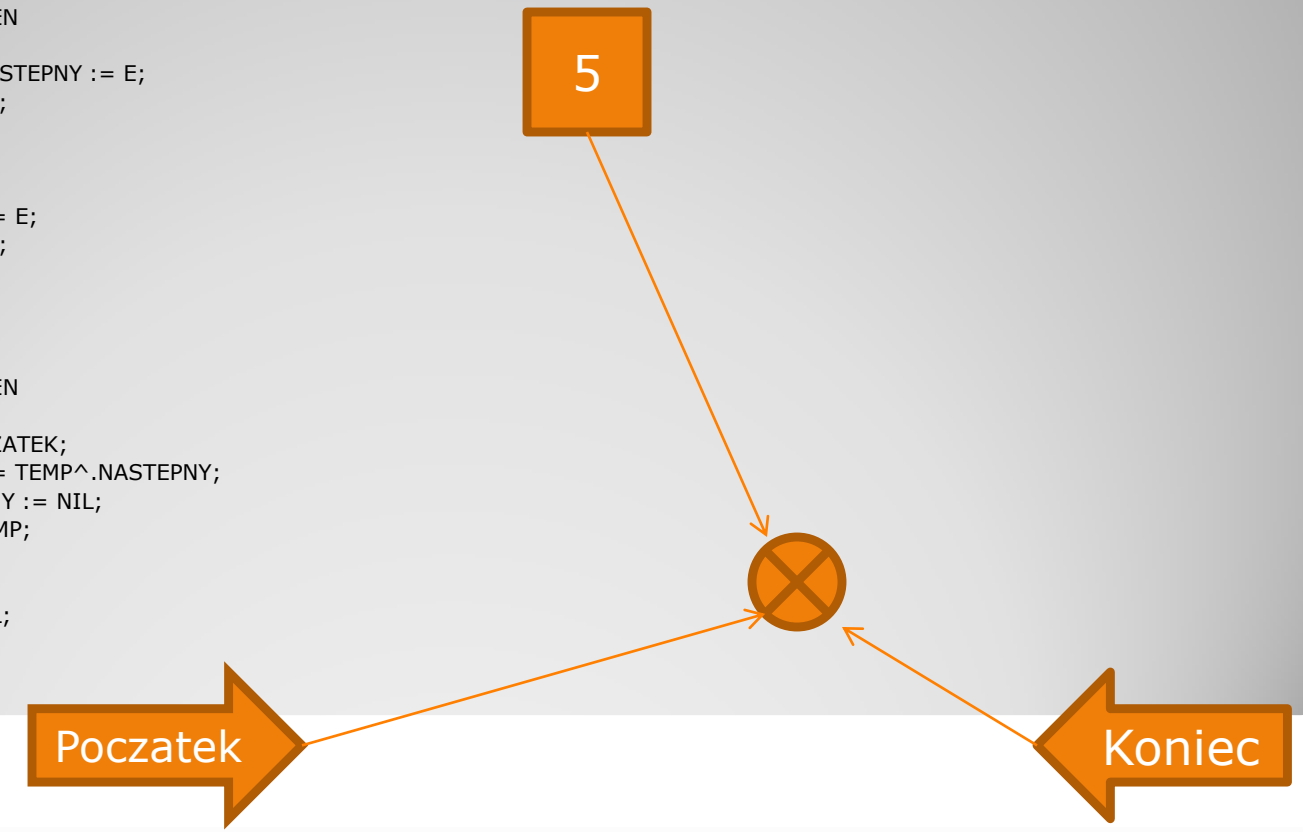
W przeciwnym wypadku

DEQUEUE przyjmuje wartość ZERA

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 5

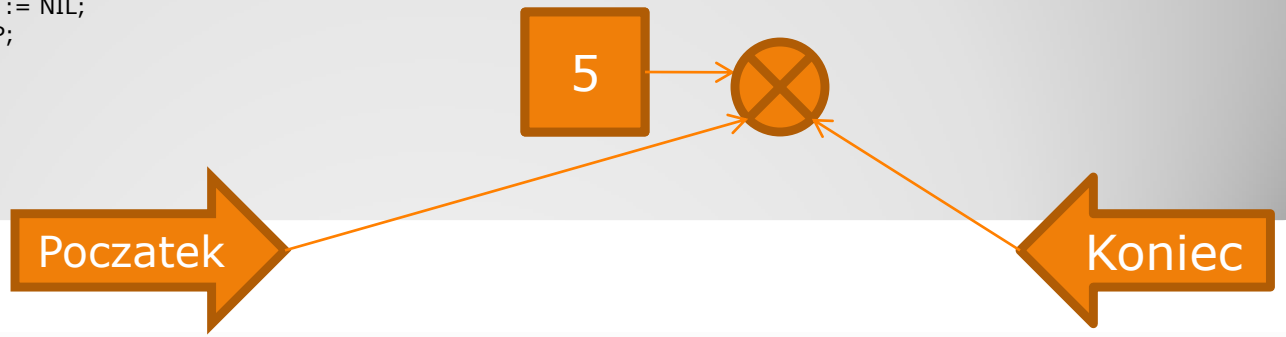


-NIL

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

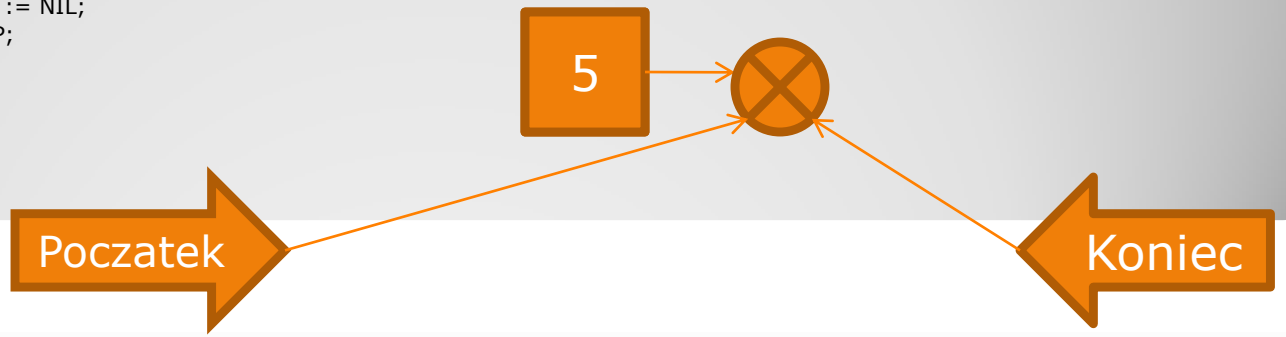
- Tworzymy nowy element z wartością 5
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

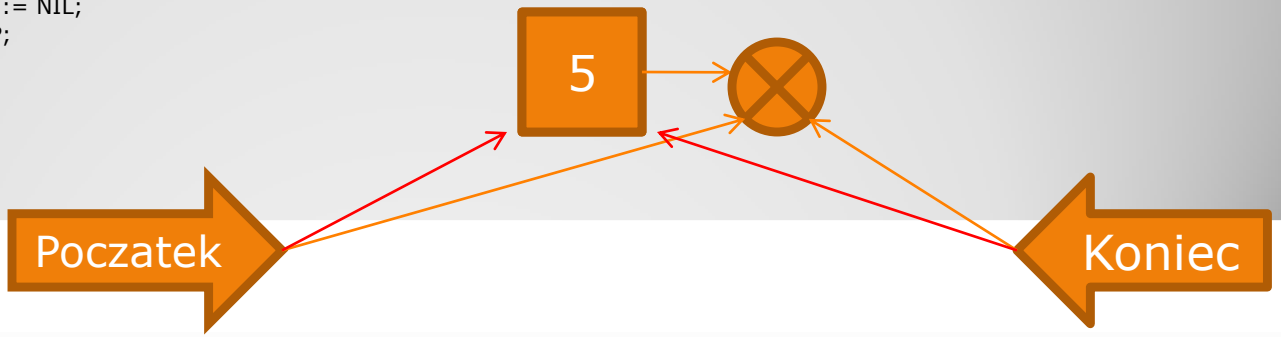
- Tworzymy nowy element z wartością 5
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

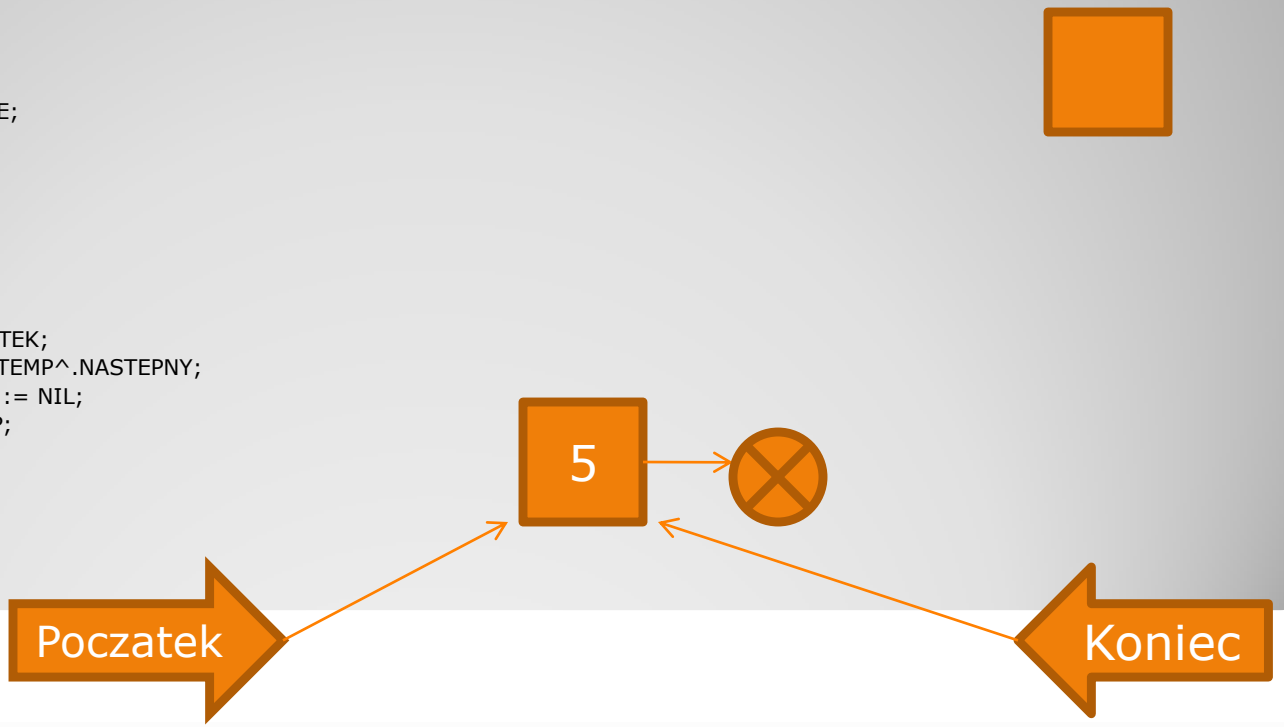
- Tworzymy nowy element z wartością 5
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- Kolejka jest pusta
- Element trafia na początek kolejki który jest zarazem jej końcem



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

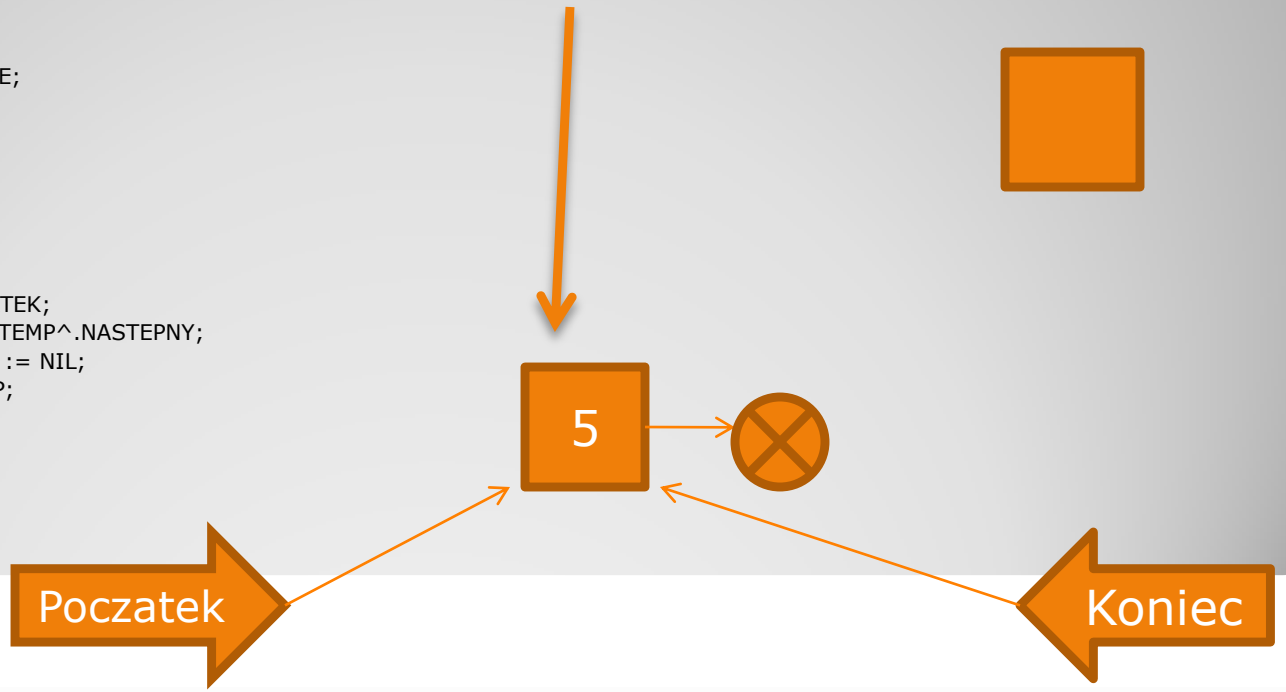
- Tworzymy nowy element z wartością 12
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 12
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już pierwszy element



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                BEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 12
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już pierwszy element
- Element staje się następnym elementem kolejki a zarazem jej nowym końcem.

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

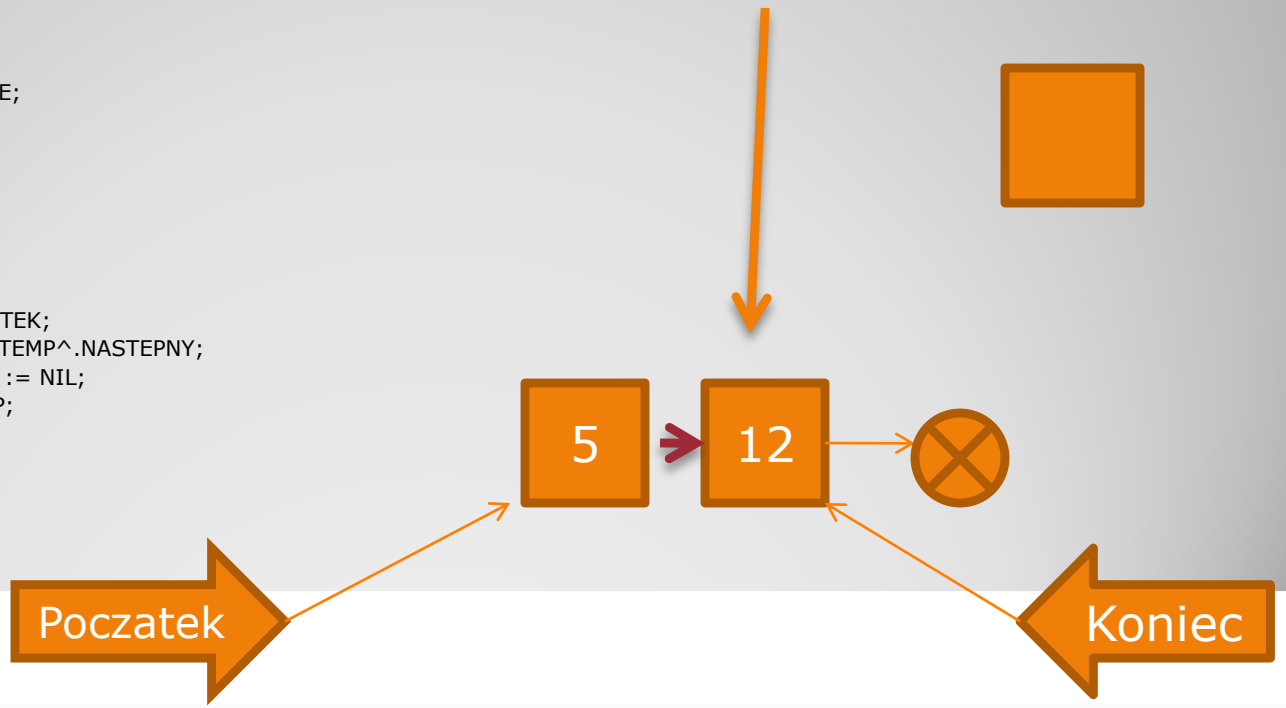
```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 23
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 23
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już kolejny element



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 23
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już pierwszy element
- Element staje się następnym elementem kolejki a zarazem jej nowym końcem.

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

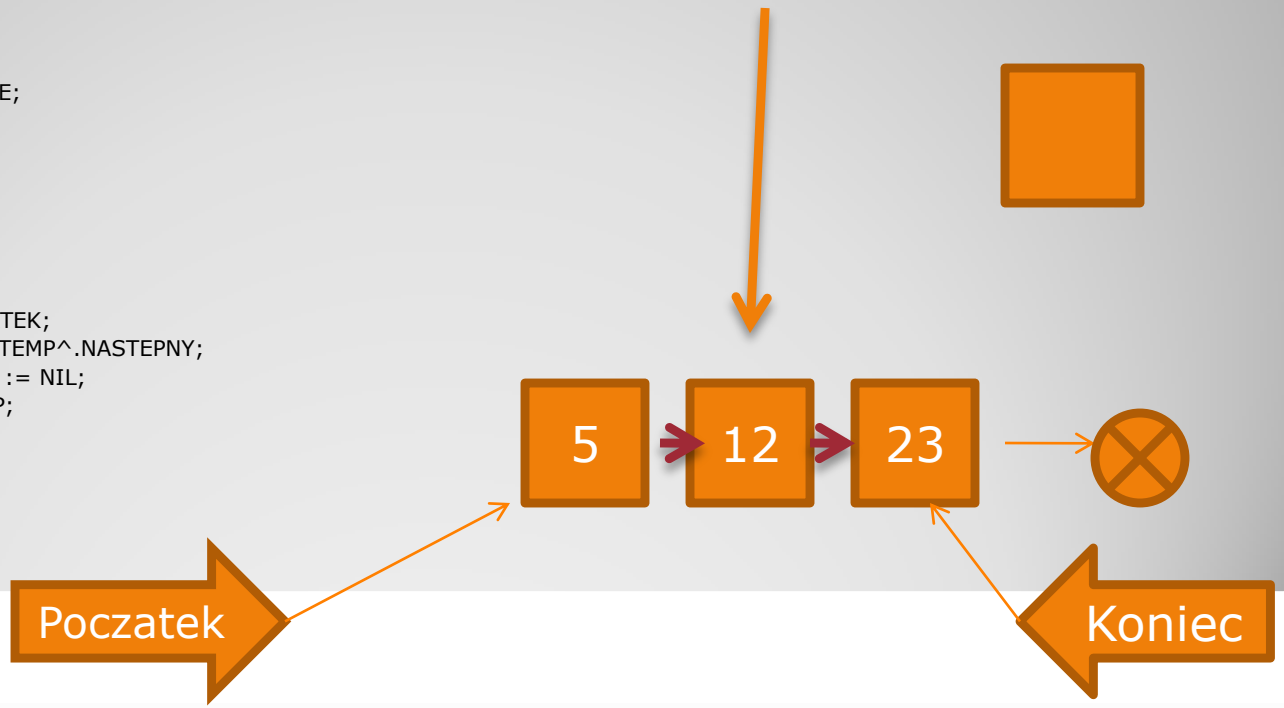
```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 6
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

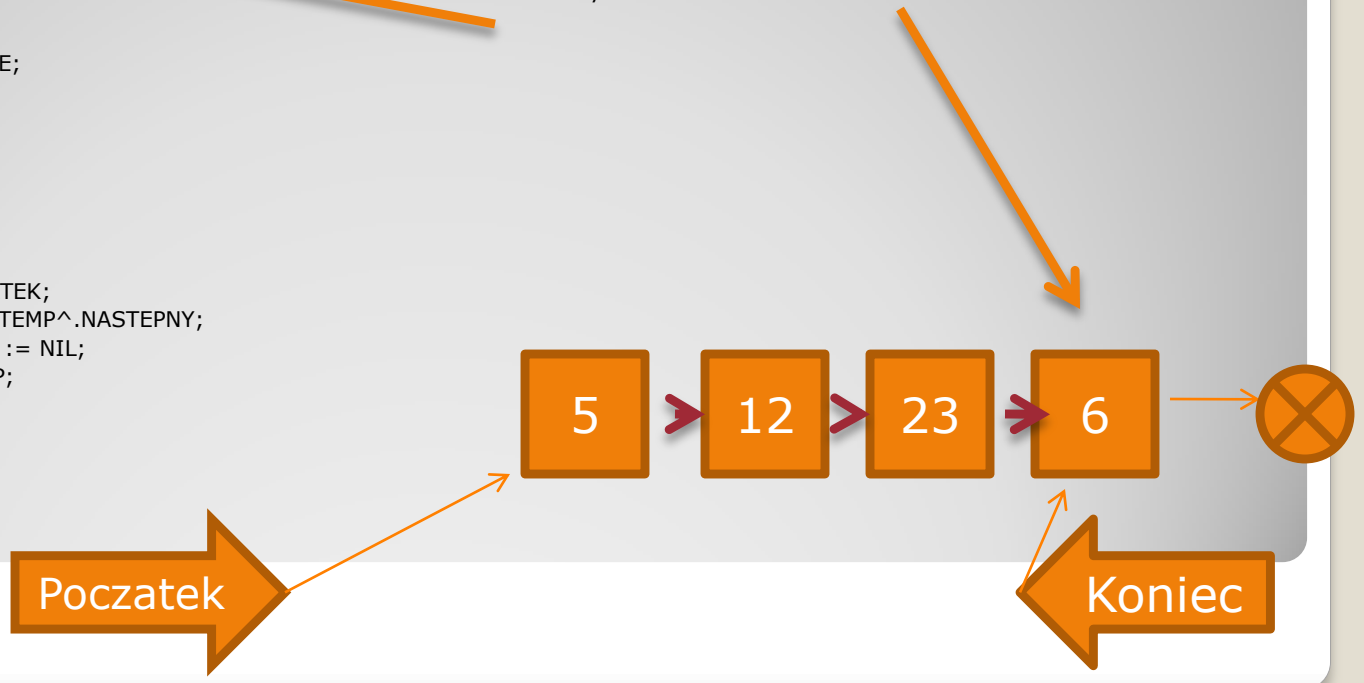
- Tworzymy nowy element z wartością 6
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już kolejny element



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Tworzymy nowy element z wartością 6
- Odwołujemy się do procedury ENQUEUE ze zmiennymi KOLEJKA I ELENENT
- Przechodzimy do procedury ENQUEUE
- Sprawdzamy czy kolejka jest pusta
- W kolejce jest już pierwszy element
- Element staje się następnym elementem kolejki a zarazem jej nowym końcem.



```
TYPE            EP=^ELEMENT;
    KP=^KOLEJKA;

    ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
    END;

    KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
    END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
    END
    ELSE
    BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
    END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
    END
    ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

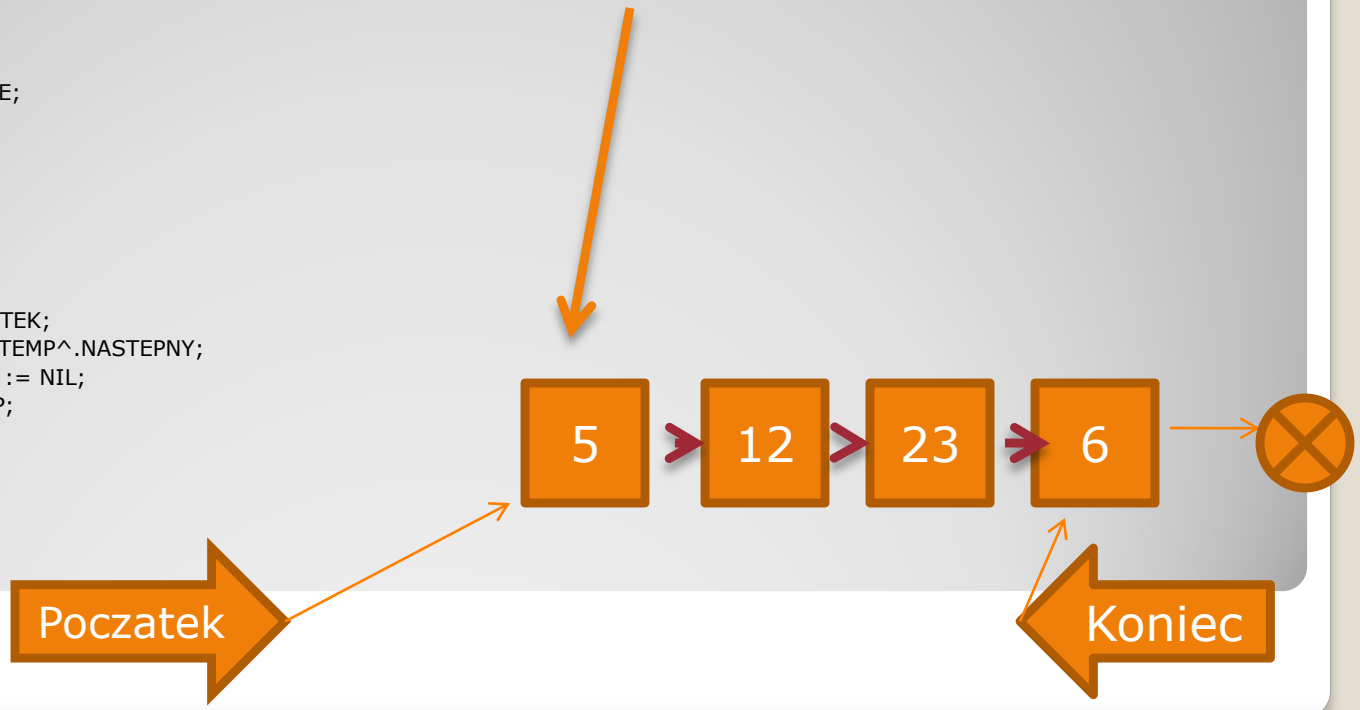
```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element

```
TYPE            EP=^ELEMENT;
    KP=^KOLEJKA;

    ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
    END;

    KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
    END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
    END
    ELSE
    BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
    END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
    END
    ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

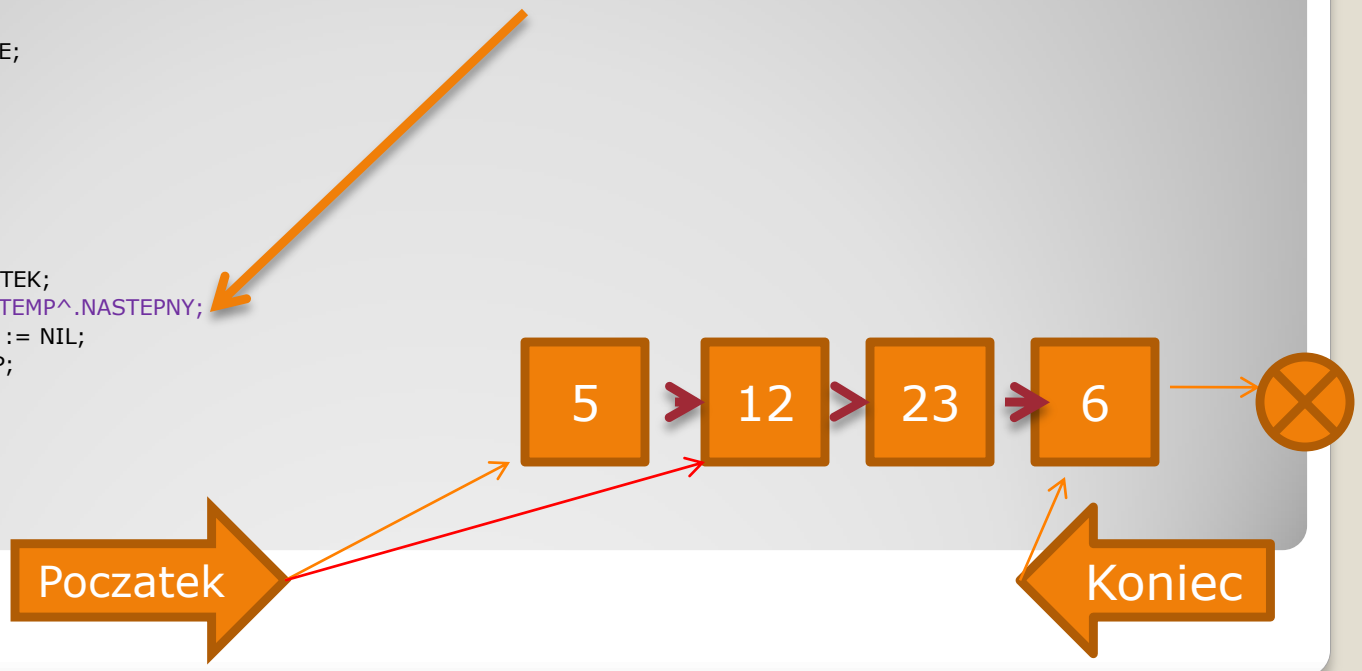
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się kolejny elementem kolejki

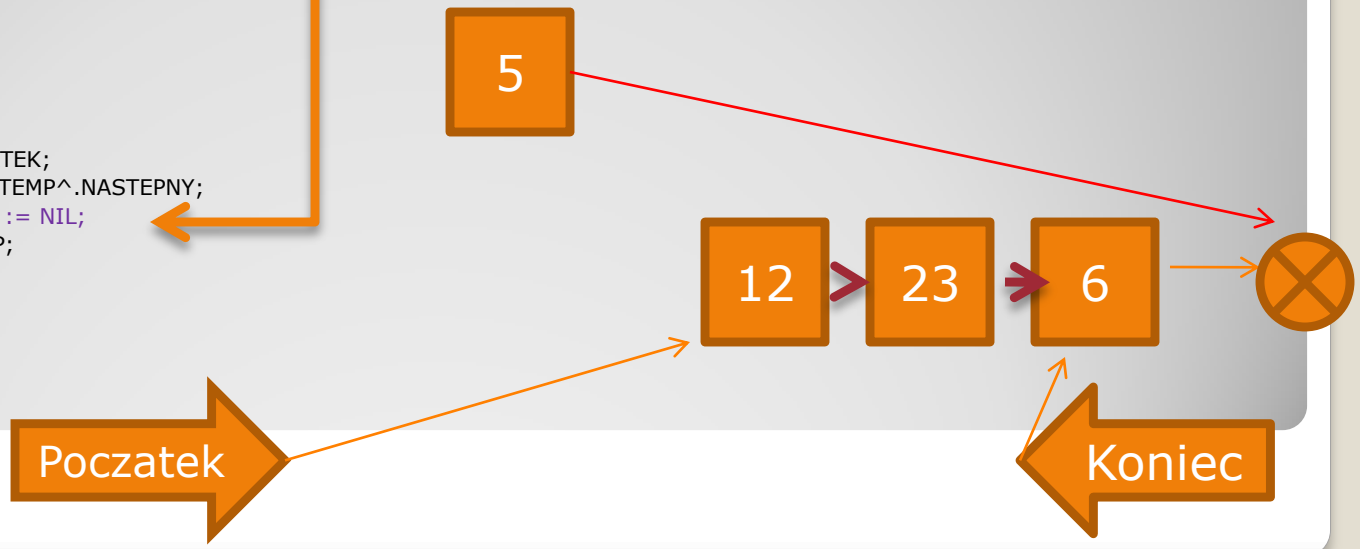


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL

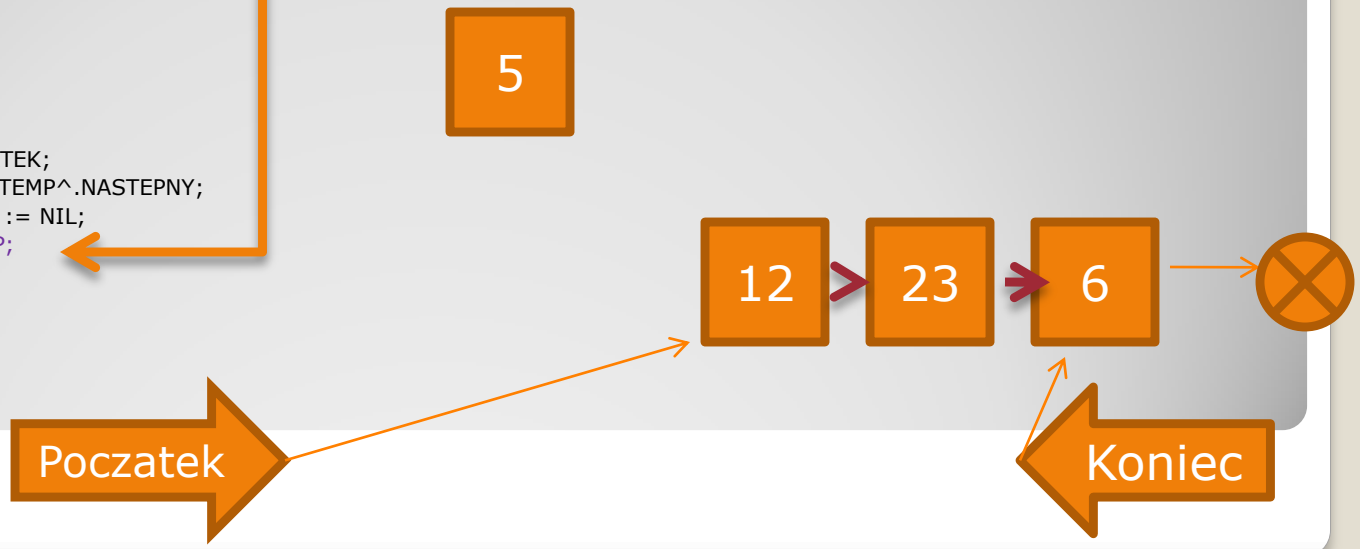


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

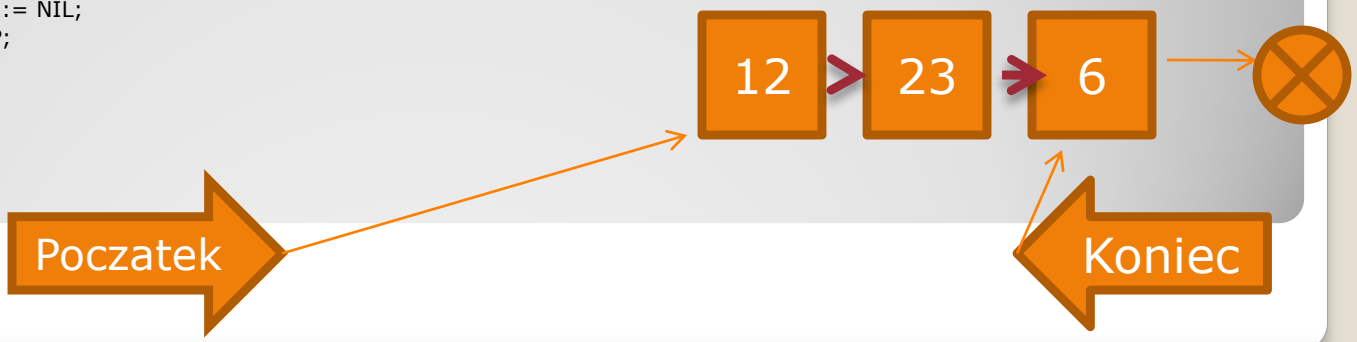
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL
- Usuwamy pierwszy element kolejki



```
TYPE            EP=^ELEMENT;
    KP=^KOLEJKA;

    ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
    END;

    KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
    END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
    END
    ELSE
    BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
    END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
    END
    ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

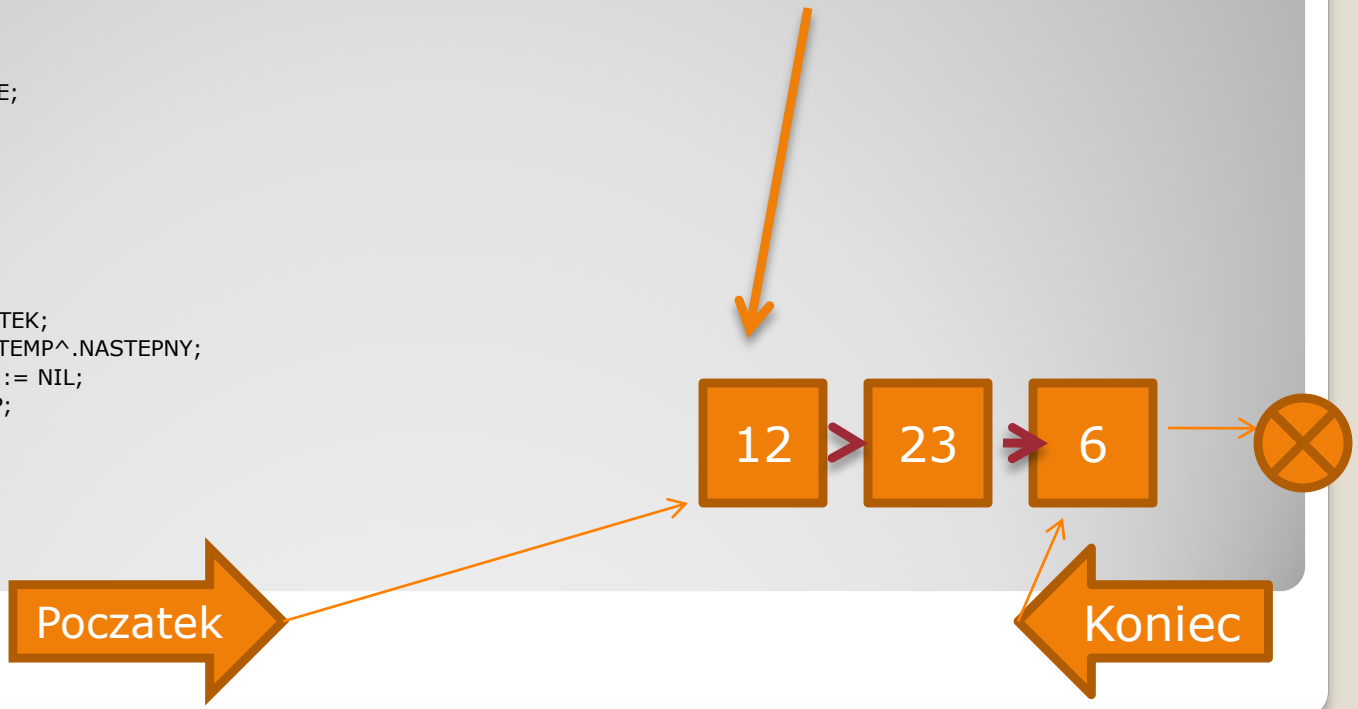
- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element

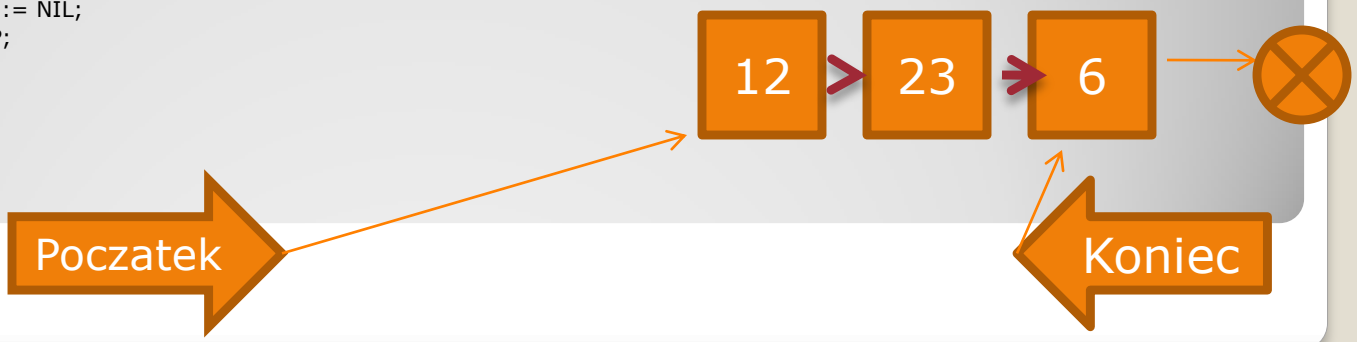


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki

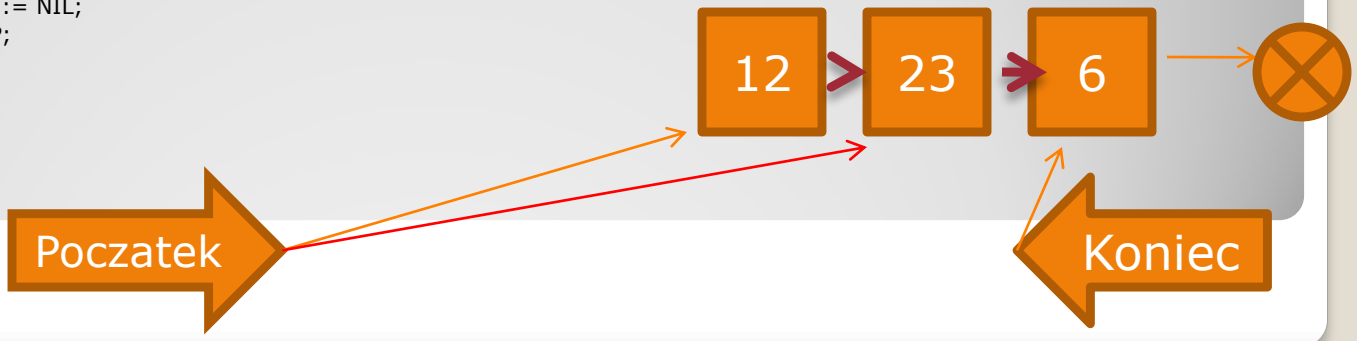


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się kolejny elementem kolejki

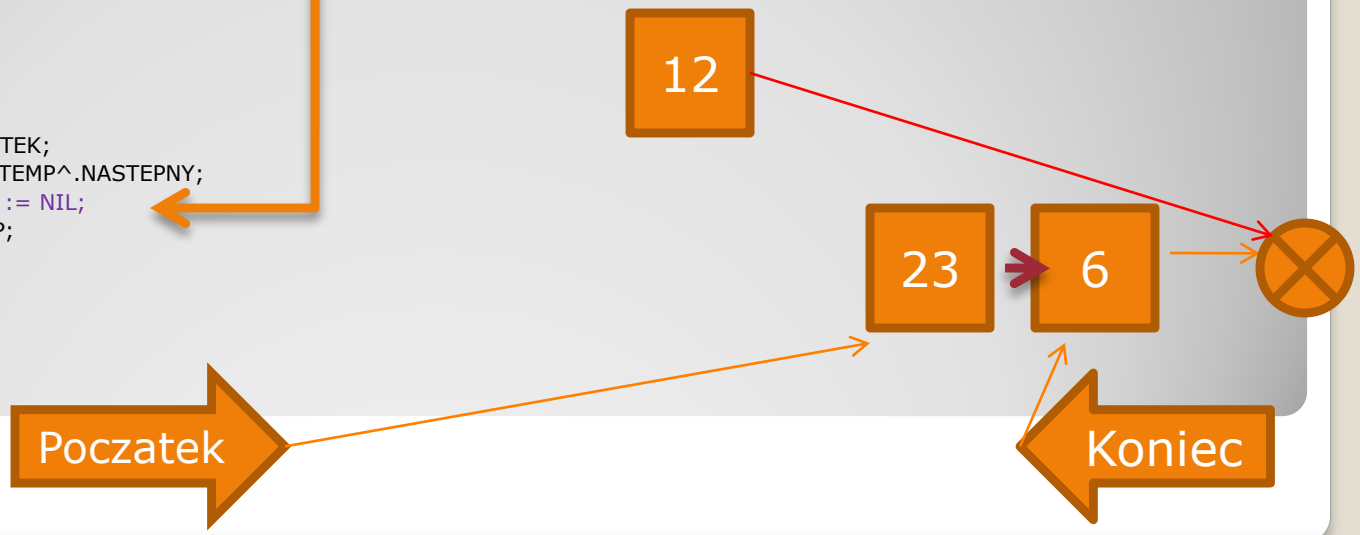


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL

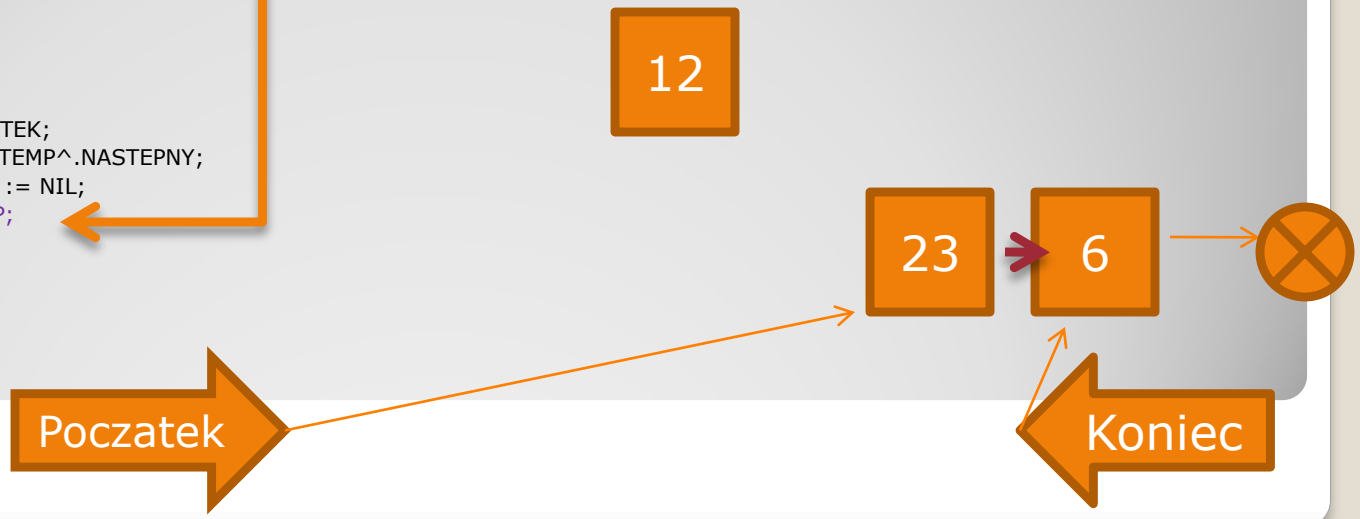


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

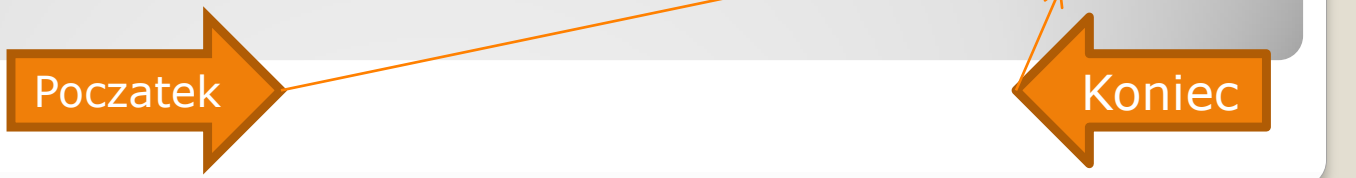
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL
- Usuwamy pierwszy element kolejki



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

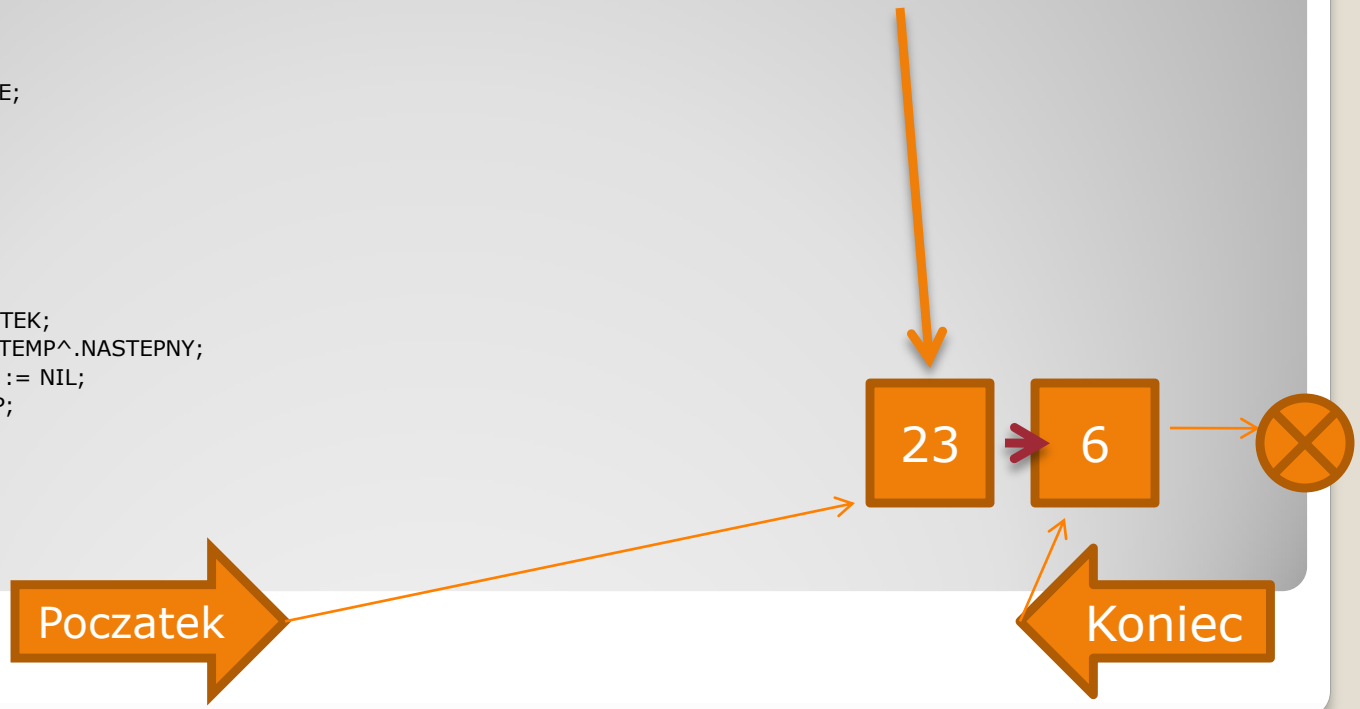
- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element



```
TYPE            EP=^ELEMENT;
    KP=^KOLEJKA;

    ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
    END;

    KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
    END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
    END
    ELSE
    BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
    END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
    END
    ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element

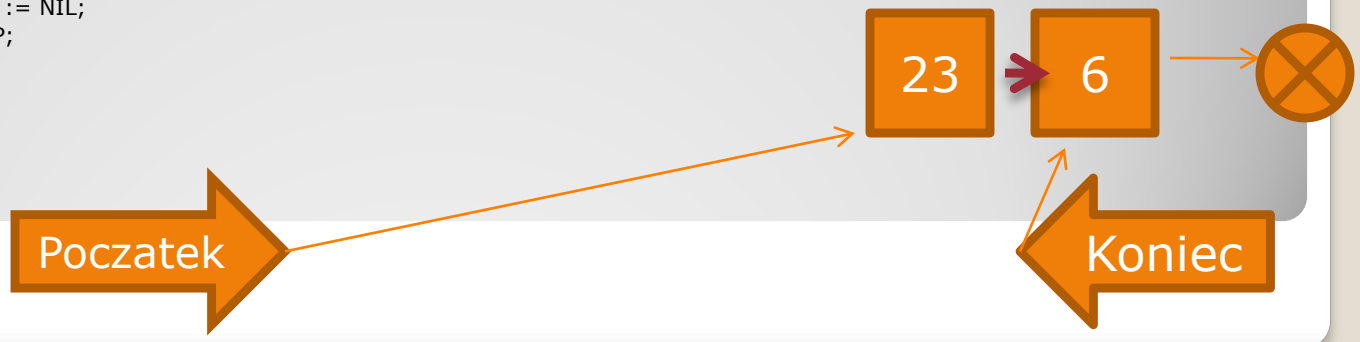


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki

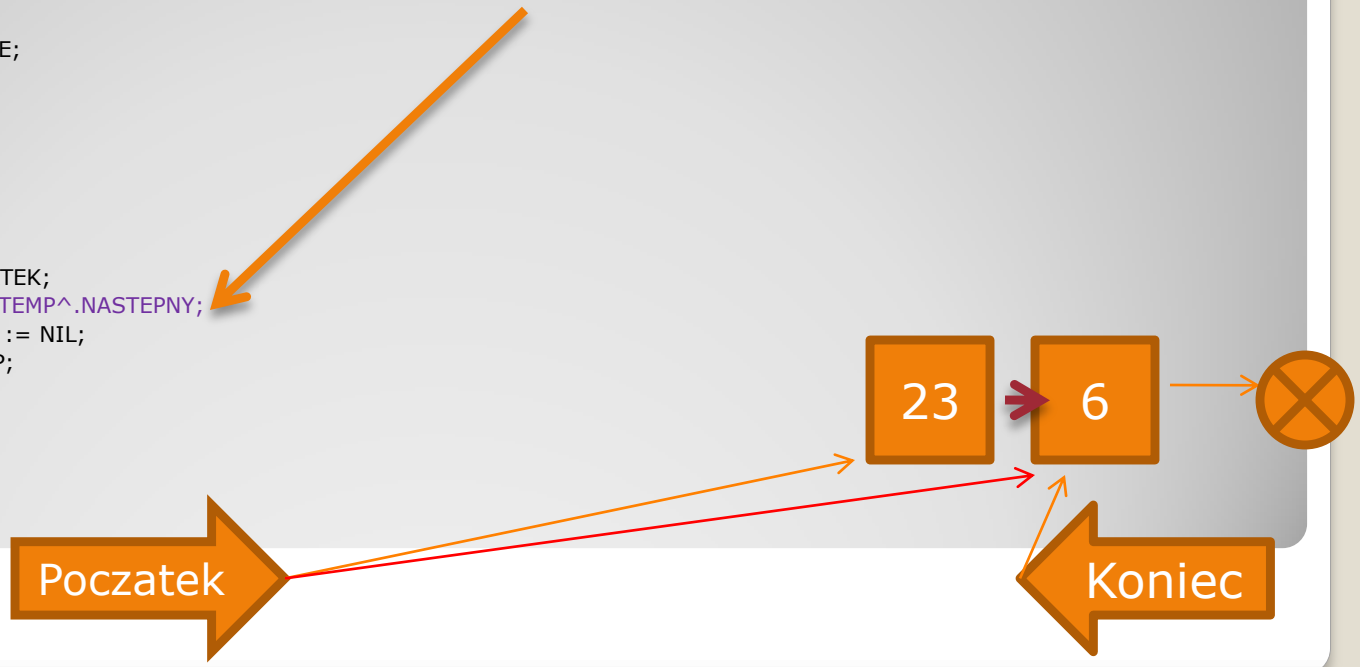


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się kolejny elementem kolejki

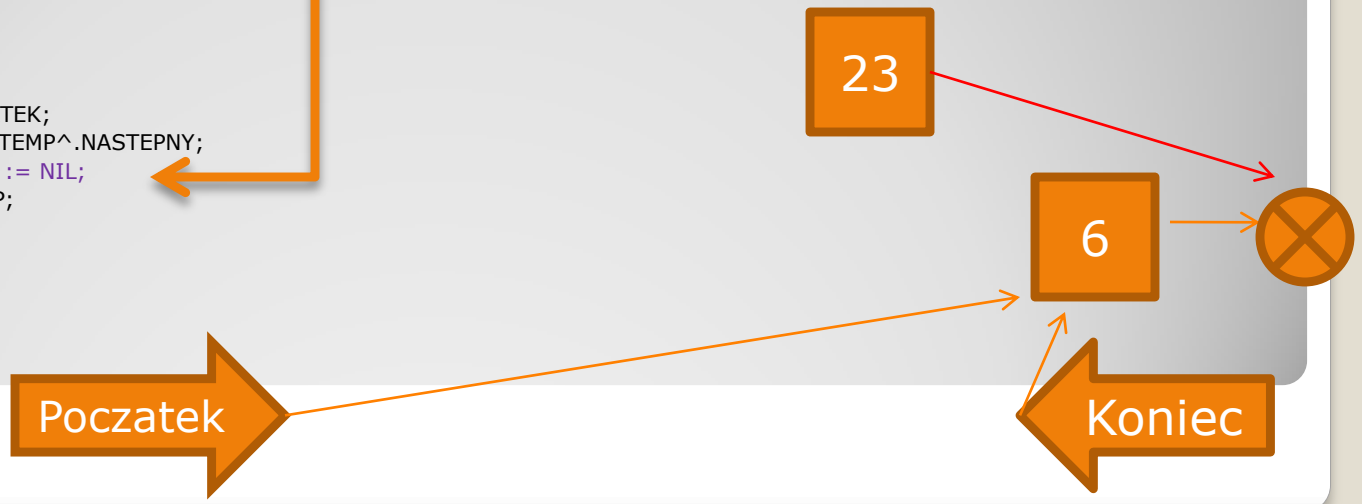


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL

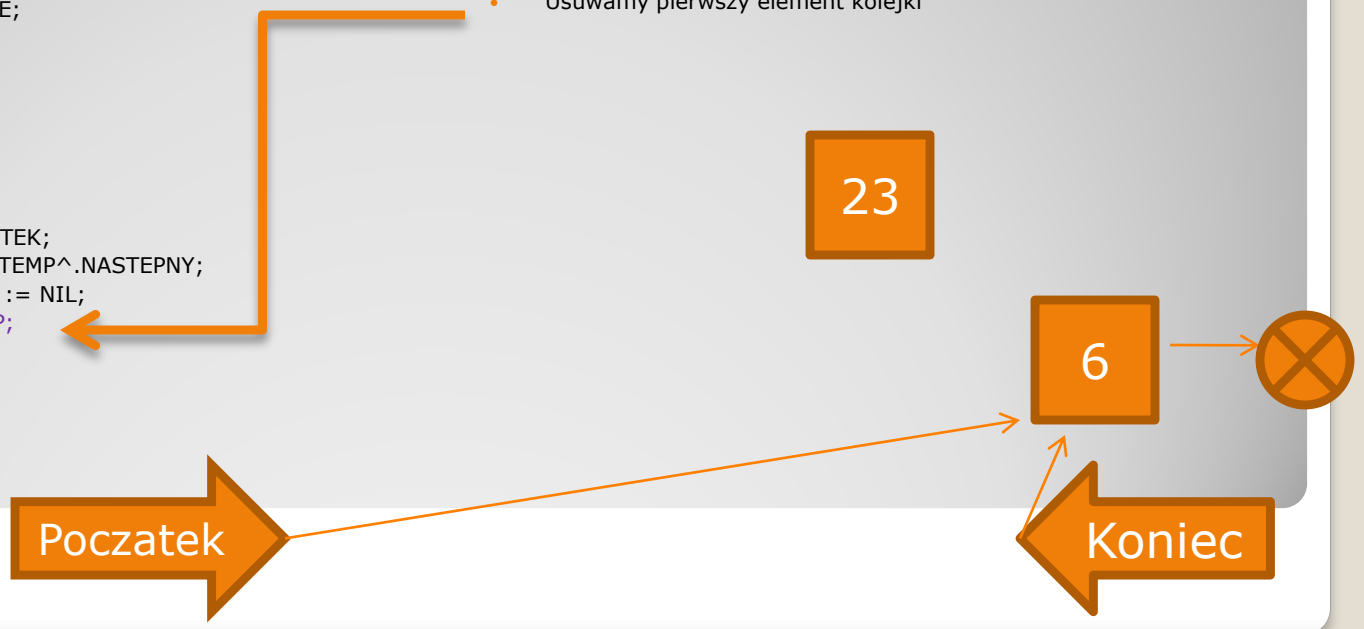


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

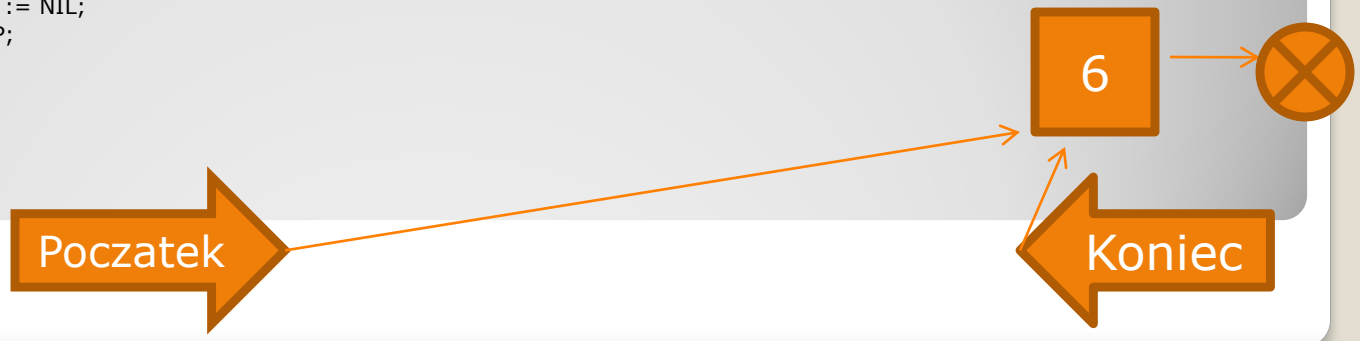
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL
- Usuwamy pierwszy element kolejki



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element



```
TYPE            EP=^ELEMENT;
    KP=^KOLEJKA;

    ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
    END;

    KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
    END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
    END
    ELSE
    BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
    END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
    IF K^.POCZATEK <> NIL THEN
    BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
    END
    ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
    FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

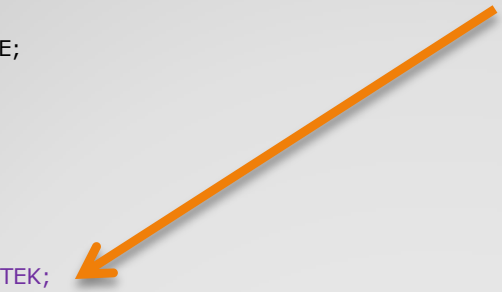
- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element

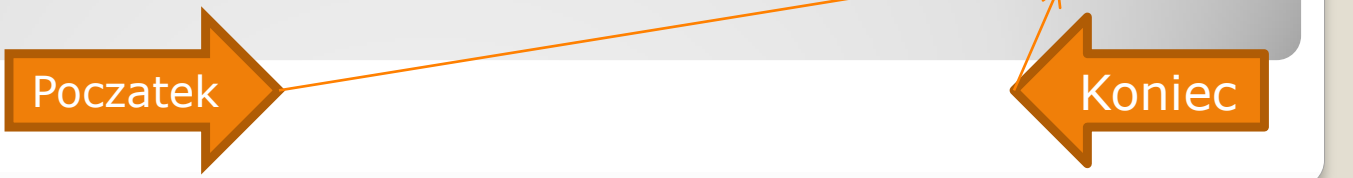
```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki



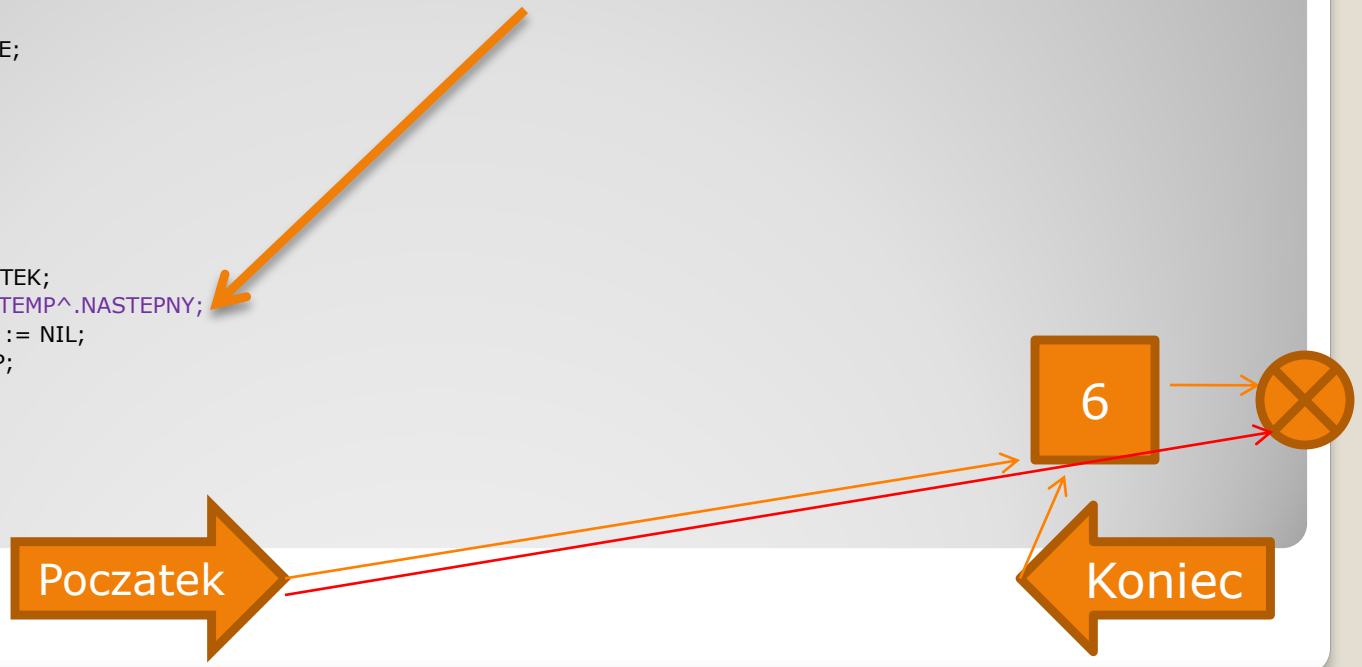


```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się kolejny elementem kolejki



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

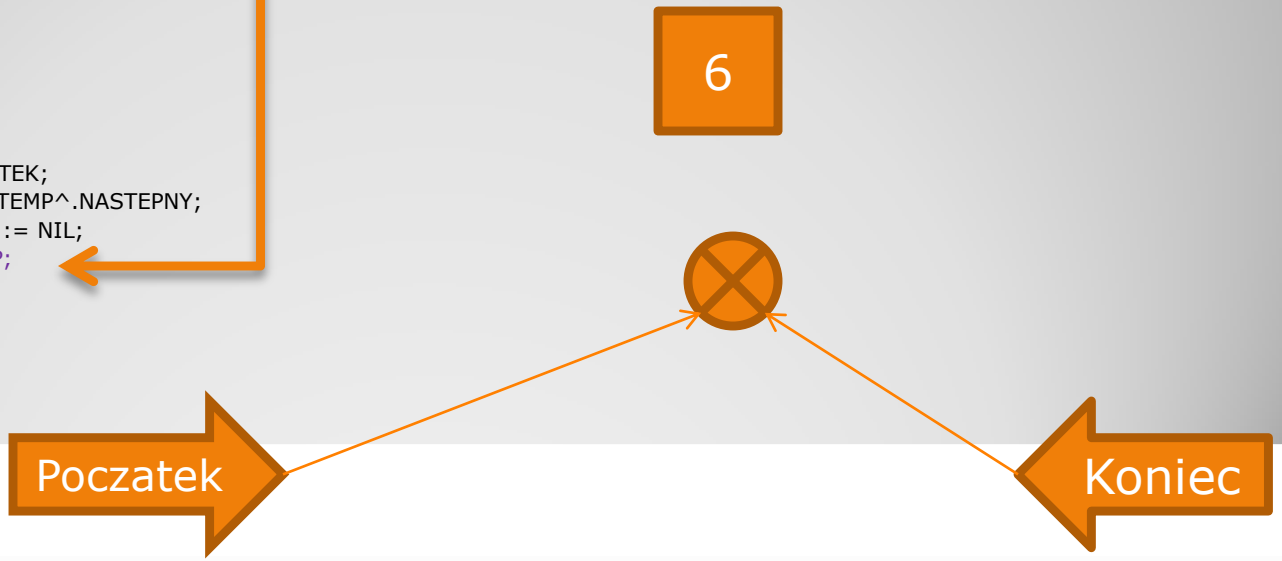
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL

```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki

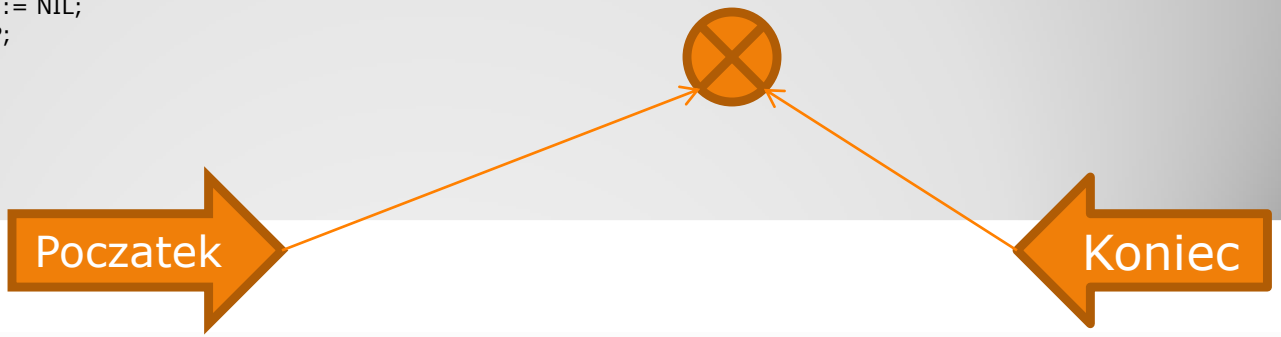
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Na początku kolejki znajduje się element
- Do zmiennej pomocniczej TEMP przypisujemy początek kolejki
- Początek kolejki staje się następnym elementem kolejki
- Następny element kolejki przyjmuje wartość NIL
- Usuwamy pierwszy element kolejki



```
TYPE            EP=^ELEMENT;
     KP=^KOLEJKA;

     ELEMENT=REKORD
                WARTOSC:BYTE;
                NASTEPNY:EP;
     END;

     KOLEJKA=REKORD
                POCZATEK:EP;
                KONIEC:EP;
     END;

PROCEDURE ENQUEUE(K: KP; E:EP);
BEGIN
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                K^.KONIEC^.NASTEPNY := E;
                K^.KONIEC := E;
     END
     ELSE
     BEGIN
                K^.POCZATEK := E;
                K^.KONIEC := E;
     END;
END;
FUNCTION DEQUEUE(K : KP):EP;
VAR TEMP:EP;
     IF K^.POCZATEK <> NIL THEN
     BEGIN
                TEMP:=K^.POCZATEK;
                K^.POCZATEK := TEMP^.NASTEPNY;
                TEMP^.NASTEPNY := NIL;
                DEQUEUE := TEMP;
     END
     ELSE
                DEQUEUE := NIL;
END;
```

```
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                NEW(ELEMENT);
                ENQUEUE(KOLEJKA, ELEMENT);
END;
BEGIN
     FOR I:=1 TO 4 DO
                BEGIN
                DEQUEUE(KOLEJKA);
END;
END.
```

- Przechodzimy do funkcji DEQUEUE która zdejmuje elementy z kolejki
- Sprawdzamy czy na początku kolejki znajduje się jakiś element
- Kolejka jest pusta



Koniec