ADT_Heap specificatie datastructuren college 5 blad 1

Elementen: bevat een veld waarde die een prioriteit aangeven

Type

```
Prioriteit        =   Integer;
PrioriteitElement =   record
                          Dataelement  :  ElementairDataType;
                          Prior        :  Prioriteit
                      end;
```

Structuur:

Prioriteit is een geordende verzameling waardes. Twee elementen vergelijken betekent de prioriteit van twee elementen vergelijken. De prioriteiten voldoen aan de heap voorwaarde.

Domein:

Aantal knopen in een heap is begrensd.

Const maxsize=N;

Type Heap = array [0..maxsize] of PrioriteitElement;

Er zijn 3 operaties gedefinieerd:

Procedure **HeapCreate**(var H:Heap;n:Integer);

Postconditie: elementen H[1] t/m H[n] voldoen aan de heap voorwaarde.

Procedure **BorrelOp**(var H:Heap;m,n:Integer);

Preconditie: elementen H[1] t/m H[n-1] voldoen aan de heap voorwaarde.

Postconditie: elementen H[1} t/m H[n] voldoen aan de heap voorwaarde.

Procedure **DwarrelNeer**(var H:Heap;n:Integer);

Preconditie: elementen H[m+1] t/m H[n] voldoen aan de heap voorwaarde.

Postconditie: elementen H[m} t/m H[n] voldoen aan de heap voorwaarde.

```
Const maxsize=N;

Type
        Prioriteit          =       Integer;
        PrioriteitElement   =       record
                                        Dataelement     :       ElementairDataType;
                                        Prior           :       Prioriteit
                                    end;
         Heap = array [0..maxsize] of PrioriteitElement;
procedure BorrelOp(var H:Heap;n:Integer);
var Kind,Ouder: o..maxsize;
begin
        H[0]:=H[n] # heap element tevens sentinel
        Kind:=n;
        Ouder:=n div 2;
        While H[Ouder].Prior >H[0].Prior do begin
                H[Kind]:=H[Ouder];
                Kind:=Ouder;
                Ouder:= Ouder div 2
        end;
        H[Kind]:=H[0]
end;
```

```
Procedure DwarrelNeer(var H:Heap;m,n:Integer);
Label:1;
Var Kind,Ouder: 0..maxsize;
begin
        H[0]:=H[m];      #kopie van te plaatsen element wordt hier bewaard
        Ouder:=m;
        Kind:=m+m;
        While Kind <= n do begin
                If Kind <n
                        Then if (H[Kind].Prior > H[Kind+1].Prior)
                                Then     Kind:=Kind+1;
                If (h[0].Prior <= H[Kind].Prior) then goto 1;
                H[Ouder}:=H[Kind];
                Ouder:=Kind;
                Kind:=Ouder+Ouder;
        end;
1:      H[Ouder]:=H[0]
end;
```

```
Procedure HeapCreate(var H:Heap;n:Integer);
var k:Integer;
begin
        k:=(n div 2) +1;
        while (K>1) do begin
                k:=k+1;
                DwarrelNeer(H,k,n)
        end
end;
```

ADT_PriorityQueue specificatie datastructuren college 5 bladzijde 1

Elementen: data elementen kunnen van allerlei vorm zijn, neem aan hier van een standard type.

Structuur: bij elk element hoort een prioriteitswaarde, hier een Integer.

Domein: aantal elementen in de rij is begrensd; als grens bereikt is de rij vol.

Er zijn 6 **Operaties**:

Procedure **CreatePQ**(var PQ:PriorityQueue;var created:Boolean);

Postconditie: als created=Y (lege) priority Queue bestaat, anders niet gelukt.

Procedure **TerminatePQ**(var PQ:PriorityQueue);

Postconditie: PQ bestaat niet meer.

Procedure **EnqueuePQ**(var PQ:PriorityQueue;e:DataType;p:Integer);

Preconditie: FullPQ =N.

Postconditie: e en bijbehorende prioriteit toegevoegd aan PQ.

Procedure **ServePQ**(var PQ:PriorityQueue;var e:DataType;var p:nteger);

Preconditie: LengthPQ>0

Postconditie: element met hoogst prioriteit in PQ weggehaald uit PQ en in e beschikbaar met prioriteit p.

Functie **LengthPQ**(PQ:PriorityQueue):Integer;

Postconditie: lengte is aantal elementen in PQ

Functie **FullPQ**(PQ:PriorityQueue):Boolean;

Postconditie: FullPQ=Y als rij vol, anders N.

ADT_PriorityQueue array implementatie met ADT_Heap datastructuren college 5 bladzijde 1

Const

```
    Maxsize=N;
```

Type

```
    Priority        =   Integer;
    PQelement       =   record
                            elem    :   DataType;
                            prior   :   Priority;
                        end;
    PQ              =   ^QueueInstanciatie;
    Index           =   1..Maxsize;
    Size            =   0..Maxsize;
    QueueInstanciatie   =   record
                                Size    :   Size;
                                Pqueue:     array[Size] of PQelement
                            end;
```

```
Procedure CreatePQ(var PQ:PriorityQueue;var created:Boolean);
begin
        new(PQ);        #neem aan lukt altijd
        PQ^.Size:=0;
        Created=Y;
end;
Procedure TerminatePQ(var PQ:PriorityQueue);
begin
        Dispose(PQ)
end;
procedure EnqueuePQ(var PQ:PriorityQueue;e:DataType;p:Priority);
begin
        with PQ^ do begin
                Size:=Size+1;
                Pqueue[Size].elem:=e;
                Pqueue[Size].prior:=p;
                BorrelOp(Pqueue,Size)           #zie ADT_Heap
        end
end;
```

ADT_PriorityQueue array implementatie met ADT_Heap datastructuren college 5 bladzijde 3

```
procedure ServePQ(var PQ:PriorityQueue;var e:DataType;var p:Priority);
begin
        with PQ^ do begin
                e:=Pqueue[1].elem;
                p:=Pqueue[1].prior;
                Pqueue[1]:=Pqueue[Size];        # laatste naar root, delete eerste
                Size:=Size-1;                   # rij 1 korter
                DwarrrelNeer(Pqueue,1,Size);
        end
end;
functie LenghtPQ(PQ:PriorityQueue):Integer;
begin
        LengthPQ:=PQ^.Size;
end;
functie FullPQ(PQ:PriorityQueue):Boolean;
begin
        FullPQ:= (PQ^.Size:=Maxsize)
end;
```