doorlopen. Een operand wordt daarbij op de stapel geplaatst. Een operator heeft als effect dat de twee laatst gestapelde operanden van de stapel worden gehaald, dat de bewerking met die twee operanden wordt uitgevoerd en dat het resultaat op de stapel wordt geplaatst. Figuur 7.2 formuleert dit algoritme in pseudocode.

Het is duidelijk dat een postfixuitdrukking bestaande uit n symbolen met behulp van een stapel in $T(n) = \Theta(n)$ tijd kan worden geëvalueerd. Voor elk symbool moet immers slechts een constant aantal stapelbewerkingen worden uitgevoerd die elk constante tijd vergen. De vereiste stapelbewerkingen kunnen door een compiler ook gemakkelijk in machine-instructies worden omgezet.

7.3.4 Omzetting van infixnotatie naar postfixnotatie

Toch worden uitdrukkingen in de meeste programmeertalen in infixnotatie geschreven. De reden hiervoor is dat er een eenvoudig algoritme bestaat om een infixuitdrukking om te zetten naar een equivalente postfixuitdrukking. Ook dat algoritme maakt gebruik van een stapel.

We hebben reeds opgemerkt dat de ordening van de operanden in de postfixnotatie dezelfde is als in de infixnotatie. Bij de omzetting van infix naar postfix wordt de uitdrukking in de leesrichting doorlopen. Wanneer een operand wordt gelezen, kan dit onmiddellijk in de output worden geplaatst. Operatoren zullen echter tijdelijk op een stapel worden bewaard. Ook de (-symbolen worden, wanneer ze optreden, op de stapel geplaatst. Er wordt verondersteld dat de stapel aanvankelijk leeg is.

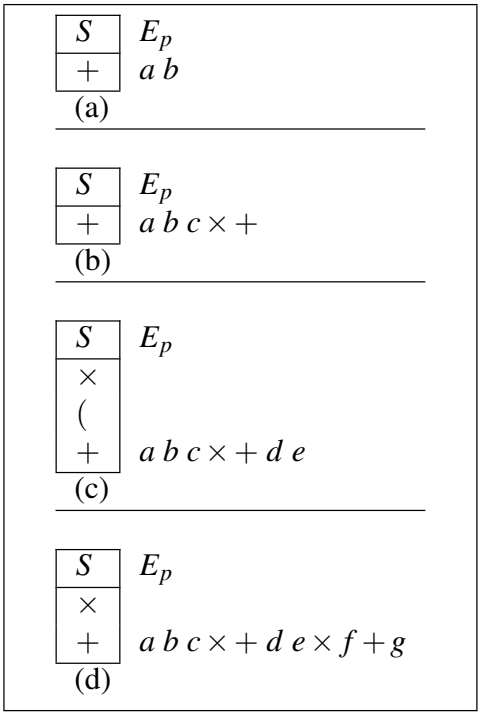
Input: infixuitdrukking E_i
Output: corresponderende postfix E_p

```
1:  S ← ∅; E_p ← ∅
2:  for all x in inputrij do
3:      if x is getal then
4:          E_p.append(x)
5:      else if x = ( then
6:          S.push(x)
7:      else if x = ) then
8:          while S.peek() ≠ ( do
9:              y ← S.pop()
10:             E_p.append(y)
11:         S.pop()
12:     else if x is operator then
13:         while S.peek() ≥_prio x do
14:             y ← S.pop()
15:             E_p.append(y)
16:         S.push(x)
17: while S ≠ ∅ do
18:     y ← S.pop()
19:     E_p.append(y)
20: return E_p
```

Figuur 7.3: INFIXNAARPOSTFIX

Ontmoet men een)-symbool, dan worden achtereenvolgens alle symbolen tot aan het eerste (-symbool, van de stapel gehaald en in de output geplaatst. Het (-symbool wordt wel van de stapel afgehaald maar niet aan de output toegevoegd. Ontmoet men een (-symbool, dan wordt dit op de stapel geplaatst. Ontmoet men ten slotte een operator, dan worden eerst achtereenvolgens operatoren van de stapel gehaald en aan de output toegevoegd, tot wanneer de top van de stapel een operator van lagere prioriteit bevat. In deze fase wordt aan het (-symbool de laagste prioriteit toegekend, zodat een (-symbool alleen maar van de stapel kan worden afgehaald nadat een)-symbool is ingelezen. Daarna wordt de gelezen operator op de stapel geplaatst. Als zo de ganse infixuitdrukking is verwerkt, worden de symbolen die zich nog op de stapel bevinden,

van de stapel afgehaald en aan de output toegevoegd. Figuur 7.3 geeft de pseudocode. Het is onmiddellijk duidelijk dat dit algoritme voor een infixuitdrukking met n symbolen uitvoeringstijd $T(n) = \Theta(n)$ heeft.



Figuur 7.4

Voorbeeld We bekijken het omzetten van $a + b \times c + (d \times e + f) \times g$. Eerst wordt het symbool a ingelezen dat onmiddellijk aan E_p toegevoegd wordt. Dan wordt $+$ ingelezen en op de stapel S geplaatst. Het volgend symbool b wordt E_p toegevoegd. Figuur 7.4(a) toont de toestand van S en E_p op dat ogenblik.

Het volgende symbool is een $\times$. Het bovenste element van S heeft lagere prioriteit en bijgevolg wordt $\times$ op S geplaatst. In de volgende stap wordt c aan E_p toegevoegd. Het volgende symbool is een $+$. Op de stapel bevinden zich $\times$ en $+$. Beide operatoren hebben geen lagere prioriteit dan de ingelezen $+$ en ze worden dus in die volgorde van de stapel gehaald en aan E_p toegevoegd. Hierna wordt de ingelezen $+$ op S gepusht (zie Figuur 7.4(b)).

Het volgende symbool (wordt op de stapel geplaatst, waarna d wordt ingelezen en aan E_p toegevoegd wordt. Volgt een $\times$-operator. De top van de stapel bevat een (, d.i. het symbool met laagste prioriteit dat slechts na inlezing van een) zal kunnen worden verwijderd. Dus wordt $\times$ op de stapel geplaatst. De hierna ingelezen e wordt aan E_p toegevoegd (zie Figuur 7.4(c)).

Een $+$ wordt ingelezen. Bijgevolg wordt het topelement $\times$, dat van hogere prioriteit is, van de stapel gehaald en aan E_p toegevoegd, waarna de ingelezen $+$ op de stapel wordt geplaatst. Het symbool f wordt aan E_p toegevoegd. Volgt het)-symbool. Dus worden $+$ en (van de stapel gehaald, maar alleen $+$ gaat naar de output. Het volgend symbool is een $\times$, die nu boven de $+$ op de stapel geplaatst wordt. Na het verwerken van g is de toestand gegeven door Figuur 7.4(d).

De infixuitdrukking is volledig verwerkt, dus de stapel wordt leeggemaakt en levert zo de postfixuitdrukking $a\ b\ c \times +\ d\ e \times f + g \times +$.