# Algoritmet

### 3.1 Metoda informatike e zgjidhjes së një problemi

Me gjithë zhvillimin e mjeteve informatike, në përgjithësi, zgjidhja e një problemi me ndihmën e një kompjuteri, *ka të automatizuar vetëm ekzekutimin e programit,* i cili duhet të konceptohet nga programuesi. Kjo është edhe arsyeja kryesore që njerëzit mësojnë gjuhët e programimit. Procesi i zgjidhjes së një problemi me ndihmën e kompjuterit kalon në disa faza:

1. Analiza e problemit dhe specifikimi i tij.
2. Ndërtimi i algoritmit.
3. Hartimi i programit.
4. Ekzekutimi i programit dhe testimi i tij.

#### 3.1.1 Analiza e problemit dhe specifikimi i tij

Përshkrimi i një problemi mund të jetë shpesh i paqartë, prandaj së pari ai duhet të analizohet me kujdes, me qëllim që të përcaktohen saktë të dhënat që e përbëjnë informacionin hyrës (input), të domosdoshme për arritjen e zgjidhjes dhe rezultatet (output) që priten nga zgjidhja e këtij problemi. Specifikimi i të dhënave dhe rezultateve për probleme të thjeshta: siç janë ato që përfshihen përgjithësisht në tekstet e programimit, është jo shumë i vështirë. Por, për problemet reale, me zgjidhjen e të cilave merren programuesit profesionistë, kërkohen përpjekje të konsiderueshme për të realizuar këtë specifikim.

#### 3.1.2 Ndërtimi i algoritmit

Pas specifikimit të problemit, duhet të bëhet përshkrimi i detajuar i të gjitha veprimeve që do të kryhen për të arritur zgjidhjen. Ky përshkrim quhet algoritëm. Ndërtimi i algoritmit përbën dhe fazën më të rëndësishme të zgjidhjes së një problemi me kompjuter.
Algoritmi është një bashkësi e fundme veprimesh rigorozisht të përcaktuara, që e ekzekutuar duke u nisur nga kushte të dhëna fillestare, jep rezultatin përkatës dhe përfundon në një kohë të fundme. Me fjalë të tjera, ekzekutimi i një algoritmi shkakton një proces sekuencial , pra një seri ngjarjesh që ndodhin njëra pas tjetrës, secila me një fillim dhe fund të përcaktuar qartë.
Një algoritëm përshkruan zgjidhjen e problemit në mënyrë të pavarur nga çdo gjuhë programimi.
Një nga vendimet më të rëndësishme që duhen marrë gjatë ndërtimit të një algoritmi është zgjedhja e strukturës më të përshtatshme të të dhënave. Të gjitha programet përpunojnë të dhëna, prandaj mënyra e organizimit të tyre ka efekte të rëndësishme në zgjidhjen përfundimtare të problemit. Struktura e të dhënave dhe algoritmet janë ngushtësisht të lidhur me njëri tjetrin.
Sipas mënyrës së organizimit të veprimeve, algoritmet mund të jenë:

1. **Linearë**: Veprimet renditen njëri pas tjetrit, çdo veprim kryhet vetëm një herë.
2. **Të degëzuar**: Sipas vlerës së kushteve logjike, kryhet një varg veprimesh apo një varg tjetër.
3. **Ciklikë:** Një ose më shumë veprime kryhen në mënyrë të përsëritur.

Këto struktura, për të cilat do të flasim hollësisht më vonë, duken shumë të thjeshta, por janë plotësisht të mjaftueshme për të ndërtuar çdo algoritëm.
Përshkrimi i algoritmeve mund të bëhet në mënyra të ndryshme, por më e natyrshmja është që algoritmet të shkruhen në një gjuhë të ngjashme me gjuhët e programimit, pasi kjo do ta lehtësonte kalimin nga algoritmi në program. Një gjuhë e tillë algoritmike quhet gjuhë " pseudo-programimi" ose ndryshe pseudo kod.
Pseudo kodi, ndryshe nga gjuhët e programimit, nuk ka rregulla precize dhe ndryshon nga një programues në tjetrin.
Ndërtimi i algoritmeve në këtë tekst ju përmbahet këtyre rregullave:

1. Për operacionet aritmetike janë përdorur simbolet e zakonshme të kompjuterit: + , -, *, / (mbledhja, zbritja, shumëzimi, pjesëtimi)
2. Madhësitë që trajtohen në algoritëm paraqiten me emra simbolikë.
3. Komentet janë shënuar midis shenjave (* dhe *) .

4. Fjalët çelës të gjuhës së programimit përdoren të përkthyera në shqip p.sh.: lexo, afisho, n.q.s., atëherë, etj.
5. Blloqet e veprimeve mbyllen ndërmjet fjalëve Fillim dhe Fund.
6. Si ndarëse e veprimeve përdoret pikëpresja ( ; ).

**Shembull 1.** Të ndërtohet algoritmi që gjen dhe afishon shumën dhe prodhimin e dy numrave me pseudo kod.

1. Lexo (numër1,numër2);
2. Shuma = numër1 + numër2;
3. Prodhimi = numër1 * numër2;
4. Afisho(Shuma,Prodhimi);
5. Fund.

Një tjetër mënyrë e paraqitjes së sekuencave të një algoritmi është ajo grafike me anë të **bllok skemave (Flow Chart).**
Për ndërtimin e tyre përdoren figura gjeometrike të llojeve të ndryshme, secila prej të cilave shërben për të paraqitur një tip specifik operacioni, si shihet dhe në figurën më poshtë:

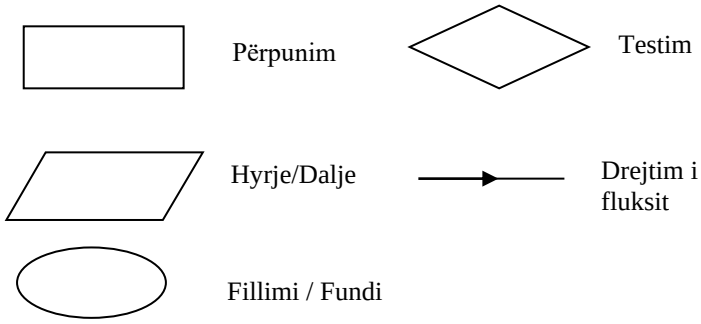


Figura 3-1. Disa simbole për bllokskemat.

Simbolet e mësipërm bashkohen nga segmente të orientuara që paraqesin sekuencën me të cilën operacionet e ndryshme do të ekzekutohen.
I paraqitur në formë grafike, algoritmi i gjetjes së shumës dhe prodhimit të dy numrave do të ishte si në figurën 3-2.

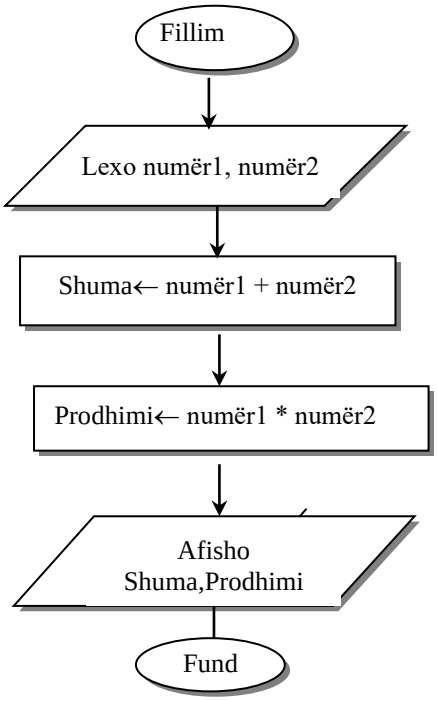


Figura 3-2. Bllok skemë për llogaritjen e shumës dhe prodhimit të dy numrave.

Simboli i testimit (romb) tregon një zgjedhje logjike që duhet kryer gjatë përpunimit. Si rezultat i zgjedhjes logjike algoritmi mund të vazhdojë sipas njërës prej rrugëve të mundshme logjike.

### 3.1.3 Hartimi i programit

Hapi i tretë në përdorimin e kompjuterit për zgjidhjen e një problemi është "përkthimi" i algoritmit në një gjuhë programimi të kuptueshme nga kompjuteri. Ndryshe nga algoritmi, gjatë hartimit të programit duhen zbatuar rigorozisht rregullat gramatikore ose siç quhet ndryshe **sintaksa e gjuhës**. Në këtë tekst, programet janë ndërtuar sipas rregullave të gjuhës C, që do të studiohen në mënyrë të detajuar në kapitujt që vijojnë.
Po japim më poshtë për ilustrim algoritmin e shkruar në pseudo kod për zgjidhjen e një ekuacioni të gradës së dytë dhe programin përkatës në gjuhën C.

```
(*Algoritmi për zgjidhjen e Ekuacionit të gradës së dytë *)
Fillim
        Lexo ( a, b, c ) ;
        d = b*b-4*a*c ;
        N.q.s. d < 0 atëherë
                shkruaj ( 's'ka rrënjë reale' )
        përndryshe
                Fillim
                        N.q.s. d > 0 atëherë
                                Fillim
                                        x1 = ( -b+√d ) / ( 2*a ) ;
                                        x2 = ( -b-√d ) / ( 2*a ) ;
                                Fund
                        përndryshe      (* d = 0 *)
                                Fillim
                                        x1 = -b / ( 2*a ) ;
                                        x2 = x1 ;
                                Fund ;
                        shkruaj ( x1 , x2 );
                Fund;
Fund.
```

Shihet që algoritmi është ndërtuar për një trajtë të përgjithshme të ekuacionit të gradës së dytë, duke përdorur parametra shkronjorë, që në momentin e ekzekutimit marrin vlerat konkrete të koeficienteve të ekuacionit real që duhet zgjidhur. I njëjti algoritëm do të shërbejë në këtë mënyrë për të zgjidhur çdo ekuacion të gradës së dytë.

### 3.1.4 Ekzekutimi i programit dhe testimi i tij

Procedura e **ekzekutimit** të një programi në kompjuter **ndryshon nga njëri sistem në tjetrin**. Sapo programi të jetë regjistruar, ai *kompilohet* dhe *ekzekutohet* duke dhënë komandat përkatëse. N.q.s. nuk detektohet asnjë gabim gjatë kompilimit, programi objekt që rezulton mund të ekzekutohet dhe rezultatet do të afishohen në ekran.
Të dhënat mund të jepen në *mënyrë interaktive* gjatë ekzekutimit të programit prej tastierës nga përdoruesi dhe rezultatet afishohen direkt tek përdoruesi (zakonisht në ekran).
Një tjetër mënyrë pune është ajo në ***batch (mode batch).*** Këtu, përdoruesi përgatit që më parë një skedar që do të përmbajë programin, të dhënat dhe disa komanda rresht. Ekzekutimi i këtij skedari bëhet pa asnjë ndërhyrje të përdoruesit.
Gjatë hartimit të programit, ashtu si dhe gjatë futjes së të dhënave dhe ekzekutimit të tij, mund të bëhen gabime që cilat shkaktojnë ndërprerjen e procesit.
Dallohen disa lloje gabimesh:

**a) Gabime sintaksore**

Këto lloj gabimesh p.sh. harrohet pikëpresja, një variabël deklarohet dy herë, nuk shkruhen në rregull fjalët çelës etj., kapen zakonisht nga kompilatori. Për këtë arsye dhe quhen syntax errors ose compile-time errors dhe zakonisht e bëjnë të pamundur kompilimin dhe ekzekutimin e programit.

**b) Gabime në ekzekutim**

Të tjera gabime si p.sh. pjesëtimi me 0, etj., nuk kapen nga kompilatori. Ato detektohen vetëm gjatë fazës së ekzekutimit. Për këtë arsye ato quhen run-time errors (gabime të kohës së ekzekutimit).

Për të dyja këto lloje gabimesh, sistemi afishon në ekran mesazhe, për të cilat përdoruesi duhet të konsultojë manualet. Në këto raste gabimet mund të korrigjohen relativisht lehtë dhe pasi të modifikohet programi ai mund të ri-kompilohet dhe riekzekutohet.

**c) Gabime logjike**

Ndryshe nga dy të parat, këto lloje gabimesh janë më të vështira për t'u identifikuar. P.sh. në qoftë se një formulë nuk zbatohet në mënyrë korrekte (në vend të shenjës + vendosim *), kjo nuk do të shkaktojë asnjë gabim gjatë fazës së kompilimit dhe ekzekutimit, por rezultati do të jetë i gabuar.

Për të qenë të bindur që rezultatet janë korrekte është e rëndësishme që përdoruesi të ekzekutojë një program disa herë duke futur të dhëna për të cilat rezultatet korrektë njihen që më parë.

Secila prej fazave që përshkruam më sipër ka rëndësinë e vet në zgjidhjen informatike të një problemi, por ndërtimi i algoritmit është vendimtar për arritjen e zgjidhjes. Për këto arsye, do ta trajtojmë pak më gjerësisht këtë problem. Theksojmë që për veprimet që përfshihen në algoritmet, do të jepen vetëm nocionet themelore. Më hollësisht, do të diskutohet për to në elementet kryesore të gjuhës C.

### 3.2 Veprimet e thjeshta. Algoritmet linearë

Algoritmi përmban të gjitha veprimet që duhet të kryhen për zgjidhjen e një problemi. Algoritmi më i thjeshtë është ai linear, ku veprimet renditen njëri pas tjetrit në vijë të drejtë, pra, secili prej tyre kryhet vetëm një herë dhe vetëm pasi është kryer veprimi paraardhës siç tregohet skematikisht në figurën 3-3.

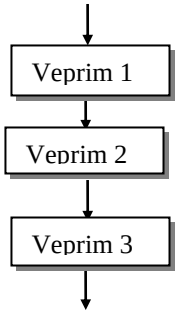


Figura 3-3.Algoritëm linear.

Madhësitë që marrin pjesë në një algoritëm e që quhen ndryshe objekte të algoritmit, janë të ndryshme. Ato mund të jenë konstante apo variabël, numerike apo shkronjore etj. Shembulli 1 është një rast i një algoritmi linear.

Më poshtë po përshkruajmë veprimet elementare që mund të kryhen me objektet në një algoritëm.

#### 3.2.1 Emërtimi i Objekteve

Për të gjitha objektet e algoritmit që mund të jenë të dhënat, rezultatet apo madhësitë që dalin nga llogaritjet e ndërmjetme, duhet të përdoren emra simbolikë. Këta emra përbëhen nga një varg germash dhe shifrash, ku karakteri i parë duhet të jetë detyrimisht germë.

Zakonisht emrat e objekteve i zgjedhim në përputhje me natyrën e tyre për ta bërë sa më të qartë algoritmin.

P.sh. emra objektesh mund të jenë : x, y, Shuma, NotaMes, Rrezja, etj.

### 3.2.2 Veprimi i Vlerëdhënies

Me anë të këtij veprimi i **jepet vlerë** ose **i ndryshohet** vlera variablave. Le të shqyrtojmë shembullin e mëposhtëm:

```
nr = 1;
```

që lexohet: variabli me emrin nr merr për vlerë numrin 1.
Mbas këtij veprimi, vlera e vjetër e variablit nr është zëvendësuar me numrin 1.
Në anën e djathtë të veprimit të vlerë dhënies mund të jetë një shprehje aritmetike p.sh:

```
shprehje = (4 + 3)*(7 - 1)
```

Në këtë rast vlerësohet shprehja.: (4 + 3)*(7 - 1) = 42 dhe i jepet si vlerë variablit shprehje duke zëvendësuar vlerën e vjetër të tij.
Në anën e djathtë mund të figurojnë edhe emra variablash të tjerë. Në këtë rast shprehja djathtas vlerësohet duke përdorur vlerat e variablave në vend të emrave të tyre.
P.sh. konsiderojmë tre veprimet e mëposhtme:

```
A = 4;
B = 12;
C = A + 2 * B;
```

Mbas këtyre tre veprimeve, variabli C do të marrë vlerën 28.
Në anën e djathtë mund të figurojë edhe variabli i anës së majtë p.sh.:

```
I = 1;
I = I + 2;
```

Pas këtyre dy veprimeve, vlera e variablit ka ndryshuar nga 1 në 3.

### 3.2.3 Veprimi i futjes së të dhënave (leximi)

Me anë të këtij veprimi realizohet futja ose dhënia e vlerave të variablave nga përdoruesi, me të cilat, do të operojë algoritmi. Rikujtojmë që specifikimi i tyre është bërë që në fazën e parë. Në algoritëm këtë veprim do ta paraqesim si më poshtë:

```
lexo ( listë objektesh );
```

P.sh.: veprimi lexo ( a, b, c ); do të thotë që përdoruesi duhet të japë një vlerë numerike variablit a dhe më pas shtyp Enter për ta ruajtur si vlerë në memorie. Më pas duhet të japë një vlerë numerike variablit b, dhe më pas shtyp Enter për ta ruajtur si vlerë në memorie, dhe në fund duhet të japë një vlerë numerike variablit c dhe me pas shtyp Enter për ta ruajtur si vlere në memorie.
Veprimi i mësipërm është njëlloj si të shkruajmë:

```
Lexo (a);
Lexo (b);
Lexo (c);
```

Gjithashtu ne mund të **inicializojmë variablat edhe si konstante** duke përdorur veprimin e vlerë dhënies. P.sh.: A=2;

```
shuma=0;        germe='S';        shenja=-1;
```

### 3.2.4 Veprimi i afishimit të rezultateve

Ky veprim paraqitet në algoritëm si më poshtë:

```
afisho ( listë objektesh );
```

P.sh.: afisho ( x, y ); do të thotë që në ekran do të afishohen përmbajtjet e variablave x dhe y ne ekran. Nëse x dhe y kanë vlerat 5 dh 8 do të kishim në ekran të afishuar vlerat: 58.
Për të afishuar një mesazh në ekran përdorim parametrin e parë. P.sh.:

```
Afisho ('Jepni numri e kufizave');
Afisho ('Shuma e dy numrave');
Afisho ('Vlera e A-se eshte:');
```

### 3.2.5 Blloku i veprimeve

Bllok veprimesh quhet një varg veprimesh elementare të përfshira ndërmjet fjalëve **Fillim** dhe **Fund** si më poshtë:

```
Fillim
 veprim elementar;
 veprim elementar;
```

.....
veprim elementar;
**Fund**;

Blloku funksionon në një algoritëm si një veprim i vetëm i pandarë, i cili konsiderohet i kryer kur janë kryer të gjitha veprimet që e përbëjnë atë nga i pari deri tek i fundit. Blloku i veprimeve mund të shfaqet i vetëm, siç është rasti i algoritmeve linearë, por ai mund të shfaqet edhe në përbërje të veprimeve të kushtëzuara apo ciklike. Në këtë rast ai përbën një segment linear në një algoritëm kompleks.

**Shembull 1.** Jepen dy numra A dhe B nga përdoruesi.
Të ndërtohet një algoritëm që llogarit shumën dhe diferencën e tyre.
Për të ndërtuar këtë algoritëm, marrim dy variabla **ndihmës**, të cilëve po u vëmë emrin **Shuma** dhe **Diferenca**, në të cilët do të vendosim përkatësisht shumën dhe diferencën e dy numrave A dhe B.

```
Algoritëm LLOGARITJE1;
fillim
 lexo (A, B);
 Shuma = A + B;
 Diferenca = A - B;
 Afisho (Shuma);
 Afisho (Diferenca);
fund.
```

### 3.3 Veprimet e përbëra dhe të kushtëzuara. Algoritmet e degëzuar

Për probleme të tjera, zgjidhja e të cilave kërkon përdorimin e kushteve logjike, struktura lineare e algoritmeve është e pamjaftueshme. Për të ndërtuar algoritme të degëzuar duhen përdorur veprime të cilat lejojnë sipas vlerës së kushtit logjik, kryerjen e një vargu veprimesh, kapërcimin e tij apo kryerjen e një vargu tjetër veprimesh. Këto quhen ndryshe dhe veprime të kushtëzuara, të cilat do të përshkruhen më poshtë.

#### 3.3.1 Veprimi i kushtëzuar N.q.s. – atëherë

Forma e këtij veprimi është:
N.q.s. < *kusht logjik* > atëherë *veprim*;
ku:
kusht logjik është një **shprehje logjike** dhe mund të marrë **vlerat: e vërtetë (po)** ose **e gënjeshtërt (jo)**
Dhe **veprim** mund të jetë një veprim elementar ose një bllok veprimesh. Në rastin e dytë do të kishim formën analitike:

```
Nqs < kusht logjik > atëherë
        Fillim
                Veprim1;
                Veprim2;
                 ..........
                VeprimN;
        Fund;
```

Skematikisht kjo strukturë paraqitet me diagramë në figurën 3-4.

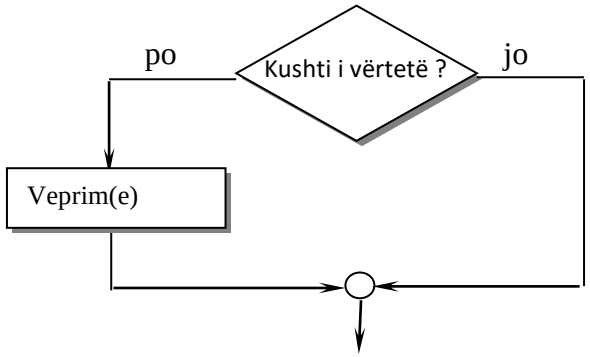


Figura 3-4. Algoritëm i degëzuar. Struktura **N.q.s.-atëherë**.

Struktura funksionon në këtë mënyrë:

Kur vlera e shprehjes logjike është **e vërtetë**, kryhet veprimi që vjen pas fjalës ***atëherë*** dhe vazhdohet me veprimin pasardhës. Kur vlera e shprehjes është **e gënjeshtërt**, nuk kryhet ky veprim (ai kapërcehet) dhe vazhdohet me veprimin pasardhës. Për të ilustruar përdorimin e këtij veprimi po japim shembullin e mëposhtëm:

**Shembull 2.** Të llogaritet vlera e funksionit:

$$y = \begin{cases} 7x - 4 & x < 0 \\ 0 & x = 0 \\ \sqrt{x} + 12 & x > 0 \end{cases}$$

në pikën x = x0 të dhënë.

Si objekte hyrëse (të dhëna) do të jenë konstantet 7, -4, 0, 12 dhe vlerën e pikës x për të cilën do të përdorim emrin simbolik x0. Rezultati që presim është vlera e funksionit në pikën e dhënë. Për këtë madhësi do të përdorim emrin simbolik y.
Edhe në këtë shembull, veprimi i parë do të jetë leximi i të dhënave. Veprimi i dytë do të jetë llogaritja e vlerës së funksionit. Por këtu ekzistojnë tre mundësi që kushtëzohen nga vlera e pikës x, prandaj do të përdorim veprimin e kushtëzuar që përshkruam më sipër.
Veprimi i fundit do të jetë afishimi i vlerës së funksionit. Algoritmi i shprehur në pseudo kod për llogaritjen e funksionit të mësipërm paraqitet si më poshtë:

```
Algoritem Funksion; (* Llogaritje funksioni *)
Fillim
        Deklarime;
        lexo (x0);
        N.q.s. x0 < 0 atëherë y = 7 * x0 - 4;
        N.q.s. x0 = 0 atëherë y = 0;
        N.q.s. x0 > 0 atëherë y = √x0 + 12 ;
        afisho (y);
Fund.
```

Në këtë rast kemi të bëjmë me algoritëm të degëzuar, ku kryerja e veprimeve nuk bëhet në vijë të drejtë, por degëzohet sipas vlerës së kushteve logjike.

### 3.3.2 Veprimi i kushtëzuar Nqs – atëherë - përndryshe

Forma e këtij veprimi është:

```
N.q.s. < kusht logjik > atëherë
        Fillim
                Veprim1;
                .....                    } blloku 1
                VeprimN;
        Fund
përndryshe
        Fillim
                Veprim1;
                .....                    } blloku 2
                VeprimN;
        Fund;
```

Në këtë strukturë, **në qoftë se kushti logjik ka vlerën e vërtetë**, atëherë kryhet blloku 1 i veprimeve, ndërsa **blloku 2 kapërcehet** dhe vazhdohet me veprimin që pason pikëpresjen e fundit.
**Nëse kushti ka vlerën e gënjeshtërt**, atëherë kapërcehet blloku 1 dhe kryhet blloku 2 i veprimeve, pastaj vazhdohet me veprimin pasues. Skematikisht ky veprim paraqitet n Kushti figurën 3-5.

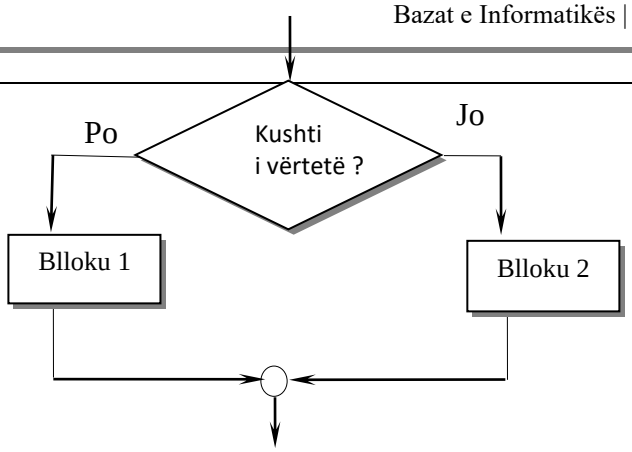


Figura 3-5. Algoritëm i degëzuar. Struktura **N.q.s.-atëherë-përndryshe**.

Në rastet kur në vend të blloqeve kemi vetëm një veprim, struktura ka formën:

N.q.s. < *kusht logjik* > atëherë
    veprim 1
përndryshe
    veprim 2;

Për të ilustruar përdorimin e këtij instruksioni marrim shembullin e mëposhtëm:

**Shembull 3. Të gjendet më i madhi ndërmjet tre numrave të dhënë.**

Numrat, që përbëjnë dhe të dhënat për algoritmin tonë po i quajmë me emrat simbolike A, B, C. Po emërtojmë gjithashtu një variabël MAX, i cili do të marrë si vlerë përfundimtare numrin më të madh. Veprimi i parë do të jetë leximi i tre numrave. Do të pasojnë dy veprime të kushtëzuara nëpërmjet të cilave bëhet krahasimi i numrave me njeri tjetrin dhe sipas vlerës logjike që do të rezultojë, variabli MAX do të marrë vlerën më të madhe. Së fundi, do të afishohet vlera e variablit MAX që përbën edhe rezultatin e problemit të shtruar për zgjidhje. Algoritmi i shprehur me pseudokod do të paraqitej si më poshtë:

```
(* Algoritëm për gjetjen e Maksimumit *)
Algoritëm Maksimum;
Fillim
    Deklarime;
    lexo (A, B, C);
    N.q.s. (A > B) atëherë
        MAX = A
    përndryshe
        MAX = B;
    N.q.s. (C > MAX) atëherë
        MAX = C;
    afisho (MAX);
Fund.
```