II. PROGRAMIRANJE ZADATAKA I POSTUPAK RAČUNANJA S ELEKTRONSKIM RAČUNSKIM AUTOMATIMA

MIRKO BRUKNER dipl. inž. — Zagreb

1. POJAM PROGRAMA I PROGRAMIRANJA

Elektronski računski automati su u pravom smislu automati, koji potpuno samostalno rješavaju i vrlo kompleksne zadatke, i to s vrlo velikom brzinom. Čovječi se rad koristi za pisanje podataka, s kojima će se vršiti računanje, tj. za priređivanje perforiranih traka i kartica, koje nose podatke računanja. Ako su ti podaci rezultat nekog prethodnog računanja, tada se može izbjeći i ovo posredstvo čovjeka. U tom slučaju automat sam izdaje podatke na trake odn. kartice, ili su oni pohranjeni u jedinici za pamćenje, odakle ih automat može direktno koristiti.

Čovjekov se rad prilikom računanja svodi na pritisak na poneku tipku, čime se starta računanje i eventuelno utiče na postupak računanja. Inače se nastoji što više isključiti čovječji rad i posredstvo, koje je sporo prema brzini kojima rade i računaju automati, te podležno greškama.

Međutim bio bi potpuno pogrešan zaključak da automati zahvaljuju samo svojoj konstrukciji tako velike sposobnosti da mogu zamijeniti umni rad čovjeka. Iako su oni u tehničkom i izvedbenom smislu vrlo komplicirani i savršeno izrađeni strojevi, oni su po svojem načinu računanja vrlo primitivni. Računski su automati u svojoj osnovnoj teničkoj izvedbi sposobni samo za ograničen broj vrlo jednostavnih vrsti operacija. Sve praktične mogućnosti automata, te njihova praktična primjena vezani su uz još jedan važan činilac, tj. program. Program je šifrirani oblik niza uputstava, koji upravlja i rukovodi rješavanjem (računanjem) određenog zadatka. On je taj koji iskorištava tehničke mogućnosti automata, te mnoštvom jednostavnih operacija rješava i komplicirane zadatke.

Pravilo da je čovjeku s većim znanjem o prirodi zadatka potrebno manje uputa za rješavanje nekog zadatka, a čovjeku s manjim znanjem mnogo više, vrijedi i za računske automate. Kako je međutim računski automat stroj koji o prirodi zadatka ne zna ništa, to je za uspješno rješavanje zadatka potreban vrlo iscrpan niz uputstava tj. naređenja. Čak i obične aritmetičke operacije (sa pomičnom komom) zahtijevaju nekoliko desetaka naređenja, dok rješavanje kompleksnijih zadataka za-

htijeva i po nekoliko tisuća. Ovu svoju primitivnost elektronski automati nadoknađuju ogromnim brzinama. Tako npr. aritmetičke operacije, i usprkos svojoj složenosti, traju samo nekoliko stotinki sekunde.

Razni automati imaju različite konstruktivne mogućnosti. Dok neki automati imaju samo uređaj za zbrajanje, a sve se ostale operacije provode pomoću programa, dotle kod drugih automata postoje uređaji i za kompliciranije operacije, npr. množenje i dijeljenje. Kod nekih su automata neki programi tehnički ugrađeni, a kod drugih se i ovi osnovni programi unose u automat poput ostalih programa. Ovi osnovni programi obuhvaćaju neke osnovne operacije, kao što su: aritmetičke operacije, čitanje i dešifriranje naređenja i podataka, pretvaranje iz decimalnog u binarni sistem i obratno, itd. Osnovni programi omogućuju praktičan rad automata, te olakšavaju i pojednostavljuju programiranje složenijih zadataka.

Za rješavanje svih praktičnih zadataka potrebno je izraditi odgovarajući program, tj. iscrpan niz naređenja, kojim se regulira cijeli tok rada automata u rješavanju dotične vrste zadataka. U programu moraju biti obuhvaćeni svi detalji i sve alternative zadatka. Kod toga svaka i najmanja pogreška može dovesti do besmislenih rezultata. Za svaku vrstu zadataka (eventualno grupu srodnih zadataka) potrebno je imati odn. izraditi odgovarajući program.

Programiranje tj. izrada programa predstavja dugotrajan posao, koji zahtijeva vrlo mnogo pažnje. On je usprkos tome ekonomičan, jer se pretpostavlja višestruko korištenje.

Kod računanja se dakako mogu koristiti gotovi programi, koje ima na raspolaganju firma, proizvođač automata ili programi drugih institucija, koje su ih izradile za određene zadatke i određeni tip automata. To omogućuje razmjenu programa među raznim korisnicima računskih automata.

Neki računski automati, naročito starijeg tipa, namijenjeni su specijalnim strukama, te imaju čvrsto ugrađene programe za računanja iz područja dotične struke. Time je omogućeno računanje ograničenog broja zadataka s tim ugrađenim programima. U ove automate spada i Zuse Z 11, namijenjen pretežno geodetskim računanjima. Ovim se automatom mogu rješavati neki geodetski zadaci, kao npr.: određivanje koordinata presijecanjem napred ili nazad, računanje dužine i smjernog kuta, rješavanje trokuta, itd. Rješavanje ovih zadataka svodi se na ukapčanje odgovarajućeg prekidača i utipkavanje podataka preko tastature. Kako su takvi automati zbog manjeg raspoloživog kapaciteta nepodesni za druge zadatke, to se danas najčešće proizvode automati bez ovakvih specijalnih namjena. U tom slučaju je korisniku automata prepušteno da koristi neke ili nečije gotove programe, ili da si izradi svrsishodne i najpodesnije vlastite programe.

Najvažniji intelektualni rad čovjeka angažiran je na izradi programa, dok je samo računanje u potpunosti automatizirano i prepušteno stroju. Samo posluživanje automata za potrebe računanja može vršiti i osoba bez naročitih kvalifikacija, jer se ono svodi na mali broj operacija mehaničke prirode.

2. PRIPREMNI RADOVI NA IZRADI PROGRAMA

Prije izrade samog programa u određenoj šifri (kodi), koji se odnosi na određenu vrstu automata i koji obrađuje određeni zadatak, potrebno je izraditi izvjesne pripremne radove, koji su općenito neovisni o tipu automata, a koji omogućuju ili olakšavaju izradu programa, naročto u slučaju kompleksnijih problema s većom razgranatošću.

Kod manjih i jednostavnijih zadataka mogu se pripremni radovi preskočiti, te direktno pristupiti izradi definitivnog programa, što u mnogome ovisi o iskustvu programera.

Pripremni radovi obuhvaćaju slijedeće operacije:

1. Analizu zadatka
2. Izradu strukturnog dijagrama
3. Izradu alogaritamskog programa.

2.1 Analiza zadatka

Analiza zadatka obuhvaća matematsko, organizaciono i logično formuliranje zadatka. Da bi se mogao izraditi program, potrebno je definirati matematske formule, po kojima će se pojedine veličine računati. Npr. ako su zadane koordinate dviju tačaka potrebno se odlučiti, da li će se dužina spojnice računati po Pitagorinom poučku ili uz pomoć trigonometrijskih funkcija i kojih. U svim tim problemima treba voditi računa o tome koje rješenje daje veću tačnost, koje je brže ili zahtijeva manji broj naređenja.

U svakom zadatku redovito postoji veliki broj logičnih formulacija, o kojima ovisi način rješavanja nekog zadatka.

U prethodnom se primjeru postavlja pitanje, da li će se dužina tačnije računati s funkcijom sin ili cos, i u kojem slučaju. Program mora biti tako sastavljen da o tome automat mora sam odlučiti i izabrati odgovarajuće rješenje.

Može se navesti veliki broj primjera logičnih formulacija, kao npr.:

1. Ako je suma kutova veća od 360^0 (400^g), da automat treba tu vrijednost odbiti od dotične sume.
2. Ako je koordinatna razlika Δx jednaka nuli, da se smjerni kut ne računa po uobičajenoj formuli, jer će Δy:o dati neizmjeran rezultat.
3. Ako je dozvoljeno odstupanje u poligonskom vlaku veće od dozvoljenog, da se takav vlak ne sračuna.
4. Da se ovisno od nekog elementa koristi strogo ili približno rješenje.
5. Da se neki postupak tako dugo ponavlja dok se ne postigne određeni broj ponavljanja ili željena tačnost.

i tako dalje.

U svim slučajevima logičnih formulacija postavlja se nekakvo pitanje, o čijoj ispunjenosti ili neispunjenosti ovisi daljnji način rješavanja. Automat treba prema odgovoru sam odabrati potrebnu varijantu, koja dakako mora biti u programu predviđena i obrađena.

2.2 Strukturni dijagram

Strukturni dijagram je grafička formulacija programa. Sa strukturnim dijagramom šematski je prikazan program: njegov tok, njegove lo-

gične formulacije i razgranatost, te matematske formulacije u više ili manje iscrpnom obliku.

Matematsko rješenje zadatka je redovito jasno definirano, te je u strukturnom dijagramu dovoljno navesti samo osnovne formule i veličine na način, da bi se znao redosljed računanja.

Od mnogo su većeg značenja logične formulacije, o kojima ovisi razgranatost programa, njegova struktura i princip rješavanja. Logične formulacije, tj. mjesta gdje se automatu postavljaju pitanja, da li je neki uvjet ispunjen ili ne, su čvorna mjesta programa, koja su u strukturnom dijagramu posebno istaknuta. Od tih čvornih mjesta računanje se dalje odvija u jednom ili nekom drugom smjeru, već prema odgovoru. To su dakle mjesta od kojih se daljnji tok ne može unaprijed znati. On je ovisan od danih podataka računanja. Program međutim mora predvidjeti sve moguće varijante.

S dobro izrađenim strukturnim dijagramom olakšana je izrada definitivnog programa, koja se tada svodi na razrađivanje problema u pojedinačne operacije i naređenja i njihovo šifriranje u određenoj kodi.

Kod izrade strukturnog dijagrama koristi se niz simbola od kojih su ovdje navedeni najbitniji.

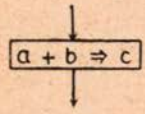
—→ Strelica je znak, koji označava smjer računanja odn. redosljed izvršavanja operacija.

=> Znak jednakosti dopunjen strelicom označava dobivanje rezultata. On osim jednakosti označava i smjer računanja (dobivanja rezultata).

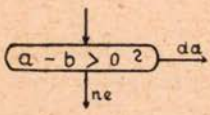
Primjer:

a : b => x znači da **a** podijeljeno s **b** daje **x**.

i + 1 => i znači da se povećanjem indeksa **i** za jedinicu dobiva nova vrijednost indeksa **i**.



U uglastim okvirima dane su operacije, koje se u jednom nizu imaju provesti bez alternacije. Strelica prikazuje tok programa, tj. nakon koje i prije koje operacije se u nizu operacija ima ova provesti.



U ovalnim okvirima su dane logične formulacije, tj. definirana pitanja odn. uvjeti. U tim čvornim mjestima program se grana u dva smjera. Jedan smjer pokazuje tok računanja ako uvjet nije ispunjen, a drugi smjer ako je uvjet ispunjen.

2.3 Alogaritamski program

Definitivno izrađeni program ima alogaritamski tj. linearni oblik. Kod toga je potrebno dijelove programa, koji se praktički nalaze paralelno jedni kraj drugih (koji trebaju teći u isto vrijeme, no samo jedan od njih, jer predstavljaju jednu varijantu tog dijela zadatka), pisati jedne iza drugih. U čvornim tačkama programa, na mjestu uvjeta, gdje se program grana, potrebno je uvesti pojam skoka. Sa skokom se tada izbjegavaju dijelovi programa, koje u danom slučaju nije potrebno provesti, a dospijeva na dio programa, gdje se nastavlja tok računanja. Kod toga

je osnovni princip da se u slučaju ispunjenosti uvjeta vrši skok, dok se u slučaju neispunjenosti uvjeta tok programa direktno nastavlja.

Za razliku od definitivnog programa, koji se sastavlja u određenoj kodi i sadrži iscrpan niz naređenja, od kojih mnoga nemaju matematski karakter, može se sastaviti alogaritamski program u sažetom obliku s općenitim matematskim simbolima. Kod toga svaka oparacija ima redni broj, koji ima karakter adrese definitivnog programa, te ga možemo smatrati simboličnom adresom.

Alogaritamski program ovakvog oblika ima sve karakteristike definitivnog programa. Pod izvjesnim uvjetima i uz primjenu odgovarajućeg programa-prevodioca mogao bi se takav program direktno koristiti za računanje na automatu. On je kod toga neovisan o vrsti automata i njegovoj šifri.

Sažeti alogaritamski programi rjeđe se koriste kao pomoćno sredstvo pri izradi konačnog programa, a češće kod publiciranja i prikaza programa.

U alogaritamskom programu upotrebljava se kao oznaka skoka broj sa strelicom. Navedeni broj označava cilj skoka tj. redni broj mjesta, na kojem se program nastavlja.

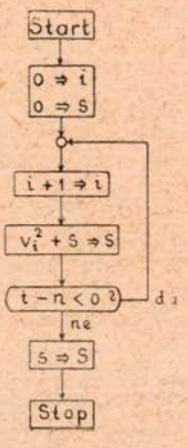
2.4 Primjeri strukturnog dijagrama i alogaritamskog programa

a) — Primjer određivanja sume kvadrata popravaka [v v]

Objašnjenja: i = redni broj popravke (tekući indeks)
s = privremena suma
S = konačna suma (rezultat)
n = broj popravaka

Veličina i i s su na početku nula. Program ima ciklički karakter, kod čega se svakim protokom povećava indeks za jedan, a nova se suma dobiva dodajući dosadanjoj sumi v_i^2

Strukturni dijagram:

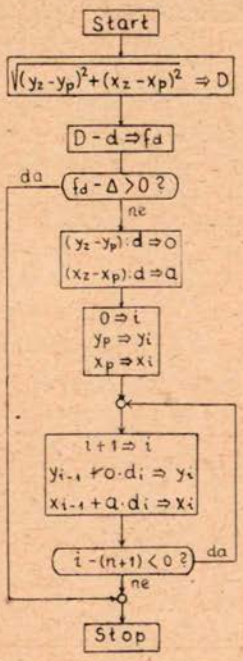


Alogaritamski program:

1. Start
2. o $=>$ i
3. o $=>$ s
4. i + 1 $=>$ i
5. $v_i \cdot v_i => v_i^2$
6. $s + v_i^2 => s$
7. i − n $<$ o ? $=>$ 4 ↓ da
8. s $=>$ S ne
9. Stop

b) — Primjer računanja koordinata malih tačaka na liniji

Strukturni dijagram:



Alogaritamski program:

1. Start
2. $y_z - y_p \Rightarrow M$
3. $M \cdot M \Rightarrow A$
4. $x_z - x_p \Rightarrow N$
5. $N \cdot N \Rightarrow B$
6. $A + B \Rightarrow R$
7. $\sqrt{R} \Rightarrow D$
8. $D - d \Rightarrow f_d$
9. $f_d - \Delta > o$? 21 ↓ — da
10. $M : d \Rightarrow o$ — ne
11. $N : d \Rightarrow a$
12. $0 \Rightarrow i$
13. $y_p \Rightarrow y_i$
14. $x_p \Rightarrow x_i$
15. $i + 1 \Rightarrow i$
16. $o \cdot d_i \Rightarrow \Delta y_i$
17. $a \cdot d_i \Rightarrow \Delta x_i$
18. $y_i + \Delta y_i \Rightarrow y_i$
19. $x_i + \Delta x_i \Rightarrow x_i$
20. $i - (n + 1) < o$? 15 ↓ — da
21. Stop — ne

Objašnjenja:

y_p, x_p	koordinate početne tačke
y_z, x_z	koordinate krajnje tačke
D =	računata dužina
d =	mjerena dužina
d_i	parcijalne dužine (od jedne male tačke do druge)
Δ =	dozvoljeno odstupanje
f_d =	uzdužna pogreška linije
o, a	koeficijenti
i =	tekući indeks
n =	broj malih tačaka na liniji
M, N, A, B, R,	su međurezultati

3. VRSTE PROGRAMA

Za razliku od programa, kojim se rješava neki cjelokupan zadatak (npr. Računanje i izjednačenje poligonskog vlaka), a koji se tretira kao »glavni program«, potrebno je istaći još dvije vrste programa: temeljni

program, kojim je omogućen osnovni rad automata, i potprograme, kojim se rješava dio glavnog programa, kao poseban problem. Glavni program može koristiti više potprograma, od kojih se svaki može javljati na više mjesta glavnog programa. Glavni program tretira bit zadatka sa svim njegovim varijantama (alternativama). Kod toga koristi temeljni program, a još se skraćuje i pojednostavljuje korištenjem raznih gotovih potprograma.

3.1 Temeljni program

Temeljni program daje automatu sposobnost za pojednostavljeno vršenje niza osnovnih radnji, što je neophodno da bi se automat uopće mogao praktično upotrebljavati. On je u automatu ili čvrsto ugrađen, ili se u automat mora učitati poput drugih programa, ali ga se ovdje može na poseban način sačuvati od brisanja ili oštećavanja. Temeljni su programi redovito standardnog oblika za pojedine tipove automata, ali se u njima ipak mogu izvesti sitnije promjene, ili ih takve naručiti kod firme.

Temeljni program izrađuje firma, a obuhvaća niz osnovnih operacija: čitanje programa i podataka, ispisivanje podataka, te aritmetičke operacije.

Program za čitanje omogućuje čitanje programa tj. njegovih naređenja, koja su pisana u nekoj određenoj kodi, dešifrira ih u internu kodu automata i pohranjuje naređenja u njemu razumljivom jeziku u jedinicu za pamćenje.

Kod čitanja brojčanih vrijednosti temeljni ih program prevodi u binarni ili kombinirani sistem, nadalje ih pretvara u polulogaritamski oblik i pohranjuje u jedinici za pamćenje.

Program za ispisivanje ima svrhu da brojeve pretvori natrag u normalni decimalni oblik, da ispisivanje vrši na određeni broj decimala i na željenom mjestu. On također propisuje ispisivanje proizvoljnog teksta, koji je u automatu pohranjen. Program za ispisivanje ima i razne pogonske znakove i mogućnosti, s kojima se postižu željeni tabelarni oblici izdanih rezultata.

Temeljni program sadrži aritmetičke operacije, koje su kod polulogaritamskog oblika brojeva složene, te za njihovu provedbu moraju postojati posebni programi. To omogućuje da se u izradi programa za aritmetičke operacije koriste jednostavna naređenja odn. simboli, a da se ova ne moraju u svakoj prilici ponovo programirati. Osim operacije zbrajanja, odbijanja, množejna i dijeljenja mogu ovdje biti obuhvaćene i druge funkcije, kao npr. vađenje drugog korjena i sl. Redovito se međutim funkcije rješavaju pomoću potprograma, jer se one ne koriste u svakoj prilici, a temeljni program tada može ostati manjeg obima, čime preostaje više kapaciteta za razne druge programe.

3.2 Potprogrami

Da bi programiranje jednog određenog zadatka (glavnog programa) bilo što jednostavnije, te da bi programi bili što kraći, za razne se zadatke i probleme sastavljaju odn. koriste potprogrami. Oni se na taj

način ne moraju svaki puta iznova programirati. Jedan određeni potprogram, koji tretira neki određeni problem, može se koristiti u raznim programima, a isto se tako može i u istom programu koristiti više puta. Time je moguće da se u programu čitavi dijelovi s nizom naređenja zamijene s jednim jedinim naređenjem, kojim se nazove dotični potprogram.

Pomoću potprograma rješava se u prvom redu pitanje određivanja različitih standardnih funkcija, kao što su: trigonometrijske, ciklometrijske, logoritamske i druge funkcije. Ove se funkcije određuju razvijanjem u redove, čime su rješive pomoću običnih aritmetičkih operacija.

U glavnom je programu potrebno na mjestu gdje se želi odrediti vrijednost neke funkcije nazvati posebnim naređenjem dotični potprogram. Time dolazi do skoka iz glavnog programa u željeni potprogram. Protokom kroz potprogram vrši se određivanje vrijednosti funkcije, nakon čega dolazi do povratnog skoka u glavni program, gdje se normalno nastavlja rješavanje glavnog zadatka.

Osnovne aritmetičke operacije, koje su u sastavu temeljnog programa, predstavljaju isto takve potprograme, samo se za razliku od drugih potprograma koriste u svakom zadatku, te su zato, kao bezuvjetno potrebni potprogrami, smješteni unutar temeljnog programa, dok se drugi potprogrami koriste od prilike do prilike.

Potprogrami mogu imati stupnjevit karakter, tj. unutar pojedinih potprograma mogu se koristiti drugi potprogrami, koji imaju značaj II. stupnja, a unutar ovih opet potprogrami III. stupnja itd. Time su potprogrami uvršteni jedni u druge.

Svaki potprogram koristi neke elemente (argumente), tzv. parametre, pomoću kojih se u potprogramu računaju vrijednosti jedne ili više funkcija. Npr. kod računanja jedne trigonomerijske funkcije parametar je veličina kuta.

Osim standardnih potprograma, koji se koriste često u raznim strukama, i koje redovito dostavlja firma, mogu sami korisnici automata sastaviti potprograme, koji se više puta javljaju kao sastavni dio jednog ili većeg broja zadataka. Tako se u geodeziji javljaju kao sastavni dio mnogih zadataka: rješavanje trokuta, određivanje dužina i smjernog kuta, transformacija koordinata, računanje polarnih koordinata, rješavanje sistema linearnih (normalnih) jednadžbi i dr. Za takve se zadatke isplati sastaviti jednom potprograme, te ih u svakoj prilici koristiti, što ubrzava i pojednostavljuje programiranje glavnih programa, koji tretiraju zadatke većeg obima.

4. OBLICI PROGRAMA

Po obliku se razlikuju dvije vrste programa: linearni i ciklički. Kod toga je potrebno naglasiti da je rijetko kada neki program u cijelosti linearan ili cikličan.Redovito svaki program sadrži linearnih i cikličkih dijelova, te bi se moglo ispravnije govoriti o linearnim i cikličkim dijelovima programa.

4.1 Linearni programi

Linearni se programi razlikuju od cikličkih po tome što ne sadržavaju ponavljanih dijelova. Kod odvijanja programa računanje protiče samo jednom kroz svaki dio programa. Takav program ne sadrži skokove unazad, kojim se tok računanja vraća na već prođeni dio programa. I svaki ciklički program mogao bi se prikazati u linearnom obliku, s tim da se ponavljani dijelovi moraju onoliko puta ponoviti koliki je predviđeni broj ponavljanja. To međutim ne dolazi u obzir kod praktičnih računanja, jer je broj ponavljanja od slučaja do slučaja različit. Računski automati dolaze do punog izražaja zahavljujući upravo mogućnosti primjene cikličkih programa. Potpuno linearni programi se stoga niti ne pojavljuju, niti su ekonomični.

4.2 Ciklički programi

Kod razrade nekog zadatka i sastava programa nastoje se pronaći svi oni dijelovi zadatka koje je moguće obuhvatiti ciklusima. Cikličkim dijelom programa moguće je obuhvatiti one dijelove zadatka u kojima se po istim formulama računaju neke veličine, svaki puta iznova, ali iz različitih podataka. Time je dužina programa znatno skraćena, a isto tako i vrijeme programiranja, te smanjena mogućnost griješenja kod sastava programa.

Kao primjer iskorištavanja ciklusa u sastavu programa može poslužiti računanje poligonskog vlaka, u kojem se mogu ciklički obraditi slijedeći dijelovi zadatka:

1. Računanje sume kutova, radi dobivanja kutnog odstupanja

$$\Sigma\beta + \beta_i \Rightarrow \Sigma\beta$$

2. .Računanje smjernih kuteva

$$v_{i-1} + \beta_i + v_\beta \pm \pi \Rightarrow v_i$$

3. Određivanje koordinatnih razlika i njihove sume

$$d_i \cdot \sin v_i \Rightarrow \Delta y_i \qquad \Sigma\Delta y + \Delta y_i \Rightarrow \Sigma\Delta y$$
$$d_i \cdot \cos v_i \Rightarrow \Delta x_i \qquad \Sigma\Delta x + \Delta x_i \Rightarrow \Sigma\Delta x$$

4. Računanje definitivnih koordinata

$$y_{i-1} + \Delta y_i + v_y \cdot d_i = y_i$$
$$x_{i-1} + \Delta x_i + v_x \cdot d_i = x_i$$

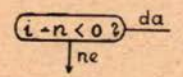
Bez primjene ciklusa većinu bi zadataka bilo vrlo teško ili nemoguće rješiti. Kad bi se npr. računanje poligonskog vlaka htjelo riješiti linearnim programom, moralo bi se sastaviti posebne programe za vlak s jednom, dvije, tri itd. tačaka, što bi predstavljalo nezgrapno i potpuno neekonomično rješenje. Primjena računskih automata došla bi tada u mnogim slučajevima u pitanje, a mnogi bi zadaci postali nerješivi.

U svakom ciklusu mora biti posebno programiran izlaz iz ciklusa. Izlaz iz ciklusa je redovito vezan uz neki uvjet. Kod toga se razlikuju dva tipa uvjeta, a prema tome i dva tipa cikličkih programa:

1. Ciklički programi s definiranim brojem protoka
2. Ciklički programi s varijabilnim brojem protoka.

Kod cikličkih programa s definiranim brojem protoka u svakom je zadatku na neki način određen broj ponavljanja, bilo da se on zadaje s ostalim podacima, bilo da ga automat iz nekih podataka može sračunati. Ponavljanje ciklusa se vrši dotle dok taj broj nije dostignut.

Logična formulacija za izlaz iz ciklusa ima tada oblik:

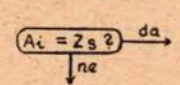


Ponovni ulaz u ciklus

Izlaz iz ciklusa

i = broj dosadanjih protoka
n = zadani broj ponavljanja

Umjesto broja protoka često se koristi neki simbol kao oznaka kraja podataka, pa prema tome i cikličkih protoka. Kako se podaci po redu dovode iz niza i obrađuju u cikličkom dijelu programa, to se ovo računanje prekida, kada se kao novi podatak pojavljuje dotični simbol. Izlaz iz ciklusa bi tada imao uvjet slijedećeg oblika:

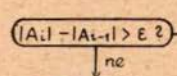


Izlaz iz ciklusa

Povratak u ciklus

A_i = slijedeći podatak iz niza
ZS = završni simbol na kraju niza podataka

Kod cikličkih programa s varijabilnim brojem protoka, broj ponavljanja je redovito vezan uz postizavanje određene tačnosti. To je naročito slučaj kada se neka veličina određuje iz razvijenog reda. U tom je slučaju broj računatih članova reda ovisan o traženoj tačnosti. To su tzv. iteracioni ciklusi. Logična formulacija, koja definira izlaz iz ciklusa, ima tada slijedeći oblik:



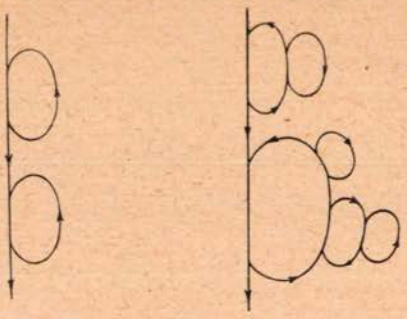
Povratak u ciklus

Izlaz iz ciklusa

A_i i A_{i-1} su dva uzastopno sračunata člana reda
ε = zadana tačnost (dozvoljena pogreška)

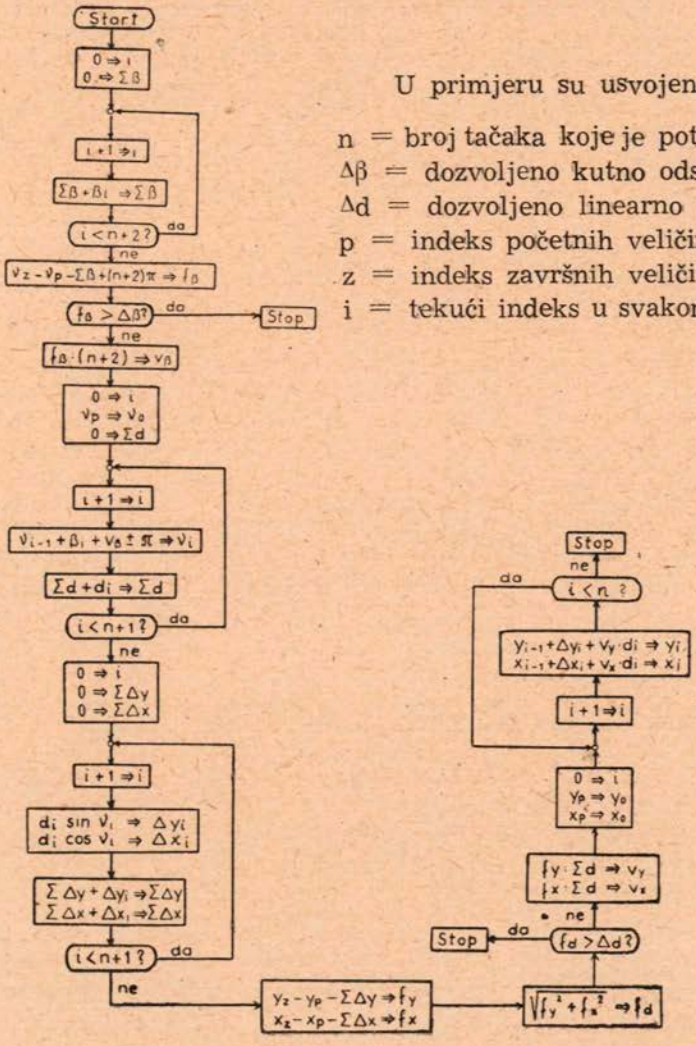
Ciklički programi mogu biti jednostavni i složeni. Kod složenih ciklusa imamo unutar ciklusa ugrađene daljnje cikluse. Na sl. 1 dan je šematski prikaz jednostavnog i složenog ciklusa.

U slijedećem primjeru za izjednačenje poligonskog vlaka dan je strukturni dijagram, koji sadrži samo jednostavne cikluse. U primjeru za sastavljanje koeficijenata normalnih jednadbi prikazan je primjer sa višestruko složenim ciklusima.



Sl. 1

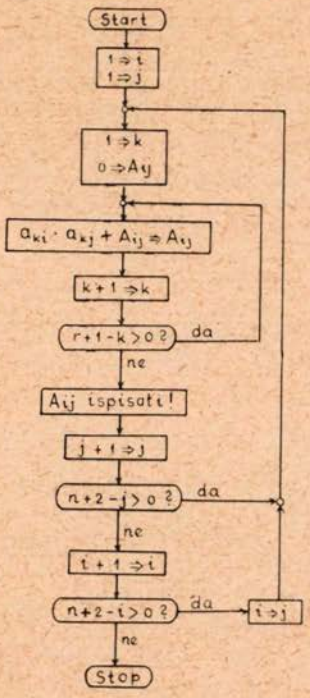
Strukturni dijagram za računanje poligonskog vlaka:



U primjeru su usvojene oznake:

n = broj tačaka koje je potrebno odrediti
$\Delta\beta$ = dozvoljeno kutno odstupanje
Δd = dozvoljeno linearno odstupanje
p = indeks početnih veličina
z = indeks završnih veličina
i = tekući indeks u svakom ciklusu

Strukturni dijagram za sastav koeficijenta normalnih jednadžbi:



Koeficijenti jednadžbi pogrešaka:

$$a_{11}\, a_{12}\, a_{13} \ldots a_{1n}\; a_{1,n+1}$$
$$a_{21}\, a_{22}\, a_{23} \ldots a_{2n}\; a_{2,n+1}$$
$$\ldots\ldots\ldots\ldots$$
$$a_{r1}\, a_{r2}\, a_{r3} \ldots a_{rn}\; a_{r,n+1}$$

Koeficijenti normalnih jednadbi:

$$A_{11}\, A_{12}\, A_{13} \ldots A_{1n}\, A_{1,n+1}$$
$$A_{22}\, A_{23} \ldots A_{2n}\, A_{2,n+1}$$
$$\ldots\ldots\ldots\ldots$$
$$A_{nn}\, A_{n,n+1}$$

gdje je općenito:

$$A_{ij} = a_{1i} \cdot a_{1j} + a_{2i} \cdot a_{2j} + a_{2i} \cdot a_{2j} + \ldots + a_{ri} \cdot a_{rj}$$

U programu su kod jednadžbi pogrešai i j indeksi stupca, a k indeks retka, a kod normalnih jednadbi je i indeks retka, a j indeks stupca.

5. IZRADA KONAČNOG PROGRAMA I KODIRANJE NAREĐENJA

Definitivna izrada programa sastoji se u tome da se cjelokupni zadatak razradi u pojedinačna naređenja, koja zajedno trebaju obuhvatiti sve operacije koje treba stroj izvršiti da izradi do kraja zadatak. Kod toga strukturni dijagram i eventualno alogaritamski program služe kao okosnica i podsjetnik na ono što se u programu ima obraditi i kojim redom, te upozoriti na mjesta gdje se program grana.

Naređenja moraju biti pisana određenom šifrom (kodom), da bi bila razumljiva automatu, te da bi ih ovaj mogao izvršiti. Kod većine automata mogu se primijeniti dvije ili više koda. Kod toga se redovito radi o jednoj internoj kodi, dok su druge tzv. eksterne. Dok je interna koda u skladu s konstruktivnim zahtjevima automata, dotle su za eksterne kode potrebni odgovarajući programi-prevodioci, koji naređenja iz eksterne kode pretvaraju u internu. Taj program-prevodilac može biti unutar temeljnog programa, u kojem slučaju čitanje eksternog programa ne predstavlja nikakvih poteškoća odn. ne iziskuje posebnih zahvata. U protivnom slučaju mora se prvo unijeti program-prevodilac, koji tada služi za čitanje i dešifriranje odgovarajućeg eksternog programa.

5.1 Interna koda

Većina automata koristi tzv. analitičku kodu. Kod ove kode izvjestan broj osnovnih operacija ima oznaku u slovima ili ciframa. Svako takvo slovo odn. cifra označuje jednu određenu operaciju. Kombiniranjem dvije ili više oznaka (slova odn. cifara) postižu se različite složenije operacije odn. naređenja. Tako je izbor naređenja znatno proširen.

Interna je koda u skladu s konstrukcijom automata, s njegovim temeljnim programom i s konstrukcijom naređenja u automatu, pa se kod čitanja programa pisanog u internoj kodi svaka oznaka direktno pretvara u električni impuls, koji se upućuje na određeno mjesto ćelije pamćenja, gdje se pohranjuje kao binarna cifra I. Ova binarna jedinica imat će isto mjesto i u registru naređenja, te je time kod odvijanja programa definirana vrst operacije.

5.2 Eksterna koda

Iz memotehničkih razloga češće se koriste kod programiranja eksterne kode. Njih je lakše zapamtiti, te je programiranje brže i jednostavnije a mnogo je lakše i dešifriranje, tj. čitanje i razumijevanje takvih programa. Osim toga može eksterna koda biti jednaka za više različitih tipova automata (barem onih od iste firme), te se na taj način može isti program koristiti na različitim automatima.

Programe pisane u eksternoj kodi mora prevodilac očitati u automat. Prevođenje se odigrava u toku samog čitanja, pa je zbog toga čitanje takvih programa nešto sporije. Unutar samog automata naređenja su u internoj kodi. Program-prevodilac, koji može ali ne mora biti u sastavu temeljnog programa, zauzima izvjestan prostor u jedinici za pamćenje, što je dakako mana eksternih koda.

Naređenja u eksternoj kodi obilježavaju se također slovima, ciframa ili simbolima. Ti simboli imaju redovito nešto zajedničko s prirodom operacije, pa ih je na taj način lakše zapamtiti. Tako npr. Zuse-ovi automati koriste tzv. Freiburšku kodu. U njoj je naređenje za operaciju zbrajanja: »+«. U internoj kodi isto naređenje za automat Zuse Z-23 glasi: Fo+454, tj. to je naređenje za naziv potprograma za zbrajanje.

Od strane firme redovito je usvojena određena eksterna koda, međutim to ne isključuje mogućnost primjene neke druge eksterne kode s alogaritamskim karakterom, samo je za takvu kodu potrebno imati ili izraditi odgovarajući program-prevodilac.

5.3 Raspodjela ćelija pamćenja i evidencija adresa

Prije izrade programa tj. ispisivanja samih naređenja, mora se izvršiti raspodjela ćelija pamćenja. Time je potrebno odrediti početnu adresu glavnog programa i eventualno nekih potprograma, tekstova itd. Odmah na početku programiranja potrebno je odrediti adrese početnih podataka s kojima se vrši računanje, u koliko se oni trebaju svi odjednom nalaziti u jedinici za pamćenje. Ako se obrađuje podatak po podatak odmah prilikom uvođenja, tada takva evidencija o podacima otpada. Evidencija adresa mora nadalje obuhvatiti sve pojedine međurezultate, jer se oni u toku računanja više puta koriste.

a) — Evidencija adresa programa. — Nakon što je odabrana početna adresa programa, može se pristupiti pisanju pojedinih naređenja. Svakom se naređenju odredi adresa, tj. redni broj, koji kontinuirano raste. Pod tom će adresom dotično naređenje biti pohranjeno u jedinici za pamćenje. Kako su programi linearan niz naređenja, to na mjestima grananja dolaze naređenja za skok. Skokovi se u programu mogu koristiti i iz drugih razloga. Kod svakog takvog naređenja za skok, mora se znati cilj skoka, tj. adresa nastavka programa. To je bitni razlog zbog kojeg se naređenja moraju numerirati tj. dodijeliti im adrese.

Nadalje je potrebno znati početne adrese svakog pojedinog potprograma. Potprogrami se koriste na taj način što se u glavnom programu nazove dotični potprogram. Kod toga se navede adresa početka potprograma. Time se vrši skok u potprogram. Nakon što je potprogram izvršio određene operacije, dolazi do povratnog skoka u glavni program.

b) — Evidencija adresa početnih podataka. — Pri izradi progarama potrebno je definirati adrese u kojima će se nalaziti početni podaci, s kojima će se vršiti rješavanje zadatka. Kod toga je poželjno da oni budu poredani u neprekinutom nizu, jer ih je na taj način najjednostavnije zadati automatu prilikom računanja. Pri izradi programa mora se za svaku pojedinu adresu tačno znati, kakav se podatak nalazi u dotičnoj ćeliji pamćenja. U tome je bit organizacije računanja. Sam automat nije u mogućnosti razlikovati vrstu podataka, npr. da li se radi o dužini ili kutu, i da li se radi baš o željenom kutu, dužini, koordinati ili sl. Zato je u naređenjima potrebno navesti adresu podatka s kojim se želi izvršiti neka računska operacija. Programer mora znati kakva se veličina u toj ćeliji pamćenja nalazi. Na taj je način omogućeno da se uvijek koristi željeni podatak. Stoga se kod sastava programa mora imati raspored podataka u jedinici pamćenja, da bi se program mogao izraditi s odgovarajućim naređenjima.

c) — Evidencija adresa rezultata i međurezultata. — Za vrijeme računanja javlja se niz međurezultata, koji se moraju privremeno pohraniti u ćelijama pamćenja, kako bi se u toku računanja opet mogli iskoristiti. Svaki se takav rezultat pohranjuje pod određenom adresom, koja se mora evidentirati. Pod tom je adresom taj podatak u svako vrijeme dohvatljiv. Evidencija adresa je potrebna, kako se u istu ćeliju ne bi pohranio i neki drugi podatak, čime bi se onaj prethodni izbrisao. Svrha je evidencije da ne bi došlo do pogrešnih računanja, tj. da se u račun ne bi uzimale pogrešne veličine.

5.4 Primjer

U slijedećem je primjeru prikazan program za određivanje dužine i smjernog kuta u Freiburškoj kodi za Zuse Z-23. Kod ovog automata mora prije svake aritmetske operacije prvi operand biti u ćeliji 6 a drugi u akumulatoru. Rezultat se nakon operacije nalazi u akmulatoru i ćeliji s adresom 6.

Plan adresa je tako sastavljen da se glavni program nalazi u adresama od 3500 dalje, potprogram za arc tg u adresama od 4000dalje, pot-

program za trigonometrijske funkcije u adresama 4500 dalje, početni se podaci (koordinate krajnjih tačaka) nalaze u adresama 3000—3003, a nastavno su ćelije za međurezultate.

Adresa	Naređenje	Značenje	Objašnenje
3000		Ya	
1		Xa	
2		Yb	
3		Xb	
4		(Yb—Ya)	
5		(Xb—Xa)	
6		ν	
3500	B6+3002	Yb→6	Dovedi veličinu iz 3002 u 6
1	B3000	Ya→a	Dovedi veličinu iz 3000 u akumulator
2	—	Yb—Ya→a,6	Odredi razliku
3	U3004	a→3004	Spremi rezultat u 3004
4	B6+3003	Xb→6	Dovedi veličinu iz 3003 u 6
5	B3001	Xa→a	Dovedi veličinu iz 3001 u akumulator
6	—	Xb—Xa→a,6	Odredi razliku
7	U3005	a→3005	Spremi rezultat u 3005
8	B6+3004	Yb—Ya→6	Dovedi veličinu iz 3004 u 6
9	:	Δy:Δx→a,6	Podijeli obje veličine
3510	F4000	arctgν→a	Naziv programa za arctg
1	D	ν	Ispiši dobiveni kut
2	U3006	a→3006	Spremi u 3006
3	B6+3004	Δy→6	Dovedi veličinu iz 3004 u 6
4	B3005	Δx→a	Dovedi veličinu iz 3005 u akumulator
5	—	Δy→Δx	Odredi razliku
6	QQE3523	a < o?	Skok na 3523 ako je rezultat negativan
7	B3006	ν→a	Dovedi veličinu u akumulator
8	F4500	sinν	Skok u potprogram za sinus
9	B6+3004	Δy→6	Dovedi veličinu iz 3004 u 6
3520	:	Δy:sinν→a,6	Podijeli obje veličine
1	D	d	Ispiši rezultat iz akumulatora
2	E3528		Skok na 3528
3	B3006	ν→a	Dovedi veličinu iz 3006 u akumulator
4	F4501	cosν→a	Skok u potprogram za cosinus
5	B6+3005	Δx→6	Dovedi veličinu iz 3005 u 6
6	:	Δx:cosν	Podijeli obje veličine
7	D	d	Ispiši rezultat iz akumulatora
3528	Zo	Stop	

5.5 Algol

U posljednje vrijeme sve se više nametao zadatak, izraditi takvu kodu koja će se praktički moći bez izmjene primijeniti na svim automatima, bez obzira na tip i firmu, a koja će biti što srodnija običnom matematskom izražavanju formula i jednadžbi. U tom su smislu bile izrađene neke alogaritamske kode, koje međutim nisu naišle na širi interes. Tek

je zadnjih godina došlo do izrade takvog međunarodnog jezika programiranja pod nazivom ALGOL. On je prihvaćen na jednom međunarodnom skupu matematičara. Tako su ovaj način programiranja prihvatile i mnoge firme. U tu su svrhu izradile odgovarajuće programe prevodioce, pomoću kojih se programi pisani u Algolu prevode prilikom očitavanja u internu kodu automata. Na taj je način došlo do sve veće primjene ovog načina programiranja. Objavljivanje i razmjena programa došla je time do većeg izražaja, jer je sada nađen zajednički jezik za različite automate.

Algol ima međutim nekoliko velikih mana. Za programe pisane u Algolu potreban je prilično velik i kompliciran prevodilac. On zauzima mnogo prostora u jedinici za pamćenje, što je kod manjih automata od presudne važnosti. Očitavanje takvih programa je zbog postupka prevođenja znatno sporije. Samo računanje je također znatno sporije, jer je mnoge organizacione funkcije preuzeo u ovom slučaju sam automat. Osim toga s ovakvim programom nije uvijek moguće postići sve ono, što je omogućeno s eksternom ili internom kodom, koja je prilagođena dotičnom tipu automata.

U algolu se zadatak ne razrađuje u pojedinačna naređenja, niti se moraju spominjati nekakve adrese. O adresama se brine sam utomat. Za računanje pojedinih veličina ispisuju se formule, po kojima se te veličine imaju sračunati. Kod toga se koriste uglavnom standardni matematski simboli za aritmetske operacije, sistem zagrada i neke standardne funkcije.

Program pisan u Algolu sadrži dva dijela: dio deklaracija i dio instrukcija (naređenja).

U dijelu deklaracija moraju biti nabrojane sve varijable koje se u programu javljaju. Time je programu prevodiocu omogućeno rezervirati za svaku veličinu po jednu ćeliju pamćenja. Varijable moraju kod toga biti razvrstane u tri grupe: cjelobrojčane (to su redovito indeksi), realne (kuda spadaju svi podaci i tražene veličine) i varijable s poljem (to su varijable s jednim ili više indeksa).

Dio instrukcija sadrži u prvom redu formule, po kojima se veličine trebaju sračunati. Tu se nadalje pružaju različite mogućnosti, kao što je: postavljanje uvjeta, vršenje skokova, slaganje ciklusa, korištenje potprograma tzv. procedura, itd.

Primjer programa u Algolu, za računanje malih tačaka na pravcu

```
»BEGIN« »COMMENT« RACUNANJE
    MALIH TACAKA;                    Komentar
    »REAL« YA, XA, YB, XB, D, A, O;  Realne varijable
    »INTEGER« N, I;                  Cjelobrojčane varijable
    »ARRAY« D, Y, X [1:30];          Varijable s poljem

    READ (YA, XA, YB, XB);           Očitaj koordinate
    PRINT (YA, XA, YB, XB);          Ispiši koordinate

    D:=SQRT((YB—YA)×(YB—YA)+
    (XB—XA)×(XB—XA));
```

```
PRINT(D);                          Ispiši dužinu
A:=(XB—XA)/D;                      Računanje koeficijenata
O:=(YB—YA)/D;                      a i o
READ(N);                           Očitaj broj tačaka n

»FOR«I:=1»STEP«1 UNTIL«N»DO«       Za i=1 do n, izvrši ope-
    »BEGIN«                        raciju koja je između
        READ(D[I]);                BEGIN i END
        Y[I]:=YA+OXD[I];           Računanje veličina
        X[I]:=XA+AXD[I];           yi i xi
        PRINT(D[I],Y[I],X[I];      Ispiši: di, yi i xi
    »END«;
»END«
```

Podaci na brojčanoj traci moraju biti pisani onim redom kako se to u programu zahtijeva; tj.: YA, XA, YB, XB, N, D_1, D_2, D_3, ... na pr.: 15649.33, 36066.62, 15605.46, 36023.15, 326.29, 42.05, 50.19

Rezultat računanja će tada imati izgled:

15649.33	36066.62	15605.46	36023.15
61.76			
26.29	15630.66	36048.12	
42.05	15619.46	36037.02	
50.19	15613.68	36031.29	

Dakle rezultati su ispisani u obliku tabele:

y_a	x_a	y_b	x_b
D			
d_1	y_1	x_1	
d_2	y_2	x_2	
d_3	y_3	x_3	

5.6 Priređivanje programa

Konačan program, sastavljen iz niza kodiranih naređenja, treba prenijeti na perforiranu traku, kartice ili magnetsku traku, već prema tome koji sistem koristi automat.

Kod sistema perforiranih traka program se redovito priređuje na posebnom stolu za programiranje. Na njemu se nalazi teleprinter, na čijoj se tastaturi ispisuju znak po znak, tj. naređenje po naređenje. Time se paralelno perforira traka, a na ispisanom protokolu može se kontrolirati ispravnost ispisanih simbola. Na istom je stolu moguće kopiranje takvih traka tj. njihovo umnažanje. Na isti način mogu se ispravljati trake. Kod ispravljanja se kopira onaj dio trake koji je ispravan, suvišni i pogrešni znakovi se ispuštaju, a potrebni novi znakovi dopisuju. Traka s programom mora biti oslobođena bilo kakve pogreške, jer i najmanja pogreška može u potpunosti promijeniti smisao programa, te dovesti do potpuno drukčijeg ili besmislenog toka operacija.

Sličan je postupak priređivanja i kod magnetnih traka i perforiranih kartica. Kod sistema kartica se na svaku karticu nanosi po jedno naređenje. Tako složeni niz kartica predstavlja tada program.

U priređivanju programskih traka ili kartica može biti korišten i sam automat. On se koristi kod pretvaranja programa u posebne sažete sisteme. Perforacije se kod tih sistema, preko uređaja za čitanje, direktno pretvaraju u električne impulse i binarne oznake, koje se jednostavno po redu pohranjuju u jedinici za pamćenje, te kod toga otpada postupak dešifriranja. U ovom sul slučaju trake kraće (kartice sadrže po više naređenja). Čitanje takvog programa vrši se s optimalnom brzinom.

Definitivno priređeni programi spremaju se i sačinjavaju tzv. biblioteku programa. Svaki program kod toga rješava jedan ili grupu određenih zadataka.

6. POSTUPAK RAČUNANJA S ELEKTRONSKIM RAČUNSKIM AUTOMATIMA

U biblioteci programa složeni su namoti s trakama ili nizovi kartica na kojima su nanešeni programi. Svaki program redovito obrađuje jedan određeni zadatak (npr. računanje poligonskog vlaka) ili grupu srodnih zadataka, što je ovisno o veličini tretiranog problema. Poželjno je da oni budu kompletni, tj. sa svim potrebnim potprogramima.

Prije samog računanja mora se u automat unijeti odgovarajući program. On ostaje sačuvan u jedinici za pamćenje, sve dok na njegovo mjesto ne dospije neki drugi program odn. neki podaci ili rezultati. Inače je moguće da istovremeno u automatu bude pohranjeno i po nekoliko (manjih) programa, dakako na različitim mjestima jedinice za pamćenje, tj. na različitim adresama. Program ostaje redovito sačuvan i nakon iskapčanja automata, tj. nakon isključenja struje.

Svaki program mora imati opis i uputstvo za upotrebu. U upustvu mora u prvom redu biti prikazan način priređivanja podataka. Podaci moraju, kao i programi, biti nanešeni na traci u obliku perforacija. Oni se pišu na teleprinteru, pri čemu se dobiva traka. Od naročite je važnosti redosljed podataka, kojim oni moraju biti pisani, a koji mora biti u skladu sa zahtjevima programa. Na početku trake mora biti naređenje s početnom adresom, počam od koje trebaju podaci da budu pohranjeni. Na kraju brojčane trake redovito se odmah navodi startna adresa programa (prvog naređenja), čime se odmah započinje računanje. Sve ovo mora biti navedeno u uputstvu.

Kod nekih se tipova zadataka uzima u računanje, tj. obradu, jedan po jedan podatak. U tom slučaju automat očita samo jedan podatak, izvrši računanje, pa tada drugi itd. Ovaj je sistem naročito u upotrebi kod sistema kartica, gdje se očita uvijek po jedna kartica s podacima koji se na njoj nalaze, npr. površina, kultura i klasa jedne katastarske čestice, iz kojih podataka stroj može npr. sračunati čisti katastarski prihod. U tom slučaju podatke nije potrebno pohraniti u jedinici za pamćenje.

Samo računanje je u potpunosti automatizirano. Rukovanje automatom, pisanje podataka, te posluživanje automata kod računanja, može

vršiti osoba bez naročitih kvalifikacija. Rezultati računanja ispisuju se na čisti papir (ili formular) sa svim naslovima, oznakama i komentarima, kako je to u programu predviđeno. Oni su redovito pisani u urednim tabelarnim oblicima, te otpada njihovo prepisivanje. Nadalje postoji mogućnost da se rezultati privremeno izdaju na trake odn. kartice, što je redovito mnogo brže i manje okupira sam računski automat. Takve se pak trake mogu naknadno ispisati u jasnom pismu pomoću drugih pomoćnih uređaja, npr. na stolu za programiranje.

Brzine, kojima automati računaju, ovisne su od vrste automata i primijenjene tehnike. One su nekoliko stotina ili tisuća puta veće nego kod upotrebe običnog računskog stroja. Kod manjih geodetskih zadataka računanja poligonskog vlaka, površina i sl.) vrijeme računanja jedva je primjetno. Ono se obično svodi samo na ispisivanje rezultata. Kod rješavanja velikog sistema jednadžbi (normalnih) vrijeme se može mjeriti u sekundama ili eventualno u minutama. Osim toga su podaci uredno ispisani, te otpada njihovo prepisivanje npr. kod predaje elaborata.

Sve ovo pruža nove mogućnosti u računskim postupcima, koje se do sada nisu koristile, zbog opsežnosti računanja koju ti postupci zahtijevaju.