Pásztor Attila

Algoritmizálás és programozás tankönyv az emeltszintű érettségihez

4. Kódolás

A **kódolás** során az algoritmust átültetjük valamely programozási nyelvre a specifikációt is szem előtt tartva. A programozási nyelv kiválasztásának számtalan szempontja lehet (melyik áll rendelkezésre, melyiket ismerjük, melyikben könnyű a feladatot megoldani,…), nekünk az előbbiekben felvázolt algoritmikus gondolkodáshoz legközelebb álló nyelvet érdemes választanunk.

A tanulmányaink során használt személyi számítógépek felépítése Neumann János elveit követik:
- kettes számrendszert használnak,
- az utasításokat szekvenciálisan hajtják végre (egy időben egy művelet),
- belső memóriájuk van,
- tárolt programmal működnek (a program is adat a memóriában eltárolva),
- univerzálisak (elvileg bármilyen feladat megoldására használhatók).

Az ilyen gépekre készült Neumann-elvű nyelvek jellemzői:
- a memória címezhető (sorszámmal),
- a program és az adatok a memóriában vannak,
- a végrehajtás nem más, mint memóriaállapotok sorozata,
- a program leírása szöveges.

Következmények:
- van változó, értékadás,
- utasítások ismételt végrehajtása és az elágazás is lehetséges,
- a beolvasás és a kiírás is memóriamásolás.

A mondatszerű algoritmusok egyszerű kódolhatósága miatt a pascal nyelvet választjuk programozási nyelvként. Természetesen választhatnánk mást (C, Basic, Visual Basic,…).

Léteznek nem Neumann-elvű nyelvek is, egyetlen konkrét példát szeretnék megemlíteni: az általános iskolában megismert LOGO nyelv egy automata-elvű nyelv (ilyen a festőrobot is).

Az automata-elvű nyelvek jellemzői:
- tevékenységorientált (állapotok vannak benne),
- az adatok állapotok illetve bemenetek,
- a végrehajtás állapotok sorozata,
- a program állapotátmenetet leíró függvény,
- a program elkülönül az adatoktól.

Következmények:
- rögzített adatszerkezet (állapotok),
- változó és értékadás nincs, csak paraméter (bár pld. a logo-ban sajnos lehetséges),
- egyszerű ciklusok (n-szer ismétlés, vagy állapotfüggő ismétlés),
- elágazás állapotfüggő, vagy paraméterfüggő,
- utasítások és eljárások paraméterezhetők,
- beolvasás nincs, állapot, vagy paraméter lekérdezhető,
- kimenet nincs, illetve az állapotváltozás maga az,
- párhuzamosság lehetséges.

Léteznek még funkcionális-elvű, logikai-elvű, objektum-elvű és adatfolyam-elvű nyelvek is, melyekre ebben a könyvben nem térünk ki.

4.1. A pascal programozási nyelv

A Neumann-elvű nyelvek közül oktatási célra a pascal a legalkalmasabb, rövid története az alábbi:

- Készítője Niklaus Wirth (Svájcban, a zurich-i egyetemen tanított fordítóprogram készítést),
- 1968-ban készült el az első változata az ALGOL60 nyelvre alapozva,
- a cél az volt, hogy könnyen lehessen fordítóprogramot írni (a programozási nyelv utasításait a végrehajtó egység által érthető utasításokká átalakítani),
- 1970-ben készült el az első pascal fordítóprogram,
- 1973-ban jelent meg az első szabvány,
- 1983-ban adták ki az újabb (szigorúbb) szabványt,
- 1983-ban jelenik meg az első PC-n futó Turbo Pascal változat (ma az 1972-ben készült Borland Pascal 7.0-t használjuk általában).

A pascal nyelv néhány jellemzője:

- Neumann-elvű, amatőr nyelv (nem nagy programrendszerek fejlesztésére készült),
- a program egy fordítási egység (a fordítási egység a program önállóan lefordítható legkisebb része), amely több programegységet tartalmazhat (a programegység egy részfeladatot old meg, egyben, egységként hajtható végre és paraméterezhető, ilyen az eljárás és a függvény),
- a paraméterek átadása történhet érték és cím szerint, az eljárásban történt érték módosítás hatásának megmaradásához a cím szerinti átadás szükséges,
- a nyelv alapszavai védettek (másra nem használhatók),
- sor ≠ utasítás, az utasítás elválasztó jel a pontosvessző,
- a típus-kompatibilitás név szerinti, mely az eljárások és függvények formális és aktuális paramétereinek megfeleltetésére és a kifejezésekre is vonatkozik: nem elegendő, hogy a típusok szerkezete azonos, a típusnévnek is azonosnak kell lennie,
- a változók típusát már a fordításkor tudni kell (erősen típusos),
- a memóriakezelés dinamikus: a felhasznált memória szükségleteknek megfelelően módosul, van kézi lefoglalás, törlés is,
- a változók hatásköre statikus (hatáskör: a programszöveg azon része, ahol a változó érvényben van), a deklarációt követően létezik,
- mindent előre definiálni kell.

Nézzük meg, hogyan lehet a mondatszerű leírással készült algoritmusokat pascal nyelvre kódolni! Az algoritmus és a pascal nyelv is egyformának tekinti a kis- és nagybetűket, csupán a szemléletesség növelésére használjuk. A következő részben táblázatos formában megadom a tanult algoritmus alapelemek pascal megfelelőjét (példával illusztrálva).
A pascal kódban jelezni fogom, hogy az utasítások elválasztására a pontosvessző karakter szolgál.

A TP azonosítói betűvel kezdődnek, és az angol abc betűit és a számokat tartalmazhatja (lehet benne néhány írásjel, például aláhúzás is). Használhatók a kisbetűk és a nagybetűk egyaránt, de a TP nem különbözteti meg őket.
Fontos már most tisztázni, hogy szövegként (magyarázat, szöveges változó értéke,…) használhatunk magyar ékezetes betűket, de készüljünk fel arra, hogy a DOS, a Windows, esetleg a Linux eltérő karakter kódolása miatt nem lesz a szöveg ékezethelyesen átvihető.

4.2. Átírási szabályok

Ebben a fejezetben megtanuljuk, hogyan lehet az algoritmusból pascal, pontosabban Borland Turbo Pascal 7.0 kódot készíteni.

4.2.1. A program

A program szerkezete	
ALGORITMUS	TURBO PASCAL KÓD

Program:
  változók és állandók deklarálása
  utasítások
Program vége.
	Program név;
	  deklarációk;
	Begin
	  utasítás_1;
	  …
	  utasítás_n;
	End.

4.2.2. Deklarációk

A szabvány szerint kötött sorrendű a deklarálás, de a TP (Turbo Pascal) szerint nem:

1. címdeklarációk
2. konstansdefiníciók
3. típusdefiníciók
4. változódeklarációk
5. eljárás- és függvénydeklarációk

Legfontosabb deklarációk	
ALGORITMUS	TURBO PASCAL KÓD

Konstans név=érték	Const név=érték; név=érték; …
	  CONST MAXN=20;
	TP ismeri a típusos konstanst:
	Const név:tipus=érték;
	  CONST MAXN:INTEGER=20;
	ebben az esetben az érték módosítható

Típus típusnév=típusdefiníció	type azonosító=típusmegadás; TYPE TombElem=Integer;
Változó név:típus	Var nev:tipus;

Egy típust használata előtt deklarálni kell!
Az eljárások és a függvények deklarálásáról később lesz szó.

4.2.3. Elemi adattípusok

Elemi adattípusok	
ALGORITMUS	TURBO PASCAL KÓD
egész Változó i:egész	Var i:Integer;
valós Változó x:valós	Var x:Real;
logikai Változó van:logikai	Var van:Boolean;
karakter Változó c:karakter	Var c:Char;
felsorolás	konstansok felsorolásával adható meg pld.: const hetvege=(szombat,vasarnap);
intervallum	határaival adható meg: pld.: 'a'..'z', vagy -10..10
mutató	csak típusos mutatót használjunk: pld.: var mut=^integer (egészre mutat)

A TP egész típusok specialitása

Típus	Tartomány	Méret (bit)
Shortint	-128..127	8
Integer	-32678..32767	16
Longint	-2147483648..2147483647	32
Byte	0..255	8
Word	0..65535	16
Minden egész típus megszámlálható.		

A TP valós típusok specialitása:

Típus	Tartomány	Pontosság (számjegy)	Méret (bájt)
Real	$2{,}9*10^{-39}..1{,}7*10^{38}$	11-12	6
Single	$1.5*10^{-45}..3{,}4*10^{38}$	7-8	4
Double	$5{,}0*10^{-324}..1{,}7*10^{308}$	15-16	8
Extended	$3{,}4*10^{-4932}..1{,}1*10^{4932}$	19-20	10
Comp	$-9{,}2*10^{18}..9{,}2*10^{18}$	19-20	8
A comp típus egy 64 bites egész valójában.			

A TP logikai típus specialitása:

A Boolean megszámlálható típus, valójában egy előre definiált felsorolt típus:
Type Boolean = (False, True);

A TP karakter típus specialitása:

Természetesen megszámlálható, az ASCII kódját tárolja, karakter konstansok két egyszerű aposztróf között adhatók meg (az aposztrófot két aposztrófként kell megadni, például:
'A', '3', vagy ''''

A TP mutató típus használata

A mutató tartalmának módosítása:
mut:=NIL; (* sehová nem mutató pointer *)
mut:=^i; (* a mut változó az i változó címét tartalmazza *)
mut^:=12; (* a mutató által címzett változó értéke lesz 12 *)

Ebben a könyvben nem fogunk mutatókat használni.

4.2.4. Összetett adattípusok

Összetett adattípusok	
ALGORITMUS	TURBO PASCAL KÓD

rekord
  Típus cikk=Rekord
  (cikkszam: egész,
  ár: valós)
	Type cikk=Record
	  cikkszam: Integer;
	  ar: Real;
	  end;

halmaz
  Típus számok=Halmaz(1..100)
  Típus betűk=Halmaz(karakter)
	set of típus;
	Type szamok= set of 1..100;
	Type betuk= set of char;

szöveg
  Változó név: szöveg
  Változó utca: szöveg(30)
	string
	Var nev:string; (* maximum 255 kar.*)
	Var utca:string[30];

sorozat
  Változó név(indextartomány:típus)
  Változó valami(1.10:valós)
	(* egydimenziós tömb *)
	Var nev:array[indextaromány] of típus;
	Var valami:array[1..10] of Real;

tömb
  Változó név(index1,..,indexn:típus)
példa:
  Változó mx(1..10,'a',,'z':valós)
	Var nev=array[index1,…,indexn] of típus;
	
	Var mx=array[1..10,'a'..'z'] of Real;

A TP rekordtípus használata:
A rekord egy mezőjére a pont segítségével lehet hivatkozni. A fenti példánál maradva:

Var cikkek:cikk;
….
cikkek.cikkszam:=12;
cikkek.ar:=123.50;
…

A TP halmaz típus specialitásai:

Az alaptípusnak megszámlálhatónak kell lennie, 256-nál nem több lehetséges értékkel. Az alaptípus alsó és felső határának 0..255 tartományban kell lennie!
[] az üres halmazt jelöli.
Halmaz felépíthető kifejezések szögletes zárójelbe tételével:
['0'..'9','A'..'Z','a'..'z']
[1, 5, i+1 .. j-1]
Felsorolt típusból is képezhető:
Type napok=(hetfo,kedd,szerda,csutortok,pentek,szombat,vasarnap);
Type naphalmaz= set of napok;

A TP string típus specialitásai:

Legfeljebb 255 karakter hosszú karaktersorozat.
Megadható hossz nélkül is.
Sorozatként lehet kezelni: 0. eleme a hossz, ezt követik a szöveg karakterei.
String konstans aposztrófok között adható meg:
’Ez egy szöveg kontans.’
Használható operátorok? +,-,<>,<,>,<=,>=

A TP sorozat típusa

Egydimenziós tömbként deklarájuk, tehát a tömböknél foglalkozunk vele..

A TP tömb típus specialitásai:

Az index típusnak megszámlálhatónak kell lennie, az elem típusa tetszőleges lehet (egyszerű, vagy összetett).
A tömb méretét a deklarációban meg kell adni, az nem változtatható.
Kerüljük a direkt tömbdeklarációt!

```
Var tomb:array[1..10] of Integer;
```

E helyett használjunk típust, és abból deklaráljuk a változót:

```
Type elemtipus=Integer;
Type tombtipus=array[1..10] of elemtipus;
Var tomb:tombtipus;
```

Miért érdemes ezt így bonyolítani?
A pascal név szerinti típusegyezést követel meg a műveletek, eljárások, függvények paramétereinél (nem elegendő a struktúra szerinti megegyezés), ezért például a tömbök közötti értékadás csak ebben az esetben valósítható meg

```
Var tomb1,tomb2:tombtipus;
a:=b;
```

Az előző értékadás így hajtható csak végre.
Többdimenziós tömbre példa:

```
Type elemtipus=Integer;
Type tombtipus=array[1..10,-10..10,’a’..’e’] of elemtipus;
Var tomb:tombtipus;

kar:char;
…
kar:=’d’;
Tomb[1,-5,kar]:=8;
…
```

4.2.5. Megjegyzések

Megjegyzések	
ALGORITMUS	TURBO PASCAL KÓD
[megjegyzés]	(* megjegyzés *) { ez is egy megjegyzés } { megjegyzések (* egymásba *) ágyazhatók}

Azt javaslom, hogy a kódolás folyamán a (* szöveg *) alakú megjegyzést használjuk, és a kapcsos zárójelet hagyjuk meg a teszteléshez: segítségével a programszöveg egy része megjegyzésbe tehető.
Példa az egymásba ágyazásra az előbbi TP kódban látható.

4.2.6. Elágazások

Elágazások	
ALGORITMUS	TURBO PASCAL KÓD
Egyágú elágazás Ha feltétel akkor utasítások	if feltétel then utasítás;
Kétágú elágazás Ha feltétel akkor utasítások_1 különben utasítások_2 Elágazás vége	if feltétel then utasítás_1 else utasítás_2;
Többágú elágazás Elágazás feltétel_1 esetén utasítások_1 … feltétel_n esetén utasítások_n egyéb esetén utasítások_m Elágazás vége	Case kifejezés of értéklilsta_1: utasítás_1; … értéklista_n: utasítás_n; else utasítás_m; End;

A TP elágazások specialitásai:

Az elágazások bármely ágában egyetlen utasítás lehet, ha többre van szükség, akkor azokat Begin … End közé kell tenni!

Példa:

```
If a>0 then
   Begin
      b:=a+1;
      c:=a-1;
   End;
```

Kétágú elágazásnál ügyelni kell arra, hogy amennyiben a then ágban több utasítás van, akkor az azokat egybefoglaló Begin … End végén nem lehet pontosvessző, mivel a pontosvessző utasítás lezáró karakter.

Példa:

```
If a>0 then
   Begin
      b:=a+1;
      c:=a-1;
   End (* nincs pontosvessző*)
Else
   Begin
      b:=a-2;
      c:=a+2;
   End;
```

A többágú elágazás kifejezése megszámlálható típusú eredményt szolgáltathat (tipikusan egész, vagy karakter). Az értéklista az értékek felsorolásával adható meg. Egy-egy ágon Begin … End között lehet több utasítás.
Ügyelni kell arra, hogy az egyes ágak egymást kizáróak legyenek!
Az ágak kiegészülnek egy egyébként ággal (nem minden nyelvben van ilyen).

Példa többágú elágazásra:

```
Case kar of (* var kar:char *)
   'a'..'z' : Begin
                    s:='kisbetű';
                    kb:=kb+1;
              End;
   '0'..'9' : Begin
                    s:='számjegy';
                    szj:=szj+1;
              End;
   '.',',','?','!','(',')' : s:='írásjel'
   else       Begin
                   s:='egyéb karakter';
                   e:=e+1;
              End;
```

4.2.7. Iterációk (ciklusok)

Iterációk (ciklusok)	
ALGORITMUS	TURBO PASCAL KÓD
számláló ciklus ciklus cv:=kezdet-től vég-ig utasítások ciklus vége	for cv:=kezdet to vég do utasítás;
elöltesztelő ciklus ciklus amíg feltétel utasítások ciklus vége	while feltétel do utasítás;
hátultesztelő ciklus ciklus utasítások amíg bennmaradási_feltétel ciklus vége	repeat utasítások; until kilépési_feltétel;

A TP ciklusok specialitásai:
A számláló ciklus ciklusváltozója csak megszámlálható típusú lehet.
Nincs lehetőség lépésköz megadására.
A ciklusváltozót a ciklusmagban nem szabad módosítani!
Ha lefelé kell számolni, akkor a kód az alábbi:

For cv:= kezdet downto vég do utasítás;

Ha több utasításból áll a ciklusmag, akkor azokat Begin … End közé kell tenni.

Példa:

```
For i:= 1 to N do
   Begin
      j:=j+i;
      k:=k*i;
   End;
```

Az elöltesztelő ciklus esetén is lehet több utasítás a ciklusmagban Begin … End között.
A ciklus előtt gondoskodjunk a feltétel beállításáról (lehetséges olyan eset, hogy már akkor hamis)!
A ciklusmagban szerepelnie kell olyan utasításnak, mely hat a feltételre, különben végtelen ciklus keletkezne.
A hátultesztelő ciklus feltételeként a kilépési feltételt kell megadni!
A hátultesztelő ciklusban több utasítást használva sincs szükség Begin … End szerkezetre.

Ciklusok átalakítása

Amennyiben az algoritmusban megfogalmazott ciklus az adott programozási nyelvben nem létezik, az algoritmus átalakításával kódolhatóvá válik. A következő táblázat erre mutat egy példát.

Iterációk (ciklusok) átalakítása egy példa alapján	
ALGORITMUS	TURBO PASCAL KÓD
kiinduló feladat: számláló ciklus ciklus i:=N-től 1-ig 2-vel utasítások ciklus vége	nem valósítható meg turbo pascalban számláló ciklussal, csak elöltesztelővel

elöltesztelő ciklussal:
i:=N
ciklus amíg i>=1
utasítások
i:=i-2
ciklus vége
	i:=N;
	while i>=1 do
	Begin
	utasítások;
	i:=i-2;
	End;

hátultesztelő ciklussal:
Ha N>=i akkor
Ciklus
utasítások
i:=i-2
amíg i>=1
ciklus vége
Elágazás vége
	if N>=i then
	Repeat
	utasítások;
	i:=i-2;
	until i<1;

4.2.8. Eljárások, függvények

Eljárások és függvények	
ALGORITMUS	TURBO PASCAL KÓD

Eljárás deklarációja:
Eljárás eljnév(formális paraméterek):
deklarációk
utasítások
Eljárás vége.
	Procedure eljnev(formális paraméterek);
	deklarációk;
	Begin
	utasítások;
	End;

Eljárás meghívása: eljárásnév(aktuális paraméterek):	procnev(aktuális paraméterek);

Függvény deklarációja:
Függvény fnév(formális paraméterek):
függvényérték típusa
deklarációk
utasítások
fnév:=kifejezés
Függvény vége.
	Function fnev(formális paraméterek):
	függvényérték típus;
	deklarációk;
	Begin
	utasítások;
	fnev:=kifejezés;
	End;

Függvény meghívása: azonosító:=fnév(aktuális paraméterek)	azonosító:=fnev(aktuális paraméterek);

Figyelem! Turbo Pascalban a deklarációnál a formális paramétereket pontosvesszővel kell elválasztani, de a meghívásnál az aktuális paramétereket vesszővel!

4.2.9. Beolvasás és kivitel

Beolvasás billentyűzetről, kiírás monitorra	
ALGORITMUS	TURBO PASCAL KÓD
Kiírás: Ki: azonosítók [formátum megkötések]	Write(kifejezések); (*kifejezések vesszővel elválasztva*) (* a kurzor az aktuális helyen marad *) WriteLn(kifejezések); (* a kurzor a köv. sor elejére kerül *) A formátum megkötéssel most nem foglalkozunk.
Beolvasás: Be: azonosítók [feltételek]	ReadLn(azonosítók); Read(atonosítók); (* végén kell ENTER *) elegánsabb egyesével, tájékoztatással: Write('Adja meg … értékét: '; ReadLn(azonosító); A beolvasási feltételekkel most nem foglalkozunk.

Alkalmazási példa:

```
Var a: string;
       i: integer;
…
Writeln('Adja meg a szöveget, és egész számot ENTER-rel zárva');
ReadLn(a,i);
WriteLn(a,i);
```

Gyakran ellenőrizni kell, hogy a beolvasott érték egy adott tartományban van-e. Ennek megvalósítása:

```
Repeat
    Write('Adjon meg egy számot 1 és 10 között: ');
    ReadLn(i);
until (i>=0) and (i<=10);
```

Megfigyelhető, hogy az until után a kilépési feltételt kellett megadni, valamint a TP műveleti precendenciája miatt a logikai művelettel összekapcsolt relációkat zárójelezni kell.

Létezik arra is megoldás, hogy ellenőrizzük, hogy a megfelelő típusú adatot adott-e meg a felhasználó (például egész helyett nem gépelt-e be valós számot). Az érettségire készülve feltételezhetjük, hogy a felhasználó a megfelelő típusú adatot adja meg.

4.3. Aritmetikai műveletek

Jel	Jelentés	Példa
+	pozitív előjel	+a
-	negatív előjel	-a
+	összeadás	a+b
-	kivonás	a-b
*	szorzás	a*b
/	(valós) osztás	a/b
div	(egész) osztás	a div b
mod	(egész) osztás maradéka	a mod b

4.4. Relációs műveletek

A műveletek eredménye logikai típusú.

Jel	Jelentés	Példa
=	egyenlő	a=b
<>	nem egyenlő	a<>b
<	kisebb	a<b
>	nagyobb	a>b
<=	kisebb, vagy egyenlő	a<=b
>=	nagyobb, vagy egyenlő	a>=b

4.5. Logikai műveletek

Egyváltozós logikai operátorok

NOT	bitenkénti negálás (egyes komplemens)
SHL	bitenkénti balra tolás
SHR	bitenkénti jobbra tolás

Kétváltozós logikai operátorok

AND	bitenkénti és (igaz, ha mindkét operandus igaz)
OR	bitenkénti vagy (igaz, ha valamelyik operandus igaz)
XOR	bitenkénti kizáró vagy (igaz, ha pontosan az egyik operandus igaz)

4.6. Precedenciák

A Turbo Pascal 7.0 a műveleteket az alábbi prioritással kezeli:

Precedencia	Művelet	Jele	Példa
1.	előjelváltás	-	-A
	tagadás	NOT	NOT(A)
2.	szorzás	*	A*B
	valós osztás	/	A/B
	egész osztás	DIV	A DIV B
	maradékképzés	MOD	A MOD B
	és	AND	A AND B
3.	összeadás	+	A+B
	kivonás	-	A-B
	vagy	OR	A OR B
	kizáró vagy	XOR	A XOR B
4.	egyenlő (reláció)	=	A = B
	nem egyenlő	<>	A <> B
	kisebb	<	A < B
	nagyobb	>	A > B
	kisebb egyenlő	<=	A <= B
	nagyobb egyenlő	>=	A >= B
	eleme (halmaznál)	IN	A IN B

Az azonos prioritású műveletek kiértékelése balról jobbra történik. Természetesen zárójel használata esetén először a zárójeles kifejezés kerül kiértékelésre. Jól látható, hogy a logikai műveletek magasabb prioritásúak a relációknál, ezért figyelni kell a zárójelezésre!

4.7. Numerikus függvények

Kivonat a leggyakrabban használt TP numerikus függvényekből:

Függvény	Jelentés	Megjegyzés
ABS(X)	X abszolút értéke	az eredmény x-szel megegyező típusú
ArcTan(X)	X arkusztangense	valós paraméter, valós eredmény
Cos(X)	X koszinusza	valós paraméter, valós eredmény
Exp(X)	X par. e alapú hatványa	valós paraméter, valós eredmény
Frac(X)	X törtrésze	X és az eredmény is valós
Int(X)	X egészrésze	X és az eredmény is valós
Ln(X)	X 10-es alapú logaritmusa	X és az eredmény is valós
Odd(X)	X páratlan-e	X egész, eredménye True, ha páratlan
Pi	a pi értéke	valós
Random	véletlen szám [0,1) között	valós
Random(X)	véletlen szám [0,X) között	valós
Randomize	vél.generátor újraindítása	Csak egyszer használjuk!
Round(X)	X egészre kerekítve	az eredmény egész
Sin(X)	X szinusza	valós paraméter, valós eredmény
Sqr(X)	X négyzete	az eredmény x-szel megegyező típusú
Sqrt(X)	X négyzetgyöke	X és az eredmény is valós
Trunc(X)	X egészrésze	az eredmény egész (levágva)

4.8. Sorszámozott típus eljárásai, függvényei

Kivonat a TP kizárólag sorszámozott típusaira használható eljárásaiból és függvényeiből:

Eljárás	**Jelentés**
Dec(X)	X-et 1-gyel csökkenti (sorszámozott számtípus)
Dec(X,n)	X-et n-nel csökenti (sorszámozott számtípus).
Inc(X)	X-et 1-gyel növeli (sorszámozott számtípus)
Inc(X,n)	X-et n-nel növeli (sorszámozott számtípus)..
Pred(X)	X-be teszi az X-et megelőző értéket.
Succ(X)	X-be teszi az X-et követő értéket.

Függvény	**Jelentés**
Ord(X)	X sorszáma (például karakternél az ASCII kódja).
Chr(X)	Az X ASCII kódú karakter.

4.9. Szövegkezelő függvények

Kivonat a leggyakrabban használt TP szövegkezelő függvényekből

Eljárás	**Jelentés**
Delete(szöveg,index,darab)	A szöveg index-edik karakterétől db-nyi karaktert kivág.
Insert(minta,szöveg,index)	A szöveg index-edik karakterétől kezdve beszúrja a mintát.
Pos(minta,szöveg)	szövegben a minta hányadik karaktertől található meg 0, ha nem illeszthető a minta a szövegre.
Str(szám:szélesség:tizedesek:szöveg)	A számot szöveggé alakítja a megadott mezőszélességgel és tizedes jegy darabszámmal. A tizedesek száma elhagyható.
VAL(szöveg,számtípusú_változó,hibakód)	A szövegesen ábrázolt számot a számtípusú_változóba teszi, ha valamelyik pozíción hiba volt, akkor hibakódban jelzi.

Függvény	**Jelentés**
Concat(S1,S2,…)	Összefűzött szöveggel tér vissza.
Copy(szöveg,index,db)	A szöveg index-edik karakterétől db-nyi karaktert tartalmazó szöveget ad vissza.
Length(S)	A szöveg karakterekben mér hosszával tér vissza.
UpCase(karakter)	A karaktert nagybetűvé alakítja – természetesen csak az angol abc betűire működik.

4.10. Összefoglalás

A fejezet első részében megismertük a Neumann elvű nyelvek jellemzőit, utaltunk a Logo, mint automata-elvű nyelv jellemzőire is. Ezt követően megismertük a mondatszerű algoritmus elemek TP-ra történő átírásának szabályait. Úgy gondolom, hogy hasznos lenne arról a néhány oldalról másolatot készíteni, hogy az első néhány program elkészítésénél majd segítségedre legyen.

Ezt követően a leggyakrabban használatos műveleteket ismertük meg, majd a műveletek kiértékelésének sorrendjét.

Összegyűjtöttük a leggyakrabban használt eljárásokat és függvényeket is.

Ez a fejezet inkább kézikönyvként használható a későbbi feladatokhoz.

4.11. Gyakorló feladatok

- Készítsd el a második fejezetben lévő mondatszerű algoritmus készítésre vonatkozó feladatok kódolását pascal nyelven!
- Próbáld meg a mondatszerű algoritmus leírásról szóló fejezet végén (18.oldal) lévő feladatokat kódolni!
- Próbáld meg kódolni a 22. oldalon lévő folyamatábrás feladatokat (ha kell, előtte tárj át mondatszerű leírásra)!
- Próbáld meg kódolni a 24. oldalon lévő struktogramos feladatokat (ha kell, előtte térj át mondatszerű leírásra)!