ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ
ՀԱՅԱՍՏԱՆԻ ՊԵՏԱԿԱՆ ՃԱՐՏԱՐԱԳԻՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ
(ՊՈԼԻՏԵԽՆԻԿ)

Քոմքյութերային համակարգերի և
ինֆորմատիկայի դեպարտամենտ
Ալգորիթմական լեզուների և
ծրագրավորման ամբիոն

PASCAL ծրագրավորում

ՄԵԹՈԴԱԿԱՆ ՑՈՒՑՈՒՄՆԵՐ ԵՎ ՍՏՈՒԳՈՂԱԿԱՆ ԱՌԱՋԱԴՐԱՆՔՆԵՐ

Երևան 2007

ՀՏԴ 681.3.06

Ստորագրված է տպագրության 26.12.06թ.

Կազմողներ՝ Ս.Ս.Ավետիսյան,
Ռ.Վ. Աղգաշյան,
Ս.Վ.Դանիելյան

Pascal ծրագրավորում: Մեթոդական ցուցումներ և ստուգողական առաջադրանքներ: Եր., Հայաստանի պետական ճարտարագիտական համալսարան, 2007.- 132 էջ:

Նախատեսված են ՀՊՃՀ բակալավրի կրթական ծրագրով սովորող բոլոր ուսանողների համար:

Գրախոս՝ Գ.Վ.Ադամյան

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

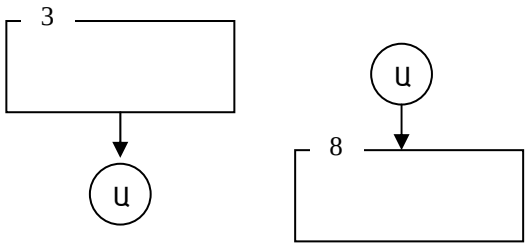
1. ՀԱՇՎՈՂԱԿԱՆ ԳՈՐԾԸՆԹԱՑԻ ԱԼԳՈՐԻԹՄԱՑՈՒՄ

1.1. Հիմնական հասկացություններ

Ալգորիթմը գործողությունների այնպիսի կարգավորված հաջորդականություն է, որը վերջավոր քայլերի ընթացքում հանգեցնում է դրված խնդրի լուծմանը:

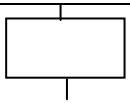
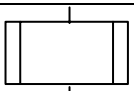
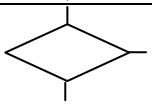
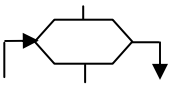
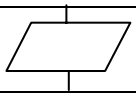
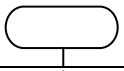
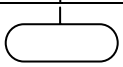
Ալգորիթմի ներկայացումը բլոկ-սխեմայի տեսքով ալգորիթմների ներկայացման ամենադիտողական եղանակն է: Այս դեպքում ալգորիթմը ներկայացվում է հատուկ բլոկների միջոցով, որտեղ յուրաքանչյուր բլոկ իրականացնում է խնդրի լուծման որոշակի փուլ: Աղյուսակ 1-ում տրված են որոշ բլոկների նկարագրությունները և նրանց միջոցով իրականացվող գործողությունները: Բլոկի ներսում գրվում է ինֆորմացիա՝ կատարվող գործողության մասին: Ամեն մի բլոկ կարող է ունենալ իր համարը, որը դրվում է բլոկի եզրագծի ընդհատված վերին ձախ անկյունում: Ալգորիթմի սխեմայում հաշվողական պրոցեսի ընթացքը ցույց է տրվում բլոկների միջև եղած կապի գծերով, որոնց ուղղությունը անպայման ցույց է տրվում սլաքով այն դեպքում, երբ նրանք ուղղված են աջից ձախ կամ ներքևից վերև: Մնացած դեպքերում սլաքները կարելի է չդնել: Ալգորիթմի սխեմայում հոսքի ընդհատման դեպքում ընդհատված գծի ծայրերում պետք է դնել միացուցիչ, որի ներսում դրվում է որևէ պայմանանշան:

Օրինակ՝



Ըստ ալգորիթմում կատարվող գործողությունների կապերի բնույթի՝ տարբերակում են գծային, ճյուղավորված և ցիկլային ալգորիթմներ:

Աղյուսակ 1.

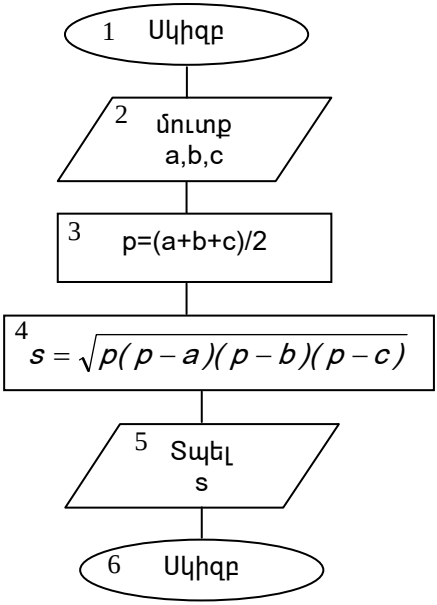
Բլոկի անվանումը	Բլոկի նշանակումը	Կատարվող ֆունկցիան
Պրոցես		Հաշվողական գործողություն կամ գործողությունների հաջորդականություն
Կանխորոշված պրոցես		Նախապես ստեղծված և առանձին նկարագրված ալգորիթմի օգտագործում
Լուծում		Պայմանի ստուգում և հաշվողական պրոցեսի շարունակության ընտրություն
Մոդիֆիկացիա		Ցիկլային պրոցեսի կազմակերպում
Մուտք-ելք		Տվյալների ներմուծում և արտածում
Սկիզբ		Ալգորիթմի սկիզբ
Ավարտ		Ալգորիթմի ավարտ
Միացուցիչ		Հոսքի ընդհատված գծերի միջև կապի ցուցիչ

1.2. Գծային ալգորիթմներ

Գծային ալգորիթմը իրար հաջորդող գործողությունների հաջորդականություն է, որոնք նախնական տվյալների բոլոր թույլատրելի արժեքների դեպքում կատարվում են խիստ միևնույն բնական հաջորդական կարգով:

Օրինակ 1.1: Կազմել բլոկ-սխեմա, որը կհաշվի տրված a, b, և c կողմերով եռանկյան մակերեսը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 1.1-ում:



Նկ. 1.1

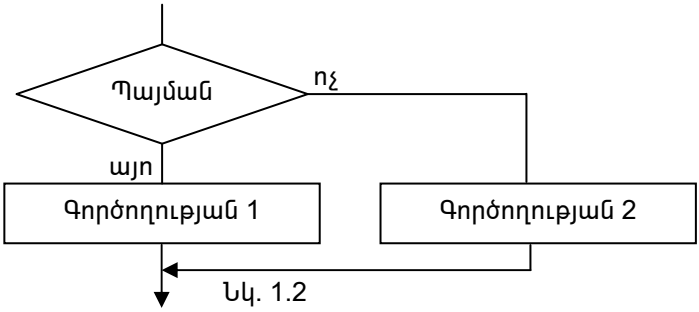
Տրված եռանկյան մակերեսը կարելի է հաշվել՝ օգտվելով Հերոնի բանաձևից: Նախ անհրաժեշտ է հաշվել տրված կողմերով եռանկյան կիսապարագիծը (p), իսկ այնուհետև՝ մակերեսը.

$$p = \frac{a+b+c}{2},\ s = \sqrt{p(p-a)(p-b)(p-c)}:$$

Բլոկ 2-ում ներմուծվել են եռանկյան կողմերի երկարությունները: Եռանկյան p կիսապարագիծը հաշվվել է 3-րդ բլոկում, իսկ s մակերեսը՝ 4-րդ բլոկում: Բլոկ 5-ում արտածվել է եռանկյան մակերեսի արժեքը:

1.3. Ճյուղավորված ալգորիթմներ

Ճյուղավորված ալգորիթմը պարունակում է լուծման բլոկ, որի իսկությունից կախված՝ իրականացվում են գործողությունների տարբեր հաջորդականություններ, որոնք կոչվում են ճյուղեր (նկ. 1.2):



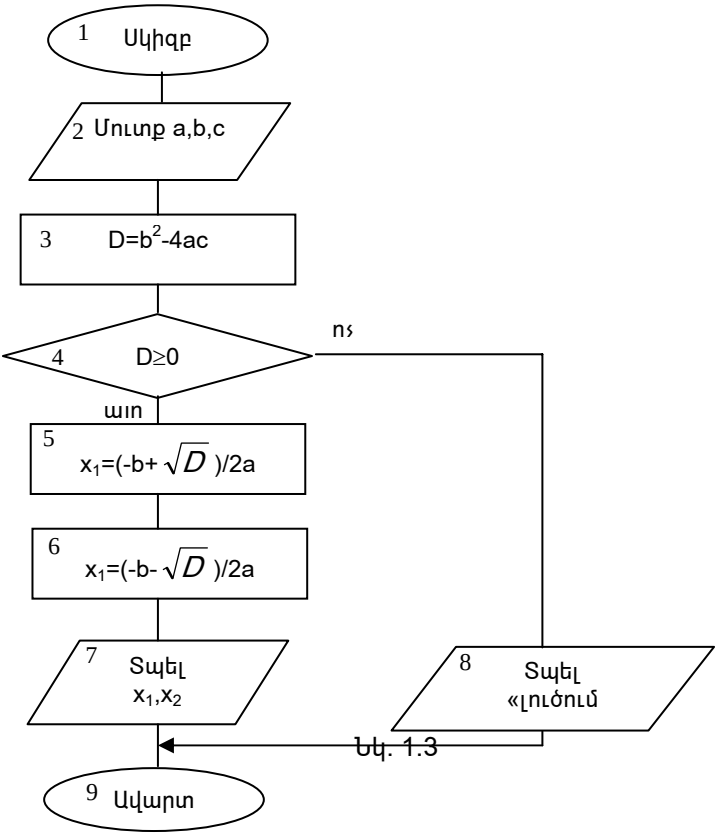
Նկ. 1.2

Սկզբում ստուգվում է գրված պայմանը, եթե այն ստանում է ճշմարիտ արժեք, ապա կատարվում է գործողություն 1-ը և բաց է թողնվում գործողություն 2-ը, իսկ եթե պայմանը ստանում է կեղծ արժեք, ապա ընդհակառակը՝ կատարվում է գործողություն 2-ը , բաց է թողնվում գործողություն 1-ը:

Օրինակ 1.2: Կազմել $ax^2+bx+c=0$ հավասարման լուծման ալգորիթմի բլոկ-սխեման:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 1.3-ում:

Բլոկում 2-ում ներմուծվում են a, b, c փոփոխականների արժեքները: 3-րդ բլոկում հաշվվում է հավասարման տարբերիչը: 4-րդ բլոկում տարբերիչը համեմատվում է 0-ի հետ: Եթե այն մեծ կամ հավասար է 0-ից, ապա կատարվում են 5-րդ և 6-րդ բլոկները, այսինքն՝ հաշվվում են հավասարման x_1, x_2 արմատները և դրանց արժեքները տպելուց հետո (7-րդ բլոկ) կառավարումը փոխանցվում է 9-րդ բլոկին, որն ավարտում է ալգորիթմը: Եթե 4-րդ բլոկում պարզվել է, որ տարբերիչը մեծ կամ հավասար չէ 0-ից, ապա նշանակում է հավասարումը իրական արմատներ չունի: Այդ դեպքում 8-րդ բլոկում տպվում է «լուծում չունի» հաղորդագրությունը և ալգորիթմն ավարտվում է:



Նկ. 1.3

1.4. Ցիկլային ալգորիթմներ

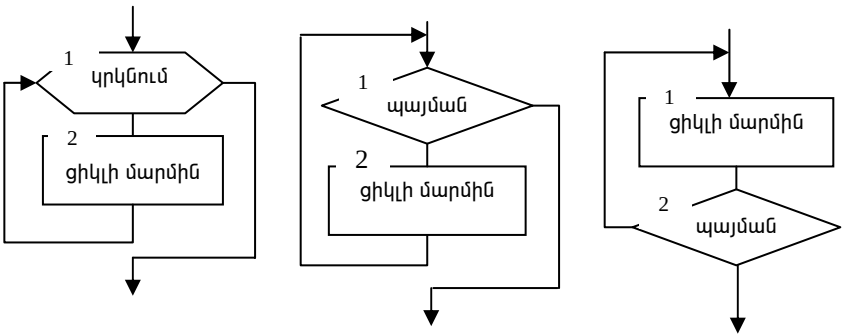
Գործնական խնդիրների լուծման ժամանակ հաճախ հանդիպում են գործողությունների որոշակի հաջորդականություններ, որոնք անհրաժեշտ է լինում կրկնել մեկից ավելի անգամ՝ մուտքային որոշակի պարամետրերի տարբեր արժեքների դեպքում: Այդ նպատակով կիրառվում են ալգորիթմական հատուկ կառուցվածքներ, որոնք կոչվում են **ցիկլային** (շրջափուլային): Ցիկլում կատարվող գործողությունները (մեկ կամ ավելի) անվանում են ցիկլի մարմին:

Ցիկլային կառուցվածքների դասակարգման համար կարևոր չափանիշը կրկնությունների թվի նախապես հայտնի կամ անհայտ լինելն է: Առաջին դեպքում, երբ հայտնի են մուտքային պարամետրերի

նախնական և **վերջնական** արժեքները, գործ ունենք, այսպես կոչված, կառուցվածքային ցիկլի հետ, որի ներկայացման համար բլոկ-սխեմայում կիրառվում է վերը բերված մոդիֆիկացիայի բլոկը: Երկրորդ դեպքում, երբ հայտնի չէ ցիկլի պարամետրի վերջնական արժեքը, հետևաբար և կրկնությունների թիվը, գործ ունենք, այսպես կոչված, իտերացիոն ցիկլային կառուցվածքի հետ: Այսպիսի կառուցվածքների ներկայացումը մոդիֆիկացիայի բլոկով, բնականաբար, անհնար է, քանի որ վերջինս պահանջում է ցիկլի պարամետրի ինչպես նախնական, այնպես էլ վերջնական արժեքների պարտադիր նշում:

Ըստ իրագործման տեխնիկական միջոցների՝ կարելի է առանձնացնել երեք հիմնական տիպի մոտեցումներ:

ա) պարամետրով ցիկլային կառուցվածք (նկ. 1.4. ա)),
բ) նախապայմանով ցիկլային կառուցվածք (նկ. 1.4. բ)),
գ) հետպայմանով ցիկլային կառուցվածք (նկ. 1.4. գ)):



ելք ցիկլից
ա)

ելք ցիկլից
բ)

ելք ցիկլից
գ)

Նկ. 1.4

Պարամետրով ցիկլային ալգորիթմը կազմակերպում է մեկ կամ մի շարք գործողությունների ցիկլային կատարումը՝ նախապես հայտնի քանակությամբ անգամ: Սկզբում ցիկլի պարամետրին վերագրվում է նրա նախնական արժեքը, որի համար կատարվում են ցիկլի մարմնում գրված բոլոր գործողությունները: Այնուհետև ցիկլի պարամետրի արժեքը փոխվում է նշված քայլով, որի համար նույնպես կատարվում է ցիկլի

մարմինը: Կրկնումները շարունակվում են այնքան ժամանակ, քանի դեռ ցիկլի պարամետրը չի գերազանցել իր համար նշված վերջին արժեքը:

Նախապայմանով ցիկլային ալգորիթմը կազմակերպում է ցիկլի մարմնում գրված գործողությունների կատարումը նաև նախապես անհայտ քանակությամբ անգամ: Ցիկլի այս կառուցվածքի իմաստն այն է, որ քանի դեռ ճշմարիտ է առաջին բլոկում գրված պայմանը, ապա կատարվում է ցիկլի մարմինը: Ցիկլը ավարտվում է, եթե գրված պայմանը ստանում է կեղծ արժեք: Քանի որ ցիկլի մարմնի կատարումը կախված է առաջին բլոկում գրված պայմանից, ապա հնարավոր է, որ ցիկլի մարմնում գրված գործողությունները չկատարվեն և ոչ մի անգամ:

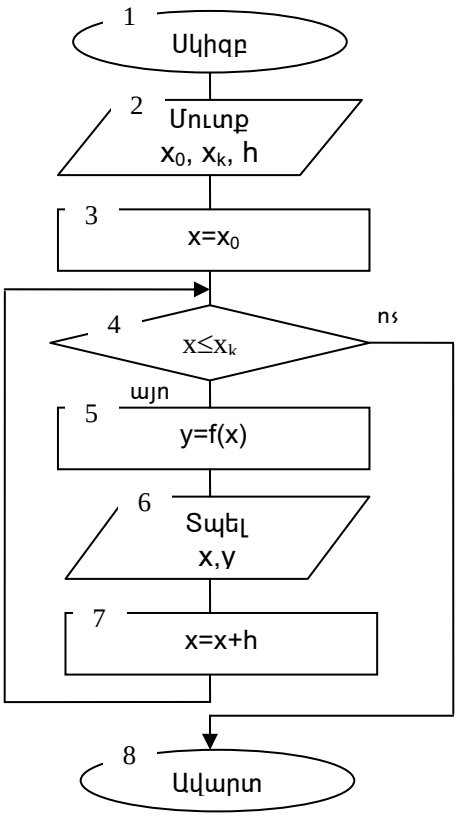
Հետպայմանով ցիկլային ալգորիթմը կազմակերպում է ցիկլի մարմնում գրված գործողությունների կատարումը նաև նախապես անհայտ քանակությամբ անգամ: Հետպայմանով ցիկլային կառուցվածքներում կրկնումները շարունակվում են այնքան ժամանակ, քանի դեռ կեղծ է 2-րդ բլոկում գրված պայմանը: Քանի որ ցիկլի մարմնում գրված գործողությունների կատարումից հետո է միայն ստուգվում ցիկլի կրկնման պայմանը, ապա ցիկլի այս կառուցվածքը կոչվում է հետպայմանով, և անկախ ցիկլի ավարտի պայմանի իսկությունից ցիկլի մարմինը կկատարվի գոնե մեկ անգամ:

Օրինակ 1.4: Կազմել y=f(x) ֆունկցիայի արժեքների հաշվման ալգորիթմը, երբ x-ը փոխվում է սկզբնական x_0 արժեքից մինչև վերջնական x_k արժեքը h հաստատուն քայլով:

Խնդիրը կարելի է լուծել նշված երեք կառուցվածքներով: Դիտարկենք խնդրի լուծման ալգորիթմը նախապայմանով ցիկլային կառուցվածքի միջոցով:

Ալգորիթմի բլոկ-սխեման բերված է նկ. 1.5-ում:

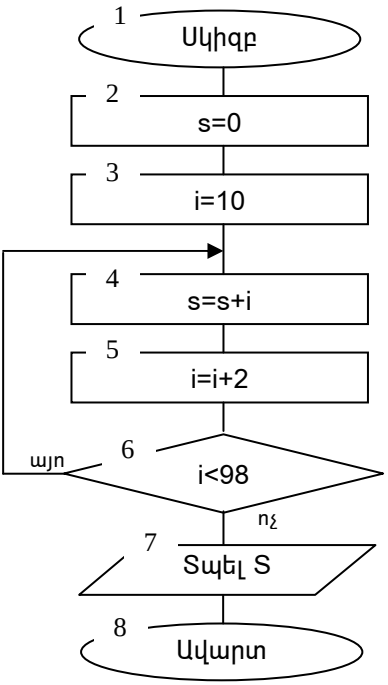
3-րդ բլոկում կատարվում է ցիկլի նախապատրաստումը՝ x փոփոխականին վերագրվում է սկզբնական x_0 արժեքը: Բլոկ 4-ում ստուգվում է ցիկլի ավարտի պայմանը: Եթե $x \le x_k$, ապա ցիկլը կրկնվում է, հակառակ դեպքում ցիկլն ավարտվում է: 5-ից 7-րդ բլոկները հանդիսանում են ցիկլի մարմինը: 5-րդ բլոկում x փոփոխականի ընթացիկ արժեքի համար y ֆունկցիայի արժեքի հաշվումից և 6-րդ բլոկում x և y փոփոխականների արժեքների արտածումից հետո բլոկ 7-ում x-ի ընթացիկ արժեքին վերագրվում է x+h արժեքը, այսինքն՝ այն մեծացվում է h քայլի արժեքով:



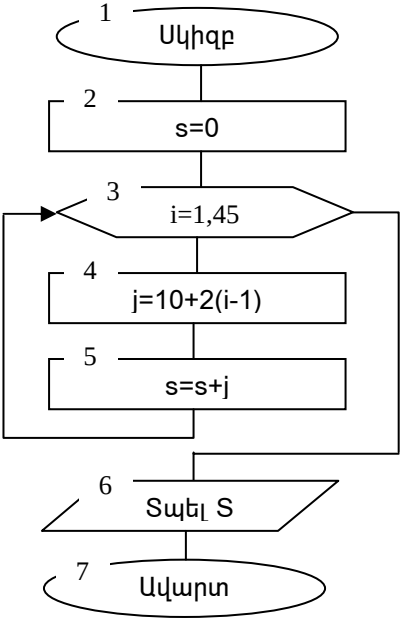
Նկ. 1.5

Օրինակ 1.5: Կազմել զույգ երկնիշ թվերի գումարի հաշվման ալգորիթմի բլոկ-սխեման:

Խնդրի լուծման երկու՝ հետպայմանով և պարամետրով ցիկլային ալգորիթմների բլոկ-սխեմաները պատկերված են համապատասխանաբար նկ. 1.6-ում և նկ.1.7-ում:



Նկ. 1.6

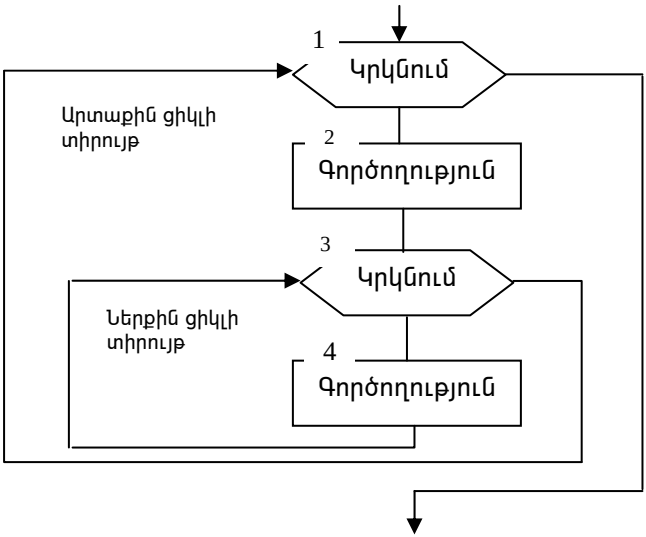


Նկ. 1.7

Գումարը կազմակերպելու համար ալգորիթմում օգտագործվում է s փոփոխականը, որին 2-րդ բլոկում վերագրվել է 0 նախնական արժեքը: Ցիկլի պարամետրի ամեն մի արժեքի համար հերթական զույգ երկնիշ թիվը գումարվում է s փոփոխականին: Նկ. 7-ում պատկերված ալգորիթմը իրականացված է հետպայմանով ցիկլի միջոցով, իսկ նկ. 8-ում պատկերվածը՝ պարամետրով ցիկլի միջոցով: Քանի որ երկնիշ թվերի քանակը 90 է, իսկ զույգ երկնիշ թվերի քանակը՝ 45, ապա պարամետրով ցիկլի կրկնության թիվը վերցված է 45: Ցիկլի մարմնի 4-րդ բլոկում մտցված է նոր j պարամետր, որը i պարամետրից կախված հաջորդաբար ընդունում է զույգ երկնիշ թվերին հավասար արժեքներ:

1.5. Ներդրված ցիկլերով ալգորիթմներ

Ցանկացած ցիկլ իր մեջ կարող է ընդգրկել մեկ կամ մի քանի այլ ցիկլեր: Ցիկլերի այդպիսի կառուցվածքը կոչվում է ներդրված ցիկլերով կառուցվածք (նկ. 1.8):

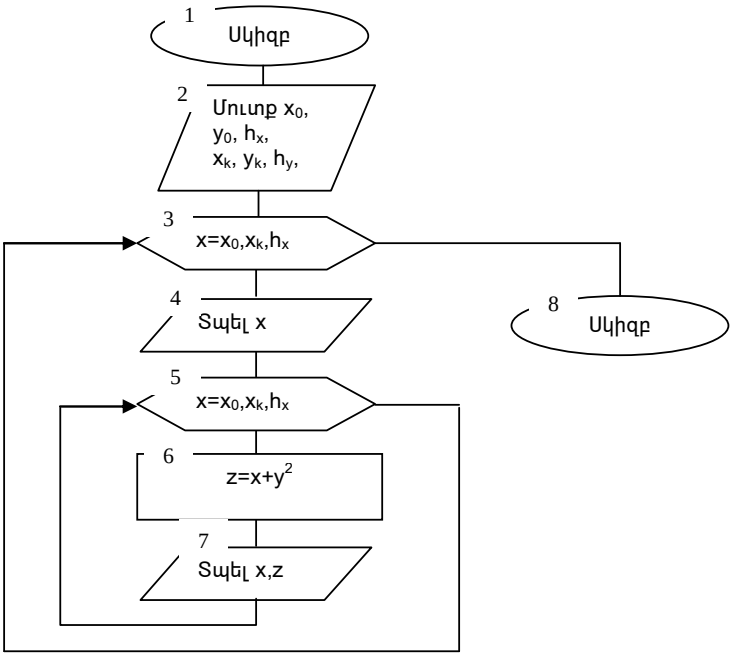


Նկ. 1.8

Օրինակ 1.6: Կազմել $z=x+y^2$ ֆունկցիայի արժեքների հաշվման ալգորիթմի բլոկ-սխեման, երբ x արգումենտը h_x քայլով փոփոխվում է x_0-ից մինչև x_k, իսկ y արգումենտը h_y քայլով y_0-ից մինչև y_k:

Խնդրի լուծման ալգորիթմը կազմելու համար անհրաժեշտ է՝ արգումենտներից մեկի ֆիքսված արժեքի համար (օրինակ x) կազմակերպել մյուս (y) արգումենտի արժեքի փոփոխման ցիկլ և հաշվել z ֆունկցիայի համապատասխան արժեքները: Այդ ցիկլի ավարտից հետո տրված քայլով պետք է փոփոխել ֆիքսված x արգումենտի արժեքը և նորից անցնել մյուս y արգումենտի փոփոխման ցիկլին: Գործընթացը կրկնել այնքան, քանի դեռ x արգումենտը դուրս չի եկել տրված միջակայքից:

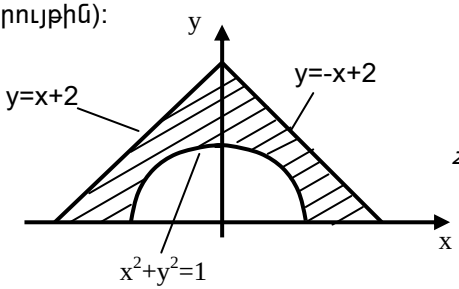
Այստեղ y պարամետրով ցիկլը կլինի ներքին, իսկ x պարամետրով ցիկլի տիրույթը՝ արտաքին: Խնդրի լուծման ալգորիթմի բլոկ-սխեման բերված է նկ. 1.9-ում:



Նկ. 1.9

1.6. Տիպային օրինակներ

Օրինակ 1.7: Տրված են կետի x և y կոորդինատները: D-ն նկարում ընդգծված տիրույթն է՝ եզրագծերով հանդերձ: Կազմել բլոկ-սխեմա, որը կհաշվի և կտպի տրված ֆունկցիայի արժեքը ((x;y)∈D գրառումը նշանակում է, որ (x;y) կոորդինատներով կետը պատկանում է D տիրույթին):



$$z = \begin{cases} x + y, \text{եթե} \quad (x;y) \in D \\ 3, \text{հակառակ դեպքում} \end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.10-ում:

Նկ. 1.10

Եթե տրված կոորդինատներով կետը ընկած է $y=kx+b$ ուղղից վերև, ապա տեղի կունենա $y> kx+b$ անհավասարությունը, հակառակ դեպքում՝ $y<kx+b$ անհավասարությունը: Եթե տրված կոորդինատներով կետը ընկած է $(x-a)^2+(y-b)^2 =r^2$ շրջանից դուրս, ապա տեղի կունենա $(x-a)^2+(y-b)^2 >r^2$ անհավասարությունը, հակառակ դեպքում՝ $(x-a)^2+(y-b)^2 <r^2$ անհավասարությունը:

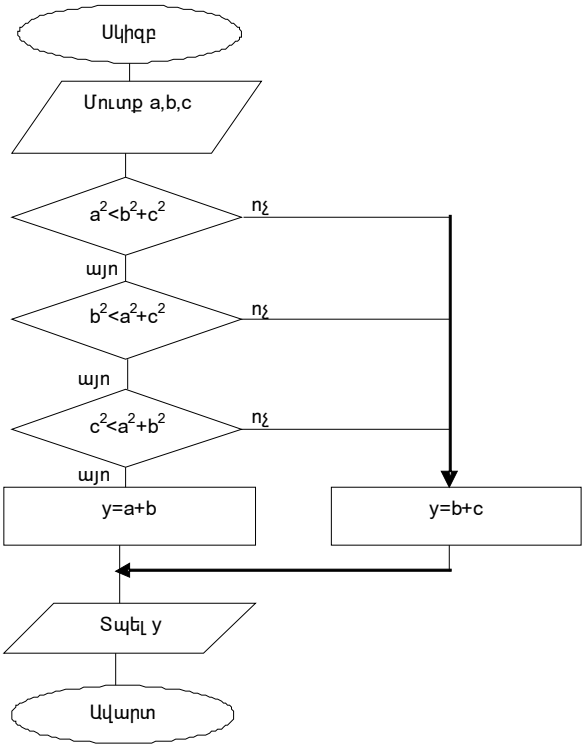
Քանի որ տրված օրինակում ընդգծված տիրույթը սահմանափակված է 4 գծերով, անհրաժեշտ է, որ կետը գտնվի ա) $x^2+y^2=1$ շրջանից դուրս ($x^2+y^2\geq 1$), բ) y=x+2 ուղղից ներքև ($y\leq x+2$), գ) y=-x+2 գծից ներքև ($y\leq -x+2$), դ) y=0 գծից վերև ($y\geq 0$): Նշված 4 պայմանների միաժամանակյա կատարման դեպքում z փոփոխականին վերագրվում է x+y արժեքը, իսկ հակառակ դեպքում, այսինքն, երբ կետը չի պատկանում տրված միջակայքին, z փոփոխականին վերագրվում է 3 արժեքը:

Օրինակ 1.8: Տրված են եռանկյան կողմերի a, b, c երկարությունները: Կազմել բլոկ-սխեմա, որը կտպի

$$y=\begin{cases} a+b & \text{, եթե եռանկյունը սուրանկյուն է ,} \\ b+c & \text{,հակառակ դեպքում:} \end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.11-ում:

Եռանկյունը սուրանկյունի է, եթե $a^2<b^2+c^2$ և $b^2<a^2+c^2$ և $c^2<a^2+b^2$: Եռանկյան կողմերի արժեքներ ներմուծումից հետո ստուգվում են նշված երեք պայմանները: Եթե ճշմարիտ են նշված երեք պայմանները, ապա եռանկյան երեք անկյուններն էլ սուր են, հետևաբար եռանկյունը սուրանկյուն է, այսինքն y փոփոխականին վերագրվում է a+b արժեք: Եթե պայմաններից մեկը չի կատարվում, նշանակում է՝ եռանկյան մեջ կա սուր անկյունից տարբեր անկյուն, և y փոփոխականին վերագրվում է b+c արժեք:



Նկ. 1.11

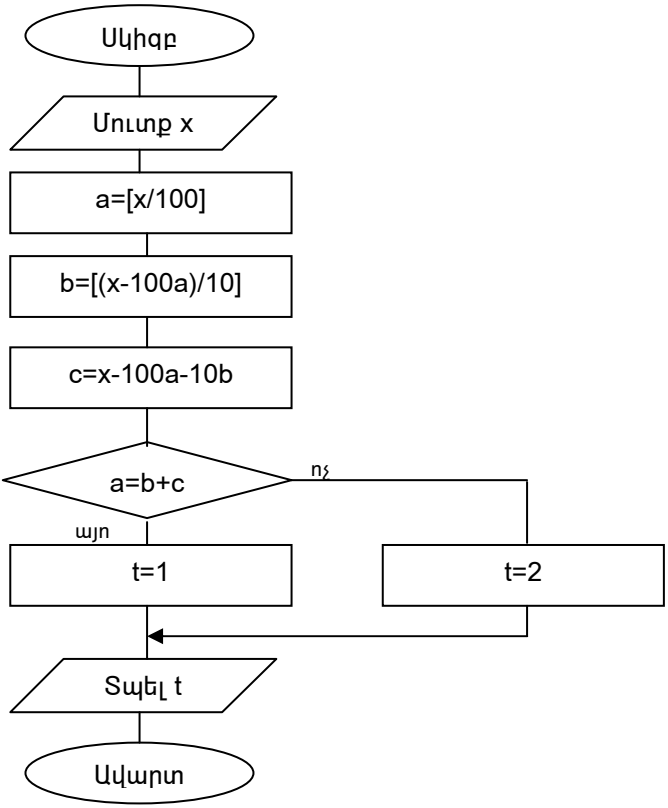
Օրինակ 1.9: Տրված է եռանիշ թիվ: Կազմել բլոկ-սխեմա, որը t փոփոխականին կվերագրի 1 արժեք, եթե եռանիշ թվի հարյուրավորների թվանշանը հավասար է միավորների և տասնավորների թվանշանների գումարին, հակառակ դեպքում՝ 2: Տպել t փոփոխականի արժեքը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.12-ում:

Տրված եռանիշ թիվը նշանակենք $x=\overline{abc}$: Եռանիշ թվի թվանշանները կարելի է գտնել հետևյալ կերպ.

a = [x/100],
b = [(x-100a)/10],
c = x-100a-10b,

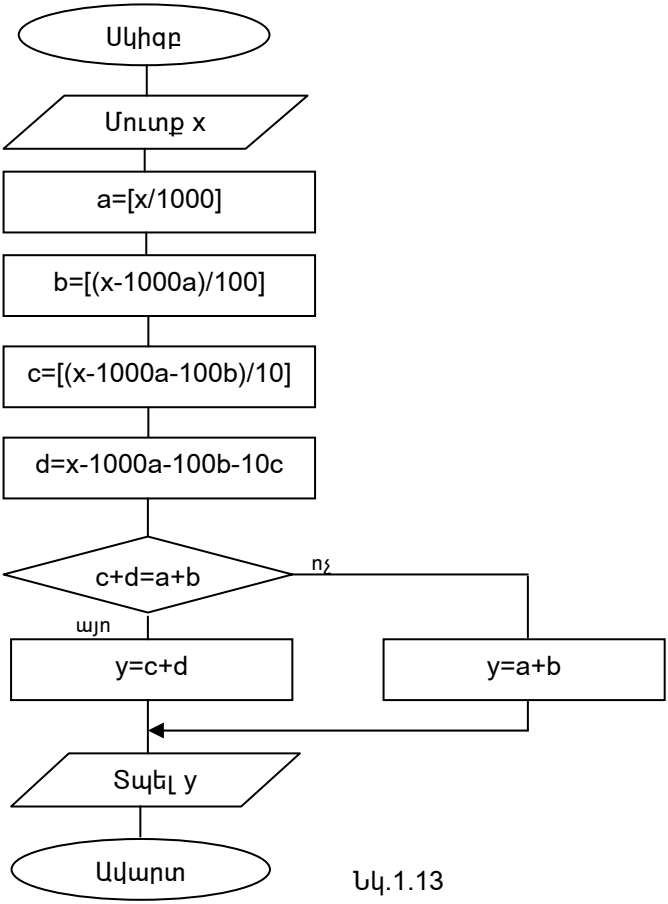
որտեղ [ ] փակագծերը նշանակում են, որ վերցվում է նրանց մեջ առնված արտահայտության արժեքի ամբողջ մասը:



Նկ.1.12

Օրինակ 1.10: Տրված է քառանիշ թիվ: Կազմել բլոկ-սխեմա, որը կհաշվի և կտպի քառանիշ թվի միավորների և տասնավորների թվանշանների գումարը, եթե թվի միավորների և տասնավորների թվանշանների գումարը հավասար է հարյուրավորների և հազավորների թվանշանների գումարին, հակառակ դեպքում՝ հարյուրավորների և հազավորների թվանշանների գումարը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.13-ում:



Նկ.1.13

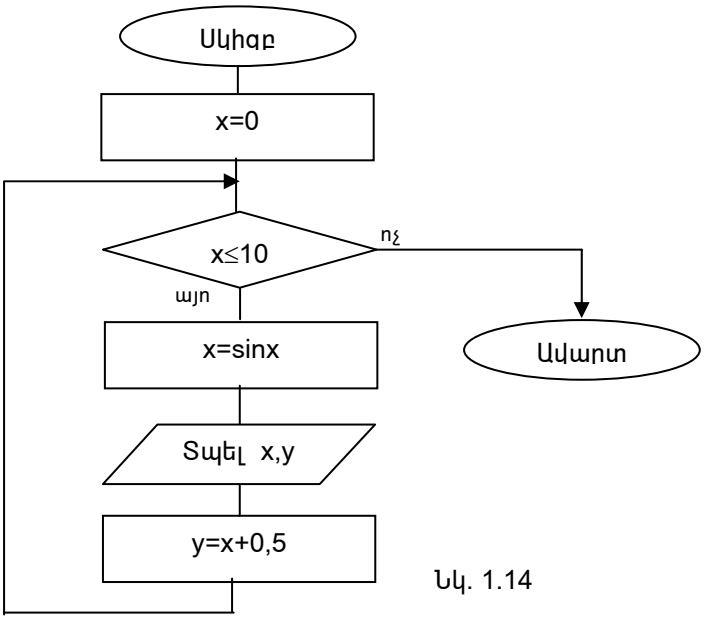
Տրված քառանիշ թիվը նշանակենք $x=\overline{abcd}$: Քառանիշ թվի թվանշանները կարելի է գտնել հետևյալ կերպ.

a = [x/1000],
b = [(x-1000a)/100],
c = [(x-1000a-100b)/10],
d = x-1000a-100b-10c,

որտեղ [ ] փակագծերը նշանակում են, որ վերցվում է նրանց մեջ առնված արտահայտության արժեքի ամբողջ մասը:

Օրինակ 1.11: Կազմել բլոկ-սխեմա, որն օգտագործելով նախապայմանով ցիկլային կառուցվածք՝ $x\in[0; 10]$ միջակայքում, $\Delta x=0,3$ քայլով կհաշվի $y=\sin x$ ֆունկցիայի արժեքները: Տպել x և y փոփոխականների արժեքների աղյուսակը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.14-ում:

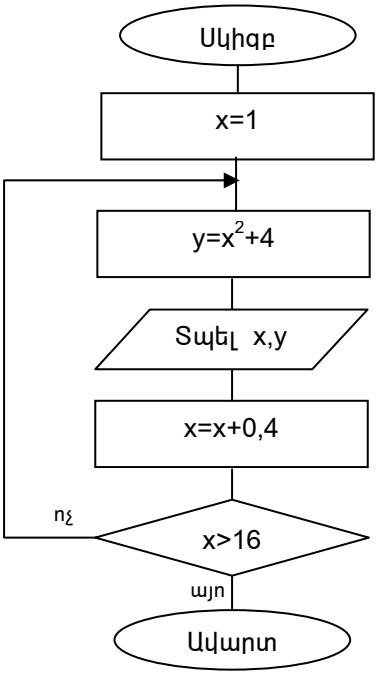


Նկ. 1.14

x փոփոխականին վերագրված է միջակայքի սկզբնական 0 արժեքը, որը նախապայմանով ցիկլի մարմնի մեջ մեծացնում է իր արժեքը 0,5 քայլով: Ցիկլի մարմնում x-ի յուրաքանչյուր արժեքի համար հաշվվում է ֆունկցիայի արժեքը, արտածվում x-ի և y-ի արժեքները: Ցիկլի մարմինը կրկնվում է այնքան, քանի դեռ ճշմարիտ է $x\le10$ պայմանը:

Օրինակ 1.12: Կազմել բլոկ-սխեմա, որը օգտագործելով հետպայմանով ցիկլային կառուցվածք՝ $x\in[1; 16]$ միջակայքում, $\Delta x=0,4$ քայլով կհաշվի $y=x^2+4x$ ֆունկցիայի արժեքները: Տպել x և y փոփոխականների արժեքների աղյուսակը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.15-ում:

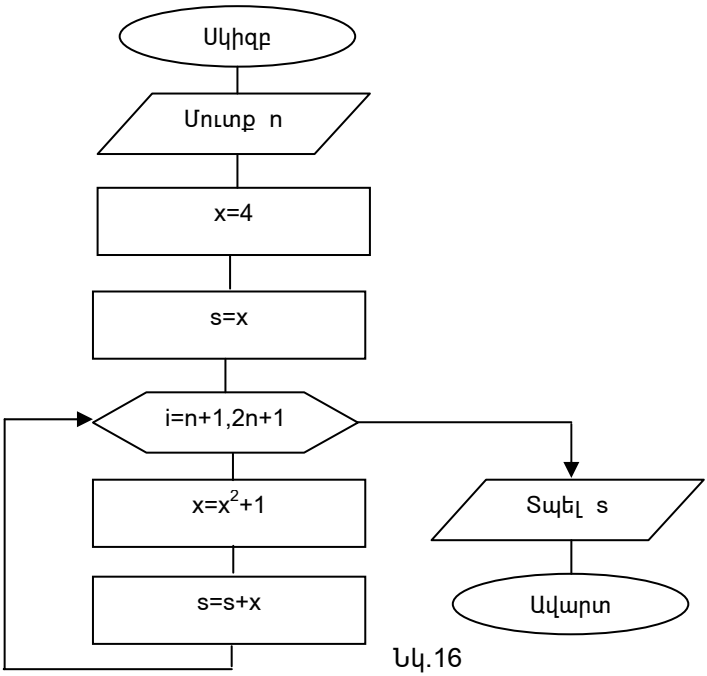


Նկ. 1.15

x փոփոխականին վերագրված է միջակայքի սկզբնական 1 արժեքը, որը հետպայմանով ցիկլի մարմնի մեջ մեծացնում է իր արժեքը 0,4 քայլով: Ցիկլի մարմնում x-ի յուրաքանչյուր արժեքի համար հաշվվում է y-ի արժեքը, արտածվում x-ի և y-ի արժեքները: Ցիկլը ավարտվում է x>16 պայմանի դեպքում:

Օրինակ 1.13: Կազմել բլոկ-սխեմա, որը տրված n բնական թվի համար կհաշվի ու կտպի $\sum_{i=n}^{2n+1} x_i$ արտահայտության արժեքը, որտեղ $x_n=4$; $x_i=x_{i-1}^2+1$:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.16-ում:

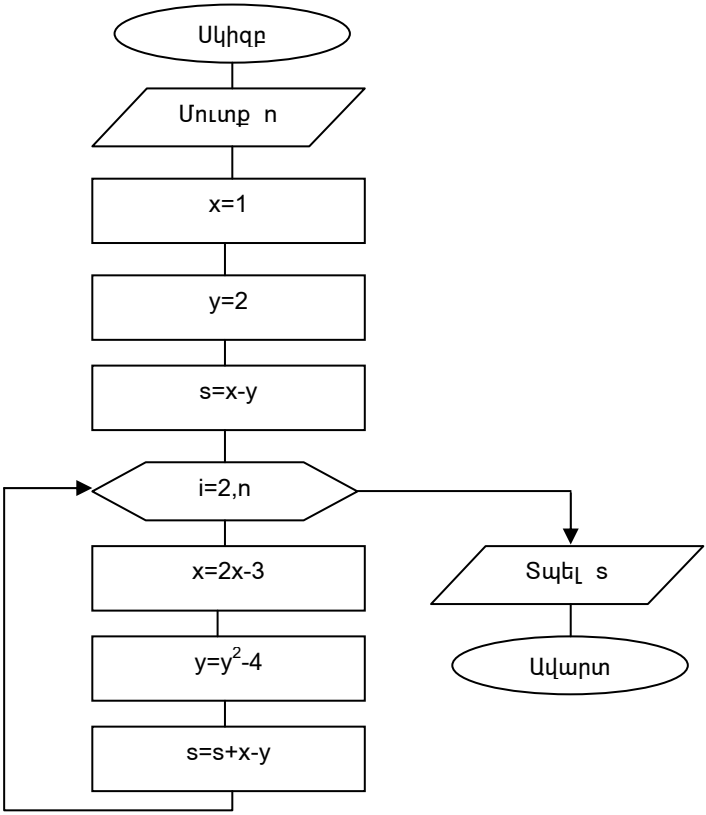


Նկ.16

Առաջին x_n գումարելիի արժեքը ցիկլից դուրս վերագրվում է գումարի համար նախատեսված S փոփոխականին: Մնացած գումարելիները հաշվվում և նրանց քառակուսիների արժեքները ավելացվում են S փոփոխականին ցիկլի մարմնում: Այստեղ պետք է հիշել վերագրման օպերատորի կատարման կարգը, այն է՝ սկզբում հաշվվում է արտահայտության աջ մասի արժեքը և այն վերագրվում ձախում գրված փոփոխականին: Խնդրի լուծման ժամանակ անհրաժեշտություն չկա x փոփոխականը պահելու որպես զանգված, քանի որ ցիկլում հաշվված x փոփոխականի ամեն մի արժեքի քառակուսին անմիջապես ավելացվում է գումարի համար նախատեսված S փոփոխականին:

Օրինակ 1.14: Կազմել բլոկ-սխեմա, որը տրված n բնական թվի համար կհաշվի ու կտպի $\sum_{i=1}^{n}(x_i - y_i)$ արտահայտության արժեքը, որտեղ $x_1=1$; $x_i=2x_{i-1}-3$; $y_1=2$; $y_i=y^2_{i-1}-4$:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.17-ում:

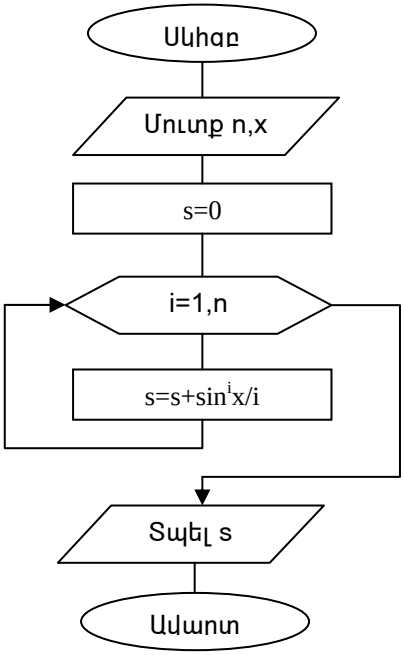


Նկ. 1.17

Առաջին x_1 և y_1 գումարելիների գումարի քառակուսու արժեքը ցիկլից դուրս վերագրվում է գումարի համար նախատեսված S փոփոխականին: Մնացած գումարելիները հաշվվում և նրանց գումարի քառակուսիների արժեքները ավելացվում են S փոփոխականին ցիկլի մարմնում: Այստեղ պետք է հիշել վերագրման օպերատորի կատարման կարգը, այն է՝ սկզբից հաշվվում է արտահայտության աջ մասի արժեքը և այն վերագրվում ձախում գրված փոփոխականին: Խնդրի լուծման ժամանակ անհրաժեշտություն չկա x և y փոփոխականները պահելու որպես զանգված, քանի որ ցիկլում հաշվված x և y փոփոխականների ամեն մի արժեքների գումարի քառակուսին անմիջապես ավելացվում է գումարի համար նախատեսված S փոփոխականին:

Օրինակ 1.15: Կազմել բլոկ-սխեմա, որը կամայական x իրական և n բնական թվերի համար կհաշվի և կտպի $\sin x + \frac{\sin^2 x}{2} + \cdots + \frac{\sin^n x}{n}$ արտահայտության արժեքը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.18-ում:



Նկ. 1.18

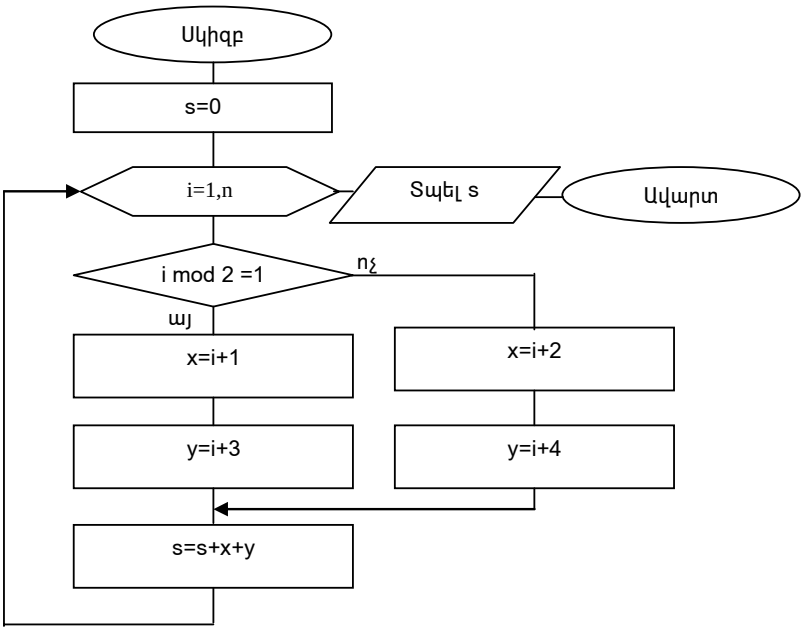
Դժվար չէ նկատել, որ $\sin x + \frac{\sin^2 x}{2} + \cdots + \frac{\sin^n x}{n}$ գումարը կարելի է ներկայացնել $\sum_{i=1}^{n} \frac{\sin^i x}{i}$ տեսքով՝ և այդպիսով՝ խնդիրը հանգեցնել պարզագույն գումարի հաշվման:

Օրինակ 1.16: Կազմել բլոկ-սխեմա, որը կհաշվի ու կտպի տրված արտահայտության արժեքը.

$\sum_{i=1}^{16}(x_i + y_i)$, որտեղ $x_i = \begin{cases} i+1 & \text{, եթե i-ն կենտ է,} \\ i+2 & \text{, հակառակ դեպքում:} \end{cases}$

, հակառակ $y_i = \begin{cases} i+3 & \text{, եթե i-ն կենտ է,} \\ i+4 & \text{դեպքում:} \end{cases}$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.1.19-ում:



Նկ. 19

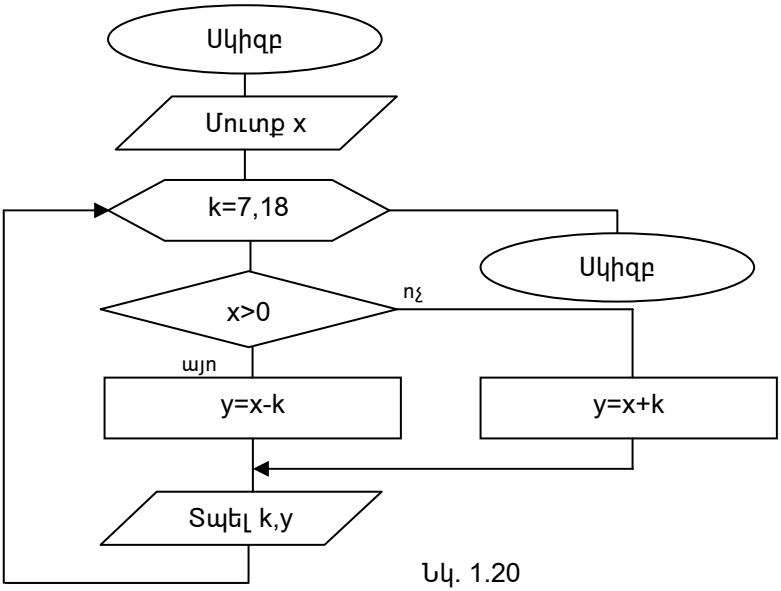
Կազմակերպվել է ցիկլ, որի մեջ հաշվվում են x և y փոփոխականների արժեքները, որոնց գումարն անմիջապես ավելացվում է գումարի համար նախատեսված S փոփոխականին:

Օրինակ 1.17: Կամայական x իրական և k պարամետրերի տրված արժեքների համար կազմել տրված ֆունկցիայի արժեքների հաշվման և արտածման բլոկ-սխեման.

$$Y=\begin{cases} x-k & \text{, եթե x>0,} \\ x+k & \text{, հակառակ դեպքում,} \end{cases}$$

որտեղ $k\in[7;18]$, $\Delta k=1$:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 1.20-ում:



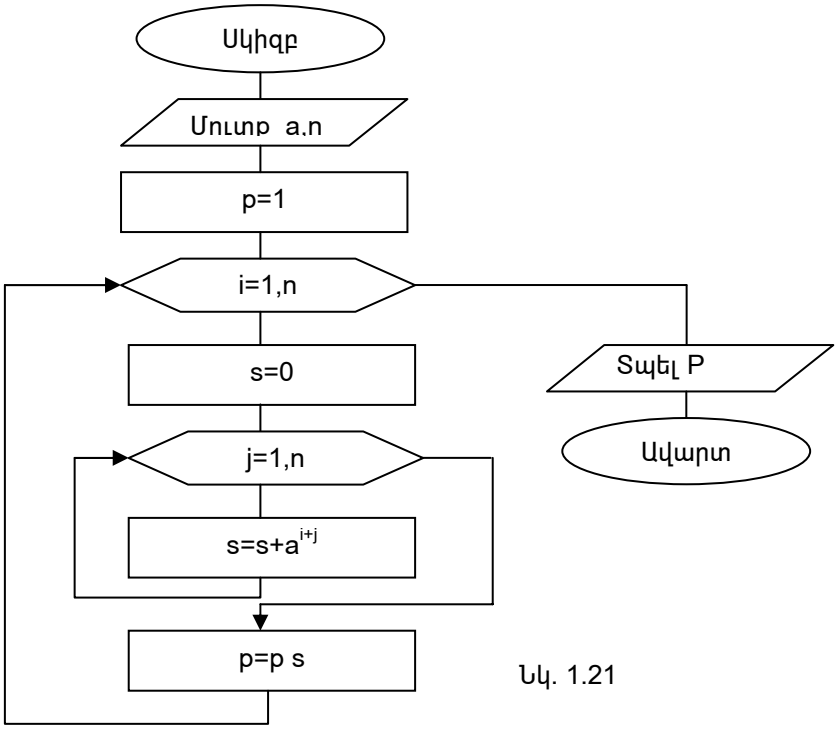
Նկ. 1.20

Քանի որ ֆունկցիայի արժեքները պետք է հաշվել k-ի տարբեր արժեքների համար, ապա կազմակերպվել է k աճող պարամետրով ցիկլ, որի մեջ հաշվվում և արտածվում են տրված ֆունկցիայի արժեքները:

Օրինակ 1.18: Կազմել բլոկ-սխեմա, որը տառային պարամետրերի կամայական թվային արժեքների համար կհաշվի և կտպի տրված արտահայտության արժեքը.

$$p=\prod_{i=1}^{n}\sum_{j=1}^{n}a^{i+j}:$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 1.21-ում:



Նկ. 1.21

Կազմակերպված են ներդրված երկու ցիկլեր, որոնցից ներքինում հաշվվում է հերթական գումարը (S), այնուհետև ստացված արժեքը արտաքին ցիկլի մեջ բազմապատկում է արտադրյալի կազմակերպման համար նախատեսված փոփոխականին (p):

1.7. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակ 2-ի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Աղյուսակ 2.

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	11, 22, 41, 70, 71, 89, 99, 103, 114, 122, 134
2	12, 21, 42, 61, 72, 88, 98, 107, 117, 123, 135
3	13, 25, 43, 62, 73, 90, 97, 105, 116, 124, 136
4	14, 26, 44, 63, 74, 84, 96, 106, 115, 125, 137
5	15, 27, 45, 64, 75, 85, 35, 108, 120, 126, 133
6	16, 28, 46, 65, 76, 83, 94, 109, 119, 121, 131
7	17, 29, 47, 66, 77, 82, 93, 101, 111, 127, 132
8	18, 30, 48, 67, 78, 81, 91, 104, 112, 128, 138
9	19, 31, 51, 68, 79, 86, 92, 110, 118, 129, 139
10	20, 32, 52, 69, 80, 87,100, 102, 113, 130, 114
11	11, 33, 53, 69, 73, 83, 95, 104, 113, 130, 140
12	12, 34, 54, 70, 72, 81, 94, 105, 112, 121, 131
13	13, 35, 55, 61, 71, 82, 96, 107, 116, 123, 131
14	14, 36, 56, 62, 74, 85, 93, 106, 111, 124, 134
15	15, 37, 57, 63, 75, 90, 97, 109, 115, 125, 133
16	16, 38, 58, 64, 76, 84, 92, 108, 114, 128, 135
17	17, 39, 59, 65, 77, 87, 98, 101, 120, 127, 136
18	18, 40, 60, 66, 78, 86, 100, 102, 119, 128, 138
19	19, 39, 41, 67, 79, 88, 99, 110, 118, 129, 140
20	20, 38, 42, 68, 80, 89, 91, 103, 117, 130, 139
21	11, 21, 43, 61, 80, 88, 96, 110, 120, 121, 132
22	12, 22, 44, 62, 79, 89, 97, 101, 111, 123, 133
23	13, 25, 45, 63, 78, 81, 98, 106, 119, 124, 140
24	14, 26, 46, 64, 77, 90, 91, 108, 118, 122, 131
25	15, 27, 47, 65, 76, 87, 99, 102, 117, 126, 139
26	16, 29, 48, 66, 75, 82, 100, 109, 112, 125, 138
27	17, 30, 51, 67, 74, 86, 92, 103, 116, 127, 134
28	18, 31, 52, 68, 73, 83, 94, 105, 115, 128, 137
29	19, 32, 53, 69, 72, 85, 93, 104, 114, 129, 135
30	20, 33, 54, 70, 71, 84, 95, 107, 113, 130, 136

2. ՃՅՈՒՂԱՎՈՐՎԱԾ ԾՐԱԳՐԵՐԻ ԿԱԶՄԱԿԵՐՊՈՒՄ

2.1. Հիմնական հասկացությունները և օպերատորները

Պարզ ծրագրի կառուցվածքը

Պարզ ծրագրի կառուցվածքն ունի հետևյալ տեսքը՝

```
PROGRAM <անուն>;
<նկարագրությունների բաժին>;
  BEGIN
  <օպերատորների հաջորդականություն>
  END.
```

Առանձին օպերատորներն իրարից անջատվում են կետ-ստորակետ (;) նշանով:

BEGIN և END բանալի բառերը օգտագործվում են ծրագրային միավորի ձևավորման համար: Յուրաքանչյուր ծրագրում բանալի բառերի այդ զույգը գոնե մեկ անգամ պետք է հանդիպի: Ամենավերջին END-ից հետո դրվում է կետ:

Նկարագրությունների բաժնում պետք է նկարագրել ծրագրում օգտագործված բոլոր հաստատունները և փոփոխականները:

Հաստատունները ծրագրի այնպիսի օբյեկտներ են, որոնք չեն կարող փոփոխել իրենց արժեքները ծրագրի կատարման ողջ ընթացքում:

Հաստատունների հայտարարումը սկսվում է CONST հատուկ բառով: Հետո հաջորդում է հավասարության նշանը (=) և այդ հաստատունի արժեքը:

Օրինակ.

```
CONST N=10; X=5.36;
```

Փոփոխականները ծրագրում հայտարարվում են հաստատուններից առաջ:

Փոփոխականների նկարագրման համար պետք է գրել VAR հատուկ բառը, որից հետո, իրարից ստորակետով անջատելով ծրագրում

օգտագործված նույնատիպ փոփոխականները, վերջում երկու կետ (:) և նշել այդ փոփոխականների տիպը:

Օրինակ.

VAR V_1, V_2, ..., V_n : t;

որտեղ V_1, V_2, ..., V_n-ը փոփոխականների անուներն են, իսկ t-ն նրանց տիպը:

Իդենտիֆիկատոր

Իդենտիֆիկատորները կիրառվում են փոփոխականների անվանման համար: Իդենտիֆիկատորները սկսվում են լատինական այբուբենի տառով, որին կարող են հաջորդել նույն այբուբենի այլ տառեր և (կամ) թվանշաններ, ինչպես նաև ընդգծման նշանը: Որևէ այլ սիմվոլի կամ նշանի առկայությունը իդենտիֆիկատորում արգելված է: ՊԱՍԿԱԼ լեզվում կան հատուկ բառեր, որոնք որպես իդենտիֆիկատոր չեն կարող գործածվել:

Մուտքի և ելքի օպերատորներ

Տվյալների ներմուծման և արտածման համար ծրագրավորման ՊԱՍԿԱԼ լեզվում նախատեսված են READ, READLN, WRITE և WRITELN մուտքի-ելքի պրոցեդուրաները: Հաճախ պարզության համար այդ պրոցեդուրաներն անվանում են օպերատորներ:

Ներմուծման համար առայժմ կօգտագործենք READ պրոցեդուրան, որն ունի հետևյալ տեսքը.

READ(<մուտքի ցուցակ>);

Մուտքի ցուցակում նշվում են այն իդենտիֆիկատորները, որոնց արժեքներն անհրաժեշտ է ներմուծել:

Մուտքի օպերատորն իրականացնելիս քոմփյութերն անցնում է սպասողական վիճակի՝ պահանջելով, որ ներմուծվեն մուտքի ցուցակում նշված բոլոր փոփոխականների արժեքները:

Տվյալների արտածման համար օգտագործում են WRITE և WRITELN պրոցեդուրաները, որոնք ունեն հետևյալ տեսքերը.

WRITE(<ելքի ցուցակ>);
WRITELN(<ելքի ցուցակ>);
WRITELN;

Սրանց տարբերությունն այն է, որ ելքի ցուցակում նշված տվյալներն արտածելուց հետո WRITELN-ը անցում է կատարում հաջորդ տողին, մինչդեռ WRITE-ի դեպքում նշիչը մնում է ընթացիկ տողի վրա: Եթե WRITELN պրոցեդուրայում նշված չէ ելքի ցուցակ, ապա պրոցեդուրան բաց է թողնում մեկ տող:

Վերագրման օպերատոր

Վերագրման օպերատորները նախատեսված են փոփոխականներին արժեքներ տալու համար:

Վերագրման օպերատորը կարելի է ներկայացնել հետևյալ ընդհանուր տեսքով.

V:=A,

որտեղ V-ն փոփոխական է, A-ն ՊԱՍԿԱԼ-ի կանոններով գրված ցանկացած արտահայտություն: A արտահայտությունը կարող է պարունակել հաստատուն, փոփոխական, ֆունկցիայի անվանում, գործողությունների նշաններ և փակագծեր:

Գործողություններ ամբողջ թվերով

Ամբողջ թվերով կարելի է կատարել գումարման (+), հանման (-), բազմապատկման (*) և բաժանման (/) գործողություններ: Առաջին երեք դեպքերում արդյունքը միշտ ստացվում է որպես ամբողջ թիվ, իսկ բաժանման դեպքում արդյունքը ստացվում է որպես իրական թիվ:

a և b ամբողջ թվերի բաժանումից ստացվող ամբողջ քանորդը և ամբողջ մնացորդը կարելի է որոշել՝ օգտվելով DIV և MOD գործողություններից.

a DIV b - a ամբողջ թիվը b ամբողջ թվին բաժանելիս ստացված ամբողջ քանորդն է:

a MOD b – a ամբողջ թիվը b ամբողջ թվին բաժանելիս ստացված ամբողջ մնացորդն է:

Իրական թվեր

Գործնական խնդիրներում հաճախ կիրառվում են իրական թվեր: ՊԱՍԿԱԼ-ում իրական թվի ամբողջ մասը կոտորակայինից անջատվում է կետով:

Իրական տիպի փոփոխականի նկարագրման համար օգտագործվում է REAL (իրական) **նկարագրիչը:**

Օրինակ՝ Var a,b:REAL;

Իրական թվերի ներկայացումը

Իրական թվերի ներկայացման (գրառման) համար ՊԱՍԿԱԼ-ում կիրառվում են երկու ձևեր: Առաջին դեպքում թվերը ներկայացվում են ինչպես մաթեմատիկայում՝ **տասնորդական կոտորակի տեսքով**՝ ստորակետը փոխարինելով տասնորդական կետով:

Օրինակ՝ 2. 36 + 13. 1 0. 19 - 1. 234 56. 0

Դրական թվի ներկայացման համար + նշանի կիրառումը պարտադիր չէ, իսկ բացասական թվից առաջ - նշան է դրվում:

Իրական թվի ներկայացման երկրորդ հնարավոր եղանակը, այսպես կոչված, **էքսպոնենտային** ձևն է: Այս դեպքում թիվը ներկայացվում է 10-ից փոքր արժեք ունեցող արտադրիչի տեսքով, որը կոչվում է **մանտիս**, բազմապատկած 10-ի անհրաժեշտ աստիճանով: Թվի ներկայացման էքսպոնենտային ձևում մանտիսը գրվում է սովորական տեսքով, իսկ 10-ի աստիճանի փոխարեն գրվում է E տառը, որին հաջորդում է աստիճանացույցին հավասար թիվը՝ իր նշանով: Օրինակ՝ 365.42 թիվը կարելի է ներկայացնել 3.6542×10^2 տեսքով: Պարզ է, որ մանտիս է հանդիսանում 3.6542 թիվը, որը բազմապատկած է 10-ի քառակուսով, հետևաբար կարելի է գրել 365.42=3.6542E2:

Ստանդարտ ֆունկցիաներ

Հաշվարկային խնդիրներ լուծելիս որոշ մաթեմատիկական ֆունկցիաներ օգտագործվում են առավել հաճախ: Որպեսզի ծրագրավորողը դրանց հաշվարկման համար անհատական ծրագիր

չկազմի, ՊԱՍԿԱԼ լեզվում նախատեսված է, այսպես կոչված, **ստանդարտ ֆունկցիաների** հավաքածու:

Աղյուսակ 2.1-ում բերված են որոշ ստանդարտ ֆունկցիաներ.

Աղյուսակ 2.1

Ֆունկցիայի անվանումը	Մաթեմատիկական գրառումը	Ներկայացումը ՊԱՍԿԱԼ-ում	Արգումենտի տիպը	Ֆունկցիայի տիպը
Սինուս	sin x	sin(x)	ամբողջ, իրական	իրական
Կոսինուս	cos x	cos(x)	ամբողջ, իրական	իրական
Արկտանգես	arctg x	arctan(x)	ամբողջ, իրական	իրական
Բնական լոգարիթմ	ln x	ln(x)	ամբողջ, իրական	իրական
e հիմքով ցուցչային ֆունկցիա	e^x	exp(x)	ամբողջ, իրական	իրական
Քառակուսի արմատ	$\sqrt{x}$	sqrt(x)	ամբողջ, իրական	իրական
X քառակուսի	x^2	sqr(x)	ամբողջ, իրական	ամբողջ իրական
Բացարձակ արժեք	|x|	abs(x)	ամբողջ, իրական	ամբողջ իրական

Ստանդարտ ֆունկցիաների գրության ժամանակ արգումենտը անպայման պետք է վերցնել փակագծերի մեջ: Որպես SIN և COS ֆունկցիաների արգումենտ ընդունվում է ռադիաններով արտահայտված անկյան մեծությունը: Եթե անկյունը տրված է աստիճաններով, ապա անհրաժեշտ է այն վերածել ռադիանների՝

$$X_{\text{ռադ.}} = X_{\text{աստ.}} \cdot \pi/180 ,$$

որտեղ π թվին թվին համապատասխանում է Pi ստանդարտ հաստատունը:

Ստանդարտ ֆունկցիայի կիրառման համար բավական է արտահայտության մեջ նշել ֆունկցիայի անունը՝ փակագծերում գրելով արգումենտը:

Իրական թվերով աշխատելիս կիրառելի են TRUNC, FRAC և ROUND ստանդարտ ֆունկցիաները:

trunc(x) - անտեսում է x իրական արգումենտի կոտորակային մասը՝ արդյունքում ստանալով ամբողջ թիվ,

frac(x) - ամբողջ մասը փոխարինում է զրոյով անփոփոխ թողնելով կոտորակային մասը: Արդյունքը կոտորակային թիվ է:

round(x) – կլորացնում է x արգումենտը մինչև մոտակա ամբողջը: Արդյունքը ամբողջ թիվ է:

Թվաբանական արտահայտություն

Թվաբանական արտահայտության արժեքը թիվ է: Թվաբանական արտահայտությունը կարող է ներառել հաստատուններ, փոփոխականներ և ֆունկցիաներ, որոնք միմյանց հետ միացված են թվաբանական գործողությունների նշաններով: Մասնավոր դեպքում առանձին թիվը, փոփոխականը, հաստատունը, ֆունկցիան նույնպես թվաբանական արտահայտություններ են: Հաճախ դրանք կոչվում են պարզ թվաբանական արտահայտություններ: Արտահայտությունը պետք է միարժեքորեն որոշի հաշվման արդյունքը: Դրա համար սահմանված է գործողությունների կատարման հստակ առաջնահերթություն.

1). Բազմապատկում, բաժանում:
2). Գումարում, հանում:

Հավասարազոր գործողությունները կատարվում են ձախից աջ: Փակագծերի կիրառումը ապահովում է գործողության առաջնայնություն:

Թվաբանական արտահայտություններում չի թույլատրվում՝

1). գրել 2 գործողության նշան իրար կողքի. A/-B կամ A+-B: Նման դեպքերում անհրաժեշտ է կիրառել փակագծեր՝ A/(-B) կամ A+(-B):

2). Բաց թողնել բազմապատկման նշանը, օրինակ 3*A – ի փոխարեն գրել 3A:

Տրամաբանական փոփոխականներ

Գործնականում կարող են հանդիպել այնպիսի իրավիճակներ, երբ, կախված որոշակի պայմանից, օպերատորների որևէ խումբ կատարվում է,

կամ չի կատարվում: Որպես պայման կարող է լինել որոշակի պնդման (ասույթի) ճշմարիտ կամ կեղծ լինելը:

Եթե պնդումը ճշմարիտ է (TRUE), ապա ասում են, որ պայմանը կատարվում է, հակառակ դեպքում՝ պայմանը չի կատարվում (FALSE):

ՊԱՍԿԱԼ-ում տրամաբանական փոփոխականների հայտարարման համար օգտագործվում է **BOOLEAN** հատուկ բառը (անգլիացի մաթեմատիկոս Ջորջ Բուլի անունով):

Օրինակ.

```
VAR  a,b:BOOLEAN;
```

Տրամաբանական արտահայտություն

Տրամաբանական արտահայտության պարզագույն ձևը վերը դիտարկված համեմատման գործողությունն է: Այդպիսի տրամաբանական արտահայտության ընդհանուր տեսքը հետևյալն է.

e_1 <առնչության նշան> e_2

որտեղ, e_1 և e_2 –թվաբանական արտահայտություններ են: ՊԱՍԿԱԼ-ում կիրառվում են առնչության հետևյալ նշանները՝ <(փոքր), <=(փոքր կամ հավասար), >(մեծ), >=(մեծ կամ հավասար), =(հավասար), < >(ոչ հավասար):

Բերված տեսքի պարզ տրամաբանական արտահայտությունում հաշվվում են e_1 և e_2 թվաբանական արտահայտությունների արժեքները և համեմատվում միմյանց հետ: Կախված այդ արժեքներից և առնչության կիրառված նշանից՝ տրամաբանական արտահայտությունը ստանում է TRUE կամ FALSE արժեքը:

Պարզ տրամաբանական արտահայտությունները, իրենց հերթին, այլ տրամաբանական արտահայտության մեջ կարող են հանդես գալ որպես օպերանդներ, որոնք կարող են միավորվել OR (կամ), AND (և), NOT (ոչ) տրամաբանական գործողություններով:

Օրինակ.

```
(X >= A)  AND ( X <= B)
(X < A)  OR  (X > B):
```

Բացասման (ժխտման) NOT («ոչ») գործողությունը իրական տրամաբանական արտահայտության ճշմարիտ արժեքը դարձնում է կեղծ կամ հակառակը:

Ինչպես թվաբանական արտահայտություններում, այստեղ ևս սահմանված է գործողությունների կատարման որոշակի առաջնահերթություն՝

- առնչության (համեմատման) գործողություն,
- բացասման NOT գործողություն,
- տրամաբանական բազմապատկման AND գործողություն,
- տրամաբանական գումարման OR գործողություն:

Պայմանական օպերատոր

Գործնական խնդիրներում հաճախ առաջանում է այնպիսի իրավիճակ, երբ կախված որոշակի պայմանի ճշմարիտ կամ կեղծ լինելուց, անհրաժեշտ է լինում իրականացնել այս կամ այն գործողությունը կամ գործողությունների շարքը:

Պայմանական օպերատորի ընդհանուր տեսքը հետևյալն է.

IF b THEN s_1 ELSE s_2 ,

որտեղ՝ b – ն տրամաբանական փոփոխական է կամ տրամաբանական արտահայտություն, s_1, s_2 – ը՝ ցանկացած օպերատորներ են (այդ թվում և պայմանական օպերատոր):

Պայմանական օպերատորն աշխատում է հետևյալ սկզբունքով. նախ հաշվվում է տրամաբանական արտահայտության արժեքը: Եթե այն ստանում է TRUE արժեք, ապա կատարվում է s_1 օպերատորը, իսկ s_2 օպերատորը բաց է թողնվում, իսկ եթե արդյունքը ստանում է FALSE արժեք, ապա ընդհակառակը՝ բաց է թողնվում s_1 օպերատորը, կատարվում s_2 օպերատորը: Ներկայացված ընդհանուր տեսքը հանդիսանում է պայմանական օպերատորի, այսպես կոչված, լրիվ ձևն է: Թույլատրվում է նաև պայմանական օպերատորի կրճատ ձևը՝

IF b THEN s :

Այս օպերատորն աշխատում է հետևյալ սկզբունքով: Սկզբում հաշվվում է b տրամաբանական արտահայտության արժեքը: Եթե այն

ստանում է TRUE արժեքը, ապա կատարվում է s օպերատորը, հակառակ դեպքում s օպերատորը չի կատարվում:

Պայմանական օպերատորի՝ վերը բերված ընդհանուր տեսքերում s, s_1 և s_2 նշանակումների տակ մինչ այժմ հասկացվում էին մեկական օպերատորներ: Հաճախ որոշակի պայմանի դեպքում կատարման ենթակա գործողությունը չի կարող ներկայացվել մեկ օպերատորով: Այդ դեպքում կիրառվում է ՊԱՍԿԱԼ-ում թույլատրված **բաղադրյալ օպերատորը:** Այն օպերատորների հաջորդականություն է, որոնք պարփակվում են BEGIN և END ծառայողական բառերի մեջ: Բաղադրյալ օպերատորի ընդհանուր տեսքը հետևյալն է՝

BEGIN s_1, s_2, ... , s_n END

որտեղ s_1, s_2, ... , s_n իրարից «;»-ով բաժանված օպերատորներ են:

Այսպիսով, պայմանական օպերատորում կարող են կիրառվել ինչպես պարզ, այնպես էլ բաղադրյալ օպերատորներ:

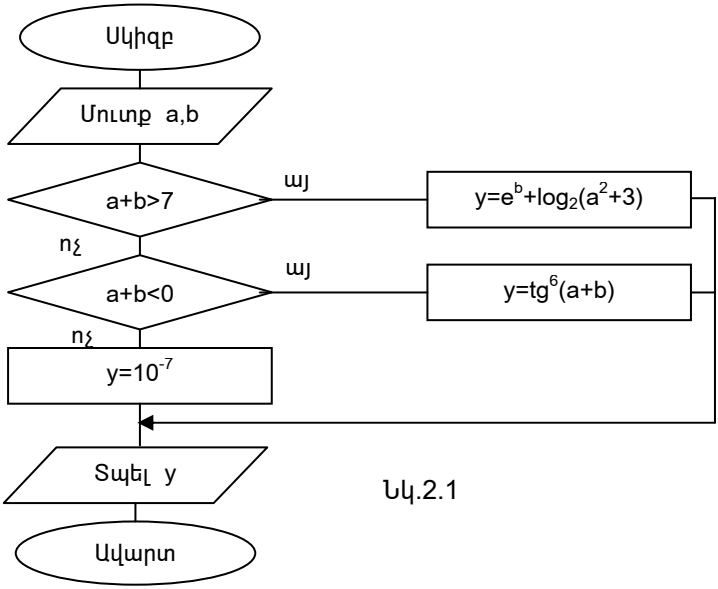
BEGIN և END ծառայողական բառերը ինչպես այստեղ, այնպես էլ ցանկացած ՊԱՍԿԱԼ-ծրագրում, փաստորեն օպերատորային փակագծերի դեր են խաղում (BEGIN-ը բացող, իսկ END-ը՝ փակող փակագծի):

2.2. Պարզ ծրագրերի տիպային օրինակներ

Օրինակ 2.1. Տառային պարամետրերի կամայական թվային արժեքների համար կազմել տրված Y ֆունկցիայի արժեքի հաշվման և տպման բլոկ-սխեման ու ծրագիրը

$$Y=\begin{cases} e^{b}+\log_2(a^2+3) & \text{եթե a+b>7} \\ tg^{6}(a+b) & \text{եթե a+b<0} \\ 10^{-7} & \text{մնացած դեպքերում} \end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.2.1-ում, իսկ ծրագիրն ունի EXAM2_1 անունը:



Նկ.2.1

```
    PROGRAM EXAM2_1;
    VAR A,B,Y:REAL;
BEGIN
  READ(A,B);
    IF A+B>7 THEN Y:=EXP(b)+LN(SQR(A)+3)/LN(2)
    ELSE IF A+B<0 THEN Y:=EXP(6*LN(SIN(A+b)/COS(A+B)))
    ELSE Y:=EXP(-7*LN(10));
    WRITE(Y)
END.
```

Պասկալ լեզվի մեջ ստանդարտ ֆունկցիա կա միայն բնական հիմքով լոգարիթմի հաշվման համար, այդ պատճառով այլ հիմքով լոգարիթմական ֆունկցիայի հաշվման համար անհրաժեշտ է անցնել տրվածին համարժեք բնական հիմքով լոգարիթմի, կիրառելով ձևափոխման հետևյալ բանաձևը՝

$$\log_a b = \frac{\ln b}{\ln a}:$$

Տվյալ խնդրում՝

$$\log_2\left(a^2+3\right)=\frac{\ln\left(a^2+3\right)}{\ln 2}:$$

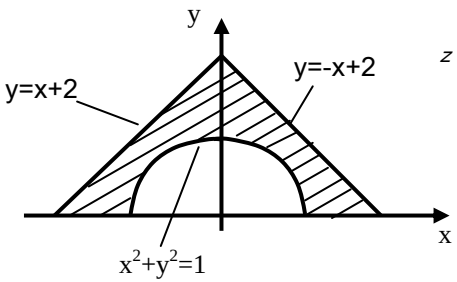
Քանի որ պասկալ լեզվում քառակուսուց բարձր աստիճանի հաշվման համար ստանդարտ ֆունկցիա չկա, ապա նպատակահարմար է օգտագործել

$$a^n = e^{n\ln(a)}$$

ձևափոխությունը: Տվյալ խնդրում՝

$$tg^6(a+b) = e^{6\ln\frac{\sin(a+b)}{\cos(a+b)}}:$$

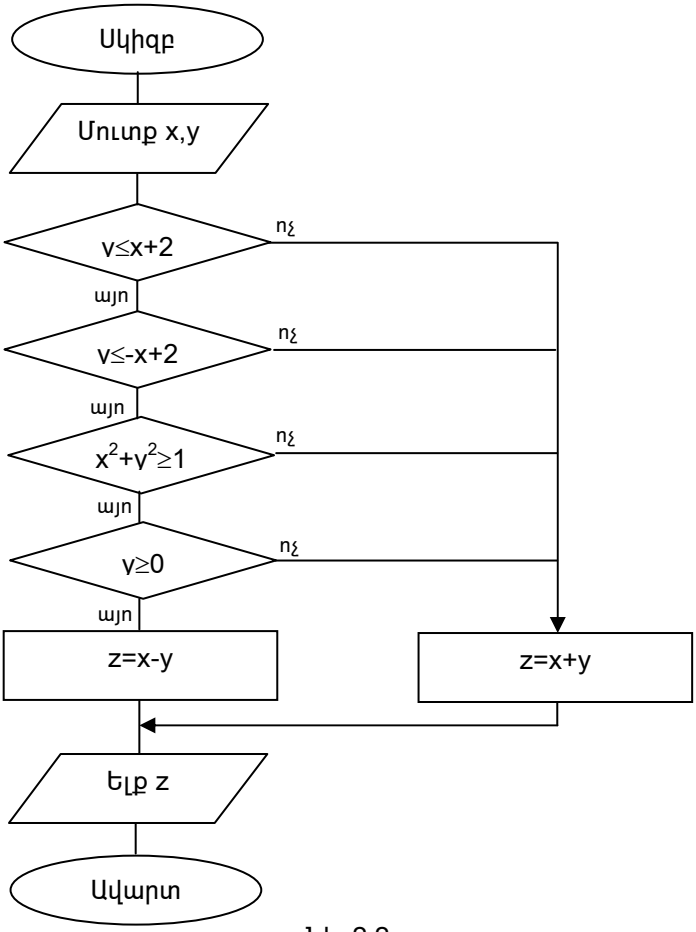
Օրինակ 2.2. Տրված են կետի x և y կոորդինատները: D-ն նկարում ընդգծված տիրույթն է՝ եզրագծերով հանդերձ: Կազմել բլոկ-սխեմա ու ծրագիր, որոնք կհաշվեն և կտպեն տրված ֆունկցիայի արժեքը ((x;y)∈D գրառումը նշանակում է, որ (x;y) կոորդինատներով կետը պատկանում է D տիրույթին):



$$z=\begin{cases} x-y, \text{եթե } (x;y)\in D \\ x+y, \text{հակառակ դեպքում} \end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 2.2-ում, իսկ ծրագիրն ունի EXAM2_2 անունը:

```
    PROGRAM EXAM2_2;
    VAR X,Y,Z:REAL;
BEGIN
  READ(X,Y);
  IF (Y<=X+2) AND (Y<=-X+2) AND (SQR(X)+SQR(Y)>=1)
     AND (Y>=0) THEN Z:=X-Y ELSE Z:=X+Y;
   WRITE(Z)
END.
```

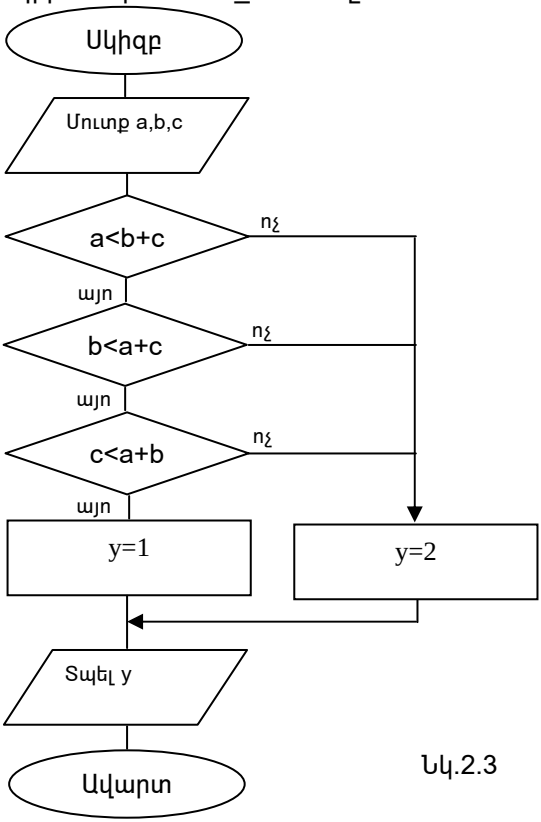


Նկ. 2.2

Որպեսզի տրված կոորդինատներով կետը ընկած լինի շտրիխված միջակայքում, անհրաժեշտ է, որ այն գտնվի ա) $y=0$ գծից վերև $y\geq 0$, բ)$x^2+y^2=1$ շրջանից դուրս ($x^2+y^2\geq 1$) , գ) $y=x+2$ ուղղից ներքև ($y\leq x+2$), դ) $y=-x+2$ ուղղից ներքև ($y\leq -x+2$): Նշված 4 պայմանների միաժամանակյա կատարման դեպքում z փոփոխականին վերագրվում է x-y արժեքը, իսկ հակառակ դեպքում, այսինքն, երբ կետը չի պատկանում տրված միջակայքին, z փոփոխականին վերագրվում է x+y արժեքը:

Օրինակ 2.3. Տրված են իրարից տարբեր a, b, c թվերը: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կտպեն y=1, եթե այդպիսի երկարություն ունեցող հատվածներով հնարավոր է կառուցել եռանկյունի, հակառակ դեպքում՝ y=2:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 2.3-ում, իսկ ծրագիրն ունի EXAM2_3 անունը:



Նկ.2.3

Տրված երեք հատվածներով (a,b,c) հնարավոր է կառուցել եռանկյունի, եթե a<b+c և c<a+b և b<a+c:

```
PROGRAM EXAM2_3;
    VAR A,B,C:REAL; Y:INTEGER;
BEGIN
  READ(A,B,C);
  IF(A<B+C) AND (B<A+C) AND (C<A+B) THEN Y:=1 ELSE Y:=2;
  WRITE(Y)
```

END.

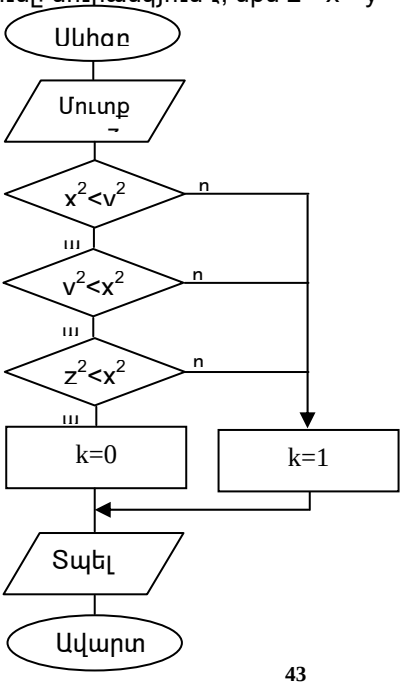
Օրինակ 2.4. Տրված են եռանկյան կողմերի x, y, z երկարությունները: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կտպեն

$$k=\begin{cases}0 & \text{, եթե եռանկյունը սուրանկյուն է ,}\\ 1 & \text{,հակառակ դեպքում:}\end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 2.4-ում, իսկ ծրագիրն ունի EXAM2_4 անունը:

```
PROGRAM EXAM2_4;
    VAR X,Y,Z:REAL; K:INTEGER;
BEGIN
  READ(X,Y,Z);
  IF (SQR(X)<SQR(Y)+SQR(Z)) AND (SQR(B)<SQR(A)+SQR(C))
       AND (SQR(C)<SQR(A)+SQR(B)) THEN K:=0 ELSE K:=1;
  WRITE(k)
END.
```

Եռանկյունը սուրանկյուն է, եթե $z^2<x^2+y^2$ և $y^2<x^2+z^2$ և $x^2<y^2+z^2$:

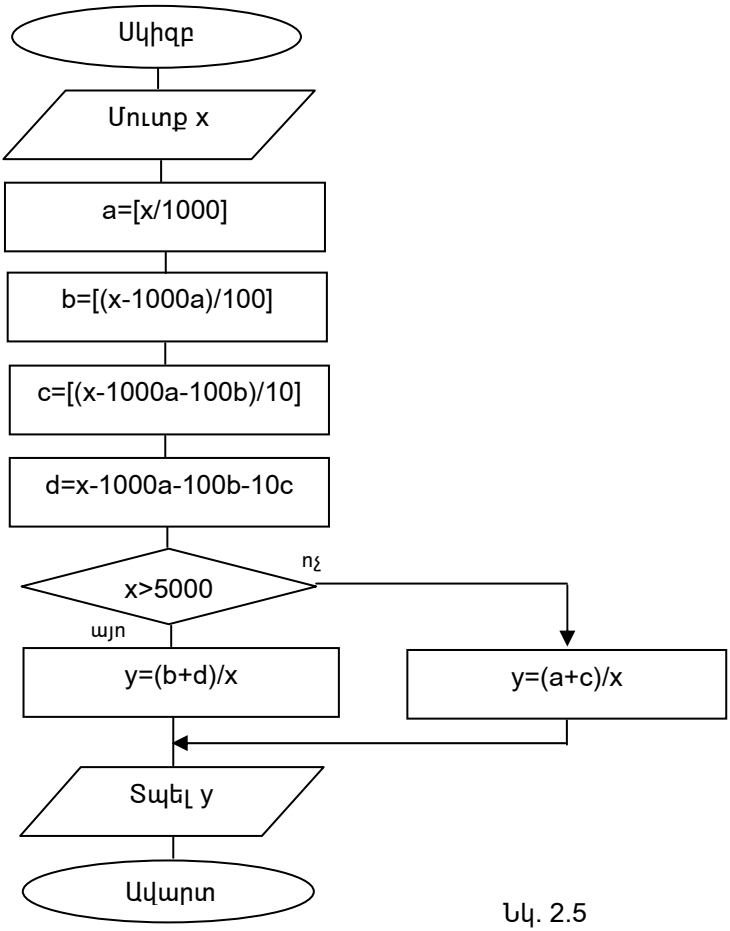


Նկ. 2.4

Օրինակ 2.5. Տրված է քառանիշ թիվ: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն քառանիշ թվի միավորների ու տասնավորների թվանշանների գումարի և քառանիշ թվի հարաբերությունը, եթե քառանիշ թիվը մեծ է 5000-ից, հակառակ դեպքում՝ հարյուրավորների և հազավորների թվանշանների գումարի և քառանիշ թվի հարաբերությունը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ. 2.5-ում, իսկ ծրագիրն ունի EXAM2_5 անունը:

```
    PROGRAM EXAM2_5;
    VAR X,A,B,C,D:INTEGER;Y:REAL;
BEGIN
 READ(X);
    A:=TRUNC(X/1000);
    B:=TRUNC((X-1000*A)/100);
    C:=TRUNC((X-1000*A-100*B)/10);
    D=X-1000*A-100*B-10*C;
    IF X>5000 THEN Y:=(B+D)/X ELSE Y:=(A+C)/X;
    WRITE(Y)
END.
```



Նկ. 2.5

Տրված քառանիշ թիվը նշանակենք $x=\overline{abcd}$: Քառանիշ թվի թվանշանները կարելի է գտնել հետևյալ կերպ.

$$a = [x/1000],$$
$$b = [(x-1000a)/100],$$
$$c = [(x-1000a-100b)/10],$$
$$d = x-1000a-100b-10c,$$

որտեղ [ ] փակագծերը նշանակում են, որ վերցվում է նրանց մեջ առնված արտահայտության արժեքի ամբողջ մասը:

Ծրագրում քառանիշ թվի թվանշանները կարելի է գտնել նաև div և mod գործողությունների օգնությամբ.

```
a:=x div 1000;
b:=(x mod 1000) div 100;
c:=((x mod 1000) mod 100) div 10;
d:=((x mod 1000) mod 100) mod 10;
```

2.3. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակ 2-ի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Աղյուսակ 2

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	1, 20, 21, 41, 51
2	2, 19, 22, 43, 53
3	3, 18, 25, 44, 56
4	4, 17, 26, 48, 57
5	5, 16, 27, 46, 55
6	6, 15, 28, 42, 59
7	7, 14, 28, 45, 54
8	8, 13, 27, 47, 53
9	9, 12, 26, 42, 60
10	10, 11, 25, 47, 60
11	1, 11, 22, 48, 52
12	2, 12, 21, 41, 57
13	3, 13, 21, 48, 58
14	4, 14, 22,43, 51
15	5, 15, 25, 42, 54
16	6, 16, 26, 44, 59
17	7, 17, 27, 49, 53
18	8, 18, 28, 46, 58
19	9, 19, 21, 45, 56
20	10, 20, 22, 42, 59
21	1, 13, 25, 41, 52
22	2, 14, 26, 49, 58
23	3, 15, 27, 48, 55
24	4, 16, 28, 47, 57
25	5, 17, 27, 45, 54
26	6, 18, 26, 43, 56
27	7, 19, 28, 42, 51
28	8, 20, 25, 47, 60
29	9, 11, 22, 46, 52
30	10, 12, 21, 44, 55

3. ՑԻԿԼԱՅԻՆ ԾՐԱԳՐԵՐԻ ԿԱԶՄԱԿԵՐՊՈՒՄ

3.1. Ցիկլային ծրագրեր

Հաճախ անհրաժեշտություն է առաջանում կրկնել գործողությունների որոշակի խումբ մեկից ավելի անգամներ: Այդպիսի պրոցեսներ նկարագրող ծրագիրը կոչվում է ցիկլային, իսկ գործողությունների կրկնվող հատվածները՝ ցիկլեր:

Ցիկլի կազմակերպման համար անհրաժեշտ է ունենալ պրոցեսի նախնական տվյալները բնորոշող որոշակի փոփոխական (պարամետր) իր արժեքների ստացման հայտնի օրենքով: Ուստի անհրաժեշտ է նախատեսել օպերատորներ, որոնք տան պարամետր-փոփոխականի ինչպես նախնական արժեքը, այնպես էլ ամեն անգամ կրկնվելուց առաջ ապահովեն պարամետրի նոր արժեքի ստացումը: Բացի այդ, ցիկլը պետք է պարունակի պայմանական օպերատոր, որն ամեն անգամ ստուգի պարամետրի վերջնական արժեքին հասնելու պայմանը, եթե այդ արժեքը դեռևս չի նվաճվել, ապա ցիկլը կրկնվում է, հակառակ դեպքում ցիկլն ավարտվում է, և կատարվում է անցում ցիկլային հատվածին հաջորդող օպերատորին: Ցիկլի ավարտի պայմանը կարող է նաև անմիջականորեն կապված չլինել ցիկլի պարամետրի վերջնական արժեքի հետ: Այն կարող է պայմանավորված լինել ցիկլում ստացվող այլ փոփոխական մեծության արժեքով:

Ցիկլային ձևերի կիրառումը, վերջին հաշվով, հնարավորություն է տալիս քոմփյութերում կատարվող բազմաթիվ գործողությունների հաջորդականությունը ներկայացնել սեղմ ձևով:

ՊԱՍԿԱԼ-ում կիրառվում են ցիկլի երեք տիպի օպերատորներ՝

- պարամետրով ցիկլի օպերատոր,
- նախապայմանով ցիկլի օպերատոր,
- հետպայմանով ցիկլի օպերատոր:

3.2. Պարամետրով ցիկլի օպերատոր

Շատ խնդիրներում նախօրոք հայտնի է, թե քանի անգամ պետք է կատարվի ցիկլում ընդգրկված գործողությունների հաջորդականությունը: Այդ տիպի խնդիրների լուծման ծրագիրը կազմելիս

նպատակահարմար է օգտագործել հատուկ տիպի ցիկլի օպերատոր, որի ներկայացման երկու հնարավոր ձևերը հետևյալներն են՝

FOR i:=m TO n DO s

կամ

FOR i:=m DOWNTO n DO s,

որտեղ՝

i - ցիկլի պարամետրն է, որը դիսկրետ տիպի (ամբողջ, սիմվոլային, տրամաբանական և այլն) փոփոխական է,

m, n - ցիկլի պարամետրի նախնական և վերջնական արժեքները որոշող արտահայտություններ են, որոնք ցիկլի պարամետրի հետ պետք է ունենան նույն տիպը,

s -ը պարզ կամ բաղադրյալ օպերատոր է:

Ցիկլի կատարման համար սկզբում հաշվվում և հիշվում են ցիկլի պարամետրի սկզբնական և վերջնական արժեքները: Այնուհետև ցիկլի i պարամետրին վերագրվում է նախնական արժեքը, որից հետո ցիկլի պարամետրի արժեքը համեմատվում է վերջնական արժեքի հետ: Քանի դեռ ցիկլի պարամետրի արժեքը փոքր է վերջնական արժեքից կամ հավասար (ցիկլի FOR ... TO տիպի օպերատորում) կամ մեծ վերջնական արժեքից կամ հավասար (ցիկլի FOR ... DOWNTO տիպի օպերատորում) կատարվում է ցիկլի հերթական կրկնումը, հակառակ դեպքում ցիկլն ավարտվում է: Ամեն մի հերթական կրկնման ժամանակ սկզբում կատարվում է s օպերատորը, իսկ հետո ցիկլի պարամետրին վերագրվում է հաջորդ մեծ արժեքը (ցիկլի FOR ... TO տիպի օպերատորում) կամ հաջորդ փոքր արժեքը (ցիկլի FOR ... DOWNTO տիպի օպերատորում):

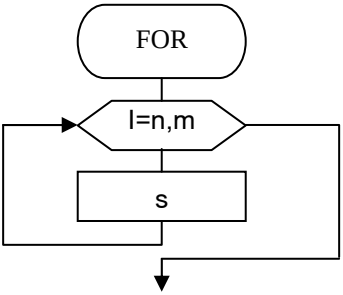
Ցիկլի աշխատանքը դիտարկենք մասնավոր դեպքում՝ ցիկլի պարամետրն ամբողջ տիպ է:

Առաջին դեպքում (FOR ... TO ձև) s օպերատորը սկզբում կատարվում է i պարամետրի m արժեքի համար, որից հետո i պարամետրի արժեքը մեծացվում է մեկով և նորից կատարվում է s օպերատորը, ապա i -ն մեծացվում է ևս մեկով և կատարվում է s-ը և այսպես այնքան ժամանակ, քանի դեռ ճշմարիտ է $i \le n$ պայմանը:

Երկրորդ դեպքում (FOR ... DOWNTO ձև) s օպերատորը սկզբում կատարվում է i պարամետրի i=m արժեքի համար, որից հետո նրա արժեքը փոքրացվում է 1-ով և այն կատարվում է այնքան ժամանակ, քանի դեռ ճշմարիտ է $i \ge n$ պայմանը:

Այսպիսով, երկու դեպքում էլ ցիկլը ամբողջությամբ կառավարվում է i պարամետրով, որի արժեքները փոփոխվում են +1 կամ -1 քայլով:

Ցիկլի FOR տիպի օպերատոր տրամաբանական կառուցվածքային սխեման, երբ ցիկլի պարամետրը ամբողջ տիպի է, բերված է նկ.3.1-ում:



Նկ.3.1. Ցիկլի FOR տիպի օպերատորի տրամաբանական սխեման

3.3. Նախապայմանով ցիկլի օպերատոր

Նախապայմանով ցիկլի օպերատորը կազմակերպում է ինչպես կրկնությունների հայտնի, այնպես էլ անհայտ թվով ցիկլերի իրագործումը:

Օպերատորի ընդհանուր տեսքը հետևյալն է՝

WHILE b DO s,

որտեղ՝

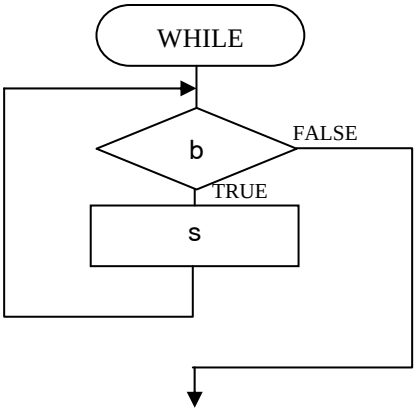
b-ն տրամաբանական արտահայտություն է (մասնավոր դեպքում՝ պայման),

s-ը ցանկացած օպերատոր է (պարզ կամ բաղադրյալ), որը կոչվում է **ցիկլի մարմին**:

Ցիկլի այս տիպի օպերատորի իմաստը հետևյալն է. s օպերատորը կրկնվում է այնքան, քանի դեռ ճշմարիտ է b տրամաբանական արտահայտությունը: Ակնհայտ է, որ ամեն անգամ, նախքան s-ի կատարվելը, հաշվվում է b-ի արժեքը: Ցիկլն ավարտվում է, եթե b տրամաբանական արտահայտությունը ստանում է կեղծ արժեք: Քանի որ տրամաբանական արտահայտության ճշմարտացիությունը ստուգվում է յուրաքանչյուր կրկնման սկզբում, ապա ցիկլի մարմինը կարող է չկատարվել ոչ մի անգամ (եթե ի սկզբանե արտահայտության արժեքը կեղծ է): Ցիկլի մարմինը կարող է ընդգրկել միայն մեկ օպերատոր: Եթե ցիկլի մարմնում օպերատորների քանակը մեկից ավելի է, ապա

անհրաժեշտ է այդ օպերատորներից ձևավորել բաղադրյալ օպերատոր, այսինքն՝ այդ օպերատորները վերցնել BEGIN և END բառերի մեջ:

WHILE ցիկլի օպերատորի տրամաբանական սխեման բերված է նկ. 3.2-ում:



Նկ.3.2. WHILE- տիպի օպերատորի տրամաբանական սխեման

3.4. Հետպայմանով ցիկլի օպերատոր

Հետպայմանով ցիկլի օպերատորը կազմակերպում է ինչպես կրկնությունների նախապես հայտնի, այնպես էլ անհայտ թվով ցիկլերի կատարումը:

Օպերատորի ընդհանուր տեսքը հետևյալն է՝

REPEAT s_1; s_2; ...; s_n UNTIL b,

որտեղ՝ s_1; s_2; ...; s_n -ը կատարվող օպերատորների հաջորդականություն է (ցիկլի մարմին), b -ն տրամաբանական արտահայտություն է (մասնավոր դեպքում պայման):

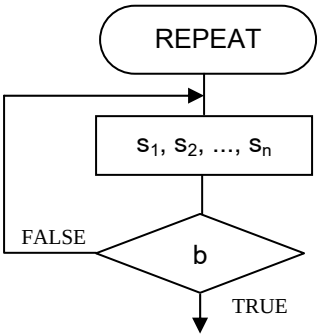
Ներկայացված ցիկլի օպերատորում սկզբում կատարվում են REPEAT և UNTIL բառերի միջև ընդգրկված s_1, s_2, ..., s_n օպերատորները, որից հետո հաշվվում է b-ի արժեքը(մասնավոր դեպքում ստուգվում է պայմանը), և եթե այն կեղծ է, նորից կատարվում են s_1, s_2, ..., s_n օպերատորները, հակառակ դեպքում ցիկլն ավարտվում է:

Ի տարբերություն նախորդ կետում դիտարկված WHILE օպերատորի, այս դեպքում պայմանը տրվում է վերջում: Այդ իսկ պատճառով ցիկլի մարմինը կատարվում է գոնե մեկ անգամ, հետևաբար կոնկրետ խնդրի

լուծման ժամանակ ցիկլի տեսքն ընտրելիս անհրաժեշտ է պարզել, թե ցիկլի մարմինը գոնե մեկ անգամ պետք է կատարվի, թե ոչ:

Ցիկլի այս ձևում s_1, s_2, ..., s_n օպերատորների հաջորդականությունը չի միավորվում BEGIN և END «օպերատորային փակագծերով» որպես մեկ բաղադրյալ օպերատոր, քանի որ ցիկլի օպերատորի REPEAT ... UNTIL գրառման պարփակ ձևը իր մեջ արդեն պարունակում է փակագծերի իմաստ: Այդ պատճառով է նաև, որ UNTIL բառից առաջ կետ-ստորակետ դնելը պարտադիր չէ:

REPEAT տիպի ցիկլի օպերատորի տրամաբանական սխեման ներկայացված է նկ.3.3-ում:



Նկ. 3.3. REPEAT տիպի օպերատորի տրամաբանական սխեման

3.5. Ցիկլի օպերատորների կիրառման հիմնական կանոնները

Խնդրի դրվածքից կախված՝ կարելի է կիրառել ցիկլի նշված ձևերից որևէ մեկը:

Ցիկլի օպերատորի բոլոր ձևերում ցիկլի բնական ավարտից հետո, այսինքն, երբ ցիկլում նշված բոլոր կրկնումները կատարված են, ավտոմատ կերպով ցիկլը ավարտվում է, և կատարում է անցնում ցիկլին անմիջապես հաջորդող օպերատորին: Սակայն հնարավոր է նաև ցիկլի վաղաժամ ընդհատում, երբ ցիկլի ներսում տեղավորված որևէ անցման օպերատոր նախքան ցիկլի լրիվ ավարտը կառավարումը փոխանցվում է ցիկլից դուրս գտնվող այլ օպերատորի: Հակառակը, այսինքն՝ կառավարման փոխանցումը արտաքին օպերատորից ցիկլի ներսում գտնվող որևէ օպերատորի չի թույլատրվում: Դա բացատրվում է նրանով, որ այս դեպքում ցիկլի սկիզբը բաց է թողնվում, հետևաբար բացակայում է տեղեկությունը պարամետրի ընթացիկ արժեքի վերաբերյալ: Այսպիսով

ցիկլի մեջ մտնել հնարավոր է միայն անցնելով նրա սկզբով, իսկ դուրս գալ ցիկլից հնարավոր է նաև՝ շրջանցելով նրա բնական ավարտը:

ՏՈՒՐԲՈ ՊԱՍԿԱԼ-ի 7-րդ տարբերակում FOR, WHILE և REPEAT ցիկլերում կարելի է օգտագործել երկու նոր ստանդարտ պրոցեդուրաներ. BREAK և CONTINUE:

BREAK պրոցեդուրան թույլատրում է վաղաժամ ընդհատել ցիկլը՝ դեռևս կատարված չլինելով ցիկլի ավարտի պայմանը: BREAK պրոցեդուրայի կատարումից հետո կառավարումը տրվում է ցիկլի օպերատորի հաջորդող օպերատորին:

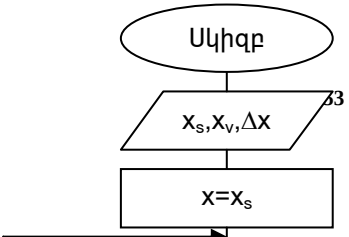
CONTINUE պրոցեդուրան թույլատրում է վաղաժամկետ ավարտել հերթական շրջափուլը: CONTINUE պրոցեդուրայի կատարումից հետո սկսվում է ցիկլի նոր շրջափուլ:

Հնարավոր է **ներդրված ցիկլերի** կիրառումը, երբ ցիկլն իր մեջ պարունակում է այլ ցիկլ: Այդ դեպքում առաջանում են «ներքին ցիկլ» և «արտաքին ցիկլ» հասկացությունները: Անհրաժեշտ է, **ներքին ցիկլը** ամբողջությամբ տեղավորվի **արտաքին ցիկլի** մեջ, այսինքն չի թույլատրվում տարբեր ցիկլերի տիրույթների հատումը: Ներդրված ցիկլերը պետք է ունենան տարբեր պարամետրեր: Արտաքին ցիկլի պարամետրի յուրաքանչյուր արժեքի համար ներքին ցիկլը իրագործվում է ամբողջությամբ, եթե ներքին ցիկլը չի պարունակում վաղաժամ ավարտի հանգեցնող անցման օպերատոր:

3.6. Պարզ ծրագրերի տիպային օրինակներ

Օրինակ 3.1. Կազմել բլոկ-սխեմա և ծրագիր, որոնք օգտագործելով ցիկլի WHILE օպերատորը՝ $x\in[-2,5;\ 4,5]$ միջակայքում, $\Delta x=0,3$ քայլով կհաշվեն $Y=x+3x^2$ ֆունկցիայի արժեքները: Տպել x և y փոփոխականների արժեքների աղյուսակը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.4-ում, իսկ ծրագիրն ունի EXAM3_1 անունը:



Նկ. 3.4

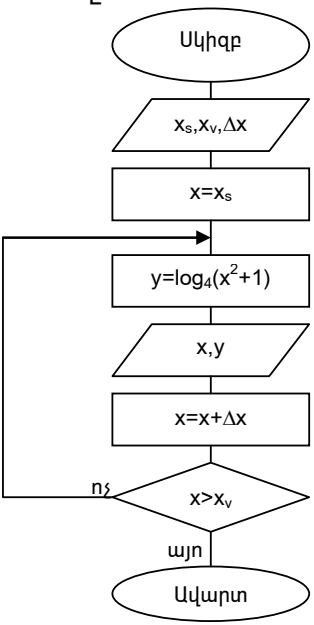
```
PROGRAM EXAM3_1;
CONST XS=-2.5;XV=4.5;DX=0.3;
VAR X,Y:REAL;
BEGIN
 X:=XS;
   WHILE X<=XV DO
   BEGIN
    Y:=X+3*SQR(X);
    WRITELN(X,Y);
    X:=X+DX
    END
END.
```

x փոփոխականին վերագրված է միջակայքի սկզբնական -2,5 արժեքը, որը նախապայմանով ցիկլի մարմնի մեջ մեծացնում է իր արժեքը dx(Δx) քայլով: Ցիկլի մարմնում x-ի յուրաքանչյուր արժեքի համար հաշվվում է ֆունկցիայի արժեքը, արտածվում x-ի և y-ի արժեքները: Ցիկլի մարմինը կատարվում է այնքան, քանի դեռ ճշմարիտ է x≤4,5 պայմանը: Ծրագրում Ֆունկցիայի հաշվումը և x ու y փոփոխականների արտածումը կազմակերպված է ցիկլի WHILE օպերատորի օգնությամբ: Քանի որ ցիկլի մարմինը պարունակում է մեկից ավելի օպերատորներ, ապա այն

կազմակերպված է որպես բաղադրյալ օպերատոր, այսինքն վերցված է BEGIN և END հատուկ բառերի միջև:

Օրինակ 3.2. Կազմել բլոկ-սխեմա և ծրագիր, որոնք օգտագործելով ցիկլի REPEAT օպերատորը՝ $x\in[-12; 12]$ միջակայքում, $\Delta x=2$ քայլով կհաշվեն $Y=\log_4(x^2+1)$ ֆունկցիայի արժեքները: Տալ x և y փոփոխականների արժեքների աղյուսակը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.5-ում, իսկ ծրագիրն ունի EXAM3_2 անունը:



Նկ. 3.5

```
    PROGRAM EXAM3_2;
    CONST XS=-12;XV=12;DX=2;
    VAR X,Y:REAL;
BEGIN
  X:=XS;
    REPEAT
     Y:=LN(SQR(X)+1)/LN(4);
     WRITELN(X,Y);
     X:=X+DX
```

```
    UNTIL X>XV
END.
```

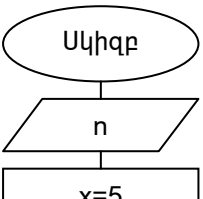
x փոփոխականին վերագրված է միջակայքի սկզբնական -12 արժեքը, որը հետպայմանով ցիկլի մարմնի մեջ մեծացնում է իր արժեքը dx(Δx) քայլով: Ցիկլի մարմնում x-ի յուրաքանչյուր արժեքի համար հաշվվում է ֆունկցիայի արժեքը, արտածվում x-ի և y-ի արժեքները: Ցիկլն ավարտվում է x>12 պայմանի դեպքում: Ծրագրում Ֆունկցիայի հաշվումը և x ու y փոփոխականների արտածումը կազմակերպված է ցիկլի REPEAT օպերատորի օգնությամբ:

Օրինակ 3.3. Կազմել բլոկ-սխեմա և ծրագիր, որոնք տրված n բնական թվի համար կհաշվեն ու կտպեն տրված արտահայտության արժեքը.

$$\sum_{i=1}^{n} x_i^{\,2}$$, որտեղ $x_1=5$; $x_i=2+\sin x_{i-1}$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.6-ում, իսկ ծրագիրն ունի EXAM3_3 անունը:

```
    PROGRAM EXAM3_3;
    VAR X,S:REAL;I,N:INTEGER;
BEGIN
    READ(N);
    X:=5;
    S:=0;
    FOR I:=1 TO N DO
    BEGIN
    S:=S+SQR(X);  X:=2+SIN(X);
    END;
    WRITE(S)
END.
```



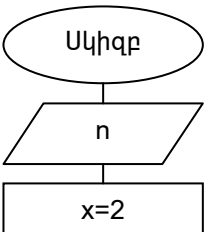
Նկ.3.6

Գումարի համար նախատեսված S փոփոխականին ցիկլից դուրս վերագրվում է 0 արժեք, իսկ x փոփոխականին՝ իր նախնական 5 արժեքը: Գումարի բոլոր գումարելիները հաշվվում և դրանց քառակուսիների արժեքները ավելացվում են S փոփոխականին ցիկլի մարմնում: Այստեղ պետք է հիշել վերագրման օպերատորի կատարման կարգը, այն է՝ սկզբից հաշվվում է արտահայտության աջ մասի արժեքը, և այն վերագրվում ձախում գրված փոփոխականին: Խնդրի լուծման ժամանակ անհրաժեշտություն չկա x փոփոխականը հայտարարել որպես զանգված, քանի որ ցիկլում հաշվված x փոփոխականի ամեն մի արժեքի քառակուսին անմիջապես ավելացվում է գումարի համար նախատեսված S փոփոխականին:

Օրինակ 3.4. Կազմել բլոկ-սխեմա և ծրագիր, որոնք տրված n բնական թվի համար կհաշվեն ու կտպեն տրված արտահայտության արժեքը.

$\prod_{i=1}^{n}\left(x_i^{\,2}+y_i^{\,2}\right)$, որտեղ $x_1=2$; $x_i=\sin x_{i-1}$; $y_1=3$; $y_i=\cos y_{i-1}$:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.7-ում, իսկ ծրագիրն ունի EXAM3_4 անունը:



Նկ. 3.7

```
    PROGRAM EXAM3_4;
    VAR X,Y,P:REAL;I,N:INTEGER;
BEGIN
    READ(N); X:=2; Y:=3;P:=1;
    FOR I:=1 TO N DO
    BEGIN
     P:=P*(SQR(X)+SQR(Y));
     X:=SIN(X);  Y:=COS(Y)
    END;
    WRITE(P)
 END.
```

Արտադրյալի համար նախատեսված P փոփոխականին ցիկլից դուրս վերագրվում է 1 արժեք, իսկ x և y փոփոխականներին իրենց նախնական՝ 2 և 3 արժեքները: Բոլոր արտադրիչները հաշվվում և նրանց քառակուսիների գումարի արժեքները բազմապատկվում են P փոփոխականով ցիկլի մարմնում: Այստեղ ևս պետք է հիշել վերագրման օպերատորի կատարման կարգը, այն է՝ սկզբում հաշվվում է

արտահայտության աջ մասի արժեքը, և այն վերագրվում ձախում գրված փոփոխականին: Խնդրի լուծման ժամանակ անհրաժեշտություն չկա x և y փոփոխականները հայտարարելու որպես զանգված, քանի որ ցիկլում հաշվված x և y փոփոխականների ամեն մի արժեքների քառակուսիների գումարը անմիջապես բազմապատկվում է արտադրյալի համար նախատեսված P փոփոխականով:

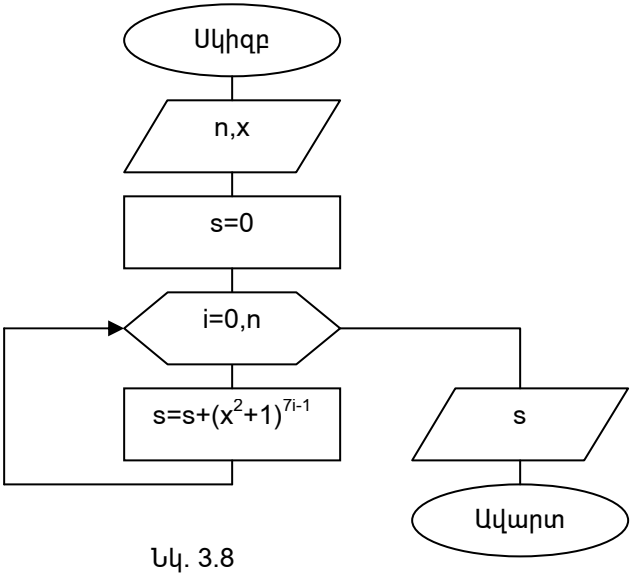
Օրինակ 3.5. Կազմել բլոկ-սխեմա և ծրագիր, որոնք x դրական և n բնական թվերի համար կհաշվեն և կտպեն

$$\left(x^2+1\right)^{-1}+\left(x^2+1\right)^{6}+\left(x^2+1\right)^{13}+\cdots+\left(x^2+1\right)^{7n-1}$$

արտահայտության արժեքը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.8-ում, իսկ ծրագիրն ունի EXAM3_5 անունը:

```
    PROGRAM EXAM3_5;
    VAR X,S:REAL;I,N:INTEGER;
BEGIN
    READ(N,X);
  S:=0;
    FOR I:=0 TO N DO
     S:=S+EXP((7*I-1)*LN(SQR(X)+1));
     WRITE(S)
 END.
```



Նկ. 3.8

Դժվար չէ նկատել, որ

$$s=\left(x^2+1\right)^{-1}+\left(x^2+1\right)^{6}+\left(x^2+1\right)^{13}+\cdots+\left(x^2+1\right)^{7n-1}$$

գումարը կարելի է ներկայացնել $s=\sum_{i=0}^{n}\left(x^2+1\right)^{7i-1}$ տեսքով և, այսպիսով՝ խնդիրը հանգեցնել պարզագույն գումարի հաշվման: Ծրագրում $\left(x^2+1\right)^{7i-1}$ արտահայտությունը հաշվված է հետևյալ կերպ. $\left(x^2+1\right)^{7i-1}=e^{(7i-1)\ln\left(x^2+1\right)}$:

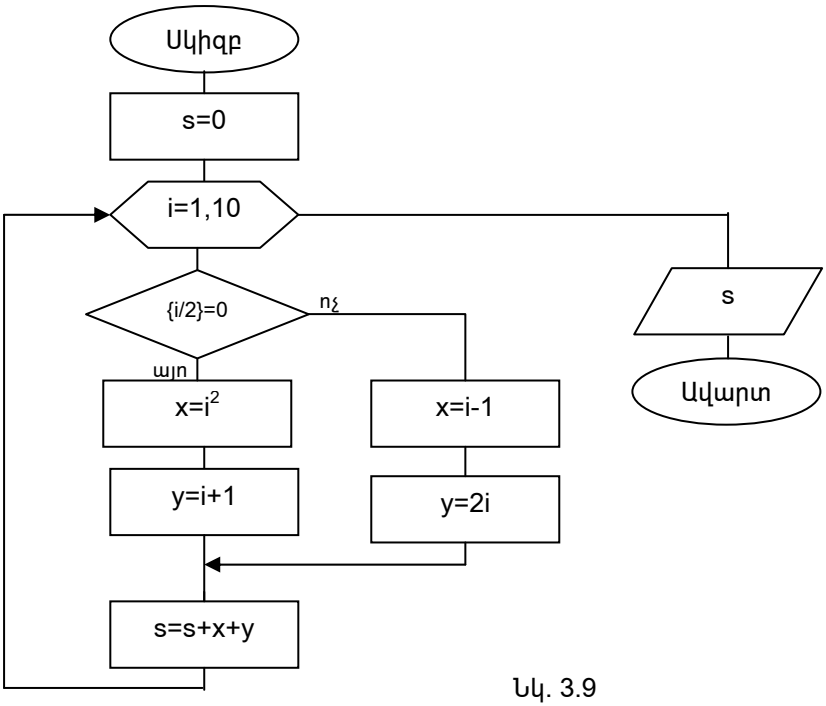
Օրինակ 3.6. Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն ու կտպեն $\sum_{i=1}^{10}(x_i+y_i)$ արտահայտության արժեքը,

որտեղ

$$x_i = \begin{cases} i^2 & \text{, եթե i-ն զույգ է,} \\ i-1 & \text{, հակառակ դեպքում:} \end{cases}$$

$$y_i = \begin{cases} i+1 & \text{, եթե i-ն զույգ է,} \\ 2i & \text{, հակառակ դեպքում:} \end{cases}$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.9-ում, իսկ ծրագիրն ունի EXAM3_6 անունը:



Նկ. 3.9

```
    PROGRAM EXAM3_6;
    VAR X,Y,S:REAL;I:INTEGER;
BEGIN
    S:=0;
    FOR I:=1 TO 10 DO
    BEGIN
```

```
    IF I MOD 2=0 THEN BEGIN X:=SQR(I); Y:=I+1 END
    ELSE BEGIN X:=I-1; Y:=2*I END;
    S:=S+X+Y
   END;
   WRITE(S)
END.
```

Կազմակերպել է մեկ ցիկլ, որի մեջ հաշվվում են x և y փոփոխականների արժեքները, որոնց գումարի արժեքը անմիջապես ավելացվում է գումարի համար նախատեսված S փոփոխականին: Քանի որ ՊԱՍԿԱԼ լեզվում պայմանի օպերատորը կարող է պարունակել միայն մեկ օպերատոր, իսկ մեր ծրագրում դրանք երկուսն են, այդ պատճառով յուրաքանչյուր ճյուղում x և y փոփոխականների ստացումը կազմակերպված է բաղադրյալ օպերատորի միջև, այսինքն այդ օպերատորները վերցված են BEGIN և END հատուկ բառերի միջև:

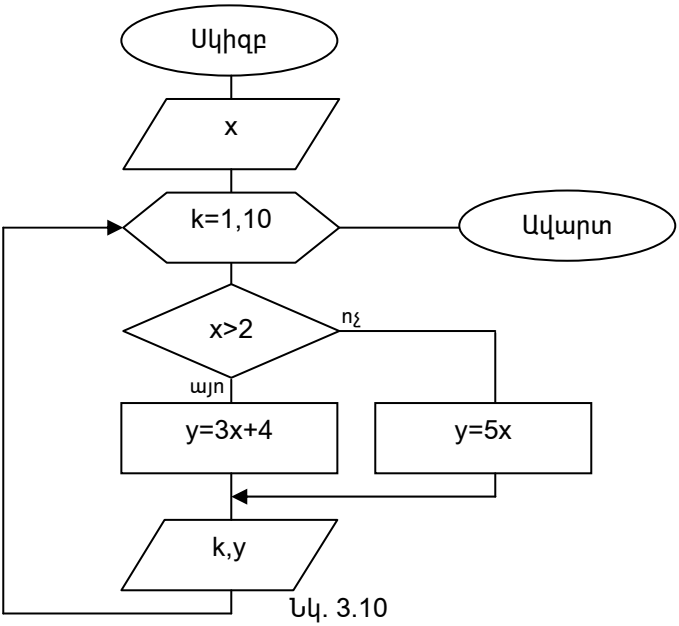
Օրինակ 3.7. Կամայական x իրական և k պարամետրերի տրված արժեքների համար կազմել տրված ֆունկցիայի արժեքների հաշվման և արտածման բլոկ-սխեմաներն ու ծրագրերը.

$$Y=\begin{cases}3x+4 & \text{, եթե x>2,}\\ 5x & \text{, հակառակ դեպքում,}\end{cases}$$

որտեղ k=1, 2, ..., 10:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.10-ում, իսկ ծրագիրն ունի EXAM3_7 անունը:

```
   PROGRAM EXAM7;
   VAR X,Y:REAL;I,K:INTEGER;
BEGIN     READ(X);
   FOR K:=1 TO 10 DO
   BEGIN
   IF X>2 THEN Y:=3*x+4 ELSE Y:=5*X;
   WRITELN(K,Y)
   END
END.
```



Նկ. 3.10

Քանի որ ֆունկցիայի արժեքները պետք է հաշվել k-ի տարբեր արժեքների համար, ապա կազմակերպվել է k աճող պարամետրով ցիկլ, որի մեջ հաշվվում և արտածվում են տրված ֆունկցիայի արժեքները:

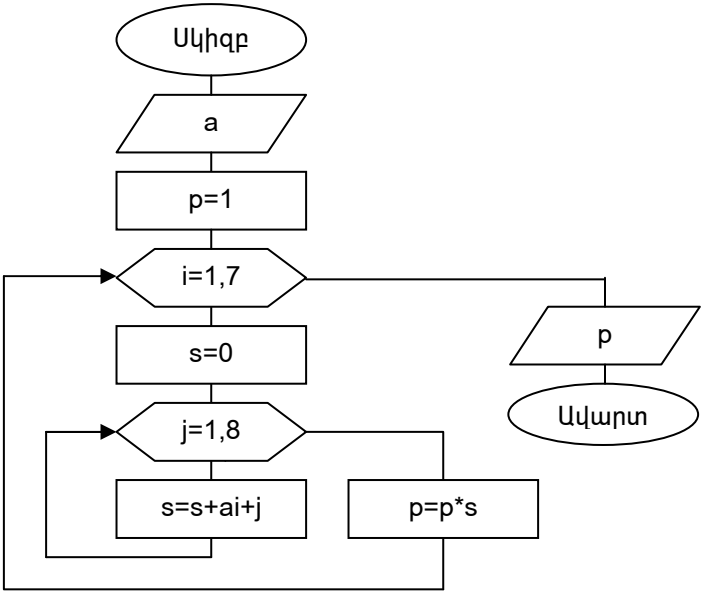
Օրինակ 3.8. Կազմել բլոկ-սխեմա ու ծրագիր, որոնք տառային պարամետրերի կամայական թվային արժեքների համար կհաշվեն և կտպեն տրված արտահայտության արժեքը.

$$p=\prod_{i=1}^{7}\sum_{j=1}^{8}(ai+j):$$

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.3.11-ում, իսկ ծրագիրն ունի EXAM3_8 անունը:

```
    PROGRAM EXAM3_8;
    VAR P,S,A:REAL; I,J:INTEGER;
BEGIN
    READ(A); P:=1;
    FOR I:=1 TO 7 DO
    BEGIN
```

```
    S:=0;
   FOR J:=1 TO 8 DO  S:=S+A*I+J;
   P:=P*S
   END;
   WRITE(P)
END.
```



Նկ.3.11

Կազմակերպված են ներդրված երկու ցիկլեր, որոնցից ներքինում հաշվվում է հերթական գումարը (S), այնուհետև ստացված արժեքը արտաքին ցիկլի մեջ բազմապատկվում է արտադրյալի կազմակերպման համար նախատեսված փոփոխականով (P):

3.7. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակ 1-ի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	61, 71, 81, 91, 101, 111, 130, 133
2	62, 73, 86, 97, 103, 112, 129, 139
3	63, 77, 82, 93, 104, 113, 128, 132
4	64, 75, 85, 94, 105, 114, 127, 140
5	65, 72, 87, 95, 102, 115, 126, 138
6	66, 78, 83, 92, 107, 116, 125, 131
7	67, 76, 88, 96, 109, 117, 124, 137
8	68, 74, 84, 99, 108, 118, 123, 134
9	69, 80, 89, 98, 106, 119, 122, 135
10	70, 79, 90, 100, 110, 120, 121, 136
11	63, 76, 84, 97, 108, 111, 130, 131
12	64, 73, 82, 93, 101, 112, 129, 133
13	65, 75, 85, 99, 105, 113, 128, 135
14	66, 71, 83, 91, 102, 114, 127, 136
15	67, 74, 87, 94, 106, 115, 126, 138
16	68, 77, 86, 95, 103, 116, 125, 138
17	69, 72, 81, 92, 104, 117, 124, 139
18	70, 78, 88, 96, 107, 118, 123, 132
19	61, 79, 90, 98, 108, 119, 122, 137
20	62, 80, 83, 100, 109, 120, 121, 134
21	63, 73, 85, 97, 110, 120, 121, 140
22	64, 79, 89, 100, 102, 119, 122, 135
23	65, 75, 89, 93, 109, 118, 123, 137
24	66, 80, 84, 99, 110, 117, 124, 134
25	67, 71, 88, 98, 107, 116, 125, 139
26	68, 78, 90, 95, 105, 115, 126, 138
27	69, 76, 81, 91, 101, 114, 127, 131
28	70, 74, 86, 94, 106, 113, 128, 136
29	61, 77, 82, 96, 103, 112, 129, 132
30	62, 72, 87, 92, 104, 111, 130, 133

4. ԶԱՆԳՎԱԾՆԵՐ

4.1. Զանգվածներ

Որոշ խնդիրների լուծման ժամանակ անհրաժեշտ է լինում օգտագործել բավական մեծ թվով նույնատիպ փոփոխականներ: Քանի որ այդպիսի փոփոխականների նկարագրումը և կիրառումը սովորական ձևերով հարմար չէ, ապա հնարավորություն է տրված դրանք սեղմ ձևով ներկայացնել որպես մեկ զանգված:

Զանգվածը այնպիսի նույնատիպ տարրերի կարգավորված հավաքածու է, որոնք ներկայացվում են միևնույն իդենտիֆիկատորով և իրարից տարբերվում են ինդեքսի (համարի) արժեքով:

Օրինակ, A, B, C, D փոփոխականները, նույնիսկ եթե նույնատիպ են, փոփոխականների զանգված չեն կազմում, իսկ x_1, x_2, ... , x_n փոփոխականները կազմում են զանգված՝ X անունով: Այդ զանգվածի տարրերը իրարից տարբերվում են ինդեքսով:

Զանգվածի առանձին տարրը որոշող ինդեքսների քանակը որոշում է զանգվածի **չափողականությունը**: Այսպես, վերը նշված X զանգվածն ունի մեկ ինդեքս, հետևաբար միաչափ զանգված է: Թույլատրվում է նաև բազմաչափ զանգվածների կիրառում:

Զանգվածի տարրը նշելու համար կիրառվում է **ինդեքսով փոփոխական**, որը ներկայացվում է զանգվածի անունով, հաջորդող ինդեքսների ցուցակով՝ վերցված քառակուսի փակագծերի մեջ: Ինդեքսները միմյանցից բաժանվում են ստորակետով: Օրինակ, X[N]; X[18]; Z[I,J]; A[1,3,K]:

Այսպիսով, առանձին տարրի տեղը զանգվածում միարժեքորեն որոշվում է ինդեքսների արժեքներով: Ինդեքսները կարող են տրվել անմիջականորեն ամբողջ թվերով, ինչպես նաև փոփոխականով կամ արտահայտությամբ: Վերջին դեպքում, որպես ինդեքսի արժեք, պետք է վերցվի արտահայտության արժեքի ամբողջ մասը:

Զանգվածներով աշխատելիս անհրաժեշտ է նախօրոք դրանք նկարագրել, ցույց տալով դրանց չափողականությունը, տարրերի թիվը և տիպը: Դա հնարավորություն է տալիս քոմփյութերին հիշողության մեջ համապատասխան դաշտ հատկացնել զանգվածի բոլոր տարրերին: Զանգվածի հայտարարման համար կիրառվում է ARRAY (զանգված) հատուկ բառ նկարագրիչը: Նկարագրումը կատարվում է պարզ տիպի փոփոխականների նկարագրման հետ (VAR բաժնում):

Օրինակ՝

```
VAR X: ARRAY[1..30] OF REAL
```

նկարագրումը ցույց է տալիս, որ X-ը միաչափ զանգված է, որն ունի 30 իրական տիպի տարրեր: Այդ զանգվածի տարրեր են՝ X[1], X[2], ... , X[30] ինդեքսով փոփոխականները: Նկարագրման ժամանակ քառակուսի փակագծերում ներկայացված 1..30 գրառումը կոչվում է սահմանային զույգ՝ 1 ներքին և 30 վերին սահմաններով:

Հնարավոր է կիրառել նաև զանգվածի հայտարարման հետևյալ ձևերը՝

```
TYPE Y=ARRAY[1..20] OF REAL;
VAR X:Y;
```

Առաջին տողում Y տիպը սահմանվում է որպես 20 տարրանոց միաչափ զանգված, որից հետո փոփոխականների VAR բաժնում X փոփոխականին վերագրվում է Y տիպը: Այսպիսով, X փոփոխականը ունենում է ARRAY[1..20] OF REAL տիպը: Այս եղանակը գերադասելի է այն դեպքում, երբ հայտարարվում են նույնատիպ մի քանի զանգվածներ, որոնք ունեն նույն թվով տարրեր: Օրինակ, X,Y և Z նմանատիպ զանգվածների նկարագրումը՝

```
VAR X:ARRAY[1..50] OF REAL;
    Y:ARRAY[1..50] OF REAL;
    Z:ARRAY[1..50] OF REAL;
```

կարող է փոխարինվել հետևյալ նկարագրությամբ՝

```
TYPE A=ARRAY[1..50] OF REAL;
VAR X,Y,Z:A;
```

Այստեղ առաջին տողում հայտարարվում է A տիպը, որպես 50 տարրանոց իրական թվերի միաչափ զանգված: Երկրորդ տողում հայտարարվում է, որ X, Y և Z փոփոխականները A տիպի են, այսինքն ունեն ARRAY[1..50] OF REAL տիպը:

ՊԱՍԿԱԼ-ում թույլատրվում է նույնատիպ զանգվածների հետ վերագրումների կատարում: Օրինակ, եթե X և Y նույնատիպ զանգվածներ են՝

```
TYPE B=ARRAY[1..25] OF INTEGER;
VAR X,Y:B;
```

ապա X:=Y գրառումով X զանգվածի բոլոր տարրերին վերագրվում են Y զանգվածի համապատասխան տարրերի արժեքները:

Ինչպես արդեն նշվել է, թույլատրվում է նաև բազմաչափ զանգվածների կիրառում: Այդ դեպքում n-չափանի զանգվածի տարրերը որոշվում են n հատ ինդեքսներով, որոնք միմյանցից բաժանվում են ստորակետերով:

Բազմաչափ զանգվածների նկարագրությունում սահմանային զույգերը նշվում են բոլոր ինդեքսների համար առանձին-առանձին, իրարից բաժանելով ստորակետերով: Այսպես, երկչափ զանգվածի դեպքում կունենանք՝

```
VAR A:ARRAY[1..6,1..8] OF INTEGER;
```

Այս հայտարարումով ներկայացվում են հետևյալ փոփոխականները՝

```
A[1,1]  A[1,2] ... A[1,8]
A[1,1]  A[1,2] ... A[1,8]
...     ...        ...
A[6,1]  A[6,2] ... A[6,8]
```

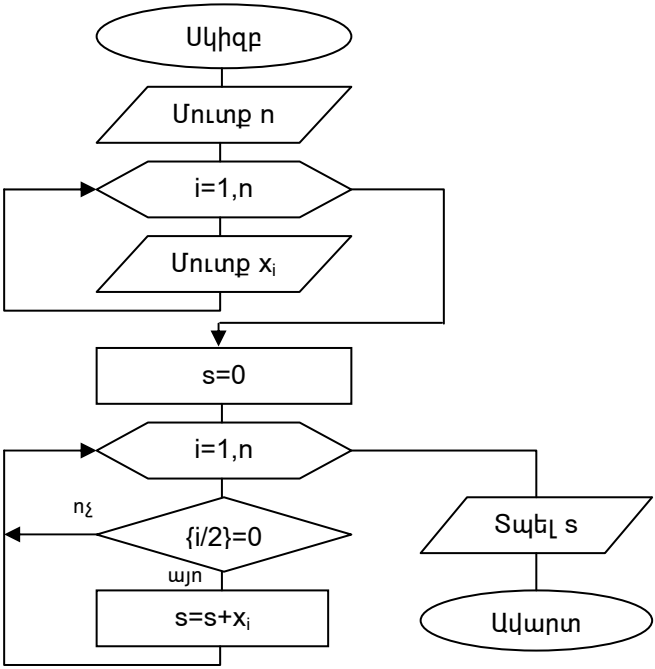
Բերված փոփոխականները այն երկչափ զանգվածի տարրերն են, որն ունի 1-ից 6 սահմանները առաջին ինդեքսով (չափումով) և 1-ից 8՝ երկրորդ չափումով: Երկչափ զանգվածը փաստորեն ներկայացնում է ուղղանկյուն աղյուսակ (մատրից), իսկ A[I,J] փոփոխականը՝ այն տարրը, որը գտնվում է I-րդ տողի և J-րդ սյան հատման դիրքում:

Զանգվածի չափողականությունը և տարրերի թիվը ՊԱՍԿԱԼ-ում չեն սահմանափակվում: Իրականում դրանք սահմանափակվում են միայն քոմփյութերի հիշողության հնարավորություններով:

4.2. Պարզ ծրագրերի տիպային օրինակներ

Օրինակ 4.1. Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն տրված n տարրեր պարունակող միաչափ զանգվածի զույգ ինդեքս ունեցող տարրերի գումարը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.1-ում, իսկ ծրագիրն ունի EXAM4_1 անունը:



Նկ. 4.1

```
    PROGRAM EXAM4_1;
    CONST N=10;
   VAR X:ARRAY[1..N]OF REAL;
        I:INTEGER; S:REAL;
BEGIN
    FOR I:=1 TO N DO READ(X[I]);
    S:=0;
    FOR I:=1 TO N DO
    IF I MOD 2=0 THEN S:=S+X[I];
    WRITE(S)
 END.
```

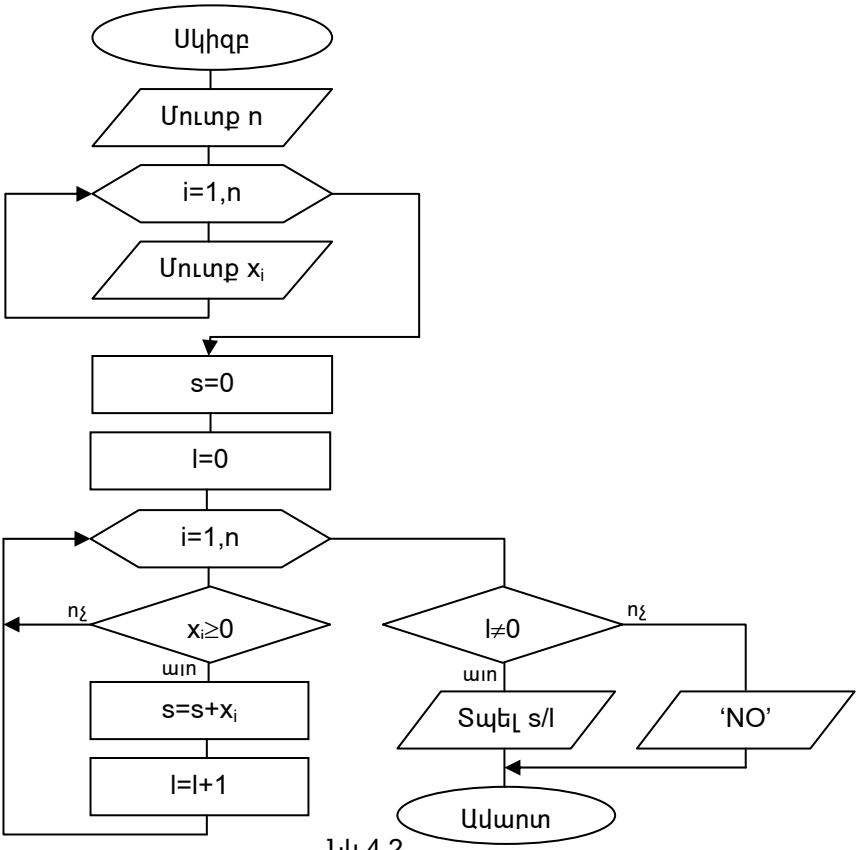
Տարրն ունի զույգ ինդեքս, եթե տարրի ինդեքսի արժեքը 2-ի բաժանելուց ստացված մնացորդը հավասար է 0-ի: Կազմակերպվել են երկու ցիկլեր, որոնցից առաջինում ներմուծվել են վեկտորի տարրերի արժեքները, իսկ երկրորդում հաշվվել է զույգ ինդեքս ունեցող տարրերի գումարը (s):

Ծրագրում զանգվածի x_i տարրի զույգության ստուգումը համարժեք է X[I] MOD 2=0 պայմանի կատարմանը:

Խնդիրը կարելի է լուծել նաև մեկ ցիկլի օգնությամբ:

Օրինակ 4.2. Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն տրված n տարրեր պարունակող միաչափ զանգվածի ոչ բացասական տարրերի միջին թվաբանականը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.2-ում, իսկ ծրագիրն ունի EXAM4_2 անունը:



Նկ.4.2

```
PROGRAM EXAM4_2;
CONST N=10;
VAR X:ARRAY[1..N]OF REAL;
    I,L:INTEGER; S:REAL;
BEGIN
  FOR I:=1 TO N DO READ(X[I]);
  S:=0;L:=0;
  FOR I:=1 TO N DO
  IF X[I]>=0 THEN BEGIN S:=S+X[I]; L:=L+1 END;
  IF L<>0 THEN WRITE(S/L) ELSE WRITE('NO')
END.
```

Ծրագրում կազմակերպված են երկու ցիկլեր, որոնցից առաջինում ներմուծվել են վեկտորի տարրերի արժեքները, իսկ երկրորդում հաշվվել են վեկտորի ոչ բացասական տարրերի գումարը և քանակը, որոնց քանորդը կտա դրական տարրերի միջին թվաբանականը: Եթե զանգվածում չկան բացասական տարրեր, ապա L-ի և S-ի արժեքները չեն փոփոխվի և կմնան հավասար զրոյի, և գումարի բաժանումը զրոյի կբերի սխալի: Այդ դեպքում տպվում է 'NO' հաղորդագրությունը:

Խնդիրը կարելի է լուծել նաև մեկ ցիկլի օգնությամբ: Այդ դեպքում նույն ցիկլի մեջ պետք է ներմուծել զանգվածի տարրերի արժեքները և հաշվել զանգվածի ոչ բացասական տարրերի գումարը և քանակը: Ցիկլից դուրս հաշվել միջին թվաբանականը:

Օրինակ 4.3. Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն n հատ ամբողջ տիպի տարրեր պարունակող միաչափ զանգվածի կենտ արժեք ունեցող տարրերի միջին քառակուսայինը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.3-ում, իսկ ծրագիրն ունի EXAM4_3 անունը:

$x_1, x_2, \dots, x_n$ տարրերի միջին քառակուսայինը հաշվվում է

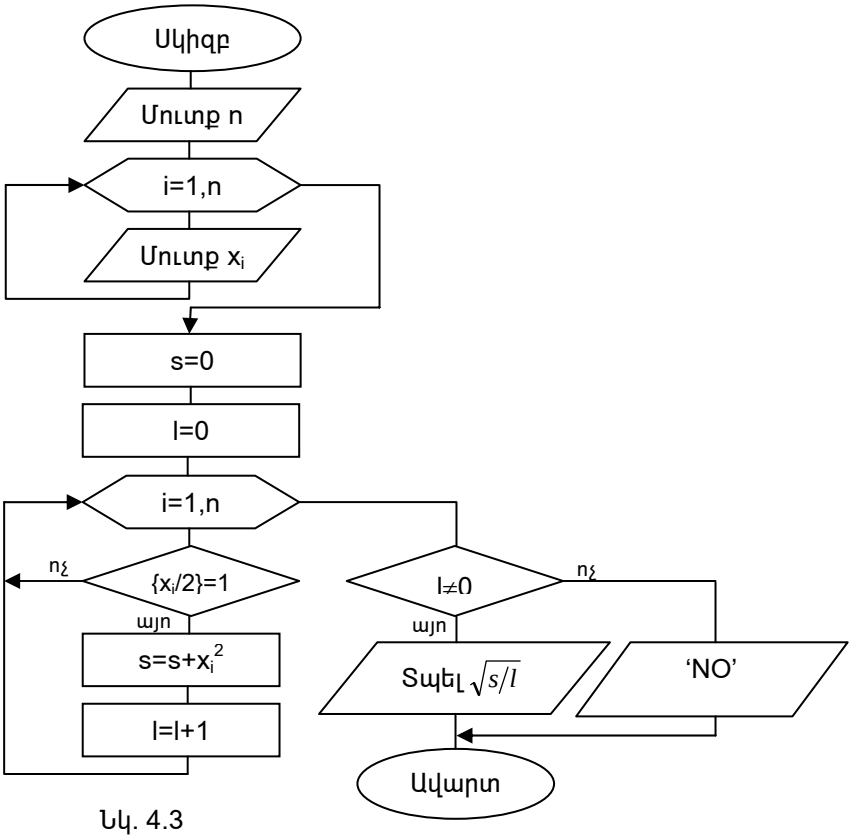
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$ արտահայտությամբ:

Կազմակերպվել են երկու ցիկլեր, որոնցից առաջինում ներմուծվել է զանգվածի տարրերի արժեքները, իսկ երկրորդում հաշվվել է զանգվածի կենտ արժեք ունեցող տարրերի քառակուսիների գումարը և քանակը: Եթե տարրի արժեքը կենտ է, այսինքն՝ 2-ի բաժանելիս տալիս է 1 մնացորդ, ապա գումարի հաշվման համար նախատեսված s փոփոխականին

գումարվում է հերթական տարրի քառակուսու արժեքը, իսկ քանակի հաշվման համար նախատեսված l փոփոխականի արժեքը մեծացվում է 1-ով: Նախապես ցիկլից դուրս s և l փոփոխականներին վերագրվել է 0 արժեք:

Ցիկլից դուրս դրանց հարաբերության արժեքից հանելով քառակուսի արմատ կստանանք վեկտորի տարրերի միջին քառակուսայինը: Եթե զանգվածում չկան կենտ արժեք ունեցող տարրեր (l=0), ապա արտածվում է 'NO' հաղորդագրությունը:

Խնդիրը կարելի է լուծել նաև մեկ ցիկլի օգնությամբ:



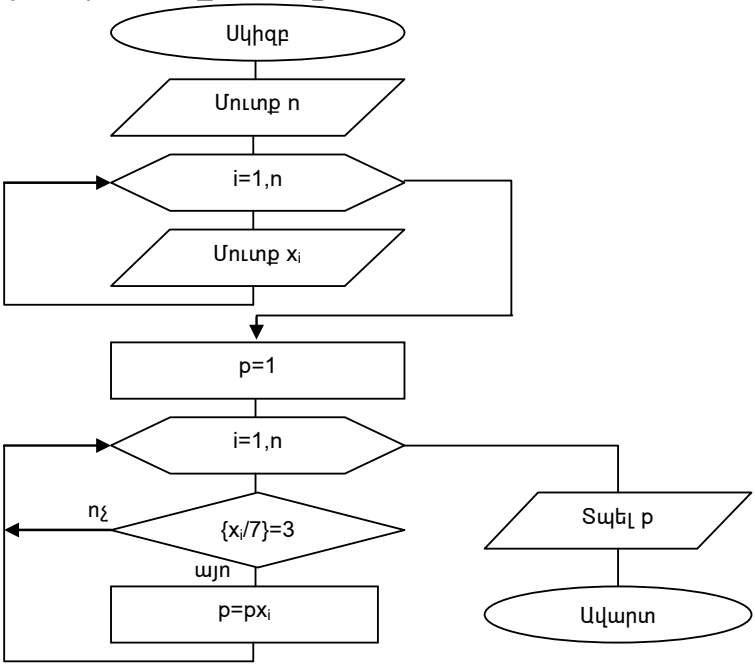
Նկ. 4.3

```
PROGRAM EXAM4_3;
CONST N=10;
VAR X:ARRAY[1..N]OF INTEGER;
```

```
        I,L:INTEGER; S:REAL;
BEGIN
   FOR I:=1 TO N DO READ(X[I]);
   S:=0;L:=0;
   FOR I:=1 TO N DO
   IF X[I] MOD 2=1 THEN BEGIN S:=S+SQR(X[I]); L:=L+1 END;
   IF L<>0 THEN WRITE(SQRT(S/I)) ELSE WRITE('NO')
END.
```

Օրինակ 4.4. Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն n հատ ամբողջ տիպի տարրեր պարունակող միաչափ զանգվածի այն տարրերի արտադրյալը, որոնք 7-ի բաժանելիս կմնա 3 մնացորդ:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.4-ում, իսկ ծրագիրն ունի EXAM4_4 անունը:



Նկ. 4.4

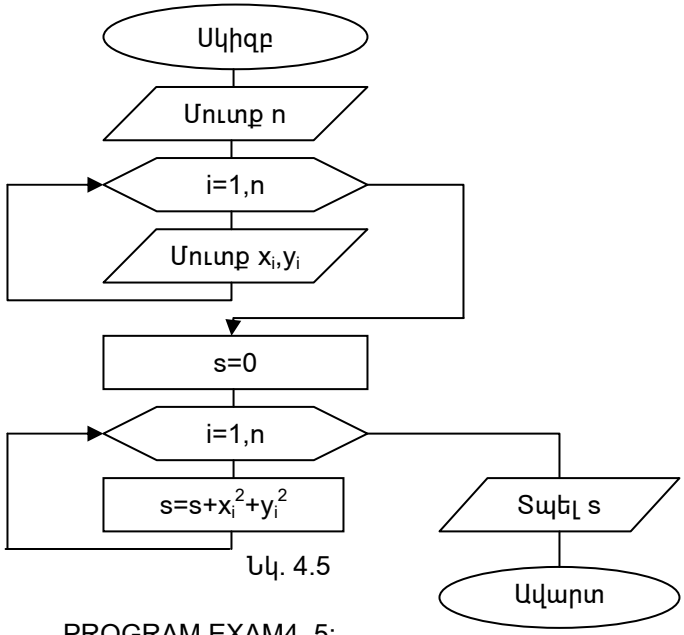
```
PROGRAM EXAM4_4;
CONST N=10;
VAR X:ARRAY[1..N]OF INTEGER;
```

```
        I:INTEGER; P:REAL;
BEGIN
    FOR I:=1 TO N DO READ(X[I]);
    P:=1; FOR I:=1 TO N DO
    IF X[I] MOD 7=3 THEN P:=P*X[I]; WRITE(P);
END.
```

Ձանգվածի տարրերի ցիկլային ներմուծումից հետո կազմակերպվել է նոր ցիկլ, ուր հաշվվում է այն տարրերի արտադրյալը, որոնք 7-ի բաժանելիս կմնա 3 մնացորդ:

Օրինակ 4.5. Տրված են n ամբողջ թիվը և n տարրեր պարունակող X և Y միաչափ զանգվածները: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն տրված զանգվածների տարրերի քառակուսայինների ընդհանուր գումարը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.5-ում, իսկ ծրագիրն ունի EXAM4_5 անունը:



Նկ. 4.5

```
PROGRAM EXAM4_5;
CONST N=10;
VAR X,Y:ARRAY[1..N] OF REAL;
```

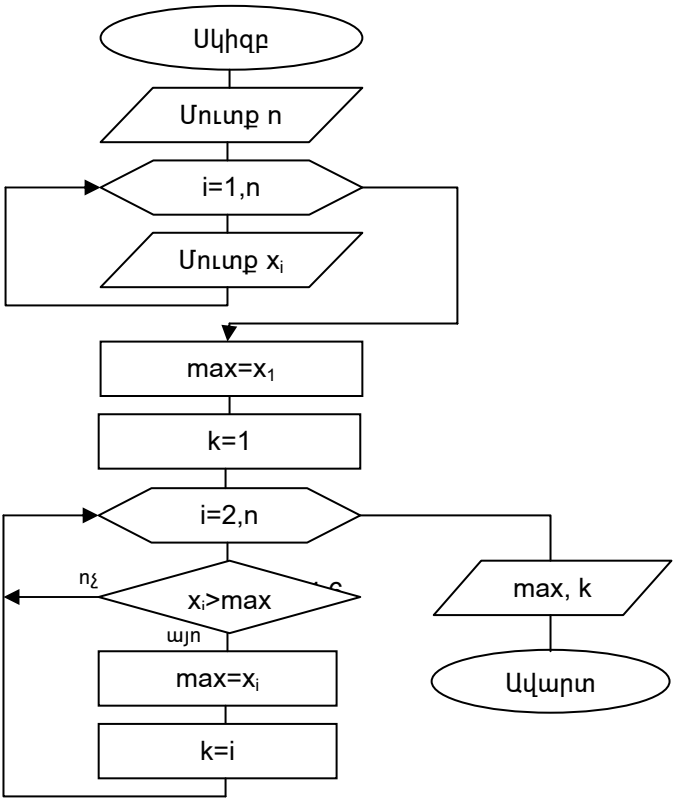
```
        I:INTEGER; S:REAL;
BEGIN
   FOR I:=1 TO N DO READ(X[I],Y[i]);
   S:=0;
   FOR I:=1 TO N DO S:=S+SQR(X[I])+SQR(Y[I]);
   WRITE(S);
END.
```

Կազմակերպվել են երկու ցիկլեր, որոնցից առաջինում ներմուծվել են x և y վեկտորների տարրերի արժեքները, իսկ երկրորդում հաշվվել է դրանց քառակուսիների ընդհանուր գումարը (s):

Օրինակ 4.6. Տրված են n բնական թիվը և n տարր պարունակող X միաչափ զանգվածը: Կազմել բլոկ-սխեմա ու ծրագիր, որոնք կհաշվեն և կտպեն զանգվածի առաջին մեծագույն տարրի արժեքը և համարը:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.6-ում, իսկ ծրագիրն ունի EXAM4_6 անունը:

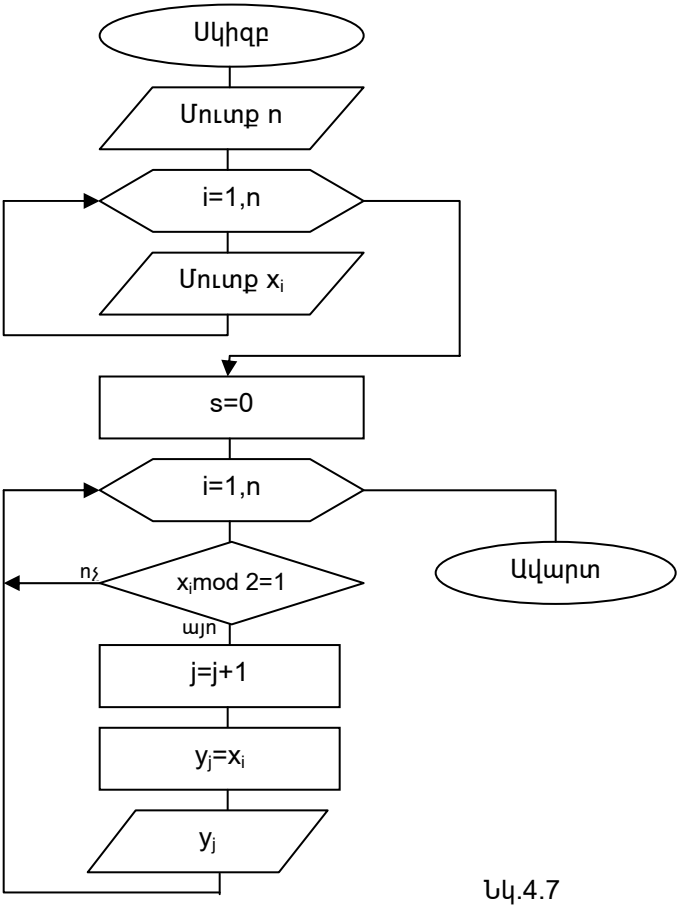
Մեծագույն տարրի համարը գտնելու համար max փոփոխականին ցիկլից դուրս վերագրվել է զանգվածի առաջին տարրի արժեքը, իսկ մեծագույն տարրի համարը հիշվում է k փոփոխականի մեջ: Ցիկլի մարմնում max-ը համեմատվում է զանգվածի մնացած բոլոր տարրերի արժեքների հետ: Եթե զանգվածի որևէ տարր մեծ է max-ից, ապա max-ին վերագրվում է այդ տարրի արժեքը, իսկ k-ին՝ նրա հերթական համարը: Քանի որ զանգվածում հնարավոր է լինեն մեկից ավելի մեծագույն, այսինքն՝ մեծագույն տարրին հավասար տարրեր, ապա առաջին մեծագույն տարրը գտնելու համար հերթական տարրը պետք է խիստ մեծ լինի max փոփոխականից, իսկ վերջին մեծագույն տարրը գտնելիս՝ մեծ կամ հավասար: Այսպիսով, ցիկլի ավարտից հետո max-ում պահված կլինի առաջին մեծագույն տարրը, իսկ k-ում նրա համարը:




```
    PROGRAM EXAM4_6;
    CONST N=10;
   VAR X:ARRAY[1..N] OF REAL;
        I,k:INTEGER; MAX:REAL;
BEGIN
    FOR I:=1 TO N DO READ(X[I]);
    MAX:=X[1]; K:=1;
    FOR I:=2 TO N DO
    IF X[I]>MAX THEN BEGIN MAX:=X[I]; K:=I END;
    WRITE(MAX,K)
 END.
```

Օրինակ 4.7. Տրված են n բնական թիվը և n տարր պարունակող X միաչափ զանգվածը: Կազմել բլոկ-սխեմա ու ծրագիր, որոնք տրված զանգվածի կենտ արժեք ունեցող տարրերից կստանան նոր զանգված: Ենթադրվում է, որ զանգվածում կան այդպիսի տարրեր:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.4.7-ում, իսկ ծրագիրն ունի EXAM4_7 անունը:



Նկ.4.7

```
    PROGRAM EXAM4_7;
    CONST N=10;
   VAR X,Y:ARRAY[1..N]OF REAL;
        I,J:INTEGER;
BEGIN
    FOR I:=1 TO N DO READ(X[I]);
    J:=0;
```

```
    FOR I:=1 TO N DO
    IF X[I] MOD 2=1 THEN
    BEGIN J:=J+1; Y[J]:=X[I]; WRITE(Y[J]) END
END.
```

Ստացվող y զանգվածը կարող է ունենալ մինչև n տարրեր: Նոր զանգվածի տարրերի ինդեքսները պահելու համար ծրագրում օգտագործվել է j փոփոխականը, որը ցիկլից դուրս ընդունում է 0 արժեք: Կազմակերպվել է ցիկլ, ուր ստացվում և արտածվում են y զանգվածի տարրերը: Եթե x զանգվածի հերթական տարրն ունի կենտ արժեք, ապա j փոփոխականը մեծանում է մեկով, և y զանգվածի j-րդ տարրին վերագրվում է x զանգվածի հերթական տարրի արժեքը:

4.3. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	1, 17, 21, 43, 54, 63, 73
2	2, 14, 24, 44, 51, 65, 71
3	3, 18, 27, 46, 56, 66, 74
4	4, 20, 25, 41, 58, 61, 80
5	5, 11, 28, 48, 57, 67, 72
6	6, 19, 23, 49, 55, 68, 79
7	7, 16, 30, 47, 59, 69, 75
8	8, 12, 29, 42, 53, 70, 78
9	9, 15, 26, 45, 60, 64, 77
10	10, 13, 22, 50, 52, 62, 76
11	10, 14, 22, 42, 51, 61, 73
12	9, 13, 25, 44, 52, 62, 74
13	8, 12, 23, 46, 53, 63, 76
14	7, 11, 26, 48, 54, 64, 77
15	6, 15, 30, 50, 55, 65, 71
16	5, 16, 29, 49, 56, 66, 79
17	4, 19, 21, 45, 57, 67, 80
18	3, 20, 24, 47, 58, 68, 72
19	2, 18, 27, 41, 59, 69, 78
20	1, 17, 28, 43, 60, 70, 75
21	1, 14, 23, 34, 59, 61, 72
22	2, 16, 24, 32, 60, 63, 73
23	3, 15, 27, 36, 58, 64, 75
24	4, 17, 28, 37, 57, 65, 78
25	5, 11, 21, 39, 56, 68, 76
26	6, 18, 29, 40, 55, 69, 71
27	7, 19, 30, 33, 54, 70, 77
28	8, 20, 26, 38, 53, 67, 79
29	9, 13, 22, 35, 52, 66, 74
30	10, 12, 25, 31, 51, 62, 80

5. ԵՆԹԱԾՐԱԳՐԵՐ

5.1. Հիմնական հասկացություններ

Որևէ խնդրի լուծման ալգորիթմը մշակելիս լինում են իրավիճակներ, երբ մուտքային արժեքների տարբեր հավաքածուների համար օպերատորների որոշակի բլոկ կրկնվում է մի քանի անգամ: Որպեսզի այդպիսի բլոկը ծրագրում մի քանի անգամ չգրվի, այն կարելի է մեկ անգամ ձևակերպել և, անհրաժեշտության դեպքում, նրան դիմել աշխատանքից հետո ապահովելով վերադարձ դիմումի կետին: Նման անցումների կազմակերպման համար օգտագործվում են **ենթածրագիր** կառուցվածքները:

Բարդ և մեծածավալ ծրագրերը միանգամից ամբողջությամբ կազմելը, ստուգելը և կարգաբերելը դժվար է: Այդպիսի ծրագրերի բաժանումը առանձին ինքնուրույն ծրագրային բլոկների՝ ենթածրագրերի հնարավորություն է տալիս հեշտությամբ կազմել և կարգաբերել առանձին բլոկներ, որից հետո դրանք միավորել մեկ ընդհանուր ծրագրում:

Մեկ անգամ արդեն կազմված ենթածրագրերը կարող են հաջողությամբ կիրառվել տարբեր խնդիրների ծրագրեր կազմելիս՝ որպես դրանց բաղադրիչ մասեր:

Յուրաքանչյուր ենթածրագիր պետք է նախօրոք նկարագրված (ներկայացված) լինի փոփոխականների հայտարարումից անմիջապես հետո՝ մինչև հիմնական ծրագրի մարմնի սկիզբը:

Կիրառվում են երկու տիպի ենթածրագրեր՝
- ենթածրագիր - պրոցեդուրա (PROCEDURE),
- ենթածրագիր - ֆունկցիա (FUNCTION):

ՊԱՍԿԱԼ - ծրագրի կառուցվածքային սխեման ընդհանուր դեպքում կարելի է ներկայացնել հետևյալ կերպ.

```
PROGRAM <անուն>;

LABEL . . . . . ;
CONST . . . . . ;
TYPE . . . . . ;
VAR . . . . . ;
```

ԵՆԹԱԾՐԱԳԻՐ 1;
ԵՆԹԱԾՐԱԳԻՐ 2;
. . . . . .
ԵՆԹԱԾՐԱԳԻՐ N;

BEGIN
. . . . .
. . . . .
. . . . .
END.

Ինչպես հիմնական ծրագիրը, այնպես էլ ֆունկցիաները և պրոցեդուրաները, ձևակերպվում են որոշակի կանոններով, որոնք դիտարկվում են ստորև:

Պրոցեդուրան ենթածրագիր է, որն իր աշխատանքի արդյունքում կարող է ստանալ և փոխանցել արժեքների հավաքածու: Այն ունի հետևյալ կառուցվածքը.

<պրոցեդուրայի վերնագիր>;
<պրոցեդուրայի նկարագրությունների բաժին>;
<պրոցեդուրայի մարմին>:
Պրոցեդուրայի վերնագիրն ունի հետևյալ տեսքը..

PROCEDURE N(p_1:t_1; p_2:t_2; ... ; p_m:t_m; VAR p_{m+1}:t_{m+1}; ... ; p_n:t_n);

որտեղ N- պրոցեդուրայի անունն է (ցանկացած թույլատրելի իդենտիֆիկատոր),

p_1, p_2, ... , p_n - ֆորմալ պարամետրեր են,

t_1, t_2, ... , t_n – համապատասխանաբար ֆորմալ պարամետրերի տիպերը:

Պրոցեդուրայի վերնագրում հայտարարված բոլոր պարամետրերը ֆորմալ (ձևական) բնույթ են կրում: Պրոցեդուրայի կատարման ժամանակ նրանցից յուրաքանչյուրը փոխարինվում է պրոցեդուրայի կանչման օպերատորում նշված որոշակի (փաստացի) պարամետրով: Պրոցեդուրայի վերնագրում օգտագործված պարամետրերը կոչվում են

ձևական պարամետրեր: Ֆորմալ պարամետրերի ցուցակում հայտարարվում են պրոցեդուրային փոխանցվող (մուտքային) և պրոցեդուրայից փոխանցվող (ելքային) պարամետրերը:

Պրոցեդուրայի վերնագրի ընդհանուր տեսքից երևում է, որ ֆորմալ պարամետրերի մի մասը (p_1, p_2, . . . , p_m) գրված է VAR բաժնից դուրս, իսկ մյուս մասը (p_{m+1}, . . . , p_n)՝ VAR բաժնում: VAR բաժնից դուրս գրված պարամետրերը մուտքային պարամետրերն են, իսկ VAR բաժնում գրված պարամետրերը՝ ելքային:

Բերենք պրոցեդուրայի վերնագրի օրինակ՝

```
PROCEDURE  SA(A:REAL; B:REAL; C:INTEGER; VAR
                    X:REAL; VAR Y:REAL);
```

Եթե ենթածրագրի ֆորմալ պարամետրերի ցուցակում կան նույն տիպի մի քանի պարամետրեր, ապա դրանք կարելի է միավորել նույն խմբի մեջ՝ իրարից անջատելով ստորակետերով, իսկ հետո նշելով ընդհանուր տիպը: Օրինակ.

```
PROCEDURE SA(A,B:REAL; C:INTEGER; VAR  X,Y:REAL);
```

Այս օրինակում A, B, X և Y պարամետրերը իրական տիպի են, իսկ C պարամետրը՝ ամբողջ տիպի: A, B, C պարամետրերը մուտքային են, իսկ X և Y պարամետրերը՝ ելքային:

Պրոցեդուրայի նկարագրությունների բաժինը կարող է պարունակել LABEL, CONST, TYPE, VAR բաժինները:

Պրոցեդուրայի մարմինը բանալի-բառերի մեջ գրված օպերատորների հաջորդականությունն է, որը տեղադրված է պրոցեդուրայի նկարագրությունների բաժնից հետո:

Օրինակ 5.1: Տրված կողմերով եռանկյան մակերեսի հաշվումը ձևակերպել պրոցեդուրայով:

```
1       PROCEDURE MAC(X,Y,Z:REAL;VAR SS:REAL);
2       VAR P:REAL;
3       BEGIN
4       P:=(X+Y+Z)/2;
```

```
5       SS:=SQRT(P*(P-X)*(P-Y)*(P-Z))
6       END;
```

Տող 1-ում գրված է պրոցեդուրայի վերնագիրը՝ MAC անվամբ: Պրոցեդուրան ունի չորս իրական տիպի ֆորմալ պարամետրեր, որոնցից երեքը (X, Y, Z) մուտքային պարամետրեր են (եռանկյան կողմերի մեծությունները), իսկ մեկը (SS) ելքային (եռանկյան մակերեսի արժեքը): Վերնագրին հաջորդում է պրոցեդուրայի նկարագրությունների բաժինը, որը պարունակում է միայն փոփոխականների նկարագրման VAR բաժինը: Բաժնում նկարագրված է եռանկյան կիսապարագծի համար նախատեսված իրական տիպի P փոփոխականը (տող 2): Տողեր 3-6-ը կազմում են պրոցեդուրայի մարմինը, որտեղ հաշվվել են X, Y և Z կողմերով եռանկյան P կիսապարագիծը (տող 4) և SS մակերեսը (տող 5):

Պրոցեդուրայի կանչի համար անհրաժեշտ է գրել կանչի օպերատոր, որը բաղկացած է պրոցեդուրայի անունից և փակագծերում գրված **փաստացի պարամետրերից**:

Օրինակ.

```
ASA(M,N,E,F,K,V);
```

Փաստացի և ֆորմալ պարամետրերը պետք է բավարարեն հետևյալ պահանջներին.

- լինեն նույն քանակությամբ,
- տեղերով իրար համապատասխանող պարամետրերը լինեն նույն տիպի:

Ֆունկցիայի կատարման արդյունքում կարող է ստացվել մեկ արժեքից ոչ ավելի: Ստացված արժեքը ֆունկցիան վերադարձնում է իր անվան միջոցով: Դա ֆունկցիայի և պրոցեդուրայի միջև եղած ամենաէական տարբերությունն է: Հաջորդ տարբերությունը վերնագրի մեջ է: Ֆունկցիայի վերնագիրը բաղկացած է FUNCTION հատուկ բառից, որին հաջորդում են ֆունկցիայի անունը և փակագծերի մեջ նկարագրված ֆորմալ պարամետրերը: Վերնագիրն ավարտվում է ֆունկցիայի անվան նկարագրությամբ:

Ֆունկցիայի վերնագիրն ունի հետևյալ տեսքը..

FUNCTION N(p_1:t_1; p_2:t_2; ... ; p_n:t_n):t;

որտեղ N- պրոցեդուրայի անունն է (ցանկացած թույլատրելի իդենտիֆիկատոր),

p_1, p_2, ... , p_n - ֆորմալ պարամետրեր են,

t_1, t_2, ... , t_n – համապատասխանաբար ֆորմալ պարամետրերի տիպերը,

t - ֆունկցիայի հաշված արժեքի տիպը:

Օրինակ.

FUNCTION SUM(X,Y,Z:REAL):REAL:

Այստեղ SUM-ը ֆունկցիայի անունն է, որը նկարագրված է որպես իրական տիպի փոփոխական: X, Y և Z փոփոխականները իրական տիպի ֆորմալ պարամետրեր են:

Ինչպես և պրոցեդուրան, ֆունկցիայի նկարագրությունների բաժինը ևս կարող է պարունակել LABEL, CONST, TYPE, VAR բաժինները:

Ֆունկցիայի նկարագրությունների բաժնին հաջորդում է ֆունկցիայի մարմինը: Քանի որ ֆունկցիան ստացված արժեքը վերադարձնում է իր անվան միջոցով, ֆունկցիայում հաշվարկված վերջնական արդյունքը անհրաժեշտ է վերագրել ֆունկցիայի անվանը: Հետևաբար ֆունկցիայի անունը պետք է ունենա նույն տիպը, ինչ վերջնական արդյունքը:

Օրինակ 5.2: Կազմել ֆունկցիա, որը կհաշվի տրված X, Y և Z կողմերով եռանկյան մակերեսը:

```
1       FUNCTION MAC(X,Y,Z:REAL):REAL;
2       VAR P:REAL;
3       BEGIN
4       P:=(X+Y+Z)/2;
5       MAC:=SQRT(P*(P-X)*(P-Y)*(P-Z))
6       END;
```

Տող 1-ում գրվել է ֆունկցիայի վերնագիրը: Ֆունկցիան ունի MAC անունը, որը իրական տիպի փոփոխական է: Ֆունկցիայում նկարագրված են երեք իրական տիպի ֆորմալ պարամետրեր (X, Y, Z); Վերնագրին հաջորդում է ֆունկցիայի նկարագրությունների բաժինը, որը պարունակում է միայն փոփոխականների նկարագրման VAR բաժինը: Այստեղ նկարագրված է եռանկյան կիսապարագծի համար նախատեսված իրական տիպի P փոփոխականը (տող 2): Տողեր 3-6-ը կազմում են ֆունկցիայի մարմինը, որտեղ հաշվվել են X, Y և Z կողմերով եռանկյան P կիսապարագիծը (տող 4) և մակերեսը (տող 5): Հաշվված մակերեսի արժեքը վերագրվել է ֆունկցիայի MAC անվանը:

Ֆունկցիայի կանչի համար անհրաժեշտ է ցանկացած արտահայտությունում նշել ֆունկցիայի անունը՝ իր փաստացի պարամետրերով:

Օրինակ.

```
Y:=MAC(4,4,4);
```

Այստեղ Y փոփոխականին կվերագրվի 4 երկարությամբ հավասարակողմ եռանկյան մակերեսը, որի արժեքը կհաշվի MAC անունով ֆունկցիան:

Ենթածրագրերի կազմակերպման համար օգտագործվում են երկու տիպի պարամետրեր. պարամետր-արժեքներ և պարամետր-փոփոխականներ:

Պարամետր-արժեքները այն պարամետրերն են, որոնք փոխանցվում են ենթածրագրին՝որպես մուտքային մեծություններ: Ենթածրագրի կանչման մասից ենթածրագրին են փոխանցվում պարամետր-արժեքների պատճենները (սրանց հետ է փաստորեն աշխատում ենթածրագիրը), հետևաբար այս կերպ փոխանցված փաստացի պարամետրի արժեքը ենթածրագրի կողմից փոխվել չի կարող: Ֆունկցիայում հայտարարված բոլոր ֆորմալ պարամետրերը պարամետր-արժեքներ են: Պրոցեդուրայում պարամետր-արժեքները այն պարամետրերն են, որոնք պրոցեդուրայի վերնագրում գրված են VAR բաժնից դուրս:

Պարամետր փոփոխականները սահմանվում են միայն պրոցեդուրաների համար: **Պարամետր-փոփոխականները** այն պարամետրերն են, որոնց միջոցով պրոցեդուրայից դուրս արժեք է փոխանցվում: Այս դեպքում հնարավոր է, որ փաստացի պարամետրերը պրոցեդուրայում ենթարկվեն փոփոխության: Պրոցեդուրայի կանչման

մասից նրանց են փոխանցվում պարամետր-փոփոխականների հասցեները, հետևաբար ենթածրագիրը հնարավորություն ունի փոփոխելու այդ հասցեների պարունակությունը:

Վերը բերված MACERES1 և MACERES2 ծրագրերում X, Y և Z պարամետրերը պարամետր-արժեքներ են, իսկ MACERES21 ծրագրի MAC անունով պրոցեդուրայում գրված SS պարամետրը՝ պարամետր-փոփոխական:

VAR բառի ազդեցության տիրույթը տարածվում է մինչև մոտակա կետ ստորակոտը (;), այսինքն՝ միայն մեկ խմբի սահմաններում: Օրինակ.

```
PROCEDURE ASA( VAR A,B:REAL; C,D:INTEGER);
```

Այստեղ A և B պարամետրերը պարամետր-փոփոխականներ են, իսկ C և D պարամետրերը՝ պարամետր-արժեքներ: Եթե անհրաժեշտ է տարբեր տիպեր ունեցող պարամետրերը հայտարարել որպես պարամետր փոփոխական, ապա ամեն մի խմբի նկարագրությունից առաջ անհրաժեշտ է գրել VAR բանալի բառը: Օրինակ.

```
PROCEDURE ASA( VAR A,B:REAL;VAR C,D:INTEGER);
```

Այս դեպքում բոլոր պարամետրերը պարամետր-փոփոխականներ են:

Ֆորմալ պարամետրերի ցուցակում հայտարարված կամայական պարամետրի տիպը կարող է լինել միայն ստանդարտ կամ նախապես հայտարարված տիպ:

Օրինակ.

```
PROCEDURE AS(X:ARRAY[1..10] OF REAL);
```

պրոցեդուրայի վերնագիրը գրված է սխալ, քանի որ ֆորմալ պարամետրերի ցուցակում X պարամետրը հայտարարվել է որպես զանգված, որը ստանդարտ տիպ չէ: Այդպիսի դեպքերում անհրաժեշտ է նախապես նկարագրել ոչ ստանդարտ տիպը, ինչպես օրինակ,

```
TYPE
    VEC=ARRAY[1..10]OF REAL;
PROCEDURE AS(X:VEC);
```

. . . . . .

Այն փոփոխականները, որոնց նկարագրությունը կատարված է ենթածրագրերի նկարագրությունների բաժնում, կարող են կիրառվել (գործել) միայն այդ ենթածրագրերի ներսում: Այսպիսի փոփոխականները կոչվում են **լոկալ** (տեղային նշանակության) **փոփոխականներ**:

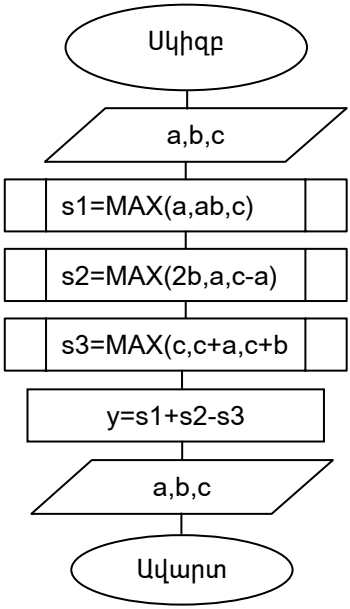
Գլոբալ (ընդհանուր նշանակության) **փոփոխականների** նկարագրությունը կատարվում է հիմնական ծրագրի փոփոխականների նկարագրությունների բաժնում: Այդ փոփոխականները կարող են օգտագործվել ինչպես հիմնական ծրագրում, այնպես էլ ենթածրագրում:

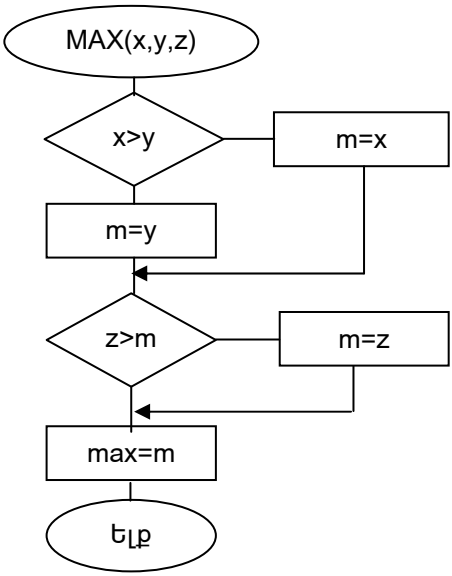
Հնարավոր է, որ գլոբալ և լոկալ փոփոխականները նշանակված լինեն միևնույն իդենտիֆիկատորներով: Այսպես, ենթադրենք, որ վերը բերված MACERES1 ծրագրի 2-րդ տողում S փոփոխականի փոխարեն կիրառված է P:REAL գլոբալ փոփոխականը: Այդ դեպքում, պրոցեդուրայի ներսում գործում է իր մեջ հայտարարված P լոկալ փոփոխականը: Այսպիսով, յուրաքանչյուր իդենտիֆիկատոր լոկալացվում է այն ծրագրային միավորի սահմաններում, որում այն հայտարարված է: Դա հնարավորություն է տալիս առանձին ծրագրավորողներին ինքնուրույն կազմել իրենց պրոցեդուրաները և վերջում դրանք միավորել մեկ ծրագրի մեջ՝ առանց մտահոգվելու իդենտիֆիկատորների համընկման մասին:

5.2. Ենթածրագրերի կիրառման օրինակներ

Օրինակ 5.3: Տրված են a, b և c փոփոխականների արժեքները: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կհաշվեն և կտպեն y=max(a,ab,c)+max(2b,a,c-a)-max(c,c+a,c+b) արտահայտության արժեքը: Մեծագույն արժեքների հաշվումը կազմակերպել ֆունկցիայի միջոցով:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.5.1-ում, իսկ ծրագիրն ունի EXAM5_1 անունը:





Նկ.5.1

```
1       PROGRAM EXAM5_1;
2       VAR A,B,S1,S2,Y:REAL;
3               FUNCTION MAX(X,Y,Z:REAL):REAL;
4               VAR M:REAL;
5               BEGIN
6               IF X>Y THEN M:=X ELSE M:=Y;
7                IF Z>M THEN M:=Z;
8               MAX:=M
9               END;
10      BEGIN
11      READ(A,B,C);
12      S1:=MAX(A,A*B,C);
13      S2:=MAX(2*B,A,C-A);
14      S3:=MAX(C,C+A,C+B);
15      Y:=S1+S2-S3;
16      WRITE('Y=',Y:6:2)
17      END.
```

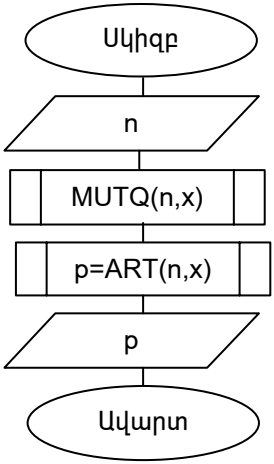
ՄՈՒՏՔ՝ A=3 B=5 C=2 ԵԼՔ՝ Y=18

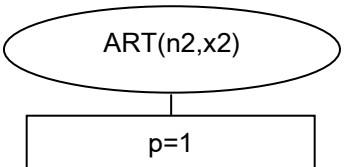
Ծրագրում կազմակերպել է MAX անունով իրական տիպի ֆունկցիա (տողեր 3-9), որն ունի x, y, z իրական տիպի 3 ֆորմալ

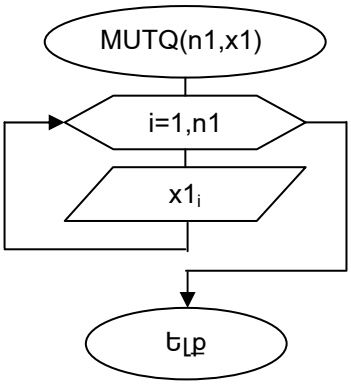
պարամետրեր: Ֆունկցիան x, y, z պարամետրերից գտնում է մեծը (M, տողեր 6-7) և քանի որ ֆունկցիան հիմնական ծրագրին իր միակ ելքային պարամետրը հիմնական ծրագրին է փոխանցում իր անվան միջոցով, ապա տող 8-ում երեք թվերից մեծագույնի արժեքը (M) վերագրվում է ֆունկցիայի MAX անվանը: Ֆունկցիայի կանչման համար տողեր 12-14-ում գրվել է ֆունկցիայի անունը, փակագծերում նշելով փաստացի պարամետրերը՝ (A,A*B,C և 2*B,A,C-A և C,C+A,C+B):

Օրինակ 5.4: Տրված են n բնական թիվը և n հատ իրական տարրեր պարունակող X վեկտորը: Կազմել բլոկ–սխեմա և ծրագիր, որոնք կհաշվեն և կարտածեն տրված վեկտորի դրական տարրերի արտադրյալը: Տարրերի արտադրյալի հաշվումը կազմակերպել ֆունկցիայի միջոցով, իսկ տրված վեկտորի տարրերի ներմուծումը՝ պրոցեդուրայի:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.5.2-ում, իսկ ծրագիրն ունի EXAM5_2 անունը:







Նկ.5.2

```
 1      PROGRAM EXAM5_2;
 2      TYPE VEC=ARRAY[1..10] OF REAL;
 3      VAR X:VEC;N:INTEGER;P:REAL;
 4        PROCEDURE MUTQ(N1:INTEGER;VAR X1:VEC);
 5        VAR I:INTEGER;
 6        BEGIN
 7        FOR I:=1 TO N1 DO READ(X1[I])
 8        END;
 9          FUNCTION ART(N2:INTEGER;X2:VEC):REAL;
10          VAR P:REAL; I:INTEGER;
11          BEGIN
12          P:=1;
13          FOR I:=1 TO N2 DO
14          IF X2[I]>0 THEN P:=P*X2[I];
15          ART:=P
16          END;
17     BEGIN
18     READ(N);
19     MUTQ(N,X);
20     P:=ART(N,X);
21     WRITE(P)
```

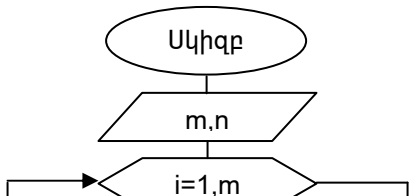
```
22      END.
```
ՄՈՒՏՔ՝ n=10 ԵԼՔ՝ 180
-2, 5, 2, -6, 0, -7, 3, 2, -8, 3

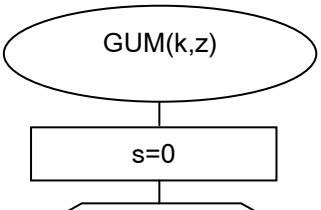
Հիմանական ծրագրի սկզբում ներմուծվում են n բնական թիվը (տող 18): Տող 19-ում դիմում է արվել MUTQ անունով պրոցեդուրային՝ նրան փոխանցելով X վեկտորի տարրերի քանակը: Պրոցեդուրայում ներմուծվում են X1 վեկտորի տարրերի արժեքները (տող 7), որոնք փոխանցվում են հիմնական ծրագրում նկարագրված X վեկտորին: Այնուհետև տող 20-ում դիմում է արվել ART անունով ֆունկցիային, նրան փոխանցելով վեկտորի տարրերի քանակը և վեկտորի տարրերի արժեքները: Ֆունկցիայում հաշվվում է զանգվածի դրական տարրերի արտադրյալը (տողեր 12-14): Քանի որ ֆունկցիան հաշվված արտադրյալի արժեքը հիմնական ծրագրին է փոխանցում իր անվան միջոցով, այդ պատճառով ֆունկցիայի անվանը վերագրվել է արտադրյալի արժեքը (տող 15):

Քանի որ պրոցեդուրայի ֆորմալ պարամետրերի ցուցակում որպես պարամետր-արժեք հանդես է գալիս X զանգվածը, դրա համար նախապես ստեղծվել է նոր VEC տիպը: Պրոցեդուրայի և ֆունկցիայի վերնագրերում ֆորմալ պարամետրերի ցուցակում X1 և X2 պարամետրերը նկարագրվել են որպես VEC տիպի: Հիմնական ծրագրի փոփոխականների նկարագրման բաժնում X վեկտորը նույնպես նկարագրվել են որպես VEC տիպի փոփոխականներ:

Օրինակ 5.5: Տրված են m և n բնական թվերը և A_{mxn} իրական տարրերով մատրիցը: Կազմել բլոկ-սխեմա և ծրագիր, որոնք կստանան Y վեկտորը, որի y_i-րդ տարրը հավասար է մատրիցի i-րդ տողի բացասական տարրերի գումարին: Յուրաքանչյուր տողի բացասական տարրերի գումարի հաշվումը կազմակերպել ֆունկցիայի միջոցով:

Խնդրի լուծման բլոկ-սխեման պատկերված է նկ.5.3-ում, իսկ ծրագիրն ունի EXAM5_3 անունը:





Նկ.5.3

```
1       PROGRAM EXAM5_3;
2       TYPE MAT=ARRAY[1..20,1..20] OF REAL;
3       VEC=ARRAY[1..20] OF REAL;
4       VAR A:MAT; B,C:VEC; I,J,M,N:INTEGER;
5               FUNCTION SUM(K:INTEGER; Z:VEC):REAL;
6               VAR I:INTEGER;S:REAL;
7               BEGIN
8               S:=0;
9               FOR I:=1 TO K DO
10                      IF Z[I]<0 THEN S:=S+X[I];
11              SUM:=S
12              END;
13      BEGIN
14      READ(M,N);
15      FOR I:=1 TO M DO
16      FOR J:=1 TO N DO READ(X[I,J]);
17      FOR I:=1 TO M DO
18      BEGIN
19      FOR J:=1 TO N DO C[j]:=A[I,J];
20      Y[I]:=SUM(N,C);
21      WRITELN(Y[I]:6:2)
22      END.
23     END.
```

ՄՈՒՏՔ՝ m=3 n=4 2 –4 0 -3 ԵԼՔ՝ -7.00
-6 3 1 -3 -9.00
2 4 0 - 2 -2.00

Մատրիցի չափի և տարրերի ներմուծումից հետո (տողեր 14-16), բացվել է I պարամետրով ցիկլ, ուր նախ I-րդ տողի տարրերը պահվում են C վեկտորի մեջ (տող 19), որից հետո տող 2-ում դիմում է արվել SUM անունով ֆունկցիային, նրան փոխանցելով մատրիցի տողում տարրերի N քանակը և C վեկտորի տարրերը: Ինչպես նախորդ օրինակում, այստեղ նույնպես ստեղծվել է նոր VEC տիպ, որպեսզի ֆունկցիայի վերնագրում X պարամետր-արժեքը, որը ներկայացնում է վեկտոր, հայտարարվի որպես այդ տիպի: Ֆունկցիան հաշվում է իրեն փոխանցված վեկտորի (մատրիցի i-րդ տողի) դրական տարրերի գումարը և իր SUM անվան միջոցով փոխանցում հիմնական ծրագրին: Հիմնական ծրագրի տող 20-ում I-րդ տողի տարրերի գումարը վերագրվում է Y վեկտորի I-րդ տարրին: Քանի

որ I պարամետրով ցիկլի մարմինը պարունակում է մեկից ավելի օպերատորներ, ապա այդ օպերատորները վերցվել են BEGIN և END հատուկ բառերի միջև, այսինքն այդ օպերատորներից ստեղծվել է մեկ բաղադրյալ օպերատոր:

Ֆունկցիայում օգտագործվող S և I փոփոխականները նկարագրված են ֆունկցիայում և լոկալ փոփոխականներ են, իսկ պրոցեդուրայի համար լոկալ փոփոխական է միայն I փոփոխականը:

5.3. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	461, 472, 481, 491, 501
2	462, 476, 482, 498, 504
3	463, 479, 483, 492, 506
4	464, 474, 484, 495, 503
5	465, 480, 485, 496, 508
6	466, 473, 486, 500, 507
7	467, 478, 487, 499, 502
8	468, 477, 488, 498, 504
9	469, 475, 489, 495, 505
10	470, 471, 490, 491, 509
11	463, 471, 485, 500, 503
12	466, 472, 483, 494, 510
13	469, 473, 481, 491, 509
14	464, 474, 488, 494, 510
15	468, 475, 490, 497, 501
16	470, 476, 484, 497, 505
17	462, 477, 487, 499, 507
18	467, 478, 489, 496, 502
19	465, 479, 486, 495, 510
20	461, 480, 482, 493, 506
21	463, 471, 490, 494, 502
22	466, 474, 489, 500, 510
23	469, 473, 488, 497, 505
24	464, 476, 487, 493, 507
25	468, 478, 486, 496, 508
26	470, 479, 485, 498, 501
27	462, 480, 484, 492, 509
28	467, 475, 483, 499, 504
29	465, 477, 482, 493, 506
30	461, 472, 481, 492, 503

6. «Տող» տիպի փոփոխականներ

6.1. Հիմնական հասկացություններ

ՏՈՒՐԲՈ ՊԱՍԿԱԼ-ի տարբերակներում սիմվոլային տողերի (այսինքն սիմվոլների հաջորդականության) մշակման համար մտցված է հատուկ՝ STRING («տող») տիպը, որը հեշտացնում է նկարագրումները և գործողությունների արտահայտությունները: STRING տիպը ՏՈՒՐԲՈ ՊԱՍԿԱԼ-ում լայնորեն կիրառվում է տեքստային ինֆորմացիայի մշակման ժամանակ: Տողը ներկայացվում է ապաթարցերի մեջ վերցված սիմվոլների հաջորդականության տեսքով: Օրինակ. S:='AB4 5>+g.6)a' :

Փոփոխականների նկարագրման բաժնում STRING տիպի փոփոխականների հայտարարման համար անհրաժեշտ է գրել փոփոխականների անունները, STRING հատուկ բառը և քառակուսի փակագծերում նշել տողի սիմվոլների հնարավոր մեծագույն քանակը (տողի երկարությունը).

<փոփոխականների ցուցակ>:STRING[<տողի երկարություն>];

եթե նշված չէ տողի երկարությունը, ապա այն ընդունվում է հավասար 255-ի, որը տողի հնարավոր մեծագույն չափն է:

Օրինակ.

```
VAR
X:STRING;          - կարող է պարունակել մինչև 255 սիմվոլ,
Y:STRING[15];      - կարող է պարունակել մինչև 15 սիմվոլ,
 Z:STRING[255];    - կարող է պարունակել մինչև 255 սիմվոլ:
```

Տողի երկարության հետ կարելի է կատարել անրաժեշտ գործողություններ և փոխել տողի երկարությունը, օրինակ տողից հեռացնել կամ ավելացնել որոշակի սիմվոլներ: Բոլոր դեպքերում տողի պայմանանշանների քանակը չի կարող գերազանցել տողի նկարագրման ժամանակ նշված N քանակը: Եթե տողի սիմվոլների քանակը ստացվում է ավելի մեծ քան N-ը, ապա վերջին ավելորդ սիմվոլներն անտեսվում են:

Տողերը կարելի է ներմուծել READLN, իսկ արտածել WRITELN ստանդարտ պրոցեդուրաների օգնությամբ:

Գոյություն ունի STRING տիպի փոփոխականի մշակման երկու տարբերակ: Առաջին տարբերակով տողը կարելի է մշակել որպես մեկ ամբողջություն: Երկրորդ տարբերակում տողը դիտարկվում է որպես բաղադրյալ օբյեկտ, բաղկացած առանձին սիմվոլներից: Մեկ սիմվոլից բաղկացած տողը կարելի է դիտել որպես CHAR ստանդարտ տիպի հաստատուն: Ընդհանուր դեպքում N սիմվոլներ պարունակող տողը ներկայացնում է N+1 սիմվոլով զանգված.

ARRAY[0..N] OF CHAR;

Զրոյական սիմվոլը պարունակում է տողի ընթացիկ երկարությունը: Տողում ցանկացած սիմվոլի կարելի է դիմել նույն կերպ, ինչպես ARRAY[0..N] OF CHAR միաչափ զանգվածի տարրին: Դրա համար պետք է նշել տողի անունը՝ քառակուսի փակագծերում նշելով տողի տվյալ տարրի դիրքի համարը: Այս դեպքում տողի տարրի հետ կարելի է կիրառել CHAR տիպի փոփոխականի հետ կիրառելի բոլոր գործողությունները:

Տողերով և նրանց սիմվոլներով կարելի է կատարել մի շարք գործողություններ՝ օգտվելով ստորև շարադրված ստանդարտ պրոցեդուրաներից և ֆունկցիաներից:

- ***LENGTH(S)*** - ֆունկցիա է, որի արժեքը ամբողջ թիվ է, S-ը տող տիպի փոփոխական է կամ հաստատուն: Այս ֆունկցիայով որոշվում է տողի փաստացի երկարությունը: Այն կիրառվում է գործնական շատ դեպքերում: Մասնավորապես, երբ փոփոխականների նկարագրման բաժնում տողը հայտարարված է որոշակի երկարության, իսկ իրականում այն կարող է պարունակել ավելի քիչ սիմվոլներ: Օրինակ,

```
VAR   X:STRING[10]; K:INTEGER;
. . . . . .
X:='ADAMYAN';
K:=LENGTH(X);
WRITE('K=',K);
```

ԵԼՔ՝ K=7

- ***CONCAT(S1, S2, ... , SN)*** - ֆունկցիա է, որի արժեքը տող է: Այն վերադարձնում է S1, S2, ... SN տողերի հաջորդական կցումից ստացված տողը, այսինքն՝ գումարում է այդ տողերը: Օրինակ,

```
X:='TURBO';
Y:=' PASCAL';
Z:=CONCAT(S1,S2);
WRITELN('Z=',Z);
```

ԵԼՔ՝ Z='TURBOPASCAL'

Տողերի համար կիրառելի է նաև "+" գործողությունը: Նախորդ դեպքի համար Z:=X+Y գործողության արդյունքը կլինի նույնը:

Եթե անհրաժեշտ է TURBO և PASCAL բառերի միջև դնել բացատանիշ, ապա կարելի է գրել.

```
Z:=CONCAT(X,' ',Y);
```

ԵԼՔ՝ Z='TURBO PASCAL'

- **COPY(S,N,L) -** ֆունկցիա է, որի արժեքը տող է: Այն S տողի N դիրքից սկսած պատճենահանում է L հատ սիմվոլ: Օրինակ,

```
X:='TURBO PASCAL';
Y:=COPY(X,7,6);
WRITELN('Y=',Y)
```

ԵԼՔ՝ Y='PASCAL'

- ***DELETE(S, N, L)*** - պրոցեդուրա է, որը S տողի N դիրքից սկսած՝ հեռացնում է L հատ սիմվոլ: Օրինակ,

```
X:='TURBO PASCAL';
DELETE(X,1,6);
WRITELN('X=',X)
```

ԵԼՔ՝ X='PASCAL'

Տող 2-ում հայտարարվել է S տողը, որոնք կարող են ունենալ մինչև 255 սիմվոլ: Տող 4-ում S-ին վերագրվել է կոնկրետ արժեք: Տող 5-ում S փոփոխականի առաջին դիրքից սկսած հեռացվել է 6 հատ սիմվոլ (TURBO): Տող 6-ում արտածվել է S-ի նոր արժեքը:

- ***INSERT(S1, S2, N)*** - պրոցեդուրա է, որը S1 տողը տեղադրում է S2 տողի մեջ՝ սկսած N-րդ դիրքից: Օրինակ.

```
X:='AAA';
Y:='123456';
INSERT(X,Y,4);
WRITELN('Y=',Y)
```

ԵԼՔ՝ Y='123AAA456'

Տող 2-ում հայտարարվել են S1 և S2 տողերը, որոնք կարող են ունենալ մինչև 255 սիմվոլ: Տողեր 4-ում և 5-ում S1 և S2 փոփոխականներին վերագրվել են կոնկրետ արժեքներ: Տող 6-ում INSERT պրոցեդուրայով S1 փոփոխականի արժեքը տեղադրվել է S2-ի մեջ՝ սկսած 4-րդ դիրքից: Տող 7-ում արտածվել է S2 փոփոխականի նոր արժեքը:

Բերված օրինակում կարող էինք չօգտագործել X փոփոխականը և ստանալ նույն արդյունքը՝ կիրառելով INSERT պրոցեդուրա՝

```
INSERT('AAA',Y,4);
```

- ***POS(S1, S2)*** - ֆունկցիա է, որի արժեքը ամբողջ թիվ է: Այն վերադարձնում է S2 տողի այն դիրքի համարը, որտեղից սկսած՝ S1 ենթատողը առաջին անգամ մտել է S2-ի մեջ: Եթե ենթատողը տվյալ տողում չի պարունակվում, ապա ֆունկցիան վերադարձնում է 0 արժեք: Օրինակ.

```
X:='12AAA456AAA65AAA6';
Y:='AAA';
K:=POS(Y,X);
WRITELN('K=',K)
```

ԵԼՔ՝ K=3

Բերված օրինակում կարող էինք չօգտագործել Y փոփոխականը, այդ դեպքում POS ֆունկցիայի կիրառման ժամանակ անհրաժեշտ էր գրել

K:=POS(‘AAA’,X);

Երկու տողերի հետ համեմատման =, <, >, <=, >=, <> գործողությունները կատարվում են առանձին սիմվոլներով՝ ձախից աջ հաջորդականությամբ: Եթե մեկ տողը պարունակում է ավելի քիչ սիմվոլներ քան մյուսը, ապա կարճ տողում թերի սիմվոլները փոխարինվում են CHR(0) արժեքներով: Օրինակ, համեմատման հետևյալ գործողությունները տալիս են TRUE արժեք.

‘B’ >‘A’,
‘BARON’<’BOR’,
‘BAR’<’BOR’,
‘TURBO’ < ‘TURBO PASCAL’:

6.2. Պարզ ծրագրերի տիպային օրինակներ

Օրինակ 6.1. Տրված տողի մեջ AB սիմվոլների հաջորդականությունը փոխարինել C սիմվոլով:

Խնդրի լուծման ծրագիրն ունի EXAM6_1 անունը:

```
1       PROGRAM  EXAM6_1;
2       VAR S:STRING;
3       I,K:INTEGER;
4       BEGIN
5       S:=‘0ABC123AB4567AB+=89AB5’;
6       WHILE POS(‘AB’,S)<>0 DO
7       BEGIN
8       K:=POS(‘AB’,S);
9       DELETE(S,K,2);
10      INSERT(‘C’,S,K)
11      END;
12      WRITELN(‘S=‘,S)
13      END.
```

ԵԼՔ՝ S=’0C123C4567C+=89C5’

Տող 2-ում հայտարարված S տողին տող 5-ում վերագրվել է որոշակի արժեք: Տողեր 6-ից 11-ում կազմակերպվել է նախապայմանով ցիկլ, որի մարմնում նախ K փոփոխականին վերագրվում է այն դիրքի համարը, որից սկսած AB սիմվոլների հաջորդականությունը առաջին անգամ մտել է S տողի մեջ (տող 8): Այնուհետև S տողի K-րդ դիրքից ջնջվում է 2 հատ սիմվոլ (տող 9) և այդ նույն դիրքում INSERT պրոցեդուրայի օգնությամբ ավելացվում C սիմվոլները: Ցիկլը շարունակվում է այնքան ժամանակ, քանի դեռ տողում կա AB սիմվոլների հաջորդականություն, այսինքն՝ POS('AB',S)≠0: Տող 12-ում արտածվել է S տողի նոր արժեքը:

Օրինակ 6.2. Տրված է տող, որի մեջ կան միայն 2 հատ H սիմվոլներ: Կազմել ծրագիր, որը կհաշվի և կտպի երկրորդ H սիմվոլի դիրքի համարը:

Խնդրի լուծման ծրագիրն ունի EXAM6_2 անունը:

```
1       PROGRAM EXAM6_2;
2       VAR X,Y:STRING;    K,K1,K2:INTEGER;
3       BEGIN
4       READLN(X);
5       K1:=POS('H',X);
6       Y:=COPY(X,1,K1);
7       DELETE(S,1,K1);
8       K2:=POS('H',X);
9       K:=K1+K2;
10      INSERT(Y,X,1);
11      WRITE('K=',K)
12      END.
```

ՄՈՒՏՔ՝ ASDHSDFGJHSDF **ԵԼՔ՝** 10

Տող 5-ում X տողի ներմուծումից հետո տող 6-ում POS ֆունկցիայի միջոցով գտնում ենք առաջին H սիմվոլի դիրքի համարը (K1): Մինչև K1-րդ դիրքում գտնվող բոլոր K1 հատ սիմվոլները տող 7-ում պատճենահանում ենք Y տող տիպի փոփոխականի մեջ: Այնուհետև DELETE պրոցեդուրայի միջոցով տրված X տողից հեռացնում ենք առաջին K1 հատ սիմվոլները, այսինքն՝ մինչև առաջին H սիմվոլը ներառյալ: Տող 9-ում նորից կիրառելով POS ֆունկցիան այս անգամ գտնում ենք հաջորդ H սիմվոլի դիրքի համարը (K2), որն արդեն առաջին

Z սիմվոլի դիրքի համարն է, քանի որ առաջին H-ը արդեն ջնջել ենք: Գտնված համարների գումարը (K) երկրորդ H սիմվոլի դիրքի համարն է:

Օրինակ 6.3. Տրված են երկու տողեր: Կազմել ծրագիր, որը առաջին տողից կհեռացնի այն սիմվոլները, որոնց հավասարը կա երկրորդ տողում:

Խնդրի լուծման ծրագիրն ունի EXAM6_3 անունը:

Խնդրի լուծման ծրագիրն ունի EXAM50 անունը:

```
1       PROGRAM EXAM6_3;
2       VAR X,Y:STRING; N,I:INTEGER;
3       BEGIN
4       READLN(X); READLN(Y);
5       N:=LENGTH(X);
6       FOR  I:=1 TO N DO
7       IF POS(X[I],Y)=0 THEN DELETE(X,I,1)
8       WRITELN('X=',X)
9       END.
```

ՄՈՒՏՔ՝ X=’A23SD8FFGH’ **ԵԼՔ**՝ X=’A23H’
Y=’G87FDS’

Տող 4-ում X և Y տողերի ներմուծումից հետո տող 5-ում հաշվել ենք X տողի փաստացի երկարությունը: Տող 6-ում կազմակերպվել է ցիկլ, որի մարմնում (տող 7) ստուգվել է X տողի հերթական I-րդ սիմվոլի Y տողի մեջ լինելը: Եթե այդ տարրին հավասար տարր Y տողի մեջ չի հայտնաբերվել, այսինքն POS(X[I],Y)=0, ապա DELETE պրոցեդուրայի օգնությամբ X տողի համապատասխան սիմվոլը հեռացվում է տողից: Տող 8-ում արտածվել է ստացված տողը:

Օրինակ 6.4. Տրված է n բնական թիվը և n երկարությամբ տող: Հաշվել տողում առկա ‘A’ սիմվոլների քանակը, եթե տողում կա գոնե մեկ հատ ‘Z’ սիմվոլ, հակառակ դեպքում՝ հաշվել ‘B’ սիմվոլների քանակը:

Խնդրի լուծման ծրագիրն ունի EXAM6_4 անունը:

```
1       PROGRAM EXAM6_4;
2       VAR X:STRING; N,L:INTEGER; Y:CHAR;
3       BEGIN
4       READLN(X);
```

```
5       N:=LENGTH(X);
6       IF POS('Z',X)<>0 THEN  Y:='A' ELSE Y:='B';
7       L:=0;
8       FOR I:=1 TO N DO
9       IF X[I]=Y THEN L:=L+1;
10      WRITE(L)
11      END.
```

Տող 4-ում X տողի տերմուծումից հետո տող 5-ում որոշել ենք նրա փաստացի երկարությունը (N): Այնուհետև տող 6-ում կազմակերպվել է համեմատություն: Եթե տրված տողում կա 'Z' սիմվոլ, ապա սիմվոլային տիպի Y փոփոխականին վերագրվում է 'A' սիմվոլը, հակառակ դեպքում՝ 'B' սիմվոլը: Այս դեպքում խնդիրը բերվում է տրված տողում Y սիմվոլային փոփոխականին հավասար սիմվոլների քանակի որոշմանը, որն իրականացվել է տողեր 7-ից 9-ում:

6.3. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	511, 517, 521, 525, 531, 536
2	514, 518, 523, 529, 533, 538
3	513, 516, 522, 526, 532, 539
4	515, 519, 524, 527, 534, 537
5	512, 520, 521, 528, 543, 540
6	515, 518, 524, 525, 534, 538
7	514, 520, 522, 530, 531, 536
8	511, 516, 522, 526, 534, 540
9	516, 520, 521, 528, 533, 539
10	513, 517, 524, 530, 532, 537
11	516, 520, 522, 525, 535, 538
12	512, 519, 524, 529, 531, 539
13	515, 518, 523, 528, 534, 537
14	514, 520, 524, 527, 533, 540
15	511, 518, 521, 530, 532, 536
16	512, 517, 524, 526, 535, 538
17	515, 520, 523, 530, 531, 540
18	513, 520, 522, 527, 533, 537
19	512, 516, 524, 528, 532, 539
20	515, 519, 523, 525, 535, 536
21	514, 516, 521, 527, 534, 537
22	515, 517, 524, 529, 531, 538
23	511, 520, 524, 526, 533, 539
24	512, 519, 524, 530, 535, 536
25	513, 518, 522, 525, 533, 540
26	515, 520, 523, 528, 531, 537
27	514, 516, 521, 527, 535, 538
28	515, 517, 524, 529, 534, 536
29	513, 519, 523, 526, 531, 539
30	511, 518, 522, 527, 532, 536

7. Գրառումներ

7.1. Ընդհանուր հասկացություններ

Պասկալ լեզվում գրառումներով հնարավոր է միևնույն համախմբության մեջ պահել տարբեր տիպի տվյալներ: Դրանք կարող են պարունակել տարբեր տիպի ֆիքսված թվով բաղադրիչներ, որոնք կոչվում են դաշտեր: Մեկ դաշտում տվյալները պետք է ունենան նույն տիպը, իսկ տարբեր դաշտերում՝ տարբեր տիպեր: Գրառում տիպը հայտարարվում է հետևյալ կերպ.

```
TYPE

   a=RECORD

        d11, d12, ...  :t1;
        d21, d22, ...  :t2;
        . . . . . .

        dK1, dk2, ...  :tK
        END;
```

որտեղ

a –ն տիպի անունն է,
d_{ij} –ն դաշտերի իդենտիֆիկատորներն են,
t_i -ն դաշտերի տիպը:

Օրինակ. դիցուք անհրաժեշտ է նկարագրել երեք փոփոխականներ՝ A, B և C, որոնցից յուրաքանչյուրն ունի չորս բաղադրիչներ՝ անունը, օրը, ամիսը և տարին:

```
VAR A,B,C:RECORD
        NAME:STRING;
        DAY,MONTH,YEAR:INTEGER
    END;
```

Այստեղ A,B և C փոփոխականները հայտարարված են որպես գրառումներ, որոնց համար բաղադրիչներ են NAME (անուն), DAY (օր),

MONTH (ամիս) և YEAR (տարի) դաշտերը: Ինչպես երևում է բերված օրինակից, դրանցից առաջինն ունի STRING տիպ, իսկ մնացած երեքը՝ ամբողջ տիպ:

Գրառումը կարող է հանդես գալ որպես այլ կառուցվածքային տիպի բաղադրիչ: Օրինակ, կարելի է կառուցել գրառումների զանգված: Դիցուք անհրաժեշտ է նկարագրել այնպիսի զանգված, ուր լինեն պահված խմբի բոլոր ուսանողների անունները, նրանց համարները դասամատյանում և ծննդյան տարեթվերը: Ենթադրենք՝ խմբում կան N ուսանողներ: Այս դեպքում նկարագրությունը կարելի է ներկայացնել հետևյալ տեսքով.

```
VAR
    X:ARRAY[1..N] OF RECORD
      NAME:STRING;
      NUMBER,YEAR:INTEGER
    END.
```

այսինքն՝ X -ը N տարր պարունակող միաչափ զանգված է: Հայտարարված զանգվածի տարրերը գրառումներ են, որոնց համար բաղադրիչներ են ուսանողի անունը՝ NAME (STRING տիպի), դասամատյանի համարը՝ NUMBER (INTEGER տիպի) և ծննդյան տարեթիվը՝ YEAR (INTEGER տիպի):

Գրառում տիպի փոփոխականին կարելի է վերագրել նույն տիպի ուրիշ գրառման արժեքը, օրինակ.

```
A:=B;
```

Գրառման յուրաքանչյուր բաղադրիչի կարելի է դիմել՝ նշելով փոփոխականի անունը, այնուհետև կետ և վերջում նշել դաշտի անունը, այսինքն՝ տալ գրառման լրիվ անունը, օրինակ.

```
A.NAME:='GOHAR';
```

որտեղ

A-ն գրառման անունն է,
NAME-ը՝ գրառման դաշտի անունը:

Եթե գրառման բաղադրիչը ներկայացնում է նորից գրառում, ապա այն կոչվում է ներդրված դաշտերով գրառում: Այդ դեպքում անհրաժեշտ է

ստեղծել գրառում-տիպ, որը կօգտագործվի փոփոխականների նկարագրման բաժնում՝ գրառման բաղադրիչի տիպը նկարագրելիս:

Ամեն անգամ գրառման բաղադրիչից օգտվելիս գրում ենք գրառման անունը, որից հետո կետով բաժանված՝ գրում դաշտի անունը: Որպեսզի ամեն անգամ չգրենք գրառման լրիվ անունը, կարելի է օգտվել WITH օպերատորից:

Այն ունի հետևյալ տեսքը.

WITH <գրառման անուն> DO <օպերատոր>;

Այս դեպքում օպերատորի ներսում կարելի է նշել միայն գրառման դաշտը:

Օրինակ, A գրառումն ունի չորս՝ D1, D2, D3, D4 բաղադրիչները: Գրառումը կարելի է ներմուծել հետևյալ երկու տարբերակներով.

```
READ(A.D1,A.D2,A.D3,A.D4);
```

կամ

```
WITH A DO READ(D1,D2,D3,D4);
```

7.2. Պարզ ծրագրերի տիպային օրինակներ

Օրինակ 7.1. Տրված է n ամբողջ թիվը և n տարր պարունակող զանգվածը: Զանգվածի տարրերը գրառումներ են, որոնց համար բաղադրիչներ են՝ խմբի յուրաքանչյուր ուսանողի ա)ազգանունը բ)մաթեմատիկա առարկայի ընթացիկ ռեյտինգային միավորը գ)մաթեմատիկա առարկայի եզրափակիչ քննության միավորը: Տպել այն ուսանողների ազգանունները, որոնք ստացել են գերազանց գնահատական (այսինքն՝ ռեյտինգային և եզրափակիչ միավորների գումարը մեծ է 80-ից):

Խնդրի լուծման ծրագիրն ունի EXAM7_1 անունը:

```
1    PROGRAM  EXAM7_1;
2    VAR X:ARRAY[1..5] OF RECORD
```

```
3     AZG:STRING;
4     REY, EZR:INTEGER;
5     END;
6     N,I:INTEGER;
7     BEGIN
8     READ(N);
9     FOR I:=1 TO N DO WITH X[I] DO
10      BEGIN
11      READLN(AZG);
12      READLN(REY);
13      READLN(EZR)
14      END;
15    FOR I:=1 TO N DO WITH X[I] DO
16      IF REY+EZRA>80 THEN WRITELN(AZG);
17    END.
```

Տողեր 2-ից 5-ում նկարագրվել է այն զանգվածը, որի տարրերը գրառում է AZG, REY և EZR բաղադրիչներով: Տողեր 9-ից 14-ում ներմուծվել են զանգվածի տարրերի արժեքները: Տող 15-ում կազմակերված ցիկլում յուրաքանչյուր կրկնման ժամանակ ստուգվում է տվյալ ուսանողի ռեյտինգային և եզրափակիչ միավորների գումարի 80-ից մեծ լինելը: Եթե այն ճշմարիտ է, ապա ցիկլի մարմնում արտածվում է այդ ուսանողի ազգանունը:

Օրինակ 7.2. Տրված է n ամբողջ թիվը և n տարր պարունակող զանգվածը: Զանգվածի տարրերը գրառումներ են, որոնց համար բաղադրիչներ են տվյալ խմբի ուսանողների՝ ա) ազգանունը բ) մի առարկայի քննական միավորը: Տպել խմբի այն ուսանողների ազգանունները, որոնց գնահատականը մեծ է խմբի միջին գնահատականից:

Խնդրի լուծման ծրագիրն ունի EXAM7_2 անունը:

```
1         PROGRAM  EXAM7_2;
2         CONST N=4;
3         VAR
4         X:ARRAY[1..N] OF RECORD
5         AZG:STRING;
```

```
6       MIAV:INTEGER
7       END;
8       I:INTEGER; S:REAL;
9       BEGIN
10      X[1].AZG:='AVAGYAN';     X[1].MIAV:=40;
11      X[2].AZG:='GASPARYAN'; X[2].MIAV:=100;
12      X[3].AZG:='PETROSYAN'; X[3].MIAV:=80;
13      X[4].AZG:='ANTONYAN';   X[4].MIAV:=60;
14      S:=0;
15      FOR I:=1 TO N DO WITH X[I] DO S:=S+MIAV:
16      S:=S/N;
17      FOR I:=1 TO N DO WHIT X[I] DO
18      IF MIAV>S THEN  WRITELN(AZG)
19      END.
```

ԵԼՔ՝ GASPARYAN
PETROSYAN

Հաստատուների նկարագրման բաժնում N փոփոխականին վերագրվել է 4 արժեքը (տող 2)։ Տողեր 4-ից 7-ում նկարագրվել է զանգվածը, որի տարրերը գրառում է AZG և MIAV բաղադրիչներով։ Տողեր 10-ից 13-ում ներմուծվել են զանգվածի տարրերի արժեքները, որոնք են. յուրաքանչյուր ուսանողի ազգանունը (AZG) և քննական միավորը (MIAV)։ Տողեր 14-ից 15-ում որոշվել է խմբի բոլոր ուսանողների միավորների գումարը (S), որի արժեքը բաժանելով խմբի ուսանողների քանակին (տող 16)՝ կստանանք խմբի միջին գնահատականը։ Տողեր 17-ից 18-ում կազմակերպվել է ցիկլ, որի մարմնում գտնվում և արտածվում են այն ուսանողների ազգանունները, որոնց գնահատականը մեծ է խմբի միջին գնահատականից։

Օրինակ 7.3. Տրված է n ամբողջ թիվը և n տարր պարունակող զանգվածը։ Զանգվածի տարրերը գրառումներ են, որոնց համար բաղադրիչներ են ձայնասկավառակում ձայնագրված երգերի ա)անունները, բ)տևողությունները (րոպեներով)։ Տպել ձայնասկավառակի առաջին երգից սկսած բոլոր այն երգերի անունները, որոնց գումարային տևողությունը չի գերազանցում 25 րոպեն։

Խնդրի լուծման ծրագիրն ունի EXAM7_3 անունը:

```
1       PROGRAM  EXAM7_3;
2       VAR X:ARRAY[1..5] OF RECORD
3       ANUN:STRING;
4       TEV:INTEGER
5       END;
6       N,I:INTEGER;
7       BEGIN
8       READ(N);
9       FOR I:=1 TO N DO WITH X[I] DO
10        BEGIN
11        READLN(ANUN);
12        READLN(TEV)
13        END;
14      I:=1;
15      S:=0;
16      WHILE S<40 DO WITH X[I] DO
17         BEGIN
18         WRITELN(ANUN);
19         I:=I+1;
20         S:=S+TEV
21         END
22      END.
```

ՄՈՒՏՔ՝

```
N=5
ERG1  8
ERG2  9
ERG3  4
ERG4  6
ERG5  8
```

ԵԼՔ՝

```
ERG1
ERG2
ERG3
```

Տողեր 2-ից 5-ում նկարագրվել է զանգվածը, որի տարրերը գրառում է ANUN և TEV բաղադրիչներով: Տողեր 9-ից 13-ում ներմուծվել են զանգվածի տարրերի արժեքները: Տող 12-ում կազմակերպվել է նախապայմանով ցիկլ, որը կրկնվում է այնքան ժամանակ, քանի դեռ

երգերի գումարային տևողությունը փոքր է 25-ից: Ցիկլի մարմնում յուրաքանչյուր կրկնման ժամանակ արտածվում է հերթական երգի տևողությունը: Գումարի համար նախատեսված S փոփոխականը մեծացվում է հերթական երգի տևողության չափով, իսկ երգի հերթական համարը՝ 1-ով: Մինչև ցիկլը տող 14-ում I փոփոխականին վերագրվել է 1 արժեք, իսկ տող 15-ում S փոփոխականին՝ 0 արժեք:

Օրինակ 7.4. Տրված է n ամբողջ թիվը և n տարր պարունակող զանգվածը: Զանգվածի տարրերը գրառումներ են, որոնց համար բաղադրիչներ են աշխատակիցների ա)ազգանունները բ)նրանց ընտանիքի անդամների քանակը գ)ընտանիքի գումարային եկամուտը: Տալել այն աշխատակցի ազգանունը, ում ընտանիքի յուրաքանչյուր անդամին ընկնող միջին եկամուտն ամանամեծն է:

Խնդրի լուծման ծրագիրն ունի EXAM7_4 անունը:

```
1    PROGRAM  EXAM7_4;
2    VAR X:ARRAY[1..5] OF RECORD
3    AZG:STRING;
4    QANAK, EKAMUT:INTEGER
5    END;
6    N,I,K:INTEGER; MAX, A:REAL;
7    BEGIN
8    READ(N);
9    FOR I:=1 TO N DO WITH X[I] DO
10      BEGIN
11      READLN(AZG); READLN(QANAK); READLN(EKAMUT)
12      END;
13   WITH X[1] DO BEGIN MAX:=EKAMUT/QANAK END; K:=1;
14   FOR I:=2 TO N DO WITH X[I] DO
15      BEGIN
16      A:=EKAMUT/QANAK;
17      IF A>MAX THEN BEGIN MAX:=A; K:=I END;
18      END;
19   WRITE(X[K].AZG)
20   END.
```

ՄՈՒՏՔ՝ N=5 **ԵԼՔ՝** EGORYAN

AVAGYAN	3	30000
GEVORGYAN	7	140000
ARAMYAN	4	100000
DURYAN	5	80000
EGORYAN	4	200000

Տողեր 2-ից 5-ում նկարագրվել է զանգվածը, որի տարրերը գրառում է AZG, QANAK և EKAMUT բաղադրիչներով: Տողեր 9-ից 12-ում ներմուծվել են զանգվածի տարրերի արժեքները: Տող 13-ում մեծագույն տարրը գտնելու համար նախատեսված MAX փոփոխականին վերագրվել է առաջին աշխատակցի ընտանիքի յուրաքանչյուր անդամին ընկնող միջին եկամուտը, այսինքն՝ գումարային եկամուտի և ընտանիքի անդամների քանակի հարաբերությունը, իսկ K փոփոխականին՝ 1: Տող Տողեր 14-ից 18-ում գտնվել է այն աշխատակցի հերթական համարը, ում ընտանիքի յուրաքանչյուր անդամին ընկնող միջին եկամուտը ամենամեծն է (K): Տող 19-ում արտածվել է K-րդ աշխատակցի ազգանունը:

7.3. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները՝ [2] խնդիրների ժողովածուից (Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.):

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	541, 546, 551, 556, 561, 567
2	542, 547, 553, 557, 563, 568
3	545, 549, 552, 559, 565, 569
4	542, 548, 555, 558, 562, 566
5	544, 546, 553, 557, 564, 567
6	541, 550, 551, 556, 563, 568
7	543, 547, 554, 560, 561, 570
8	545, 548, 552, 558, 562, 567
9	542, 549, 553, 556, 565, 566
10	544, 547, 551, 560, 563, 569
11	543, 548, 555, 557, 562, 570
12	541, 550, 552, 559, 564, 567
13	543, 546, 553, 558, 563, 569
14	545, 549, 554, 560, 561, 568
15	542, 547, 551, 556, 562, 567
16	544, 550, 555, 557, 565, 566
17	543, 549, 552, 559, 564, 569
18	541, 546, 553, 560, 562, 570
19	545, 547, 554, 558, 563, 567
20	543, 548, 551, 559, 561, 569
21	545, 550, 555, 560, 565, 568
22	544, 549, 553, 556, 564, 567
23	541, 546, 552, 557, 562, 566
24	542, 547, 551, 558, 565, 569
25	543, 548, 554, 556, 561, 568
26	544, 549, 555, 559, 564, 567
27	543, 546, 553, 557, 563, 570
28	541, 547, 551, 558, 565, 569
29	545, 550, 554, 560, 561, 566
30	542, 546, 552, 556, 562, 568

8. ՖԱՅԼԵՐ

8.1. Ընդհանուր հասկացություններ

Ֆայլը արտաքին հիշողության այն տիրույթն է, որն ունի անուն: Ցանկացած ֆայլում կարող են գրված լինել միևնույն տիպի բաղադրիչներ: Նոր ստեղծվող ֆայլի հնարավոր ծավալը կախված է միայն համակարգչի այն արտաքին հիշող սարքի ծավալից, որտեղ պետք է գրվի տվյալ ֆայլը:

Տարբերում են ֆայլերի երեք տեսակ.

1. **Տիպայնացված ֆայլ:** Այնպիսի տվյալների հաջորդականություն է, որոնց տիպը ֆիքսվել է ֆայլի հայտարարման պահին: Այս դեպքում ֆայլային փոփոխականը կարելի է նկարագրել հետևյալ կերպ.

```
VAR
    < անուն> : FILE OF <տիպ>;
```

2. **Տեքստային ֆայլ:** Տողերում միավորված պայմանանշանների հավաքածու է: Այս դեպքում Ֆայլային փոփոխականը կարելի է նկարագրել հետևյալ կերպ.

```
VAR
    < անուն>: TEXT;
```

3. **Ոչ տիպայնացված ֆայլ:** Այնպիսի տվյալների հաջորդականություն է, որոնց տիպը նախապես չի ֆիքսվել: Այս դեպքում Ֆայլային փոփոխականը կարելի է նկարագրել հետևյալ կերպ.

```
VAR
    < անուն> : FILE;
```

Նշված երեք ձևերի մեջ <անուն> -ը ֆայլային փոփոխականի անունն է (ցանկացած թույլատրելի իդենտիֆիկատոր), file, of, text- բանալի-բառեր են, իսկ <տիպ>-ը ՏՈՒՐԲՈ ՊԱՍԿԱԼ-ում սահմանված որևէ տիպ է, բացի ֆայլայինից: Օրինակ,

```
F1: FILE OF REAL;
```

F2: TEXT;
F3: FILE;

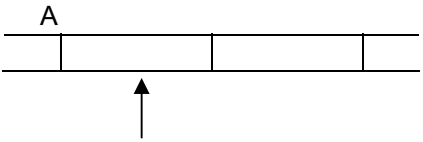
Բերված օրինակում f1-ը տիպայնացված ֆայլ է, որի բաղադրիչները իրական տիպի թվեր են, f2-ը տեքստային ֆայլ է, իսկ f3-ը՝ ոչ տիպայնացված ֆայլ:

Տվյալ աշխատանքի շրջանակներում մենք գործ ենք ունենալու միայն տիպայնացված ֆայլերի հետ:

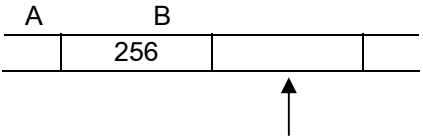
Ֆայլային փոփոխականների հետ չի կարելի կատարել ոչ մի գործողություն (վերագրում, համեմատում և այլն): Դրանք կարելի է օգտագործել միայն ֆայլերի հետ կապված գործողությունների մեջ (ֆայլի ընթերցում, ձայնագրում, ոչնչացում և այլն):

Ֆայլում ինֆորմացիայի գրանցման կամ ֆայլից ինֆորմացիայի ընթերցման սկզբունքի նկարագրման լավ մոդել է մագնիսական ձայնագրիչի դեպքը, եթե պատկերացնենք, որ կարդացող կամ գրող սարքը (մագնիսական գլխիկը) շարժվում է անշարժ մագնիսական ժապավենի երկայնքով (իրականում հակառակն է):

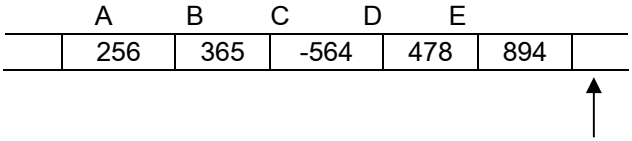
Դիցուք մագնիսական գլխիկը, որը պատկերված է սլաքով ↑), գտնվում է մագնիսական ժապավենի A տիրույթի դիմաց.



Եթե ժապավենի վրա ձայնագրենք որևէ թիվ, օրինակ 256, ապա մագնիսական գլխիկի դիմաց կկանգնի ժապավենի հաջորդ B ազատ տիրույթը.



Ենթադրենք ժապավենի վրա արդեն գրվել են 256, 365, -564, 478, 894 հինգ թվերը, և սրանով էլ ավարտվել է ֆայլի բաղադրիչների գրանցումը:



Այս դեպքում ասում են, որ մագնիսական ժապավենի վրա գրվել է 5 երկարությամբ ֆայլ: Ֆայլի ձայնագրման վերջում մագնիսական գլխիկը գտնվում է ֆայլից դուրս:

Ընդհանրացնելով ասվածը՝ նկատենք, որ եթե իրականացվել է ֆայլի n-րդ տարրի ձայնագրումը, ապա ֆայլի ցուցիչը (մագնիսական գլխիկը) կանգնում է հաջորդ ազատ դիրքի վրա, որտեղ կարելի է գրանցել հաջորդ՝ n+1-րդ տարրը:

Ֆայլի երկարություն է կոչվում ֆայլում գրված բաղադրիչների քանակը: Ֆայլը կոչվում է դատարկ (զրոյական), եթե չի պարունակում ոչ մի բաղադրիչ: Ֆայլի մեջ տարրի ներմուծումը և ընթերցումը կատարվում է բաղադրիչ առ բաղադրիչ:

8.2. Ֆայլերին դիմելու հնարավորությունը

ՊԱՍԿԱԼ ծրագրում կամայական ֆայլի կարելի է դիմել՝ նախապես հայտարարված ֆայլային տիպի փոփոխականը կապելով գոյություն ունեցող կամ նոր ստեղծվող ֆայլի անվան հետ: Այդ իրականացվում է ASSIGN ստանդարտ պրոցեդուրայի միջոցով, որն ունի հետևյալ տեսքը.

ASSIGN(f,name);

Այս պրոցեդուրան ֆայլային տիպի F փոփոխականը կապում է Name անունով արտաքին ֆայլի հետ: NAME-ը տեքստային (STRING տիպի) արտահայտություն է, որը պարունակում է ֆայլի անվանումը:

Ֆայլի անունը գրվում է հետևյալ կանոններով.

ա) անունը պետք է պարունակի մինչև 8 պայմանանշաններ (թույլատրելի պայմանանշաններն են լատինական այբուբենի տառերը, արաբական թվերը, հետևյալ պայմանանշանները՝ ! @ # $ % ^ & ( ) ‘ ~ - _):

բ) անվանը, կետով բաժանված, կարող է հաջորդել ընդլայնումը (եթե անունն ունի ընդլայնում), որը կարող է պարունակել մինչև երեք թույլատրելի պայմանանշաններ:

գ) Ֆայլի անվանը կարող է նախորդել ֆայլը գտնելու ուղին, այսինքն՝ արտաքին հիշող սարքի անունը, և սկսած հիմնային կատալոգից այն կատալոգների անունները, որոնց շղթայի վերջնական օղակում գտնվում է տվյալ ֆայլը: Եթե ֆայլը գտնվում է ընթացիկ կատալոգի մեջ, ապա ուղին կարող է սկսվել ընթացիկ կատալոգից: Եթե արտաքին հիշող սարքը կամ կատալոգը նշված չէ, ապա հասկացվում է ընթացիկ արտաքին հիշող սարքը, ընթացիկ կատալոգը:

Օրինակ.

ASSIGN(X,’C:\BP\BIN\AA1.DAT’)- ֆայլային X փոփոխականը կապվում է C: կուտակիչի հիմնային կատալոգի BP ենթակատալոգի BIN ենթակատալոգի AA1.DAT ֆայլի հետ:

ASSIGN(Y1,’AA2.DAT’)- ֆայլային Y1 փոփոխականը կապվում է ընթացիկ կուտակիչի ընթացիկ կատալոգի AA2.DAT ֆայլի հետ:

8.3. Ֆայլերի նախապատրաստումը աշխատանքի

Ֆայլը նախապատրաստել աշխատանքի նշանակում է այդ ֆայլի համար նշել տվյալների փոխանակման ուղղությունը: ՏՈՒՐԲՈ ՊԱՍԿԱԼ-ում ֆայլը կարելի է բացել տվյալներ կարդալու, գրելու, կամ կարդալու ու միևնույն դեպքում գրելու համար:

Ֆայլից տվյալներ կարդալու նախապատրաստման համար կիրառում են RESET ստանդարտ պրոցեդուրան, որն ունի հետևյալ տեսքը.

RESET(F);

Այս պրոցեդուրայի օգնությամբ արդեն գոյություն ունեցող ֆայլը նախապատրաստվում է ընթերցման: RESET-ին պետք է նախորդած լինի ASSIGN պրոցեդուրան, որով F ֆայլային փոփոխականը պետք է կապակցվեր անհրաժեշտ ֆայլի հետ: RESET պրոցեդուրայի կատարումից հետո ֆայլի ընթացիկ տարրը ցույց տվող ցուցիչը կանգնում է ֆայլի սկզբում (առաջին տարրի վրա):

Օրինակ,

```
ASSIGN(Y1,'T1.DAT');
RESET(Y1);
```

Բերված օրինակում ASSIGN պրոցեդուրայի միջոցով Y1 ֆայլային փոփոխականը կապվել է T1.DAT ֆայլի հետ, այնուհետև T1.DAT ֆայլը բացվել է ընթերցման նպատակով:

ՏՈՒՐԲՈ ՊԱՍԿԱԼ տարբերակում թույլատրվում է RESET պրոցեդուրայի օգնությամբ բացած տիպայնացված ֆայլերում կատարել նոր տվյալների ձայնագրում:

Ֆայլում տվյալներ գրելը նախապատրաստելու համար կիրառում են REWRITE ստանդարտ պրոցեդուրան, որն ունի հետևյալ տեսքը.

REWRITE(F);

Պրոցեդուրան բացում է ֆայլը, որի հետ նախապես ASSIGN պրոցեդուրայով կապվել է F ֆայլային փոփոխականը: Եթե այդ անունով ֆայլ արդեն գոյություն ունի, ապա REWRITE-ը ջնջում է եղած տվյալները: Նոր ստեղծվող ֆայլում ցուցիչը կանգնում է ֆայլի սկզբում (առաջին տարրի վրա) և սպասում է ինֆորմացիայի ընդունման:

Օրինակ.

```
ASSIGN(Y2,'T2.DAT');
REWRITE(Y2);
```

Բերված օրինակում ASSIGN պրոցեդուրայի միջոցով Y2 ֆայլային փոփոխականը կապվում է T2.DAT ֆայլի հետ, իսկ REWRITE-ը նախապատրաստում է ֆայլը՝ տվյալների ձայնագրման համար:

8.4. Ֆայլերի փակումը

CLOSE(F) պրոցեդուրան փակում է նախապես բացած ֆայլը, որի հետ կապված է F ֆայլային փոփոխականը: Այդ դեպքում պրոցեդուրան ապահովում է բոլոր նոր կատարված փոփոխությունների գրանցումը արտաքին ֆայլում: Տվյալ ֆայլի հետ աշխատանքը շարունակելու համար պետք չէ նորից կիրառել ASSIGN պրոցեդուրան, քանի որ CLOSE(F) -ը չի խզում կապը ֆայլային փոփոխականի և արտաքին ֆայլի միջև:

Օրինակ.

```
CLOSE(Y1);
```

Այս պրոցեդուրան փակում է այն բաց ֆայլը, որի հետ կապված է Y1 ֆայլային փոփոխականը:

Չի կարելի մեկ CLOSE պրոցեդուրայի կիրառմամբ փակել մեկից ավելի ֆայլեր:

Ավելորդ բարդություններից (ինֆորմացիայի կորուստ, ինֆորմացիայի ոչ ճիշտ ներմուծում) խուսափելու համար խորհուրդ է տրվում ծրագրի վերջում անպայման փակել բոլոր բացված ֆայլերը:

8.5. Տվյալների գրանցումը և ընթերցումը

Ֆայլի մեջ տվյալների ներմուծման համար օգտագործում են READ և READLN, իսկ ֆայլից տվյալների ընթերցման համար՝ WRITE և WRITELN պրոցեդուրաները:

READ(f, <մուտքի ցուցակ>)- պրոցեդուրան f ֆայլային փոփոխականի հետ կապված արդեն գոյություն ունեցող ֆայլից տվյալներ է կարդում և վերագրում մուտքի ցուցակում նշված փոփոխականներին:

Օրինակ,

READ(F1,A);

F1 ֆայլային փոփոխականի հետ կապված ֆայլից ընթերցում և A փոփոխականին է վերագրում ֆայլի հերթական բաղադրիչի արժեքը:

Մուտքի ցուցակում փոփոխականների քանակը սահմանափակված չէ:

READ(F,A,B,C) պրոցեդուրայի կատարումը համարժեք է

```
BEGIN
READ(F,A);
READ(F,B);
READ(F,C)
END;
```

բաղադրյալ օպերատորի կատարմանը:

Օրինակ,

READ(F1,A,B);

F1 ֆայլային փոփոխականի հետ կապված ֆայլից ընթերցում է հերթական երկու բաղադրիչները, որոնցից առաջինի արժեքը վերագրում է A փոփոխականին, իսկ երկրորդի արժեքը՝ B փոփոխականին:

Օրինակ, դիցուք F ֆայլային փոփոխականի հետ կապված ֆայլում գրված են երկու տողեր, յուրաքանչյուրում՝ երեքական թիվ.

1	6	4	⊗	2	3	7	⊗	•

Աղյուսակում • -ով ցույց է տրված ֆայլի վերջը, իսկ ⊗- ով՝ «տողի վերջը» պայմանանշանը:

```
REZET(F);
READ(F,A,B);
```

A և B փոփոխականները կընդունեն համապատասխանաբար 1 և 6 արժեքները:

```
REZET(F);
READ(F,A,B,C,D,E);
```

A, B և C փոփոխականները կընդունեն առաջին տողի երեք պայմանանշանները (1, 6 և 4), որից հետո չորրորդ և հինգերորդ դիմումների ժամանակ վերադարձվում է դատարկ տող, և D ու E փոփոխականները արժեքներ չեն ստանում:

Տիպայնացված ֆայլերի համար մուտքի ցուցակում նշված փոփոխականները պետք է ունենան ֆայլի բաղադրիչների տիպը:

Օրինակ. Գրել ծրագրի հատված, որը կստանա 10 տարրեր պարունակող X զանգվածը: Զանգվածի տարրերը նախորդ օրինակում բերված ֆայլի առաջին 10 բաղադրիչներն են:

```
RESET(F1);
FOR I:=1 TO 10 DO   READ(F1,X[I]);
```

RESET(F1) պրոցեդուրայի օգնությամբ արդեն գոյություն ունեցող ֆայլը նախապատրաստվում է ընթերցման: RESET-ին պետք է նախորդած լինի ASSIGN պրոցեդուրան, որով F ֆայլային փոփոխականը պետք է կապակցվեր անհրաժեշտ ֆայլի հետ: RESET պրոցեդուրայի կատարումից հետո ֆայլի ընթացիկ տարրը ցույց տվող ցուցիչը կանգնում է ֆայլի սկզբում: Ցիկլի մեջ I պարամետրի յուրաքանչյուր արժեքի համար ֆայլից ընթերցվում է նրա I -րդ բաղադրիչը (I=1,10) և վերագրվում զանգվածի I -րդ տարրին:

READLN(f, <մուտքի ցուցակ>)- պրոցեդուրան f ֆայլային փոփոխականի հետ կապված արդեն գոյություն ունեցող ֆայլից կարդում և մուտքի ցուցակում նշված փոփոխականներին է վերագրում արժեքներ: Այս պրոցեդուրան նման է READ պրոցեդուրային, միայն այն տարբերությամբ, որ տողի չկարդացված մասը և տողի վերջի նշանը բաց է թողնվում և հաջորդ READ կամ READLN պրոցեդուրաներից որևէ մեկին դիմելու դեպքում կկարդա սկսած նոր տողի առաջին սիմվոլից:

READLN պրոցեդուրայում կարելի է նշել կամայական թվով պարամետրեր:

READLN(F,A,B,C,D) պրոցեդուրան համարժեք է

```
BEGIN
READ(F,A);
READ(F,B);
READ(F,C);
READ(F,D);
READLN;
END
```

բաղադրյալ օպերատորի կատարմանը, որտեղ սկզբում իրականացվում է պրոցեդուրայի մուտքի ցուցակում նշված բոլոր փոփոխականների արժեքների ներմուծումը, որից հետո կատարում է անցում հաջորդ տողին:

Օրինակ. դիցուք մուտքային տվյալները գրված են երեք տողերում, յուրաքանչյուրում՝ երեքական թիվ.

3	5	7	⊗	4	6	1	⊗	2	8	9	⊗	•

Եթե անհրաժեշտ է ծրագրի տարբեր հատվածներում ֆայլից կարդալ և A փոփոխականին վերագրել յուրաքանչյուր տողի առաջին թիվը, ապա այն կարելի է կազմակերպել READ պրոցեդուրայի միջոցով՝ ներմուծելով ավելորդ B և C փոփոխականները, որոնք ծրագրում չեն օգտագործվում.

```
READ(F,A,B,C);
```

Պրոցեդուրայի առաջին կատարման ժամանակ կկարդացվեն առաջին տողի երեք թվերը, երկրորդ կատարման ժամանակ՝ երկրորդ տողի երեք թվերը և այլն: Նման իրավիճակներից խուսափելու համար հարմար է օգտվել READLN պրոցեդուրայից, որը մուտքային տվյալներից կարդալով մուտքի ցուցակում նշված փոփոխականների քանակին հավասար տվյալներ՝ կատարում է անցում հաջորդ տողի սկզբին:

Նախորդ օրինակում ամեն անգամ գրելով

```
READLN(F,A);
```

հերթական տողից կարդալով առաջին տվյալը՝ կանցնենք հաջորդ տողի առաջին տվյալին: Այսպիսով, առաջին կիրառման դեպքում A փոփոխականին կվերագրվի 3 արժեքը, երկրորդ կիրառման արդյունքում՝ 4 արժեքը, իսկ երրորդ կիրառման դեպքում՝ 2 արժեքը:

Եթե վերը բերված տվյալներից անհրաժեշտ է կարդալ առաջին տողի առաջին թիվը և այնուհետև երրորդ տողի առաջին թիվը, ապա կարելի է գրել

```
READLN(A);
READLN;
READLN(A);
```

Առանց պարամետրի READLN պրոցեդուրան կատարում է անցում հաջորդ տողի սկզբին:

WRITE(f, <ելքի ցուցակ>)- պրոցեդուրան f ֆայլային փոփոխականի հետ կապված արդեն գոյություն ունեցող ֆայլում գրում է ելքի ցուցակում նշված փոփոխականների արժեքները:

Ինչպես READ պրոցեդուրայի դեպքում, այստեղ նույնպես ելքի ցուցակում փոխականների քանակը սահմանափակված չէ, և

```
WRITE(F,A,B,C)
```

պրոցեդուրայի կատարումը համարժեք է

```
BEGIN
WRITE(F,A);
WRITE(F,B);
WRITE(F,C)
END;
```

բաղադրյալ օպերատորի կատարմանը:

Տեքստային ֆայլերի համար ելքի ցուցակը կարող է բաղկացած լինել CHAR, STRING, ինչպես նաև ամբողջ կամ իրական տիպի մեկ կամ մի քանի փոփոխականներից: Տիպայնացված ֆայլերի համար ելքի ցուցակում նշված փոփոխականները և ֆայլի բաղադրիչները պետք է ունենան նույն տիպը:

WRITELN(f, <ելքի ցուցակ>)- պրոցեդուրան f ֆայլային փոփոխականի հետ կապված արդեն գոյություն ունեցող ֆայլում գրում է ելքի ցուցակում նշված փոփոխականների արժեքները միայն այն տարբերությամբ, որ գրվող ինֆորմացիան ավարտվում է տողավերջի նշանով:

8.6. Տիպայնացված ֆայլերի համար կիրառվող պրոցեդուրաներ և ֆունկցիաներ

ՊՐՈՑԵԴՈՒՐԱՆԵՐ

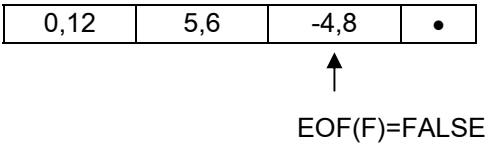
SEEK(f, n) պրոցեդուրան ֆայլային f փոփոխականի հետ կապված ֆայլի ընթացիկ տարրի ցուցիչը տեղադրում է ֆայլի n համարով բաղադրիչի վրա: Բաղադրիչների համարակալումը սկսվում է զրոյից:

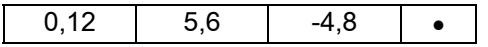
TRUNCATE(f) պրոցեդուրան ֆայլային f փոփոխականի հետ կապված ֆայլից հեռացնում է, ընթացիկ բաղադրիչից սկսած, մինչև վերջին բոլոր բաղադրիչները:

ՖՈՒՆԿՑԻԱՆԵՐ

EOF(f) ֆունկցիան ընդունում է TRUE արժեքը, եթե ընթացիկ տարրի ցուցիչը գտնվում է ֆայլի վերջին տարրից հետո (ցույց է տալիս ֆայլի վերջը), հակառակ դեպքում՝ FALSE:

Օրինակ.





EOF(F)=TRUE

որտեղ - ֆայլի ցուցիչն է, իսկ աղյուսակում • -ով ցույց է տրված ֆայլի վերջը:

Ակնհայտ է, որ եթե ֆայլը դատարկ է, ապա EOF ֆունկցիան կընդունի TRUE արժեք.

EOF(F)=TRUE

FILEPOS(f) ֆունկցիան վերադարձնում է f ֆայլային փոփոխականի հետ կապված ֆայլի այն ընթացիկ բաղադրիչի համարը, որը պետք է մշակվի մուտքի-ելքի հաջորդ գործողությամբ: Ֆայլի առաջին բաղադրիչն ունի 0 հերթական համարը:

FILESIZE(f) ֆունկցիան վերադարձնում է f ֆայլային փոփոխականի հետ կապված ֆայլի բաղադրիչների քանակը:

8.7. Ֆայլերի կիրառման օրինակներ

Օրինակ 8.1: C կուտակիչի հիմնային կատալոգում գրված է ամբողջ բաղադրիչներով DD1.DAT ֆայլը: Դիցուք F1 ֆայլային փոփոխականի հետ կապված DD1.DAT ֆայլում գրված են 10 ամբողջ թվեր:

Գրել ծրագիր, որն ամբողջ բաղադրիչներով ֆայլից կընթերցի առաջին 8 բաղադրիչների արժեքները և կհաշվի նրանց գումարը:

Խնդրի լուծման ծրագիրն ունի EXAM8_1 անունը:

```
1       PROGRAM EXAM8_1;
2       VAR F:FILE OF INTEGER;S,I,A:INTEGER;
3       BEGIN
```

```
4          ASSIGN(F,'C:\DD1.DAT');
5          RESET(F);
6          S:=0;
7          FOR I:=1 TO 8 DO
8          BEGIN
9          READ(F,A);
10        S:=S+A
11        END;
12      CLOSE(F);
13      WRITE('S=',S)
14      END.
```

Մուտք՝ **ԵԼՔ՝ S=24**

1	6	4	1	2	3	7	4	4	4	•

Տող 2-ում հայտարարվել է ամբողջ տիպի F ֆայլային փոփոխականը, որը տող 4-ում ASSIGN պրոցեդուրայով կապվել է տրված ֆայլի հետ: Տող 5-ում կարդալու համար բացվել է ֆայլը: Պահանջվող գումարի հաշվման համար օգտագործվել է S փոփոխականը, որին նախապես վերագրվել է 0 արժեք (տող 6): Կազմակերպվել է I պարամետրով ցիկլ, որի յուրաքանչյուր արժեքի համար ֆայլից ընթերցվում և A փոփոխականին է վերագրվում ֆայլի հերթական տարրի արժեքը (տող 9): Ֆայլից ընթերցված արժեքը գումարվում է գումարի հաշվման համար նախատեսված S փոփոխականին (տող 10): Տող 12-ում փակվել է բացված ֆայլը:

Օրինակ 8.2: C կուտակիչի հիմնային կատալոգի AA ենթակատալոգում ստեղծել DD1.DAT ֆայլը, որի մեջ գրված լինեն առաջին 12 բնական թվերը:

Խնդրի լուծման ծրագիրն ունի EXAM8_2 անունը:

```
1     PROGRAM EXAM8_2;
2     VAR F1:FILE OF INTEGER;
3     I:INTEGER;
4     BEGIN
5     ASSIGN(F1,'C:\AA\DD1.DAT');
```

```
6     REWRITE(F1);
7     FOR I:=1 TO 12 DO WRITE(F1,I);
8     CLOSE(F1);
9  END.
```

ԵԼՔ՝

1	2	3	4	5	6	7	8	9	10	11	12	•

Տող 2-ում հայտարարվել է ամբողջ տիպի ֆայլային F1 փոփոխականը, որը տող 5-ում կապվել է C կուտակիչի հիմնային կատալոգի AA ենթակատալոգի DD1.DAT ամբողջ բաղադրիչներով ֆայլի հետ: Տող 6-ում բացվել է F1 ֆայլային փոփոխականի հետ կապված DD1.DAT ֆայլը՝ տվյալների ներմուծման համար: Տող 7-ում բացվել է ցիկլ, որտեղ ցիկլի պարամետրի ընդունած 1-ից մինչև 12 արժեքները գրվել են ստեղծված ֆայլի մեջ: Տող 8-ում փակվել է ստեղծված ֆայլը:

Օրինակ 8.3: C: կուտակիչի հիմնային կատալոգի B1 ենթակատալոգի B2 ենթակատալոգում ստեղծված է n հատ ամբողջ տիպի բաղադրիչներով DD1.DAT ֆայլը: Նույն ֆայլում գրել կենտ արժեք ունեցող տարրերը:

Խնդրի լուծման ծրագիրն ունի EXAM8_3 անունը:

```
1     PROGRAM  EXAM8_3;
2     VAR F:FILE OF INTEGER;
3         I,N,INTEGER;
4         X:ARRAY[1..100] OF INTEGER;
5     BEGIN
6     ASSIGN(F,'C:\B1\B2\DD1.DAT');
7     READ(N);
8     RESET(F);
9     FOR I:=1 TO N DO READ(F,X[I]);
10    REWRITE(F);
11    FOR I:=1 TO N DO
12    IF X[I] MOD 2=1 THEN WRITE(F,X[I]);
13    CLOSE(F)
14    END.
```

ՄՈՒՏՔ՝ **N=10**

2	3	4	6	7	6	7	8	1	4	•

ԵԼՔ՝

3	7	7	1	•

Տող 2-ում հայտարարվել է ամբողջ տիպի F ֆայլային փոփոխականը: Տող 6-ում F ֆայլային փոփոխականը կապվել է C կուտակիչի հիմնային կատալոգի B1 ենթակատալոգի B2 ենթակատալոգի DD1.DAT ֆայլի հետ: Տող 7-ում ներմուծվել է ֆայլում բաղադրիչների քանակը (N): Տող 8-ում կարդալու համար բացվել է ֆայլը: Տող 9-ում ստեղծված ցիկլի մարմնում ֆայլից կարդացվել են ֆայլի բոլոր տարրերը և պահվել X ամբողջ տիպի զանգվածի մեջ: Տող 10-ում տվյալների ներմուծման համար բացվել է F ֆայլային փոփոխականի հետ կապված ֆայլը: Տողեր 11-ից 12-ում զանգվածի կենտ արժեք ունեցող տարրերը գրվել են նույն ֆայլի մեջ: Այն իրականացվել է պարամետրով ցիկլի մարմնում: Տող 13-ում փակվել է ֆայլը:

Օրինակ 8.4: C կուտակիչի հիմնային կատալոգի B1 ենթակատալոգի B2 ենթակատալոգում ստեղծված են n հատ իրական տիպի բաղադրիչներով DD1.DAT և DD2.DAT ֆայլերը: C կուտակիչի հիմնային կատալոգում ստեղծել նոր DD3.DAT ֆայլը, որում գրված լինեն DD1.DAT և DD2.DAT ֆայլերի դրական բաղադրիչները:

Խնդրի լուծման ծրագիրն ունի EXAM8_4 անունը:

```
1       PROGRAM  EXAM8_4;
2       VAR F1,F2,F3:FILE OF REAL;
3            I,N,INTEGER; A:REAL;
4       BEGIN
5       ASSIGN(F1,'C:\B1\B2\DD1.DAT');
6       ASSIGN(F2,'C:\B1\B2\DD2.DAT');
7       ASSIGN(F3,'C:\DD3.DAT');
8       READ(N);
9       RESET(F1);
10      RESET(F2);
11      REWRITE(F3);
12      FOR I:=1 TO N DO
13      BEGIN
```

```
14      READ(F1,A]); IF A>0 THEN WRITE(F3,A);
15      READ(F2,A]); IF A>0 THEN WRITE(F3,A);
16      END;
17      CLOSE(F1);
18      CLOSE(F2);
19      CLOSE(F3);
20      END.
```

Տող 2-ում հայտարարվել են իրական տիպի F1, F2, F3 ֆայլային փոփոխականները: Տող 5-ում F1 ֆայլային փոփոխականը կապվել է C կուտակիչի հիմնային կատալոգի B1 ենթակատալոգի B2 ենթակատալոգի DD1.DAT ֆայլի հետ, իսկ տող 6-ում F2 ֆայլային փոփոխականը՝ նույն կատալոգի DD2.DAT ֆայլի հետ: Տող 7-ում F3 ֆայլային փոփոխականը կապվել է C կուտակիչի հիմնային կատալոգի DD3.DAT ֆայլի հետ: Տող 8-ում ներմուծվել է ֆայլում բաղադրիչների քանակը (N): Տող 9-ում և 10-ում կարդալու համար բացվել են համապատասխանաբար DD1.DAT և DD2.DAT ֆայլերը: Տող 11-ում տվյալների ներմուծման համար բացվել է F ֆայլային փոփոխականի հետ կապված DD3.DAT ֆայլը: Տող 12-ում բացվել է պարամետրով ցիկլ, որի մարմնում ֆայլերից A փոփոխականի մեջ է բերվում հերթական տարրի արժեքը, և եթե այն դրական է, ապա գրվում է DD3.DAT ֆայլի մեջ: Տողեր 13-ից 19-ում փակվել են բոլոր ֆայլերը:

8.8. Տնային առաջադրանքներ

Տնային առաջադրանքում ուսանողը պետք է կատարի աղյուսակի դասամատյանի՝ իր համարին համապատասխան տողում նշված խնդիրները, որոնք բերված են բաժին 8-ում:

Մատյանի համարը	Առաջադրվող խնդիրների համարները
1	571, 581, 591
2	572, 586, 593
3	573, 587, 594
4	574, 584, 595
5	575, 590, 596
6	576, 585, 597
7	577, 589, 592
8	578, 583, 598

9	579, 588,599
10	580, 582, 600
11	571, 590, 593
12	572, 581, 594
13	573, 584, 591
14	574, 587, 595
15	575, 582, 596
16	576, 588, 597
17	577, 589, 592
18	578, 586, 598
19	579, 585, 599
20	580, 583, 600
21	571, 590, 592
22	572, 583, 591
23	573, 585, 595
24	574, 587, 593
25	575, 588, 596
26	576, 581, 597
27	577, 589, 598
28	578, 582, 599
29	579, 586, 600
30	580, 584, 594

Գրականություն

1. Աղգաշյան Ռ., Ավետիսյան Ս. PASCAL ծրագրավորում: ՀՊ�ճՀ, 2001թ.
2. Աղգաշյան Ռ., Առաքելյան Ա., Ավետիսյան Ս., Դանիելյան Ս. Քոմփյութերների կիրառում և ծրագրավորում / խնդիրների ժողովածու / : ՀՊՃՀ, 1998թ.
3. Առաքելյան Ա., Աղգաշյան Ռ., Ավետիսյան Ս., Դանիելյան Ս., Մանուկյան Լ. ՀՊՃՀ ընդհանուր կրթության պետական ամփոփիչ քննության քննահարցերի շտեմարանի խնդիրների լուծման ուղեցույց / էՀՄ և ծրագրավորում / : ՀՊՃՀ, 1996թ.
4. Աղգաշյան Ռ., Ղուկասյան Վ., Ծրագրավորում բոլորի համար. Մաս1, Ալգորիթմների կառուցում, ՀՊՃՀ, 2000թ.

Պատվեր՝ 636 Տպաքանակ՝ 100

Տպագրված է Հայաստանի Պետական Ճարտարագիտական Համալսարանի տպարանում

Երևան, Տերյան 105