***ՀՀ կրթության, գիտության, մշակույթի եվ սպորտի նախարարություն***

***Հայաստանի Ազգային պոլիտեխնիկական Համալսարան (Հիմնադրամ)***

***Ալգորիթմական լեզուների և ծրագրավորման ամբիոն***

*Մասնագիտություն* Տեղեկատվական տեխնոլոգիաներ

**ԿՈՒՐՍԱՅԻՆ ԱՇԽԱՏԱՆՔԻ ԱՌԱՋԱԴՐԱՆՔ**

Առարկա: ***Տվյալների կառուցվածքներ և ալգորիթմներ***

ՏՏ924-1 ակադ. խմբի ուսանող \_\_ \_Գալստյան Խորեն Կարենի\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*.*

(ուսանողի ազգանուն, անուն, հայրանուն).

1. Աշխատանքի թեման` Ստեղծել տվյալների ցուցակ C++ ծրագրի միջոցով

2. Հաշվեբացատրագրի բովանդակությունը

2.1. Խնդրի դրվածքը*.*

2.2. Ալգորիթմի նկարագրությունը

2.3. Ծրագրի կոդ

2.4 . Ծրագրի աշխատանքի արդյունք

3. Գրաֆիկական մասի ծավալը\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Կատարման ժամանակացույցը\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Աշխատանքի ղեկավար՝ \_\_\_\_\_Խաչատրյան Ա. Ռ.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ա.Հ.Ա., գիտական աստիճան, տարակարգ, ստորագրություն, ամսաթիվ)

6. Ամբիոնի վարիչ՝ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ս.Ս. Ավետիսյան

(ստորագրություն)

7. Ուսանող \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ամսաթիվ, ստորագրություն)

Երևան – 2021 թ.

***Հ կրթության, գիտության, մշակույթի եվ սպորտի նախարարություն***

***Հայատանի Ազգային Պոլիտեխնիկական Համալսարան (Հիմնադրամ)***

***Ալգորիթմական լեզուների և ծրագրավորման ամբիոն***

***ՀԱՇՎԵԲԱՑԱՏՐԱԳԻՐ***

***Տվյալների կառուցվածքներ և ալգորիթմներ***

առարկայի կուրսային աշխատանքի

Թեմա ՝

Ստեղծել երկկապ գծային ցուցակ, որի մեկ միավորն ունի կառուցվածք տիպի ինֆորմացիա:Կառուցվածքի դաշտերն են՝ ազգանուն,գնահատական,բացակայություն:

Պետք է կազմել ուսանողների միջին գնահատականը հաշվող ֆունկցիա,ինչպես նաև ֆունկցիա,որը կտպի այն ուսանողների բացակայությունները և ազգանունները,որոնք ունեն միջին գնահատականից ցածր գնահատական:

Ուսանող՝ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Գալստյան Խորեն Կարենի\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ազգանուն, անուն, հայրանուն, ամսաթիվ, ստորագրություն).

Ղեկավար՝ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Խաչատրյան \_\_\_Ա. Ռ.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ա.Հ.Ա., ստորագրություն, ամսաթիվ)

6. Ամբիոնի վարիչ՝ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ս.Ս. Ավետիսյան

(ամսաթիվ, ստորագրություն)

Երևան – 2021թ.

**ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ**

[**ՆԵՐԱԾՈՒԹՅՈՒՆ** 4](#_Toc70624686)

[**1** **ԱՌԱՋԱԴՐԱՆՔ** 5](#_Toc70624687)

[1.1 ԾՐԱԳՐԻ ՆԿԱՐԱԳՐՈՒԹՅՈՒՆ 6](#_Toc70624688)

[1.2 ՖՈՒՆԿՑԻԱՆԵՐԻ ՆԿԱՐԱԳՐՈՒԹՅՈՒՆ 10](#_Toc70624689)

[1.2.1 Ցուցակի երկարություն 10](#_Toc70624690)

[1.2.2 Տվյալների ներմուծում 10](#_Toc70624691)

[1.2.3 Փնտրվող ինֆորմացիայի ստացում 11](#_Toc70624692)

[1.2.4 Հանգույցի ավելացում ցուցակի սկզբում 13](#_Toc70624693)

[1.2.5 Հանգույցի ավելացում ցուցակի վերջում 14](#_Toc70624694)

[1.2.6 Ցուցակի հանգույցի ավելացում ցուցակի տրված տեղում 15](#_Toc70624695)

[1.2.7 Ցուցակի սկզբից ցուցակի հանգույցի ջնջում 17](#_Toc70624696)

[1.2.8 Ցուցակի վերջից ցուցակի հանգույցի ջնջում 17](#_Toc70624697)

[1.2.9 Ցուցակի տրված տեղից ցուցակի միավորի հեռացում 18](#_Toc70624698)

[1.2.10 Ցուցակի արտածում 20](#_Toc70624699)

[1.2.11 Ցուցակի կամայական երկու հանգույցների փոխատեղում 21](#_Toc70624700)

[1.2.12 Դեստրուկտոր 22](#_Toc70624701)

[1.2.13 Միջին գնահատաանից ցածր գնահատական ունեցողների տվյալներ 23](#_Toc70624702)

[1.3 ԾՐԱԳՐԻ ԿՈԴ 25](#_Toc70624703)

[1.4 ԾՐԱԳՐԻ ԱՇԽԱՏԱՆՔԻ ԱՐԴՅՈՒՆՔ 37](#_Toc70624704)

[**ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ** 55](#_Toc70624705)

[**ԳՐԱԿԱՆՈՒԹՅՈՒՆ** 56](#_Toc70624706)

# **ՆԵՐԱԾՈՒԹՅՈՒՆ**

Հաճախ մեծ ծրագրերում անհրաժեշտ է լինում օգտագործել այնպիսի տվյալներ, որոնց չափերը և կառուցվածքն անհրաժեշտ է փոփոխել աշխատանքի ընթացքում: Նման դեպքերում օգտագործվում են դինամիկ տվյալների կառուցվածքներ: Սրանց հիմնական բնութագրիչը այն է, որ ինֆորմացիան հիշողության (RAM) մեջ պահվում է ոչ թե ինդեքսավորված կողք կողքի, այլ միմյանց կցված սլաքների կամ ցուցիչների միջոցով։ Յուրաքանչյուր էլեմենտ իր մեջ պահում է ինֆորմացիա, թե որտեղ է գտնվում իրեն հարևան էլեմենտը կամ էլեմենտները։­

Կառուցվածքի յուրաքանչյուր տարր բաղկացած է երկու դաշտից.

* Ինֆորմացիոն դաշտ կամ տվյալների դաշտ, որում պարունակվում են այն տվյալները, որոնց համար ստեղծվել է կառուցվածքը,
* Կապերի դաշտ, որում պարունակվում են մեկ կամ մի քանի ցուցիչներ, որոնց միջոցով տվյալ տարրը կապված է կառուցվածքի այլ տարրերի հետ:

Տվյալների կառուցվածքների տարածված տիպերից են կապակցված ցուցակները, որոնց տարատեսակներից է երկկապ գծային ցուցակը:

Այս աշխատանքի նպատակն է երկկապ գծային ցուցակների ստեղծումը և մշակումը:

Աշխատանքի արդյունքում երկկապ գծային ցուցակների ուսումնասիրության համար C++ ծրագրավորման լեզվով գրվել է ծրագիր: Ծրագրում ցուցակը ներկայացվում է գրառումների (struct) միջոցով: Ցուցակի հետ աշխատանքի համար ստեղծվել են մեթոդներ, որոնք իրականացնում են ցուցակի հանգույցի ավելացում, ցուցակի հանգույցի հեռացում, ցուցակի երկու կամայական հանգույցի փոխատեղում, ցուցակի արտածում, ցուցակում փնտրվող հանգույցի մասին ինֆորմացիայի ստացում, ցուցակից ըստ տրված պահանջի տվյալների արտածում:

# **ԱՌԱՋԱԴՐԱՆՔ**

Ստեղծել երկկապ գծային ցուցակ, որի մեկ միավորը ունի գրառում տիպ, որի դաշտերն են ուսանողի

* Ազգանուն
* Բացակայություն
* Գնահատական

Ցուցակի համար սահմանել հետևյալ ֆունկցիաները՝

1. Ստեղծել ցուցակը տպող ֆունկցիա:
2. Ստեղծել ցուցակի սկզբում ցուցակի միավոր ավելացնող ֆունկցիա:
3. Ստեղծել ցուցակի վերջում ցուցակի միավոր ավելացնող ֆունկցիա:
4. Ստեղծել ցուցակի տրված տեղում ցուցակի միավոր ավելացնող ֆունկցիա:
5. Ստեղծել ցուցակի սկզբից ցուցակի միավոր հեռացնող ֆունկցիա:
6. Ստեղծել ցուցակի վերջից ցուցակի միավոր հեռացնող ֆունկցիա:
7. Ստեղծել ցուցակի տրված տեղից ցուցակի միավոր հեռացնող ֆունկցիա:
8. Ստեղծել ցուցակի երկու կամայական միավոր փոխանակող ֆունկցիա:
9. Ստեղծել փնտրվող միավորի մասին ինֆորմացիա վերադարձնող ֆունկցիա:
10. Ստեղծել ցուցակի ուսանողների գնահատականների միջինը հաշվող ֆունկցիա:
11. Ստեղծել ֆունկցիա, որը կարտածի այն ուսանողների ազգանունները և բացակայությունները, որոնց գնահատականը փոքր է ցուցակի միջին գնահատականից:

## ԾՐԱԳՐԻ ՆԿԱՐԱԳՐՈՒԹՅՈՒՆ

Երկկապ գծային ցուցակի յուրաքանչյուր հանգույց ունի երկու ցուցչային դաշտ, որոնցից մեկը (prev) պահում է հանգույցի նախորդող հանգույցը իսկ մյուսում (next)՝ հաջորդող հանգույցը:

Ցուցակի առաջին հանգույցի prev ցուցիչն ունի NULL արժեք: Ցուցակի վերջին հանգույցի next ցուցիչւ նույնպես ունի NULL արժեք:

Ցուցակի յուրաքանչյուր հանգույց ներկայացվում է գրառման (struct) տեսքով:

struct node

{

usanox x;

node\* next;

node\* prev;

};

usanox-ը ինֆորմացիոն դաշտի տիպն է, որը նույնպես ներկայացված է գրառման տեսքով:

struct usanox

{

char azganun[20];

int gnahatakan;

int bacakayutyun;

};

Այստեղ char azganun[20]-ը ուսանողի ազգանունի դաշտն է, 20 թիվը ցույց է տալիս, որ ազգանունը պետք է պարունակի մինչև 20 սիմվոլ:

Int gnahatakan-ը ուսանողի գնահատականների դաշտն է, այն պետք է գտնվի [0; 100] միջակայքում:

Int bacakayutyun-ը ուսանողի բացակայությունների քանակն է, որը չպետք է լինի բացասական:

class list

{

node\* head;

public:list() :head(NULL) {}

void add\_last();

void add\_first();

void add\_med();

void del\_first();

void del\_med();

void del\_last();

void show();

int length();

void info();

void swap();

double average();

void newlist\_average();

usanox read();

~list();//

};

Ծրագիրը գրված է դասի (class) օգտագործմամբ։ Դասի փակ (private) անդամն է head-ը իսկ բաց (public) անդամներն են ծրագրի հիմնական ֆունկցիաները, ինչպես նաև կոնստրուկտորը և դեստրուկտորը։

int main()

{

int l;

list ob;

do {

cout << endl;

cout << " ՄԵՆՅՈՒ " << endl;

cout << "1.Ցուցակի տպում" << endl;

cout << "2.Ցուցակի սկզբից ցուցակի միավորի ավելացում" << endl;

cout << "3.Ցուցակի վերջից ցուցակի միավորի ավելացում" << endl;

cout << "4.Ցուցակի տրված տեղում ցուցակի միավորի ավելացում" << endl;

cout << "5.Ցուցակի սկզբից ցուցակի միավորի հեռացում" << endl;

cout << "6.Ցուցակի վերջից ցուցակի միավորի հեռացում" << endl;

cout << "7.Ցուցակի տրված տեղից ցուցակի միավորի հեռացում" << endl;

cout << "8.Ցուցակի երկու կամայական միավորի փոխանակում" << endl;

cout << "9.Միավորի փնտրում" << endl;

cout << "10.Ուսանողների միջին գնահատական" << endl;

cout << "11.Միջինից ցածր գնահատական ունեցողների տվյալ" << endl;

cout << "12.Ելք" << endl;

{

cin >> l;

switch (l)

{

case 1:

{

ob.show();

break;

}

case 2:

{

ob.add\_first();

break;

}

case 3:

{

ob.add\_last();

break;

}

case 4:

{

ob.add\_med();

break;

}

case 5:

{

ob.del\_first();

break;

}

case 6:

{

ob.del\_last();

break;

}

case 7:

{

ob.del\_med();

break;

}

case 8:

{

ob.swap();

break;

}

case 9:

{

ob.info();

break;

}

case 10:

{

ob.average();

break;

}

case 11:

{

ob.newlist\_average();

break;

}

case 12:

{

cout << "Ավարտ";

break;

}

default:

{

cout << "Սխալ է ընտրվել" << endl;

break;

}

}

}

} while (l != 12);

return 0;

}

Գլխավոր (main) ֆունկցիայում ստեղծվել է դասի օբյեկտը: Ամբողջ աշխատանքը կատարվում է ընտրության (switch) օպերատորի միջոցով: Գլխավոր (main) ֆունկցիան կատարվում է այնքան ժամանակ, քանի դեռ չենք ընտրել մենյուի 12 կետը: Համապատասխան թվերի ներմուծումով կատարվում են մեթոդները: Եթե ընտրել ենք այնպիսի կետ, որը մենյուում չունենք, այսինքն ներմուծված թիվը փոքր է մեկից կամ մեծ 12-ից, ապա արտածվում է սխալ է ընտրվել հաղորդագրությունը:

## ՖՈՒՆԿՑԻԱՆԵՐԻ ՆԿԱՐԱԳՐՈՒԹՅՈՒՆ

### Ցուցակի երկարություն

int length()

{

node\* temp = head;

int l = 0;

while (temp != NULL)

{

l++;

temp = temp->next;

}

return l;

}

Այս մեթոդի միջոցով կատարվում է ցուցակում առկա հանգույցների քանակի հաշվում: Կատավում է հետևյալ կերպ՝ քանի դեռ չենք հասել ցուցակի վերջը կատարվում է հաշվիչի արժեքի ավելացում մեկով և անցում հաջորդ հանգույցին: Հաշվիչի արժեքը կլինի ցուցակի երկարությունը: Անհրաժեշտ է այլ ֆունկցիաներում ճշտությունն ապահովելու համար:

### Տվյալների ներմուծում

usanox read()

{

usanox y;

cout << "Ազգանուն:";

cin >> y.azganun;

do

{

cout << "Բացակայություն:";

cin >> y.bacakayutyun;

} while (y.bacakayutyun < 0);

do

{

cout << "Գնահատական:";

cin >> y.gnahatakan;

} while (y.gnahatakan < 0 || y.gnahatakan>100);

return y;

}

Այս մեթոդը կատարում է հանգույցի ինֆորմացիայի ներմուծում: Այստեղ պաշտպանվում է դաշտերի ինֆորմացիայի ճշտությունը, այսինքն պետք է գնահատականը լինի [0; 100] միջակայքի թիվ, իսկ բացակայությունների քանակը չպետք է լինի բացասական:

### Փնտրվող ինֆորմացիայի ստացում

void info()

{

node\* temp = head;

if (temp != NULL)

{

int l;

char pnt[50]; int gnah, bacaka;

do

{

cout << "1.Փնտրում ըստ ազգանվան" << endl;

cout << "2.Փնտրում ըստ գնահատականի" << endl;

cout << "3.Փնտրում ըստ բացակայության" << endl;

cout << "4.Ելք" << endl;

cin >> l;

switch (l)

{

case 1:

{

int qanak = 0;

cout << "Փնտրվող ազգանուն:"; cin >> pnt;

temp = head;

while (temp != NULL)

{

if (strcmp(temp->x.azganun, pnt) == 0)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak++;

}

temp = temp->next;

}

if (qanak == 0) cout << "Այդպիսի մարդ չկա" << endl;

break;

}

case 2:

{

int qanak1 = 0;

temp = head;

cout << "Փնտրվող գնահատական:"; cin >> gnah;

while (temp != NULL)

{

if (temp->x.gnahatakan == gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak1++;

}

temp = temp->next;

}

if (qanak1 == 0) cout << "Այդպիսի մարդ չկա" << endl; break;

}

case 3:

{

int qanak2 = 0;

temp = head;

cout << "Փնտրվող բացակայություն: "; cin >> bacaka;

while (temp != NULL)

{

if (temp->x.bacakayutyun == bacaka)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak2++;

}

temp = temp->next;

}

if (qanak2 == 0) cout << "Այդպիսի մարդ չկա" << endl; break;

}

case 4:

{ cout << "Փնտրումն ավարտվեց" << endl;

return;

}

default:

{

cout << "Սխալ հրաման" << endl; break;

}

}

} while (l != 4);

}

else

cout << "Դատարկ է" << endl;

}

Այս ֆունկցիայի միջոցով ստացվում է ինֆորմացիայի արտածում ըստ փնտրվող ազգանվան, գնահատակի կամ բացակայությունների: Ֆունկցիան առաջարկում է ընտրել, թե ըստ որ դաշտի ենք ուզում կատարել փնտրում, ընտրելուց հետո պետք է ներմուծել փնտրվելիք արժեքը: Եթե փնտրվող արժեքով ուսանող (ուսանողներ) ունենք, ապա կսստանանք դրանց տվյալները, հակառակ դեպքում, եթե չունենք փնտրվող տվյալով ուսանող, կտպվի հաղորդագրություն, որ այդպիսի ուսանող չունենք:

### Հանգույցի ավելացում ցուցակի սկզբում

void add\_first()

{

node\* temp = new node;

temp->x = read();

temp->prev = NULL;

if (head != NULL)

{

temp->next = head;

head->prev = temp;

head = temp;

}

else

{

temp->next = NULL;

head = temp;

}

}

Այս ֆունկցիան ավելացնում է միավոր ցուցակի սկզբում: Կատարվում է ուսանողի տվյալների ներմուծումը և իրեն նախորդող հանգույցի ցուցչի դաշտում պահվում է NULL: Այնուհետև ստուգվում ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ չէ, ապա next ցուցչում, որում պահվում է իրեն հաջորդող հանգույցի հասցեն, պահվում է ցուցակի առաջին տարրը (head), իսկ ցուցակի առաջին տարրի prev ցուցչում պահվում է ավելացվող հանգույցը: Այնուհետև ցուցակի առաջին տարրը դառնում է ավելացված հանգույցը: Իսկ եթե ցուցակը դատարկ չէ, ապա ավելացվող հանգույցը լինում է առաջինը, նշանակում է, որ հանգույցի next ցուցչում պետք է պահվի NULL:

### Հանգույցի ավելացում ցուցակի վերջում

void add\_last()

{

node\* temp = new node;

node\* temp1 = head;

node\* temp2 = head;

temp->x = read();

temp->next = NULL;

if (temp2 != NULL)

{

while (temp1 != NULL)

{

temp1 = temp1->next;

}

temp->prev = temp1;

temp1 = head;

while (temp1->next != NULL)

{

temp1 = temp1->next;

}

temp1->next = temp;

}

else

{

temp->prev = NULL;

head = temp;

}

}

Այս ֆունկցիան կատարում է հանգույցի ավելացում ցուցակի վերջում: Սկզբում կատարվում ինֆորմացիոն դաշտի տվյալների ներմուծումը և ավելացվող հանգույցի next ցուցչում, որում պահվում է իրեն հաջորդող հանգույցը, պահվում NULL: Սրանից հետո ստուգվում է ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ չէ, ապա նախապայմանով ցիկլի միջոցով հասնում ենք ցուցակի վերջին տարրին: Այժմ գտնվում ենք ցուցակի վերջին տարրի վրա: Ավելացվող հանգույցի prev ցուցչում, որը իրեն նախորդող հանգույցն է պահում, պահվում է ցուցակի վերջին հանգույցը: Դրանից հետո նախապայմանով ցիկլի միջոցով հասնում ենք ցուցակի վերջը: Վերջին հանգույցի next ցուցչում պահվում է ավելացվող հանգույցը: Հակառակ դեպքում, եթե ցուցակը դատարկ է, ավելացվող հանգույցը կլինի առաջինը, որը նշանակում է, որ prev ցուցչում նույնպես պետք է պահվի NULL:

### Ցուցակի հանգույցի ավելացում ցուցակի տրված տեղում

void add\_med()

{

node\* head1 = head;

node\* head2 = head;

if (head != NULL)

{

int hasce;

do

{

cout << "Հասցե=";

cin >> hasce;

} while (hasce<1 || hasce>length() + 1);

if (hasce == 1)

{

add\_first();

return;

}

if (hasce == length() + 1)

{

add\_last();

return;

}

{

for (int i = 1; i < hasce - 1; i++)

head1 = head1->next;

node\* temp = new node;

node\* head2 = head1->next;

head1->next = temp;

temp->prev = head1;

temp->next = head2;

head2->prev = temp;

temp->x = read();

return;

}

}

else

{

node\* temp = new node;

temp->next = NULL;

temp->x = read();

head = temp;

}

{

return;

}

}

Այս ֆունկցիայի միջոցով կատարվում է հանգույցի ավելացում ցուցակի տրված դիրքում:Նախ ստուգվում է ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ չէ, ապա պետք է նորմուծենք այն դիրքը, որում ուզում ենք ավելացնել հանգույցը: Պարզ է որ տրված դիրքը պետք է լինի ցուցակի սահմաններում, այսինքն դիրքը պետք է լինի [1;երկարություն+1] սահմաններում: Եթե դիրքը հավասար է մեկի, ապա կատարվում է ցուցակի սկզբից ցուցակի հանգույցի ավելացման մեթոդը : Եթե դիրքը հավասար է լինում երկարություն+1 (length()+1), ապա կատարվում է ցուցակի վերջից ցուցակի միավոր ավելացնող մեթոդը: Եթե դիրքը հավասար չէ ոչ մեկի, ոչ երկարություն+1 (length() + 1), ապա տրված դիրքում հանգույցը ավելացվում է հետևյալ կերպ՝ պարամետրով ցիկլի միջոցով հասնում ենք տրված դիրքից մի հատ հետ, որից հետո հանգույցի տիպի փոփոխականի մեջ (head2) պահվում է տրված դիրքից մեկ քայլով հետ գտնվող հանգույցի հաջորդ հանգույցը : Այսինքն ավելացվող հանգույցը պետք է լինի այս երկուսի արանքում: Ավելացվող հանգույցի prev ցուցչում, որում պահվում է հանգույցին նախորդող հանգույցը, պահվում է պարամետրով ցիկլի միջոցով մեր հասած հանգույցը (head1), իսկ այդ head1 հանգույցի next ցուցչում պահվում է ավելացվող հանգույցը (temp): Ավելացվող հանգույցի (temp) next ցուցչում պահվում է head2-ը, իսկ head2-ի prev ցուցչում պահվում է ավելացվող հանգույցը: Եթե ցուցակը դատարկ է լինում, ապա ավելացվող հանգույցը լինում է առաջինը, այսինքն դրա next և prev ցուցիչներում պահվում է NULL:

### Ցուցակի սկզբից ցուցակի հանգույցի ջնջում

void del\_first()

{

int erk = length();

node\* temp = head;

if (temp != NULL)

{

if (erk == 1)

{

cout << "Ցուցակը դատարկվեց" << endl;

head = NULL;

delete head;

return;

}

temp = head->next;

temp->prev = NULL;

delete head;

head = temp;

}

else cout << "Դատարկից ջնջել չի կարելի" << endl;

}

Այս մեթոդը ջնջում է հանգույց ցուցակի սկզբից: Նախ ստուգվում է ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ է, ապա արտածվում է դատարկից ջնջել չի կարելի հաղորդագրությունը: Եթե ցուցակը դատարկ չէ, ստուգում ենք թե ցուցակում 1 հանգույց ունենք թե՞ ոչ: Եթե ցուցակում 1 տարր կա, որը head-ն է, ապա դրան վերագրվում է NULL և այն դինամիկ հիշողությունից ջնջվում է, որից հետո արտածվում է հաղորդագրություն, որ ցուցակը դատարկվել է: Եթե ցուցակում ունենց մեկից ավելի հանգույցներ, ապա հանգույց տիպի ցուցչում պահվում է ցուցակի երկրորդ հանգույցը: Երկրորդ հանգույցի prev ցուցչում պահվում է NULL, իսկ head-ը ջնջվում է դինամիկ հիշողությունից: Պարզ է, որ առաջին հանգույցը հեռացնելուց հետո մեր առաջին հանգուցը պետք է լինի երկրորդը (temp), դրա համար head-ին վերագրվել temp-ը:

### Ցուցակի վերջից ցուցակի հանգույցի ջնջում

void del\_last()

{

int l = 0;

node\* temp = head;

node\* temp1 = head;

if (head != NULL) {

if (length() == 1)

{

cout << "Ցուցակը դատարկվեց" << endl;

head = NULL;

delete head;

return;

}

do

{

temp1 = temp1->next;

l++;

} while (temp1->next != NULL);

for (int i = 1; i < l; i++)

temp = temp->next;

temp->next = NULL;

delete temp->next;

}

else

{

cout << "Դատարկից ջնջել չի կարելի" << endl; return;

}

}

Այս մեթոդը կատարում է հանգույցի ջնջում ցուցակի վերջից: Նախ ստուգում ենք արդյոք ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ է, արտածվում է հաղորդագրություն, որ դատարկ ցուցակից ջնջել չի կարելի: Եթե ցուցակը դատարկ չէ, ստուգվում է թե ցուցակում 1 հանգույց կա (length()==1) թե՞ ոչ: Եթե ցուցակը 1 հանգույց է պարունակում, որը head-ն է, ապա նրանում պահվում է NULL և դինամիկ հիշողությունից հեռացվում է, որից հետո արտածվում է հաղորդագրություն, որը ցուցակը դատարկվել է: Եթե ցուցակում կան մեկից ավելի հանգույցներ, ապա պարամետրով ցիկլի միջոցով հասնում ենք ցուցակի նախավերջին հանգույցի վրա: Նրա next ցուցչում պահում ենք NULL և դինամիկ հիշողությունից ջնջում ենք վերջին հանգույցը:

### Ցուցակի տրված տեղից ցուցակի միավորի հեռացում

void del\_med()

{

int tex;

if (head != NULL)

{

if (length() == 1)

{

head = NULL;

delete head;

cout << "Ցուցակը դատարկվեց" << endl; return;

}

do

{

cout << "Դիրք=";

cin >> tex;

} while (tex<1 || tex>length());

node\* temp = head;

node\* temp1 = head;

node\* temp2 = head;

if (tex == 1)

{

del\_first();

return;

}

if (tex == length())

{

del\_last();

return;

}

{

for (int i = 1; i < tex; i++)

temp = temp->next;

for (int i = 1; i < tex - 1; i++)

temp1 = temp1->next;

for (int i = 1; i < tex + 1; i++)

temp2 = temp2->next;

}

temp1->next = temp2;

temp2->prev = temp1;

delete temp;

}

else cout << "Դատարկից ջնջել չի կարելի" << endl;

}

Այս մեթոդով կատարվում է տրված դիրքից ցուցակի հանգույցի ջնջում: Նախ ստուգվում ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ է, արտածվում է հաղորդագրություն այդ մասին: Հակառակ դեպքում, եթե ցուցակում կա հանգույց, ստուգվում է արդյոք ունենք միակ հանգույց (length()==1) թե՞ ոչ: Եթե կա միայն մեկ հանգույց, որը head-ն է, նրան տալիս ենք NULL արժեք, դինամիկ հիշողությունից ջնջում և արտածում, որ ցուցակը դատարկվեց: Եթե ցուցակում կան մեկից ավելի հանգույցներ, ապա ներմուծում ենք այն հանգույցի դիրքը (պահելով այն ցուցակի հանգույցների քանակի սահմանում) որը պետք է հեռացվի: Ստուգում ենք թե արդյո՞ք մեր տված դիրքը հավասար է 1-ի, եթե այո, ապա կատարվում է ցուցակի սկզբից հանգույցի հեռացում կատարող մեթոդը: Այնուհետև ստուգում ենք թե արդյո՞ք այդ դիրքը հավասար է մեր հանգույցների քանակին, եթե այո, ապա կատարվում է ցուցակի վերջից հանգույց հեռացնող մեթոդը: Եթե դիրքը սրանցից ոչ մեկին չի համապատասխանում, ապա տրված դիրքից հանգույցի հեռացումը կատարվում է հետևյալ կերպ՝ պարամետրով ցիկլի միջոցով temp-ը հասցնում ենք մեր ուզած դիրքին, temp1-ը մեր ուզած դիրքից մեկով հետ իսկ temp2-ը մեր ընտրած դիրքից մեկով առաջ: Այնուհետև temp1-ի և temp2-ի կապերը temp-ի հետ խզում ենք: Դրա համար temp1-ի next ցուցչում պահվում է temp2-ը, իսկ temp2-ի prev ցուցչում temp1-ը: Սրանից հետո temp-ը ջնջվում է դինամիկ հիշողությունից:

### Ցուցակի արտածում

void show()

{

node\* temp = head;

if (temp != NULL) {

while (temp != NULL)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

temp = temp->next;

}

}

else cout << "Դատարկ է" << endl;

}

Այս մեթոդը կատարում է ցուցակի հանգույցների արտածում: Նախ ստուգում ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ չէ, ապա նախապայմանով ցիկլի միջոցով կատարվում է հանգույցի ինֆորմացիոն դաշտի արտածումը և անցումը հաջորդ հանգույցին, քանի դեռ չենք հասել ցուցակի վերջը: Հակառակ դեպքում, եթե ցուցակը դատարկ է, արտածվում է դատարկ է հաղորդագրությունը:

### Ցուցակի կամայական երկու հանգույցների փոխատեղում

void swap()

{

int l, k;

node\* temp = head;

node\* temp1 = head;

if (length() >= 2)

{

do

{

cout << "l=";

cin >> l;

cout << "k=";

cin >> k;

} while (l<1 || l>length() || k<1 || k>length());

if (l < k)

{

for (int i = 1; i < l; i++)

temp = temp->next;

for (int i = 1; i < k; i++)

temp1 = temp1->next;

usanox t;

t = temp->x;

temp->x = temp1->x;

temp1->x = t;

}

else

{

for (int i = 1; i < k; i++)

temp = temp->next;

for (int i = 1; i < l; i++)

temp1 = temp1->next;

usanox t;

t = temp->x;

temp->x = temp1->x;

temp1->x = t;

}

if (l == k) cout << "Բան չփոխվեց" << endl;

}

else cout << "Բավական տարրեր չկան";

}

Այս մեթոդը կատարում է կամայական երկու հանգույցների փոխատեղում: Նախ ստուգվում է ցուցակում գոնե երկու տարր լինելու փաստը: Եթե ցուցակում կա երկուսից քիչ տարր, ապա արտածվում է, որ բավարար տարրեր չկան, իսկ եթե երկու կամ ավելի տարրեր կան, ապա պետք է ներմուծել փոխատեղվող տարրերի դիրքերը: Պարզ է, որ այդ դիրքերը պետք է լինեն ցուցակում առկա տարրերի քանակի սահմաններում, այսինքն պետք է լինեն զրոյից մեծ և ցուցակի երկարությունից փոքր կամ դրան հավասար: Եթե այդ դիրքերը իրար հավասար են եղել, ապա արտածվում է, որ բան չփոխվեց, հակառակ դեպքում կատարվում է տարրերի փոխատեղումը հետևյալ կերպ՝ պարամետրով ցիկլի միջոցով temp փոփոխականը հասցնում ենք մեր տրված առաջին դիրքին իսկ temp1 փոփոխականը մեր տված երկրորդ դիրքին: Այնուհետև մեր ինֆորմացիոն դաշտի տիպի փոփոխականում (t) պահում ենք առաջին դիրքի հանգույցի ինֆորմացիոն տվյալները: Առաջին դիրքի հանգույցի ինֆորմացիոն դաշտում պահում ենք երկրորդ դիրքի հանգույցի ինֆորմացիոն տվյալները: Իսկ երկորդ դիրքի հանգույցի ինֆորմացիոն դաշտում պահում ենք t-ում պահված տվյալները:

### Դեստրուկտոր

~list()

{

while (head != NULL)

{

node\* temp = head->next;

delete head;

head = temp;

}

};

Այս մեթոդով կատարվում է դինամիկ հիշողությունից տարրերի հեռացումը, քանի դեռ տարրեր կան ցուցակում, ինչպես նաև ստեղծած օբյեկտի ոչնչացումը: Այս մեթոդն ավտմոտ կանչվում է, երբ ոչնչանում է օբյեկտը: Այս դեպքում օբյեկտը ոչնչանում է, երբ ավարտվում է գլխավոր (main) ֆունկցիայի աշխատանքը:

**Ուսանողների միջին գնահատականի հաշվում**

double average()

{

node\* temp = head;

double mij = 0;

int erk = length();

if (temp != NULL)

{

while (temp->next != NULL)

{

mij += temp->x.gnahatakan;

temp = temp->next;

}

mij += temp->x.gnahatakan;

cout << "Միջին գնահատական = " << mij / erk << endl;

return mij / erk;

}

else

{

cout << "Դատարկ է" << endl;

return NULL;

}

}

Այս մեթոդով կատարվում է ցուցակում առրկա ուսանողների միջին գնահատականի հաշվում: Նախ ստուգվում է ցուցակը դատարկ է թե՞ ոչ: Եթե ցուցակը դատարկ է, արտածվում է դատարկ է հաղորդագրությունը: Եթե ցուցակում ունենք հանգույց (հանգույցներ), ապա բոլոր հանգույցների ինֆորմացիոն դաշտի գնահատականները գումարվում են իրար և պահվում double տիպի mij փոփոխականում: Այժմ միջին գնահատականը գտնելու համար պետք է ստացված գումարը բաժանել հանգույցների քանակին: Քանի որ ստացված քանորդը double տիպի է, ապա այս մեթոդը նույնպես պետք է լինի double տիպի:

### Միջին գնահատաանից ցածր գնահատական ունեցողների տվյալներ

void newlist\_average()

{

{

node\* temp = head;

int qanak = 0;

double gnah = average();

if (temp != NULL)

{

while (temp->next != NULL)

{

if (temp->x.gnahatakan < gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << endl;

qanak++;

}

temp = temp->next;

}

if (temp->x.gnahatakan < gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << endl;

qanak++;

}

if (qanak == 0)

{

cout << "Այդպիսիք չկան" << endl;

}

}

}

}

Այս մեթոդը արտածում է այն ուսանողների ազգանունները և բացակայությունները, ովքեր ստացել են միջին գնահատականից ցածր գնահատական: Նախապայմանով ցիկլի միջոցով հերթով ստուգում ենք բոլոր ուսանողների գնահատականները միջին գնահատականի հետ: Եթե ընթացիկ ուսանողի գնահատականը փոքր է միջին գնահատականից, ապա արտածվում է այդ ուսանողի ազգանունը և բացակայությունները և հաշվիչի արժեքը մեկով ավելացվում է: Վերջում ստուգում ենք հաշվիչի արժեքը, եթե այն հավասար է զրոյի, ուրեմն միջին գնահատականից ցածր գնահատական ստացած ուսանող չունենք, արտածվում է հաղորդագրություն այդ մասին:

## ԾՐԱԳՐԻ ԿՈԴ

#include <iostream>

#include <stdlib.h>

using namespace std;

struct usanox

{

char azganun[20];

int gnahatakan;

int bacakayutyun;

};

struct node

{

usanox x;

node\* next;

node\* prev;

};

class list

{

node\* head;

public:list() :head(NULL) {}

void add\_last();

void add\_first();

void add\_med();

void del\_first();

void del\_med();

void del\_last();

void show();

int length();

void info();

void swap();

double average();

void newlist\_average();

usanox read();

~list();

};

int list::length()

{

node\* temp = head;

int l = 0;

while (temp != NULL)

{

l++;

temp = temp->next;

}

return l;

}

usanox list::read()

{

usanox y;

cout << "Ազգանուն:";

cin >> y.azganun;

do

{

cout << "Բացակայություն:";

cin >> y.bacakayutyun;

} while (y.bacakayutyun < 0);

do

{

cout << "Գնահատական:";

cin >> y.gnahatakan;

} while (y.gnahatakan < 0 || y.gnahatakan>100);

return y;

}

double list::average()

{

node\* temp = head;

double mij = 0;

int erk = length();

if (temp != NULL)

{

while (temp->next != NULL)

{

mij += temp->x.gnahatakan;

temp = temp->next;

}

mij += temp->x.gnahatakan;

cout << "Միջին գնահատական = " << mij / erk << endl;

return mij / erk;

}

else

{

cout << "Դատարկ է" << endl;

return NULL;

}

}

void list::newlist\_average()

{

{

node\* temp = head;

int qanak = 0;

double gnah = average();

if (temp != NULL)

{

while (temp->next != NULL)

{

if (temp->x.gnahatakan < gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << endl;

qanak++;

}

temp = temp->next;

}

if (temp->x.gnahatakan < gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << endl;

qanak++;

}

if (qanak == 0)

{

cout << "Այդպիսիք չկան" << endl;

}

}

}

}

void list::info()

{

node\* temp = head;

if (temp != NULL)

{

int l;

char pnt[50]; int gnah, bacaka;

do

{

cout << "1.Փնտրում ըստ ազգանվան" << endl;

cout << "2.Փնտրում ըստ գնահատականի" << endl;

cout << "3.Փնտրում ըստ բացակայության" << endl;

cout << "4.Ելք" << endl;

cin >> l;

switch (l)

{

case 1:

{

int qanak = 0;

cout << "Փնտրվող ազգանուն:"; cin >> pnt;

temp = head;

while (temp != NULL)

{

if (strcmp(temp->x.azganun, pnt) == 0)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak++;

}

temp = temp->next;

}

if (qanak == 0) cout << "Այդպիսի մարդ չկա" << endl;

break;

}

case 2:

{

int qanak1 = 0;

temp = head;

do

{

cout << "Փնտրվող գնահատական:"; cin >> gnah;

} while (gnah < 0 || gnah>100);

while (temp != NULL)

{

if (temp->x.gnahatakan == gnah)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak1++;

}

temp = temp->next;

}

if (qanak1 == 0) cout << "Այդպիսի մարդ չկա" << endl; break;

}

case 3:

{

int qanak2 = 0;

temp = head;

cout << "Փնտրվող բացակայություն: "; cin >> bacaka;

while (temp != NULL)

{

if (temp->x.bacakayutyun == bacaka)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

qanak2++;

}

temp = temp->next;

}

if (qanak2 == 0) cout << "Այդպիսի մարդ չկա" << endl; break;

}

case 4:

{ cout << "Փնտրումն ավարտվեց" << endl;

return;

}

default:

{

cout << "Սխալ հրաման" << endl; break;

}

}

} while (l != 4);

}

else

cout << "Դատարկ է" << endl;

}

void list::add\_first()

{

node\* temp = new node;

temp->x = read();

temp->prev = NULL;

if (head != NULL)

{

temp->next = head;

head->prev = temp;

head = temp;

}

else

{

temp->next = NULL;

head = temp;

}

}

void list::add\_last()

{

node\* temp = new node;

node\* temp1 = head;

node\* temp2 = head;

temp->x = read();

temp->next = NULL;

if (temp2 != NULL)

{

while (temp1 != NULL)

{

temp1 = temp1->next;

}

temp->prev = temp1;

temp1 = head;

while (temp1->next != NULL)

{

temp1 = temp1->next;

}

temp1->next = temp;

}

else

{

temp->prev = NULL;

head = temp;

}

}

void list::add\_med()

{

node\* head1 = head;

node\* head2 = head;

if (head != NULL)

{

int hasce;

do

{

cout << "Հասցե=";

cin >> hasce;

} while (hasce<1 || hasce>length() + 1);

if (hasce == 1)

{

add\_first();

return;

}

if (hasce == length() + 1)

{

add\_last();

return;

}

{

for (int i = 1; i < hasce - 1; i++)

head1 = head1->next;

node\* temp = new node;

node\* head2 = head1->next;

head1->next = temp;

temp->prev = head1;

temp->next = head2;

head2->prev = temp;

temp->x = read();

return;

}

}

else

{

node\* temp = new node;

temp->next = NULL;

temp->x = read();

head = temp;

}

{

return;

}

}

void list::del\_first()

{

int erk = length();

node\* temp = head;

if (temp != NULL)

{

if (erk == 1)

{

cout << "Ցուցակը դատարկվեց" << endl;

head = NULL;

delete head;

return;

}

temp = head->next;

temp->prev = NULL;

delete head;

head = temp;

}

else cout << "Դատարկից ջնջել չի կարելի" << endl;

}

void list::del\_last()

{

int l = 0;

node\* temp = head;

node\* temp1 = head;

if (head != NULL) {

if (length() == 1)

{

cout << "Ցուցակը դատարկվեց" << endl;

head = NULL;

delete head;

return;

}

do

{

temp1 = temp1->next;

l++;

} while (temp1->next != NULL);

for (int i = 1; i < l; i++)

temp = temp->next;

temp->next = NULL;

delete temp->next;

}

else

{

cout << "Դատարկից ջնջել չի կարելի" << endl; return;

}

}

void list::del\_med()

{

int tex;

if (head != NULL)

{

if (length() == 1)

{

head = NULL;

delete head;

cout << "Ցուցակը դատարկվեց" << endl; return;

}

do

{

cout << "Դիրք=";

cin >> tex;

} while (tex<1 || tex>length());

node\* temp = head;

node\* temp1 = head;

node\* temp2 = head;

if (tex == 1)

{

del\_first();

return;

}

if (tex == length())

{

del\_last();

return;

}

{

for (int i = 1; i < tex; i++)

temp = temp->next;

for (int i = 1; i < tex - 1; i++)

temp1 = temp1->next;

for (int i = 1; i < tex + 1; i++)

temp2 = temp2->next;

}

temp1->next = temp2;

temp2->prev = temp1;

delete temp;

}

else cout << "Դատարկից ջնջել չի կարելի" << endl;

}

void list::show()

{

node\* temp = head;

if (temp != NULL) {

while (temp != NULL)

{

cout << temp->x.azganun << " " << temp->x.bacakayutyun << " " << temp->x.gnahatakan << endl;

temp = temp->next;

}

}

else cout << "Դատարկ է" << endl;

}

void list::swap()

{

int l, k;

node\* temp = head;

node\* temp1 = head;

if (length() >= 2)

{

do

{

cout << "l=";

cin >> l;

cout << "k=";

cin >> k;

} while (l<1 || l>length() || k<1 || k>length());

if (l < k)

{

for (int i = 1; i < l; i++)

temp = temp->next;

for (int i = 1; i < k; i++)

temp1 = temp1->next;

usanox t;

t = temp->x;

temp->x = temp1->x;

temp1->x = t;

}

else

{

for (int i = 1; i < k; i++)

temp = temp->next;

for (int i = 1; i < l; i++)

temp1 = temp1->next;

usanox t;

t = temp->x;

temp->x = temp1->x;

temp1->x = t;

}

if (l == k) cout << "Բան չփոխվեց" << endl;

}

else cout << "Բավական տարրեր չկան";

}

list::~list()

{

while (head != NULL)

{

node\* temp = head->next;

delete head;

head = temp;

}

};

int main()

{

int l;

list ob;

do {

cout << endl;

cout << " ՄԵՆՅՈՒ " << endl;

cout << "1.Ցուցակի տպում" << endl;

cout << "2.Ցուցակի սկզբից ցուցակի միավորի ավելացում" << endl;

cout << "3.Ցուցակի վերջից ցուցակի միավորի ավելացում" << endl;

cout << "4.Ցուցակի տրված տեղում ցուցակի միավորի ավելացում" << endl;

cout << "5.Ցուցակի սկզբից ցուցակի միավորի հեռացում" << endl;

cout << "6.Ցուցակի վերջից ցուցակի միավորի հեռացում" << endl;

cout << "7.Ցուցակի տրված տեղից ցուցակի միավորի հեռացում" << endl;

cout << "8.Ցուցակի երկու կամայական միավորի փոխանակում" << endl;

cout << "9.Միավորի փնտրում" << endl;

cout << "10.Ուսանողների միջին գնահատական" << endl;

cout << "11.Միջինից ցածր գնահատական ունեցողների տվյալ" << endl;

cout << "12.Ելք" << endl;

{

cin >> l;

switch (l)

{

case 1:

{

ob.show();

break;

}

case 2:

{

ob.add\_first();

break;

}

case 3:

{

ob.add\_last();

break;

}

case 4:

{

ob.add\_med();

break;

}

case 5:

{

ob.del\_first();

break;

}

case 6:

{

ob.del\_last();

break;

}

case 7:

{

ob.del\_med();

break;

}

case 8:

{

ob.swap();

break;

}

case 9:

{

ob.info();

break;

}

case 10:

{

ob.average();

break;

}

case 11:

{

ob.newlist\_average();

break;

}

case 12:

{

cout << "Ավարտ";

break;

}

default:

{

cout << "Սխալ է ընտրվել" << endl;

break;

}

}

}

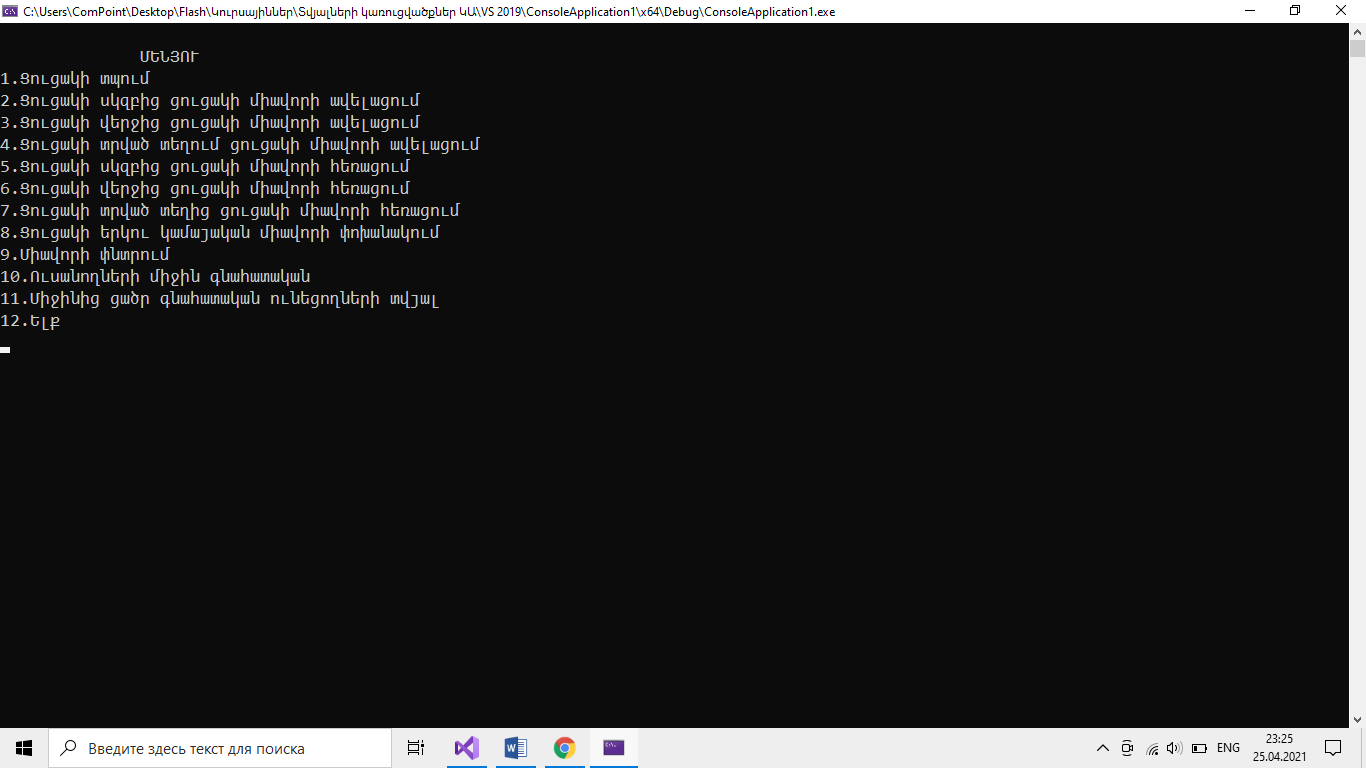
} while (l != 12);

return 0;

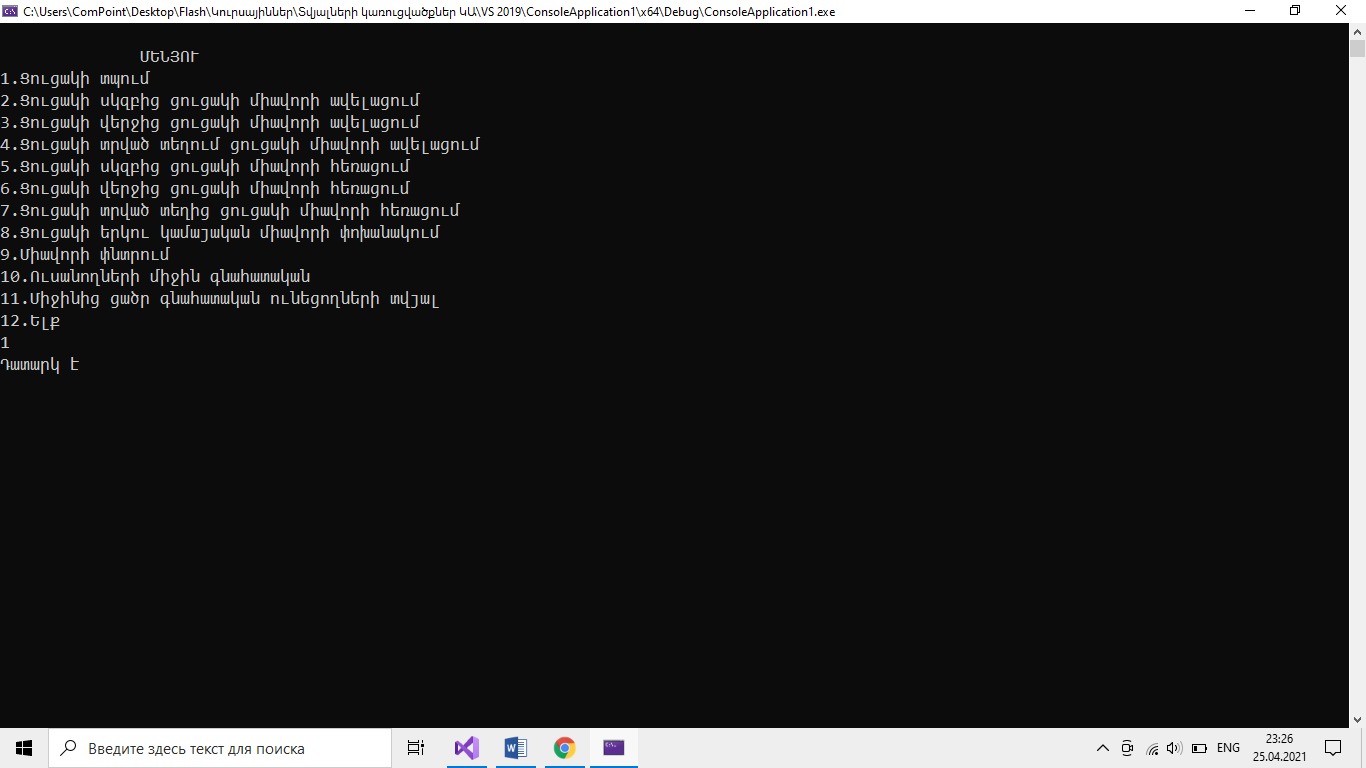
}

## ԾՐԱԳՐԻ ԱՇԽԱՏԱՆՔԻ ԱՐԴՅՈՒՆՔ

­­Աշխատանքի սկզբում կտեսնենք հետևյալ պատուհանը:

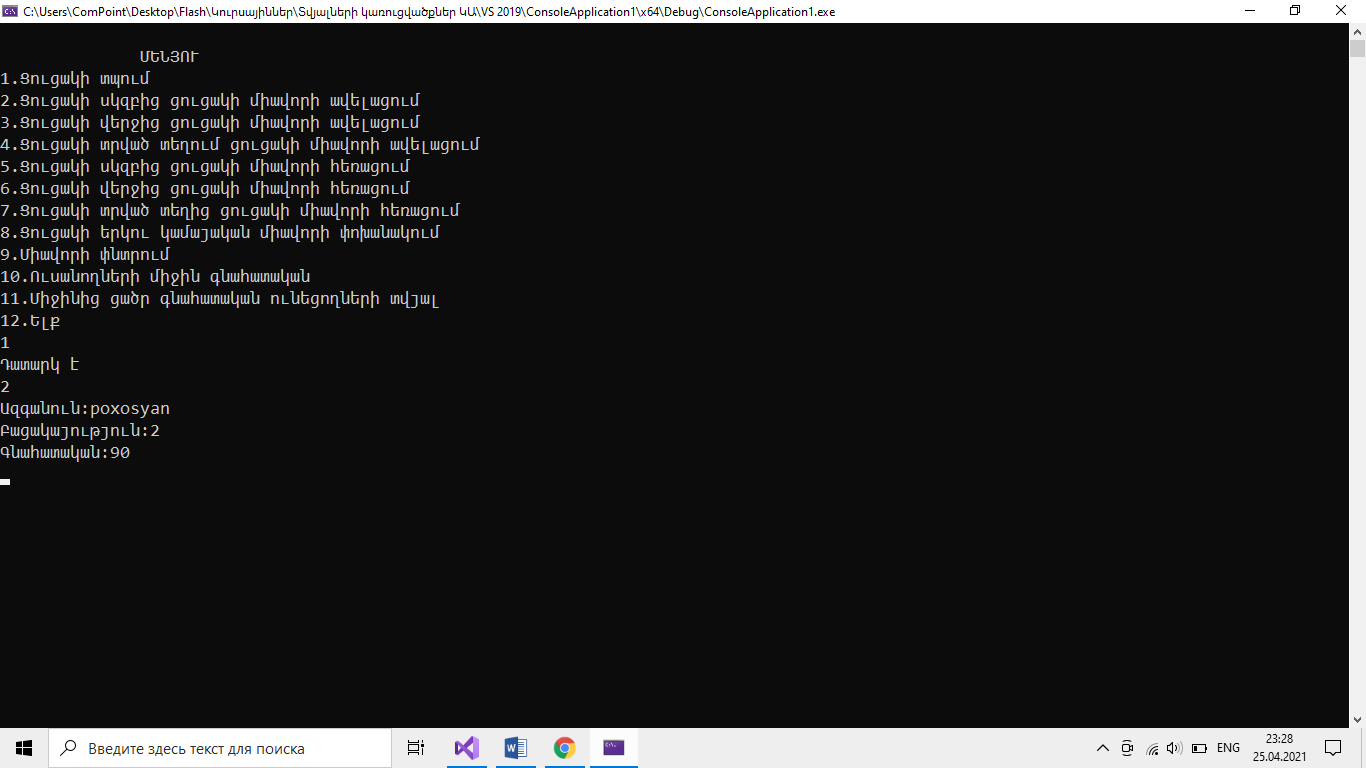


Նախ ընտրենք մենյուի 1 կետը:

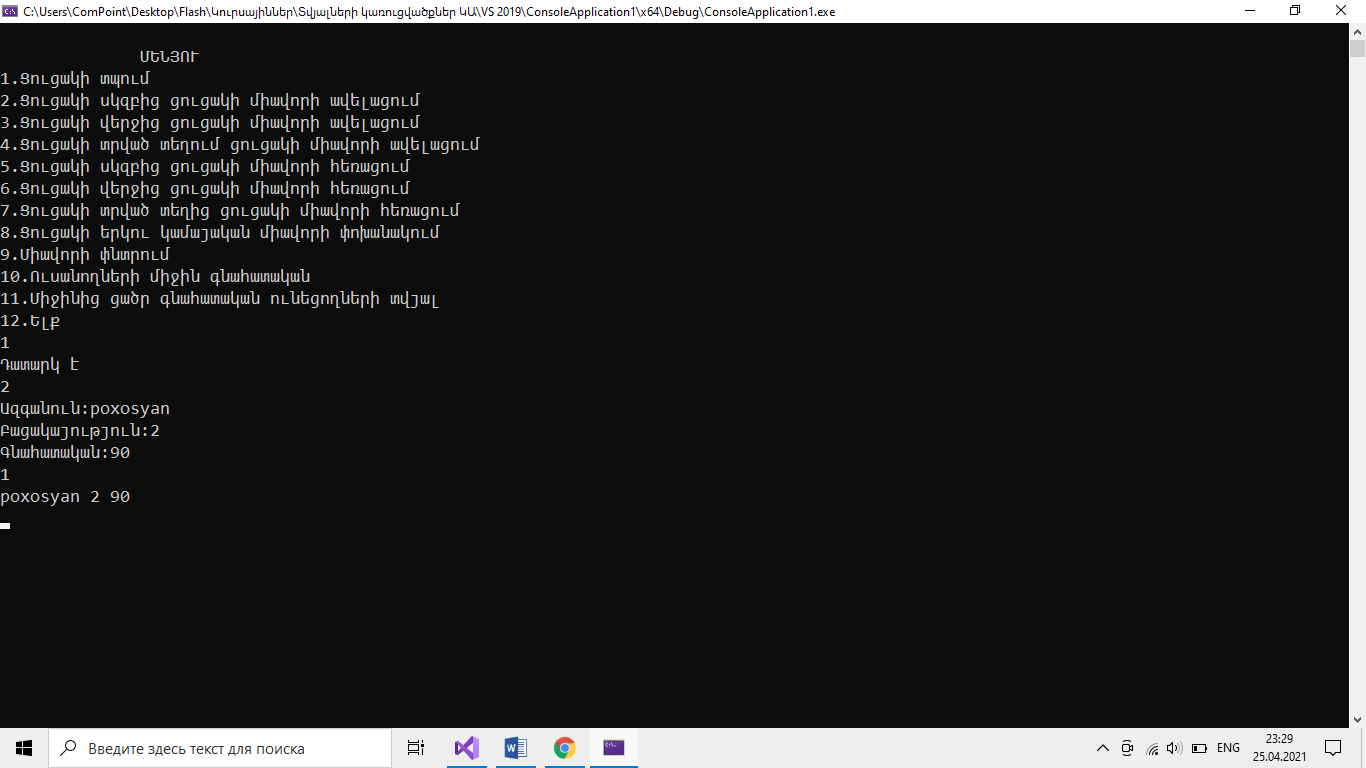


Կտեսնենք դատարկ է հաղորդագրությունը, որը նշանակում է, որ ցուցակում դեռ տարրեր չկան:

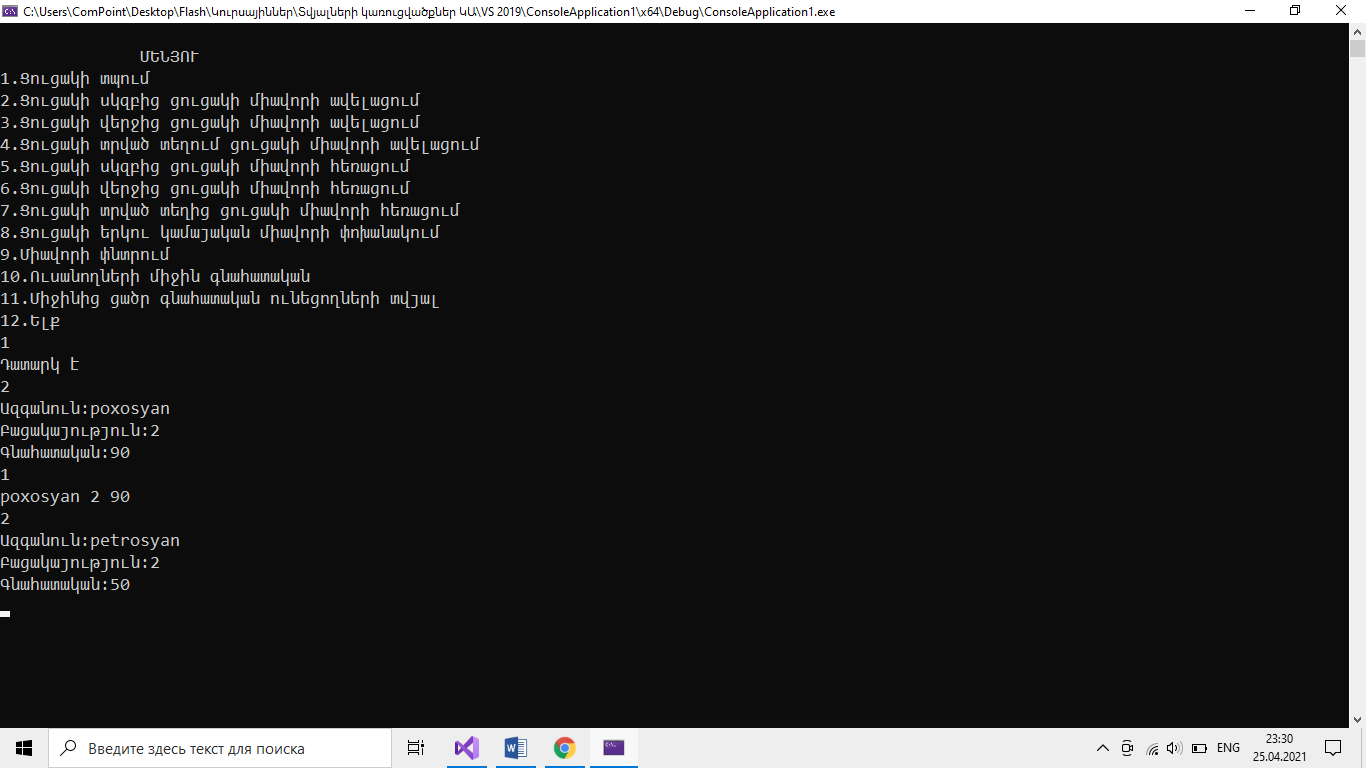
Ընտրենք մենյուի 2 կետը և կատարենք ներմուծումը:



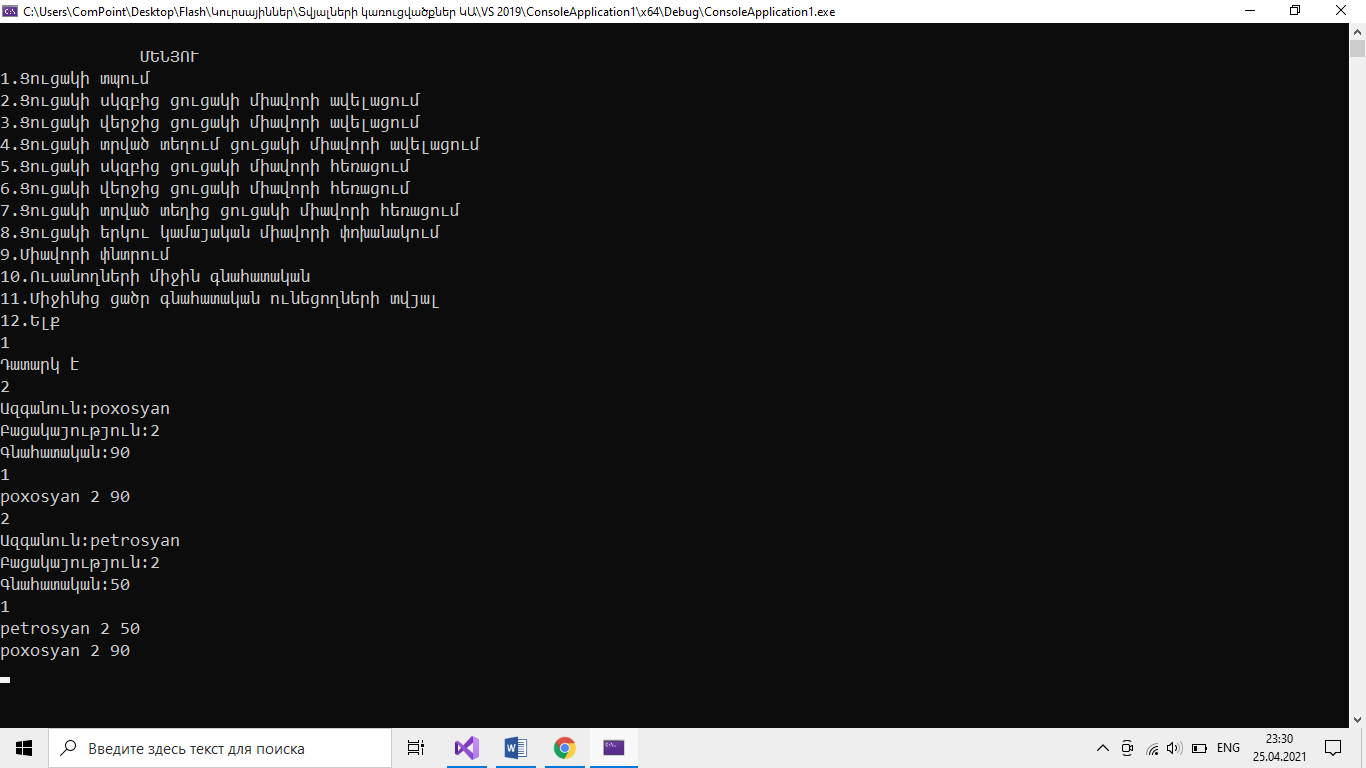
Հիմա կատարենք ցուցակի արտածումը:



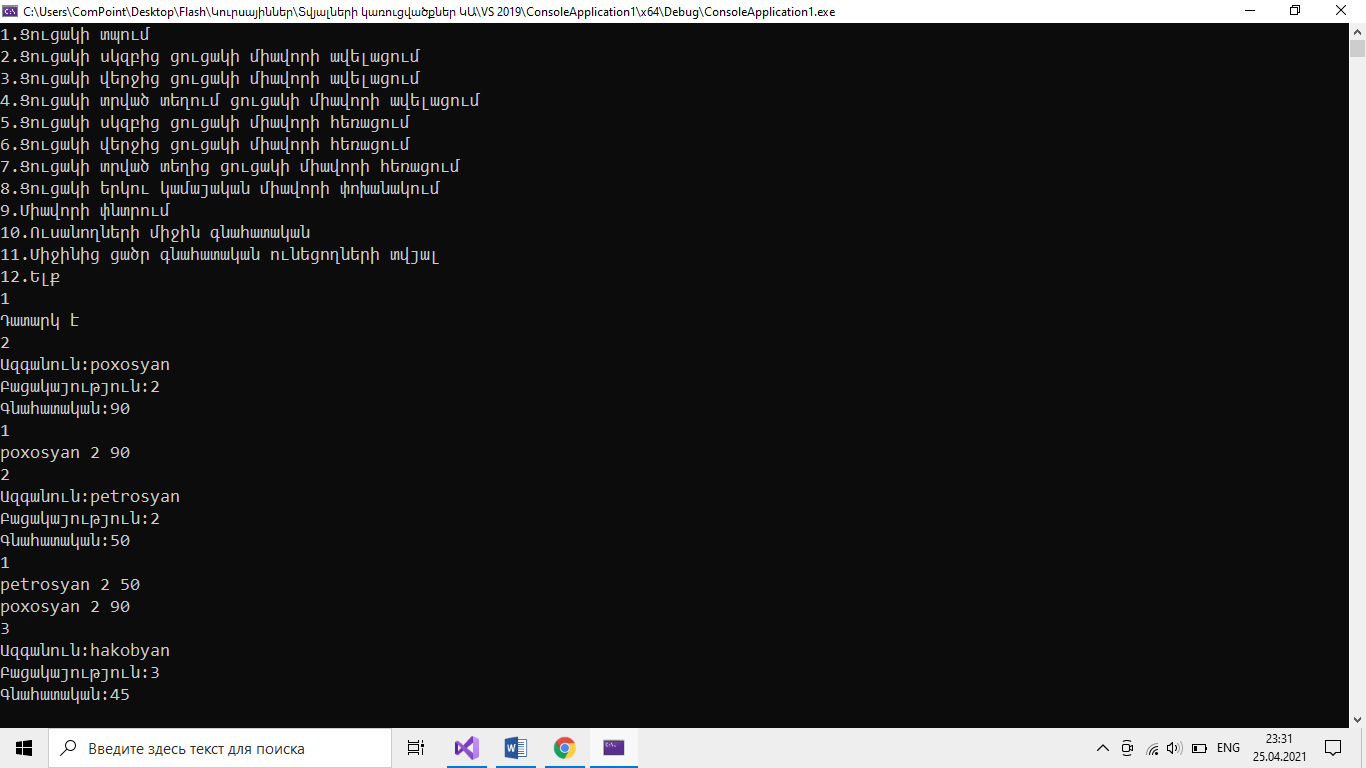
Նորից կատարենք ցուցակի սկզբից միավորի ավելացումը:



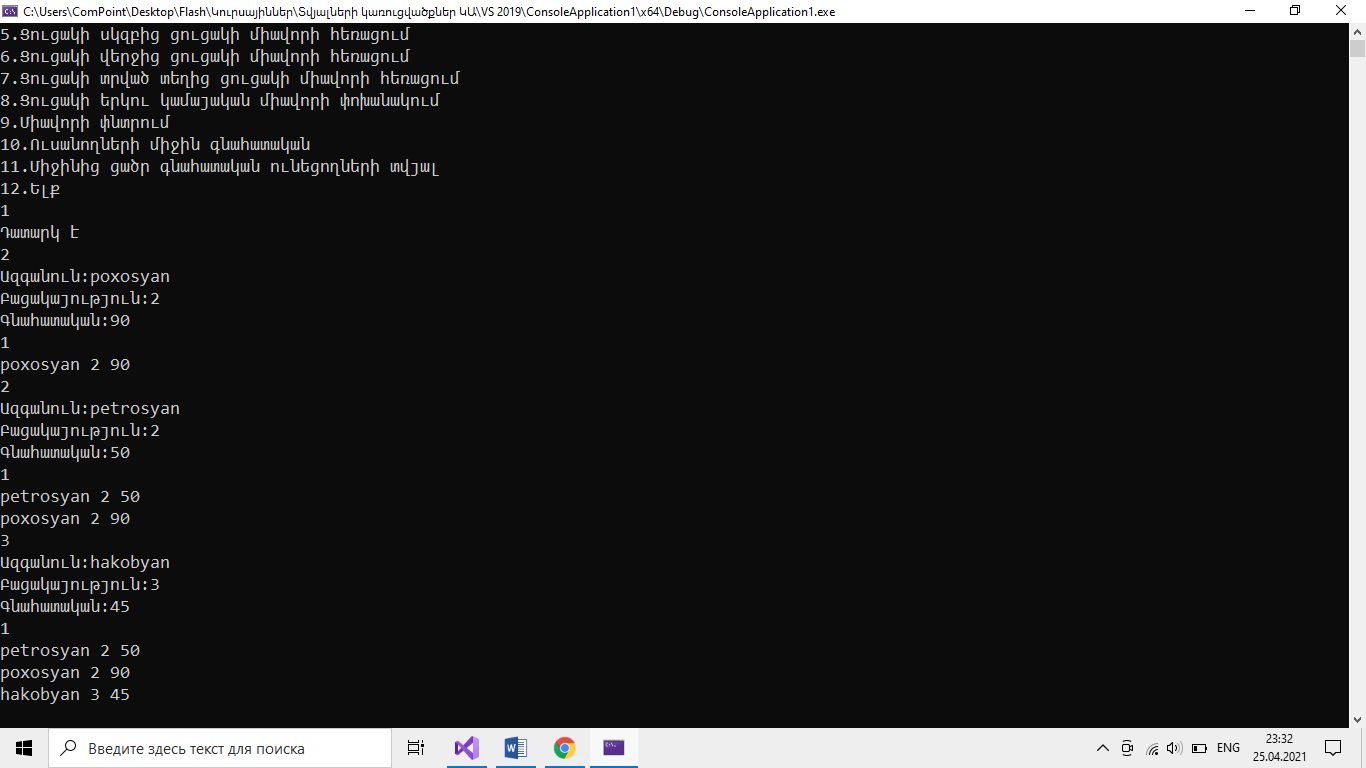
Նորից կատարենք ցուցակի արտածում:



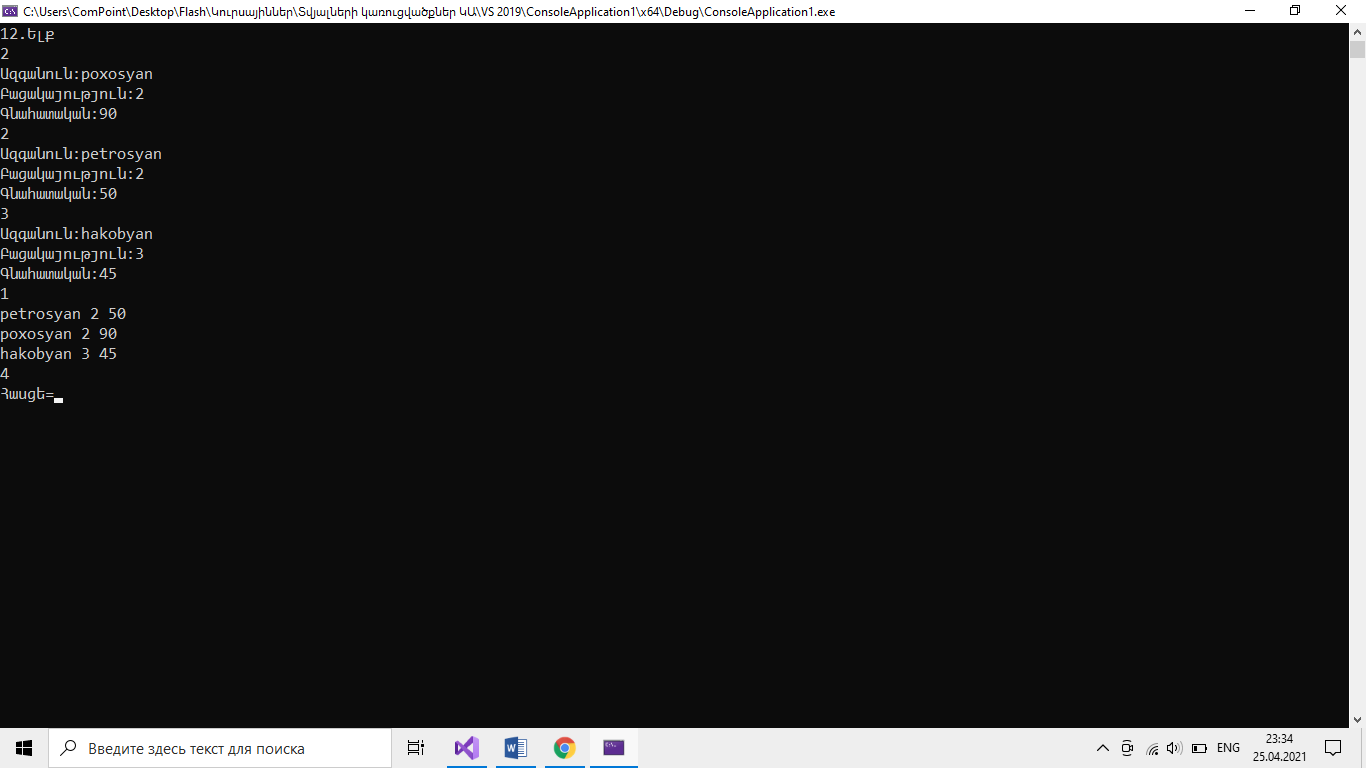
Կատարենք ցուցակի վերջից ցուցակի միավորի ավելացում:



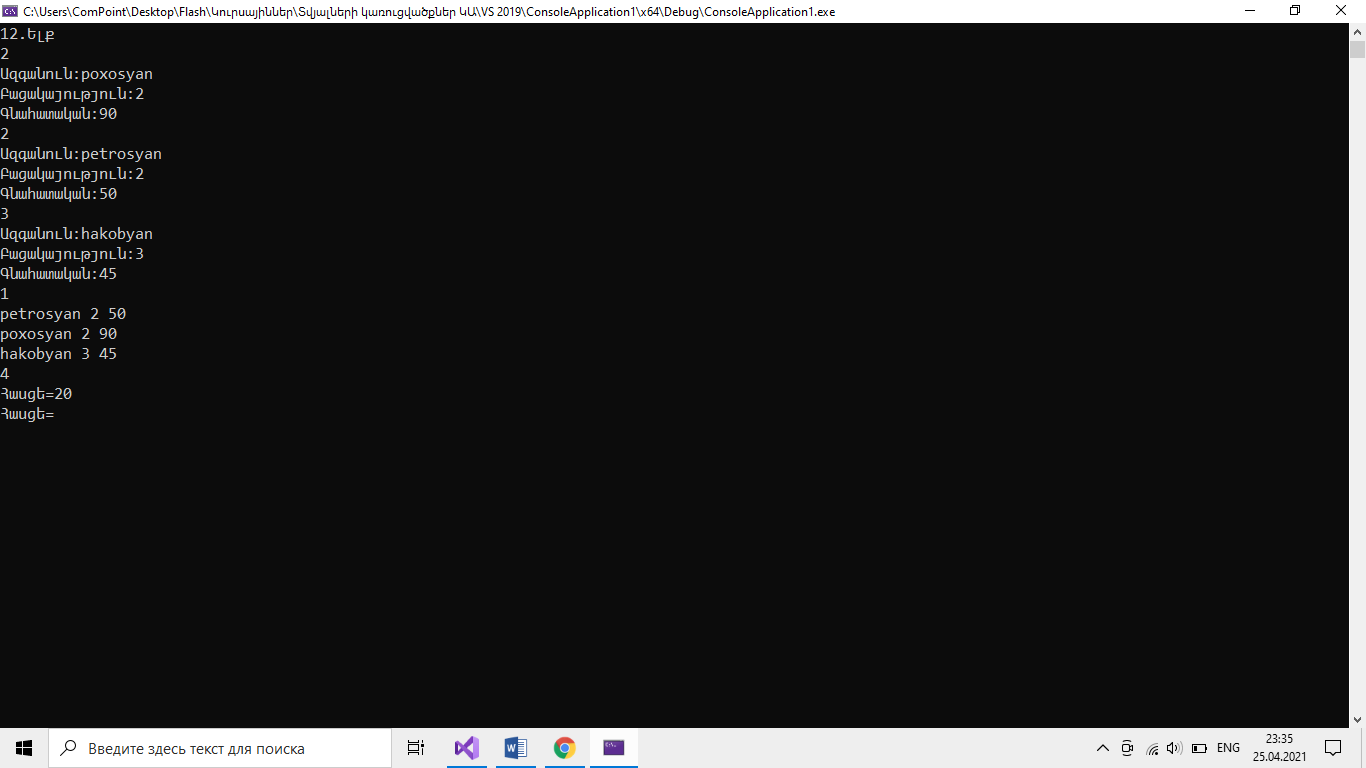
Այժմ տպենք ցուցակը:



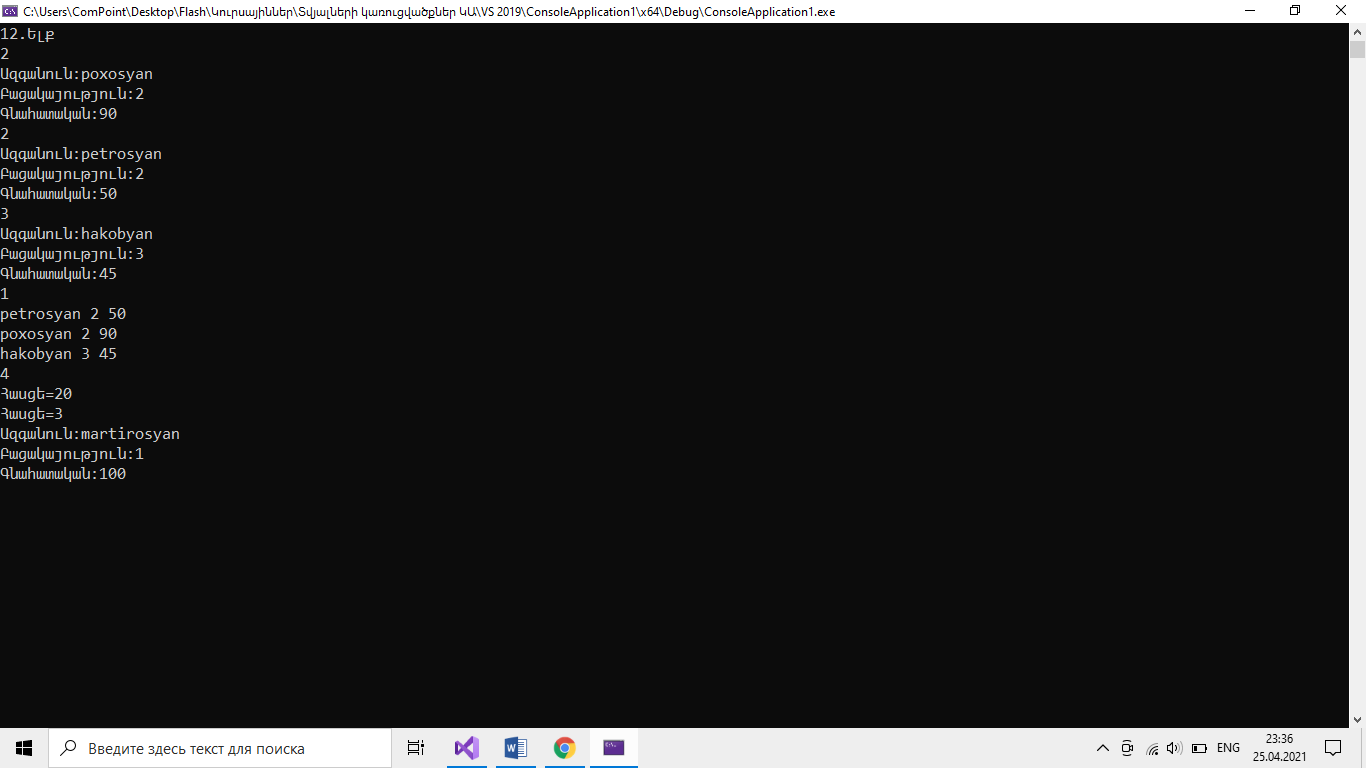
Կատարենք տրված տեղում ցուցակի միավորի ավելացում:



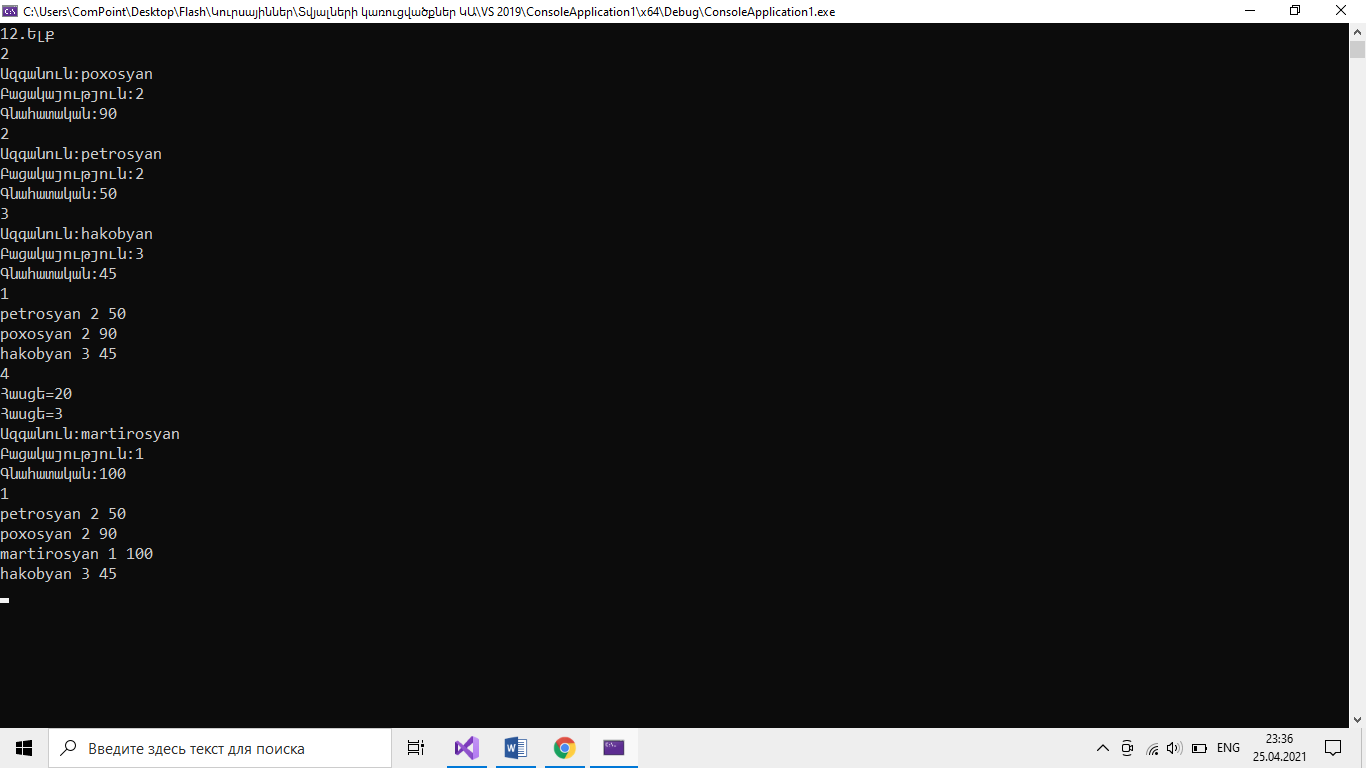
Ներմուծենք հասցեն 20:



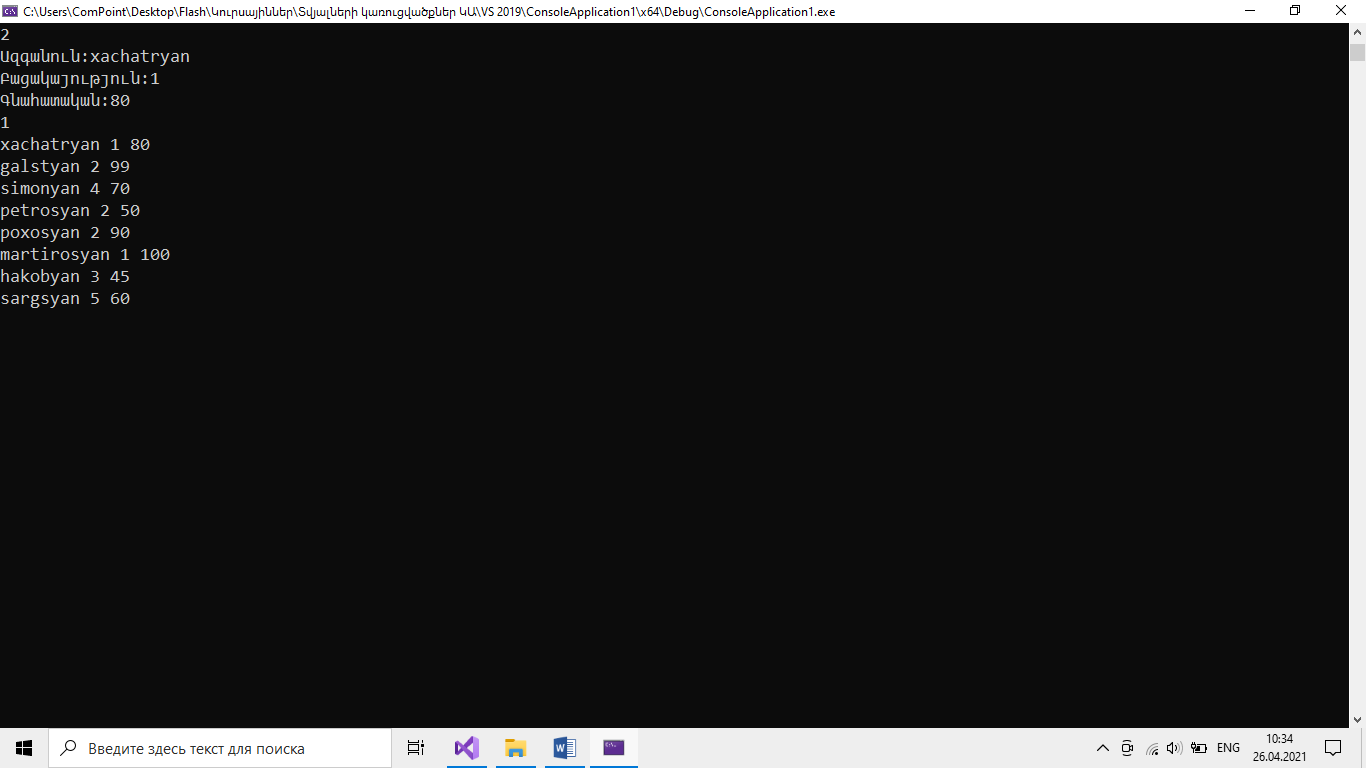
Պետք է նորից կատարել հասցեի ներմուծում, քանի որ 20-ը թույլատրելի սահմանում չէ:



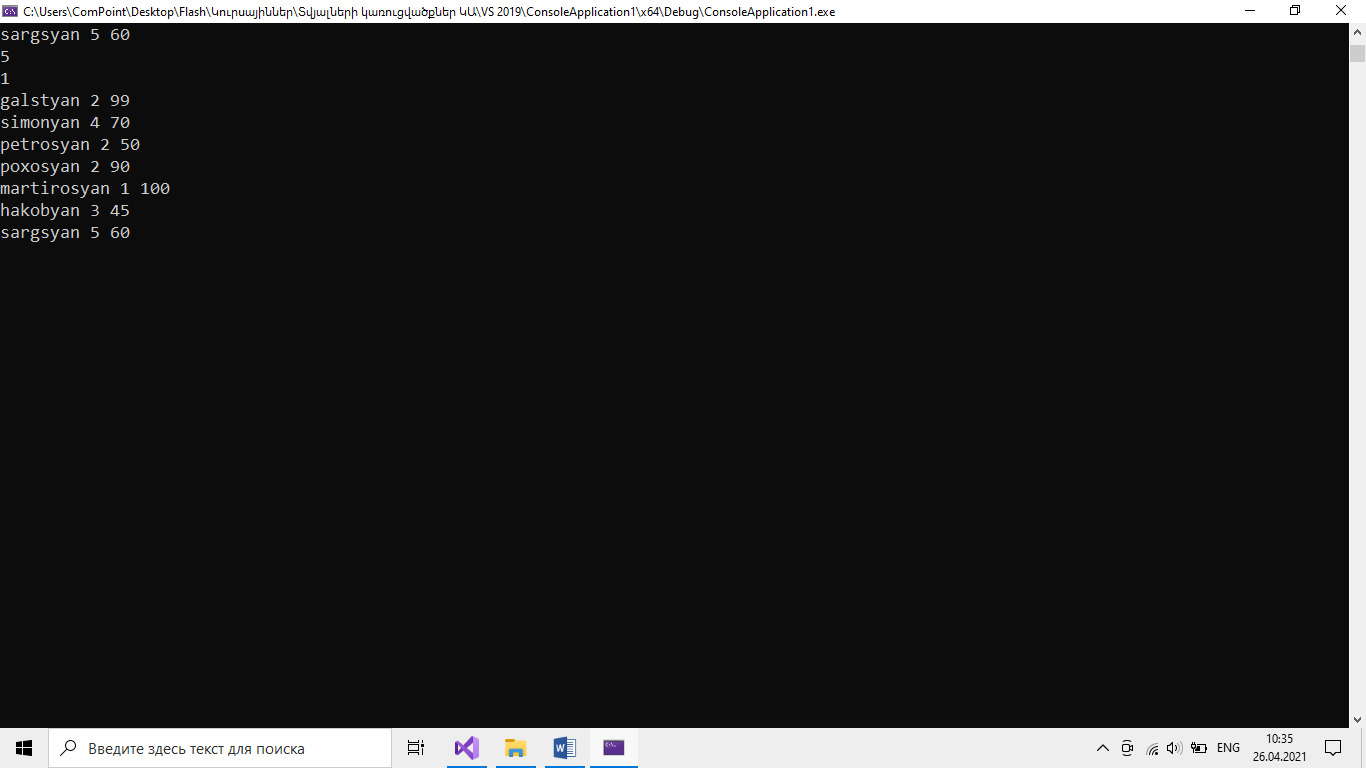
Արտածելուց հետո կտեսնենք, որ ցուցակի երրորդ դիրքում ավելացել է martirosyan ազգանունով ուսանողը:



Ցուցակում տարբեր դիրքերում ավելացնենք տվյալներ հետագա աշխատանքի համար:

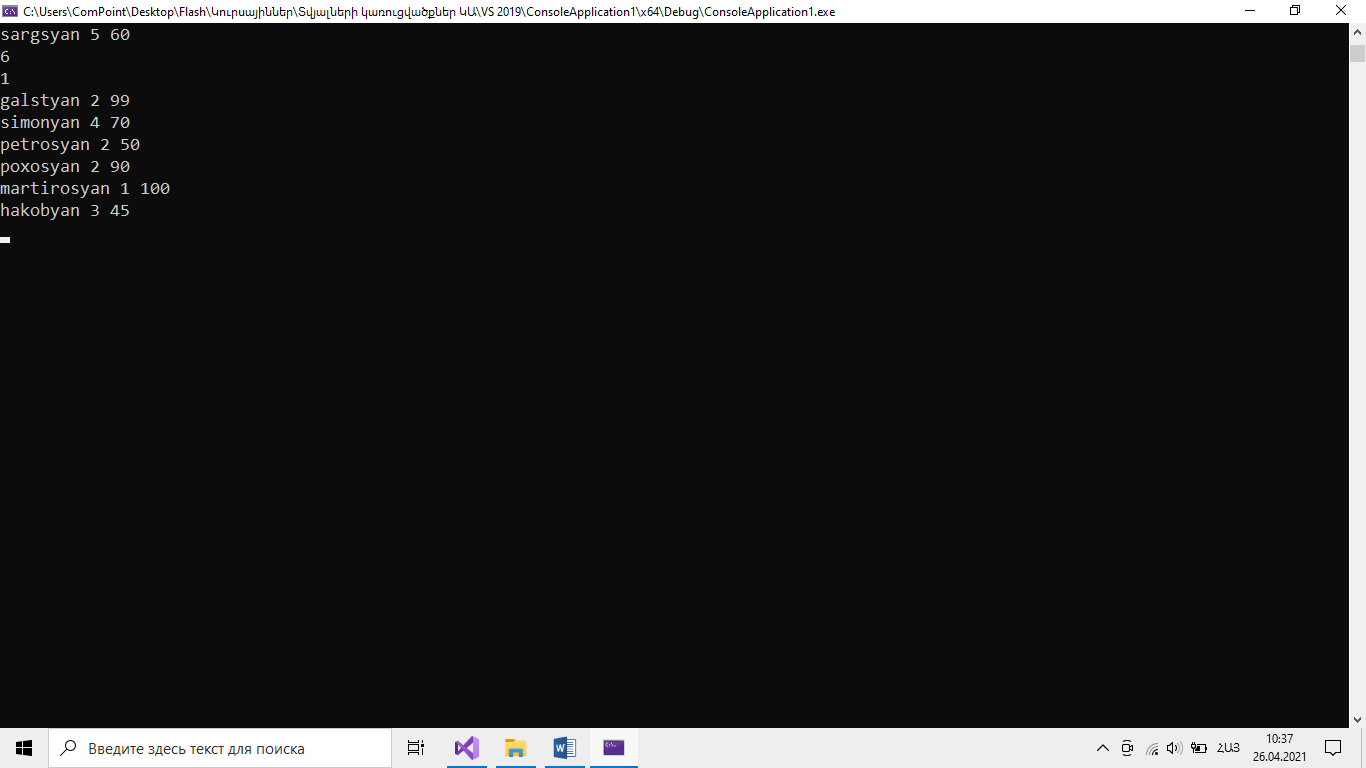


Կատարենք հանգույցի ջնջում ցուցակի սկզբից այնուհետև տպենք ստացված ցուցակը:



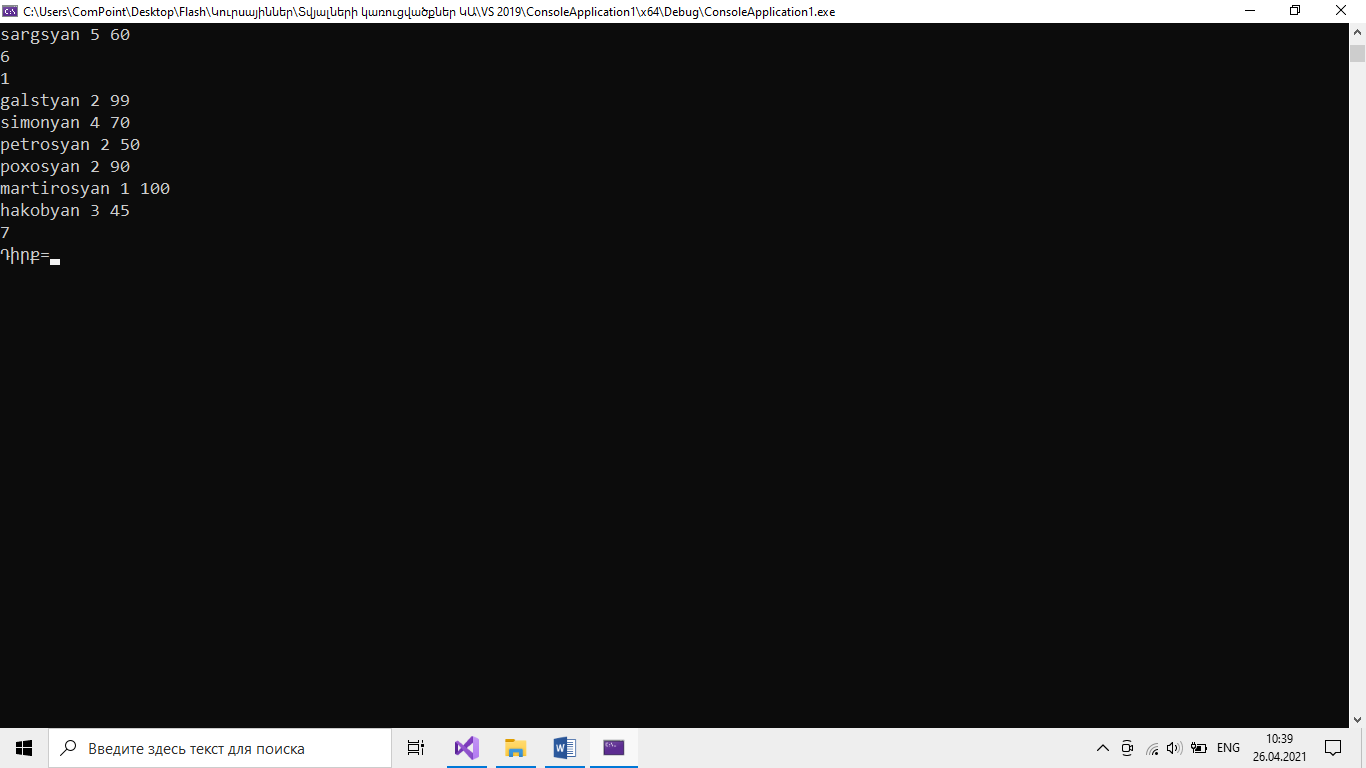
Տեսանք, որ xachatryan ազգանունով ուսանողը ջնջվեց ցուցակից:

Կատարենք հանգույցի ջնջում ցուցակի վերջից և տպենք ցուցակը:

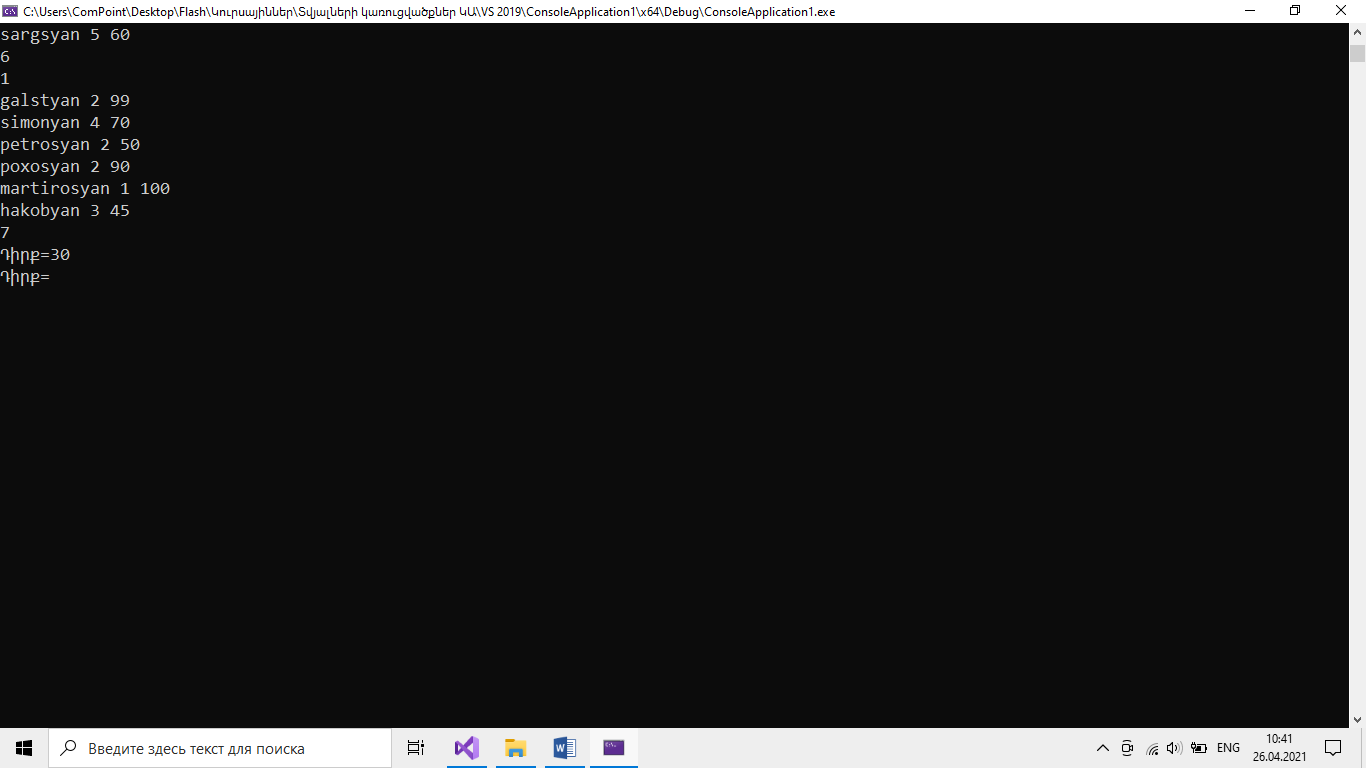


Տեսանք, որ sargsyan ազգանունով ուսանողը, ով վերջինն էր, ջնջվեց ցուցակից:

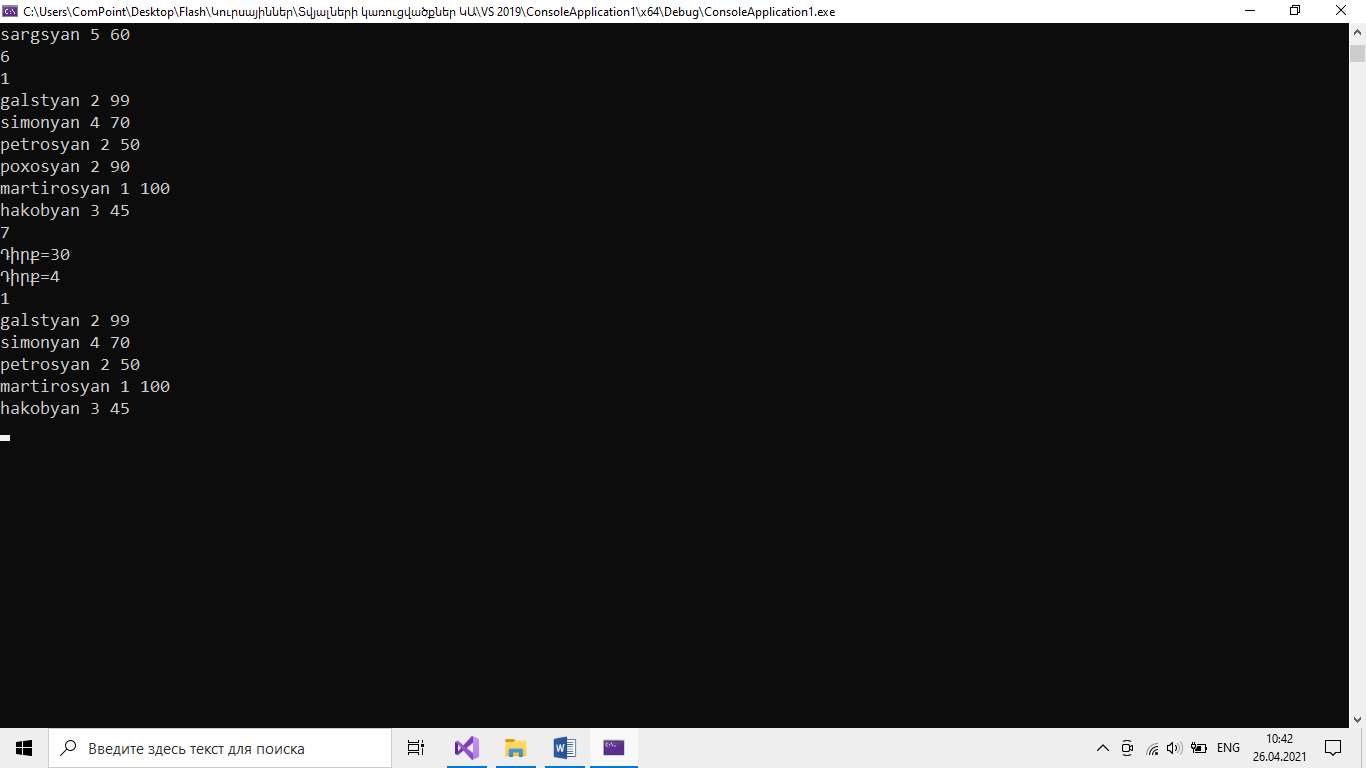
Կատարենք տրված տեղից հանգույցի ջնջում:



Պետք է ներմուծենք դիրքը, որից ուզում ենք հեռացնել հանգույցը: Պարզ է, որ ներմուծված դիրքը պետք է լինի ցուցակի սահմաններում: Ներմուծենք դիրքը 30:

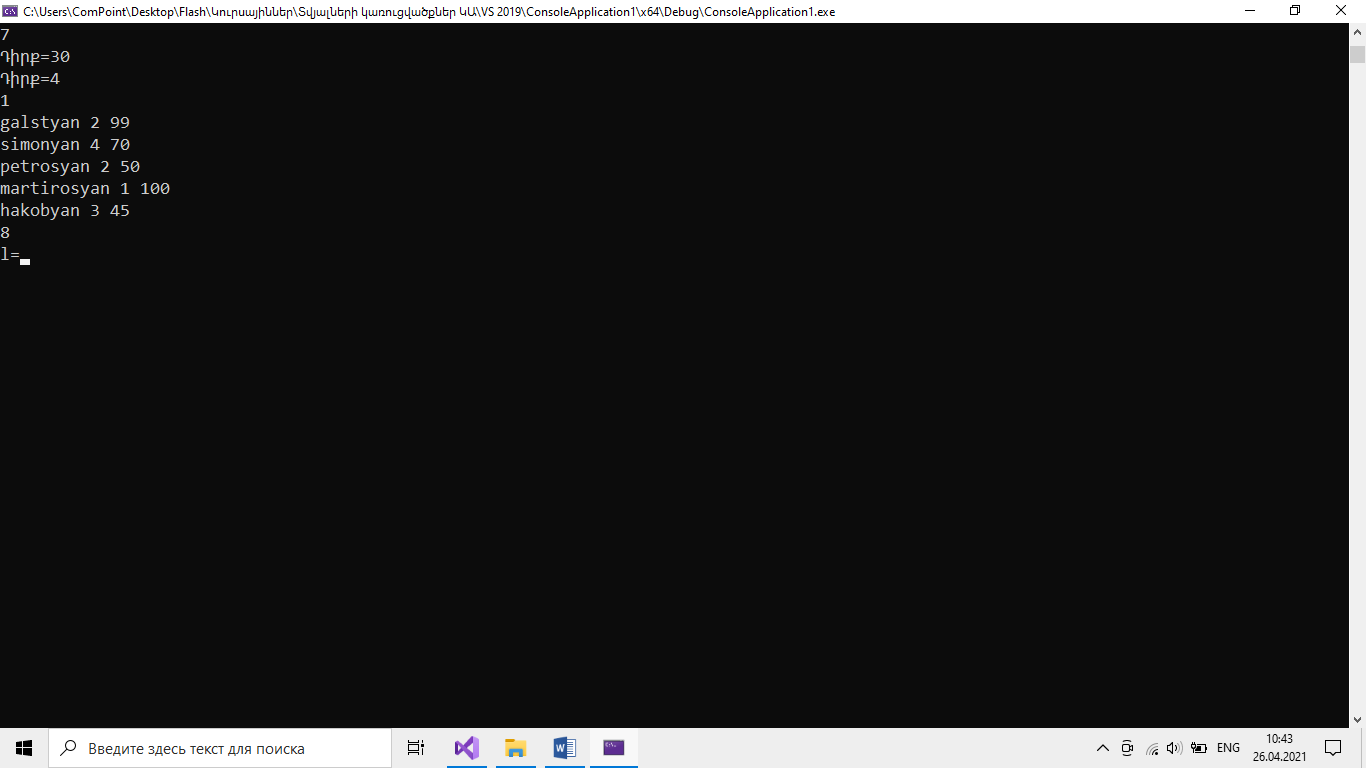


Քանի որ ցուցակում կան երեսունից քիչ ուսանողներ, ապա դիրքը պետք է ներմուծենք նորից: Այժմ ներմուծենք դիրքը 4 և տպենք ցուցակը:

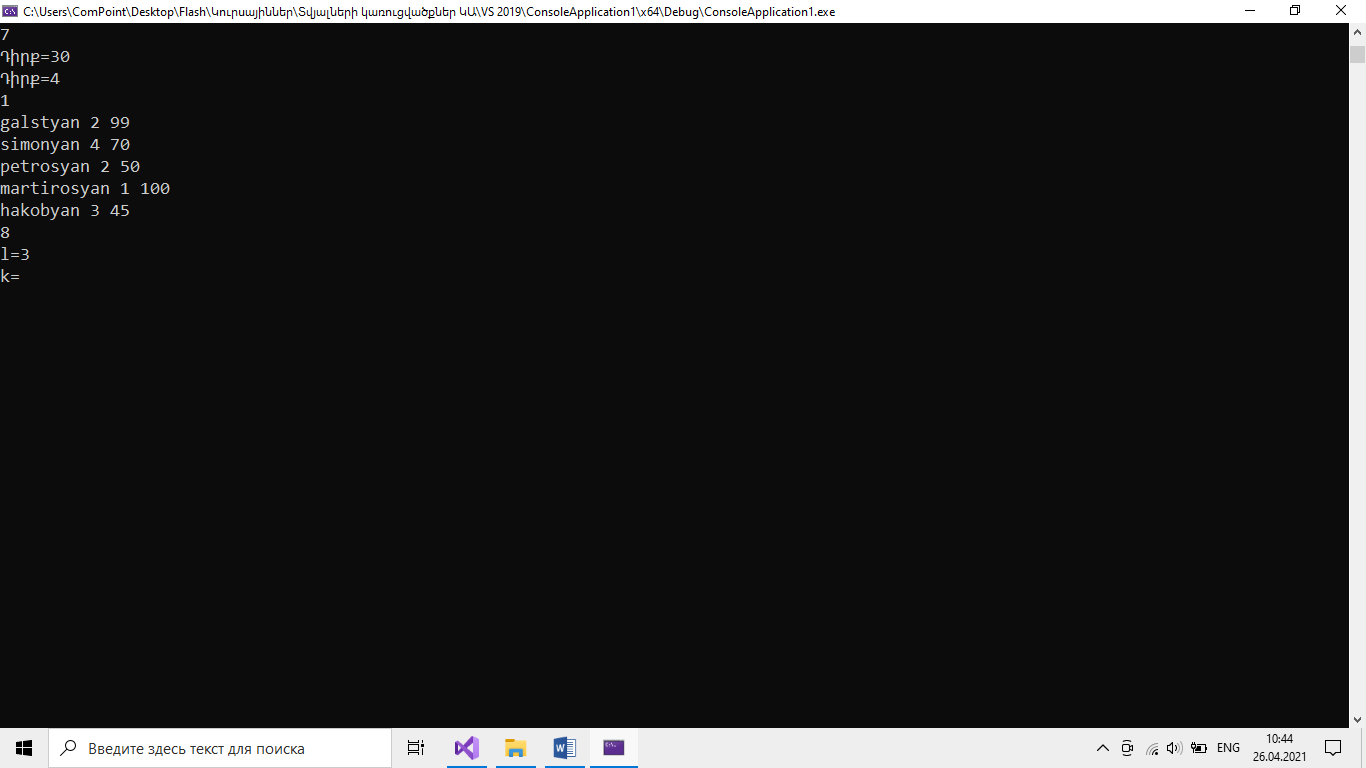


Տեսանք, որ ցուցակի չորրորդ դիրքում գտնվող ուսանողը, ով poxosyan-ն էր, ջնջվեց ցուցակից:

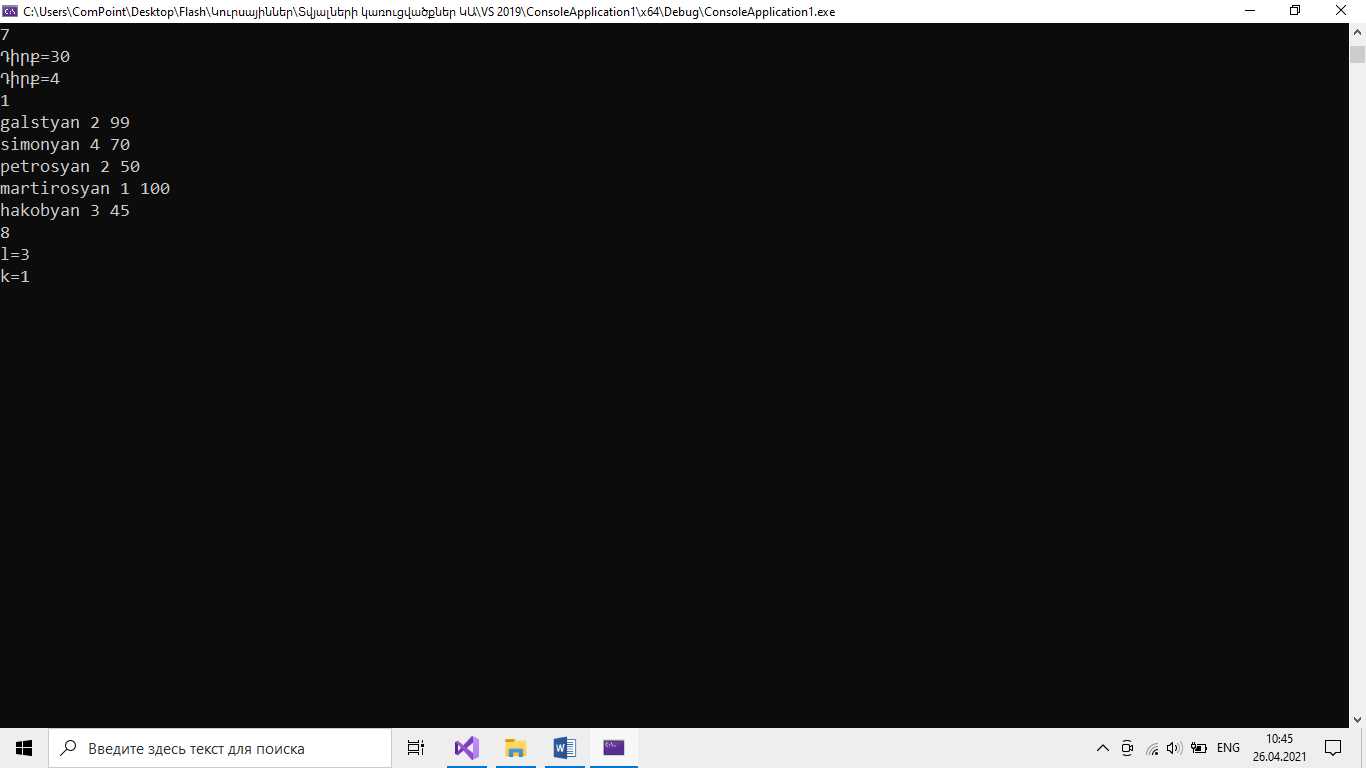
Այժմ կատարենք ցուցակի երկու կամայական միավորի փոխատեղում:



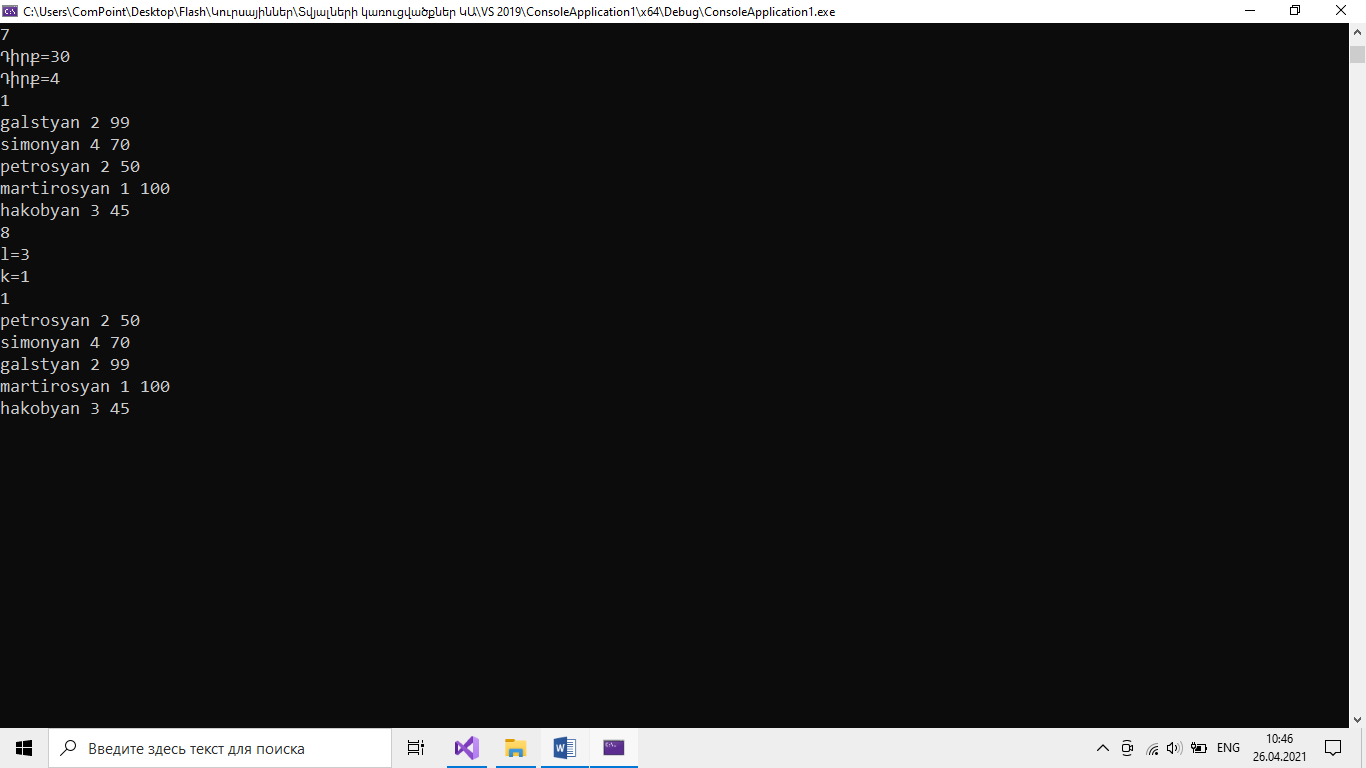
Պետք է ներմուծենք փոխատեղվող հանգույցներից առաջինի դիրքը: Այն նույնպես պետք է լինի ցուցակի սահմաններում: Ընտրենք դիրքը 3:



Քանի որ 3 դիրքը ցուցակի սահմաններում էր, ապա պետք է ներմուծենք փոխատեղվող երկրորդ հանգույցի դիրքը: Ընտրենք այդ դիրքը 1:

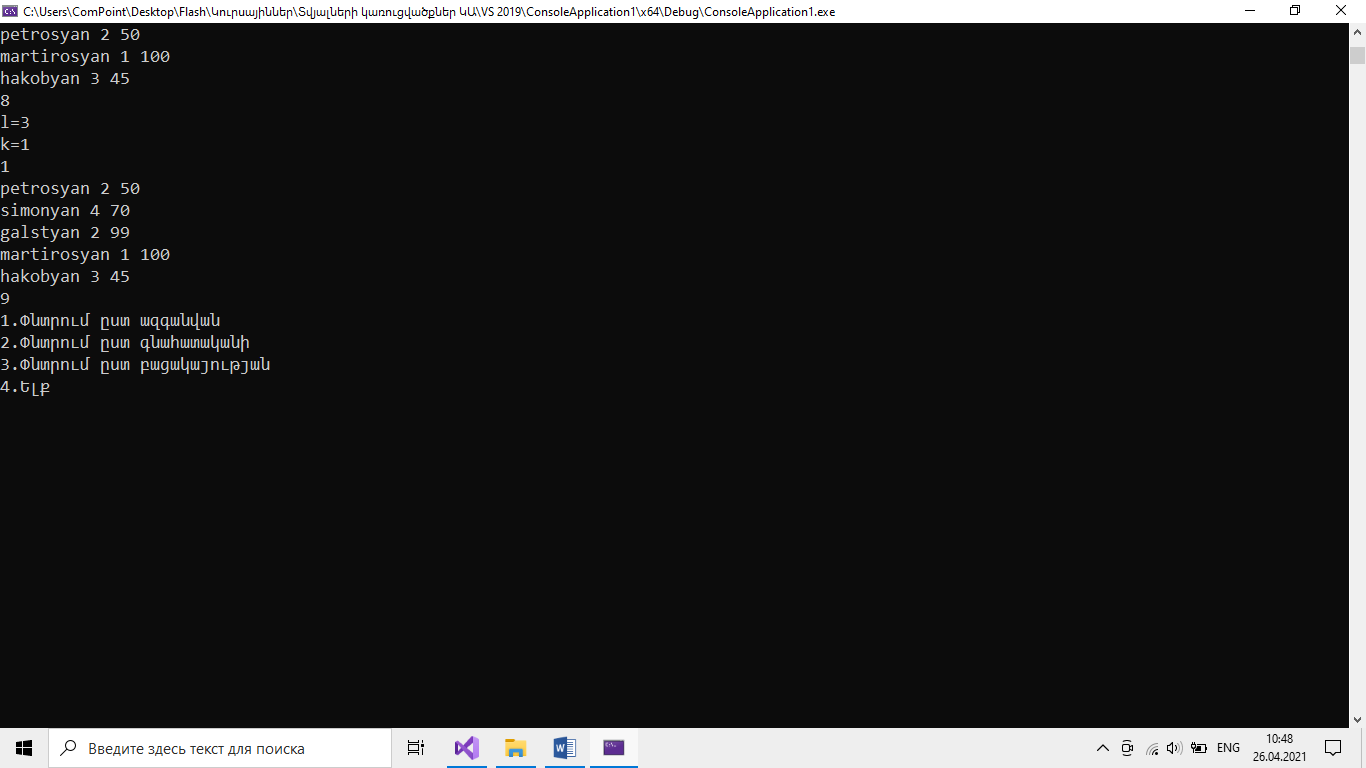


Արդեն փոխատեղումը կատարված է, այժմ տպենք ցուցակը արդյունքը տեսնելու համար:

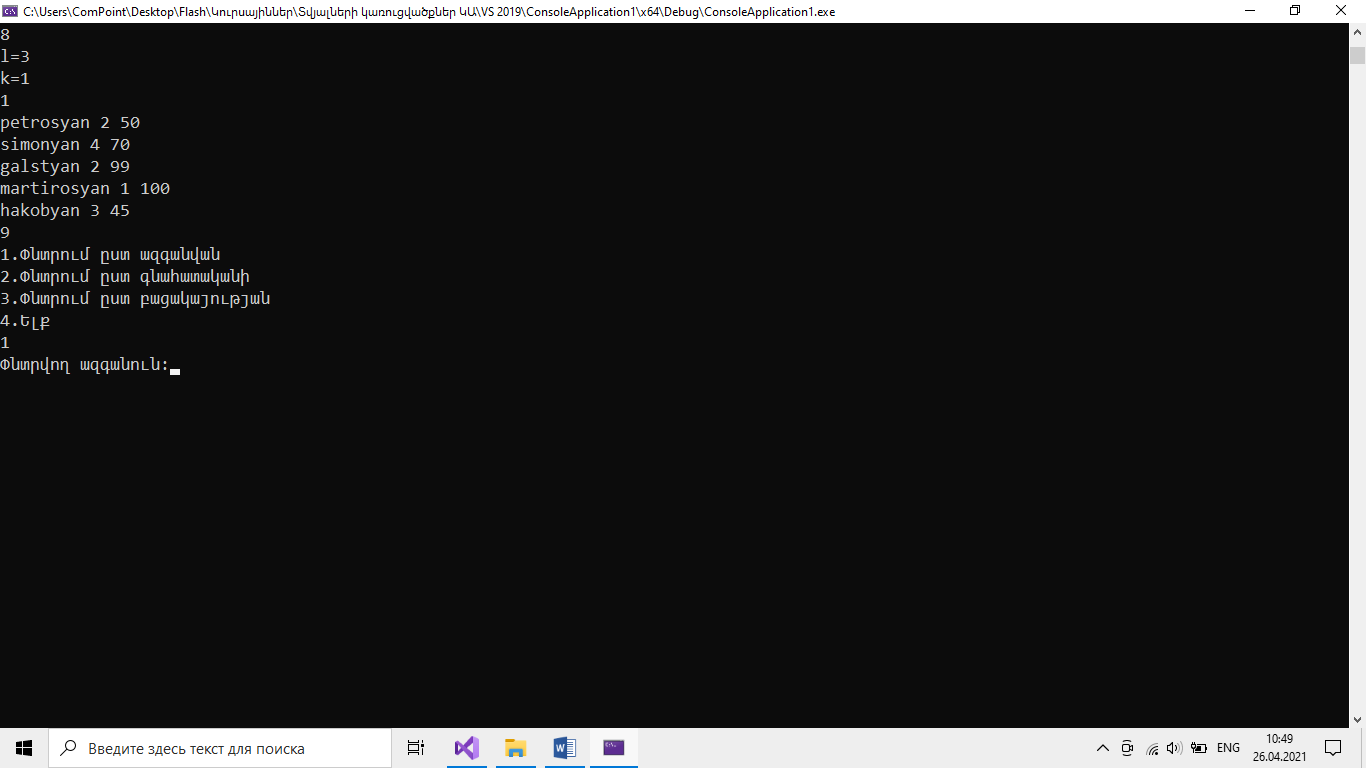


Տեսնում ենք, որ ցուցակի առաջին և երրորդ դիրքում գտնող ուսանողները տեղերով փոխվել են:

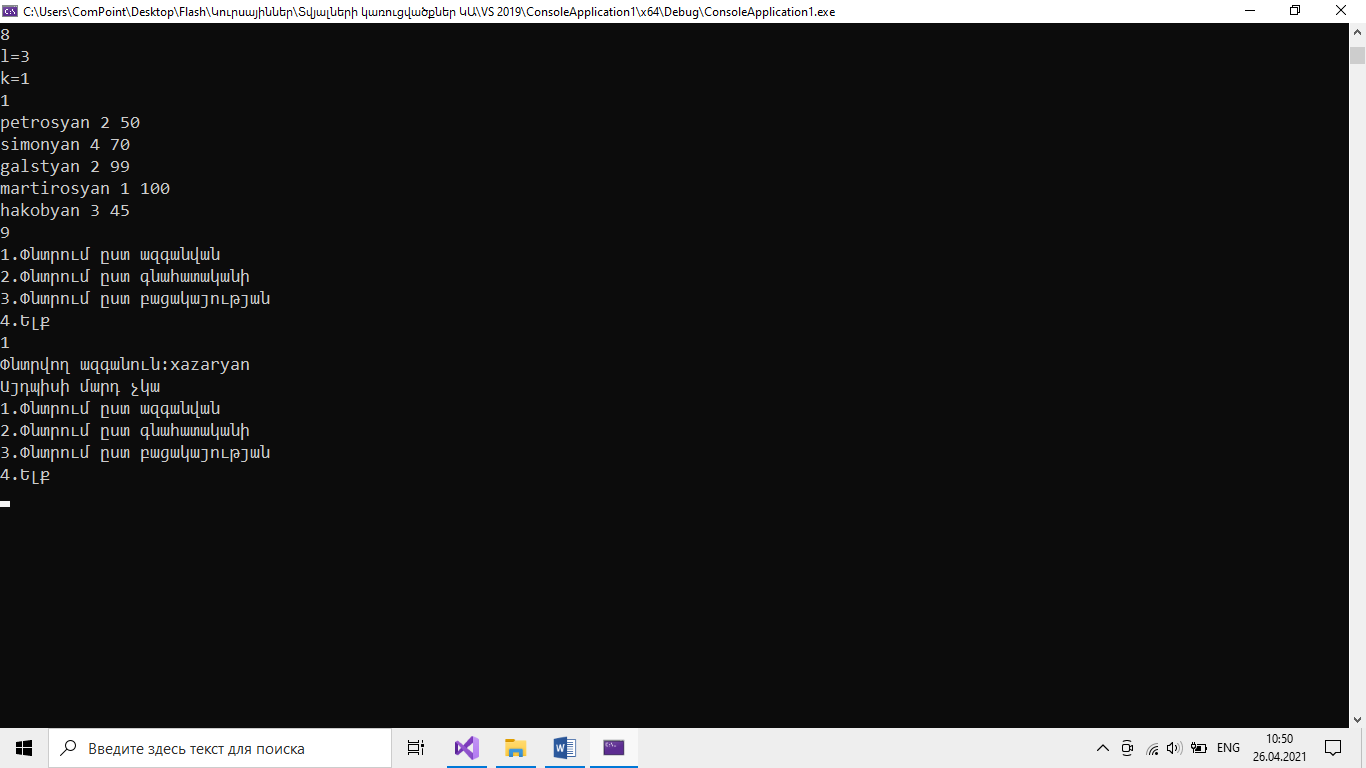
Այժմ կատարենք փնտրվող ինֆորմացիայի արտածումը:



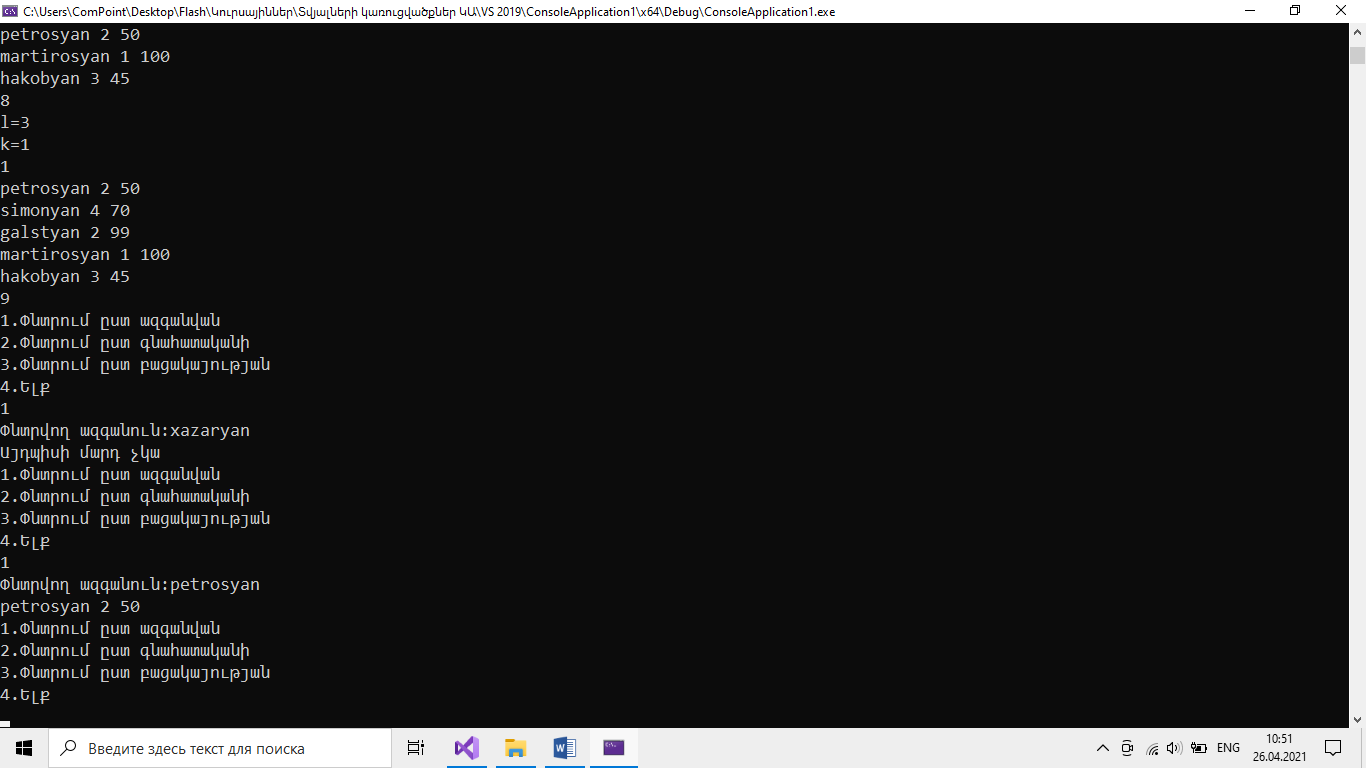
Այժմ ունենք հնարավորություն փնտրել տվյալներ ըստ ազգանվան, գնահատականի կամ բացակայության: Կատարենք արժեքի փնտրում ըստ ազգանվան:



Հիմա պետք է ներմուծենք փնտրվող ազգանունը: Ներմուծենք այնպիսի ազգանուն, որից ցուցակում չունենք:

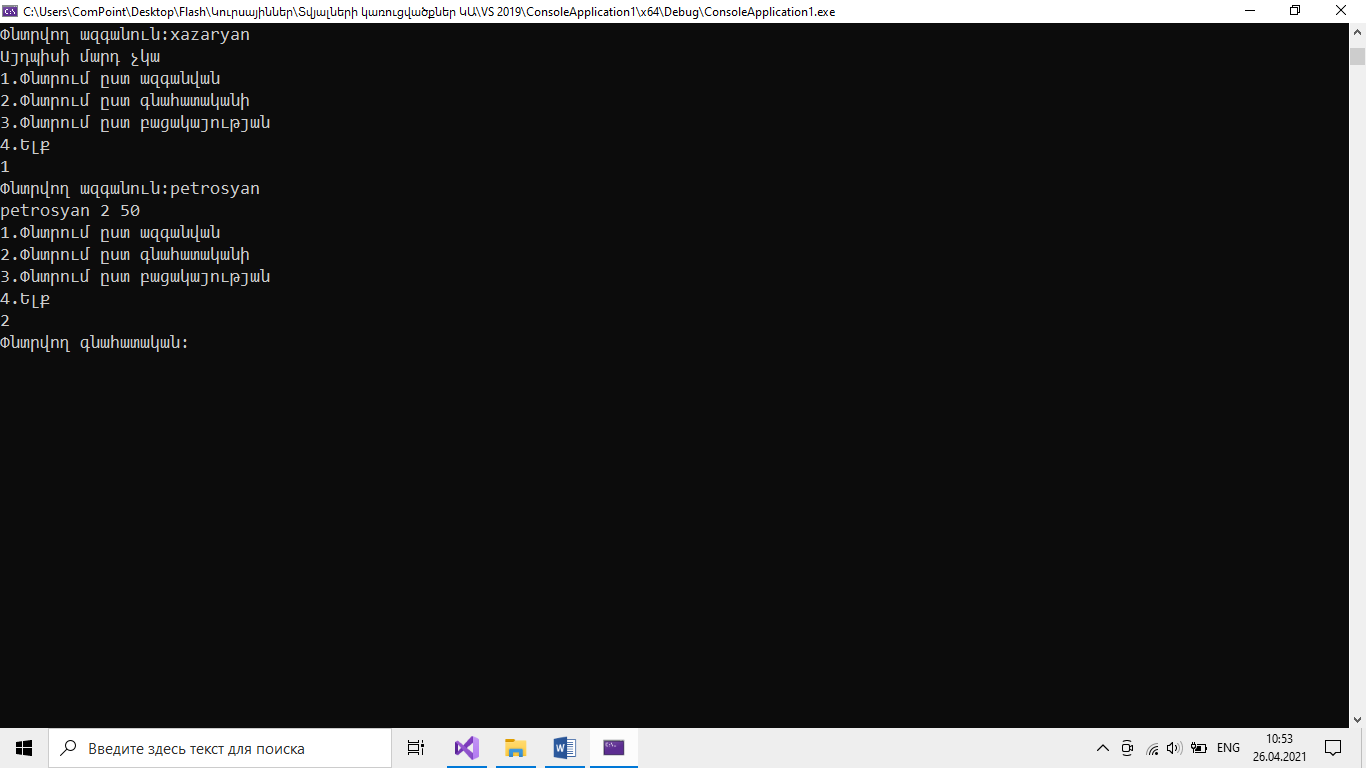


Քանի որ xazaryan ազգանունով ուսանող ցուցակում չունեինք, ստացանք այդպիսի մարդ չկա հաղորդագրությունը: Այժմ նորից կատարենք փնտրում ըստ ազգանվան: Ներմուծենք աjնպիսի ազգանուն, որից ցուցակում ունենք:

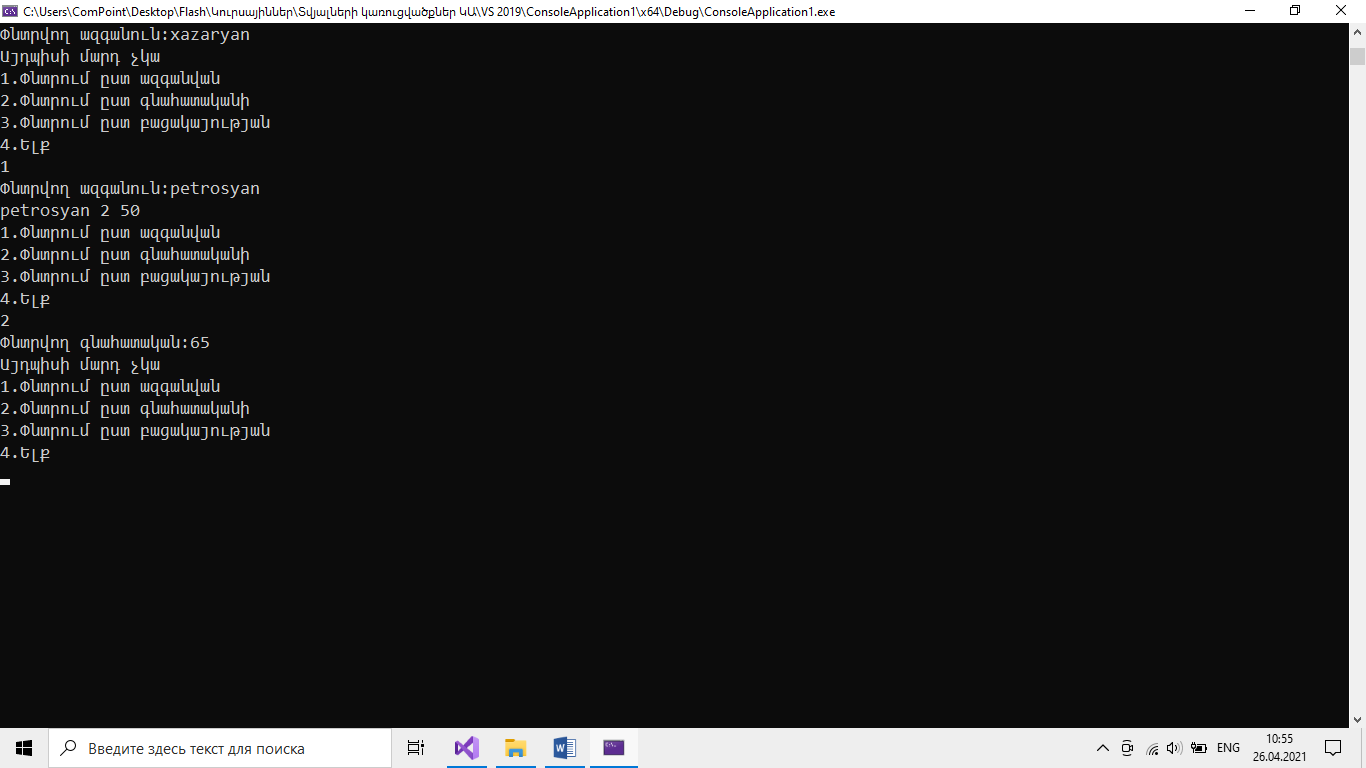


Ստացանք petrosyan ազգանունով ուսանողի տվյալները:

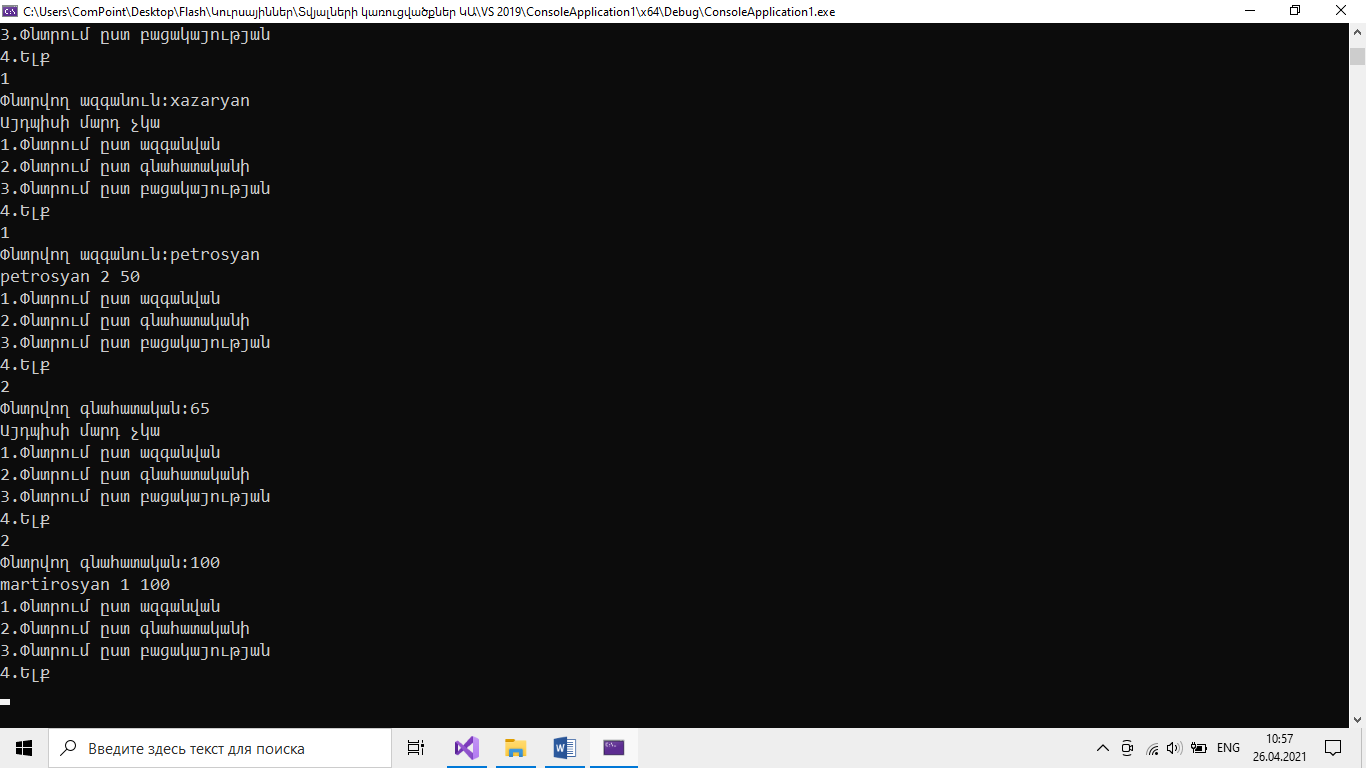
Այժմ կատարենք արժեքի փնտրում ըստ գնահատականի:



Պետք է ներմուծենք [0; 100] միջակայքի թիվ, հակառակ դեպքում նորից կսպասվի գնահատակի մուտք։ Ներմուծենք գնահատականը 65:

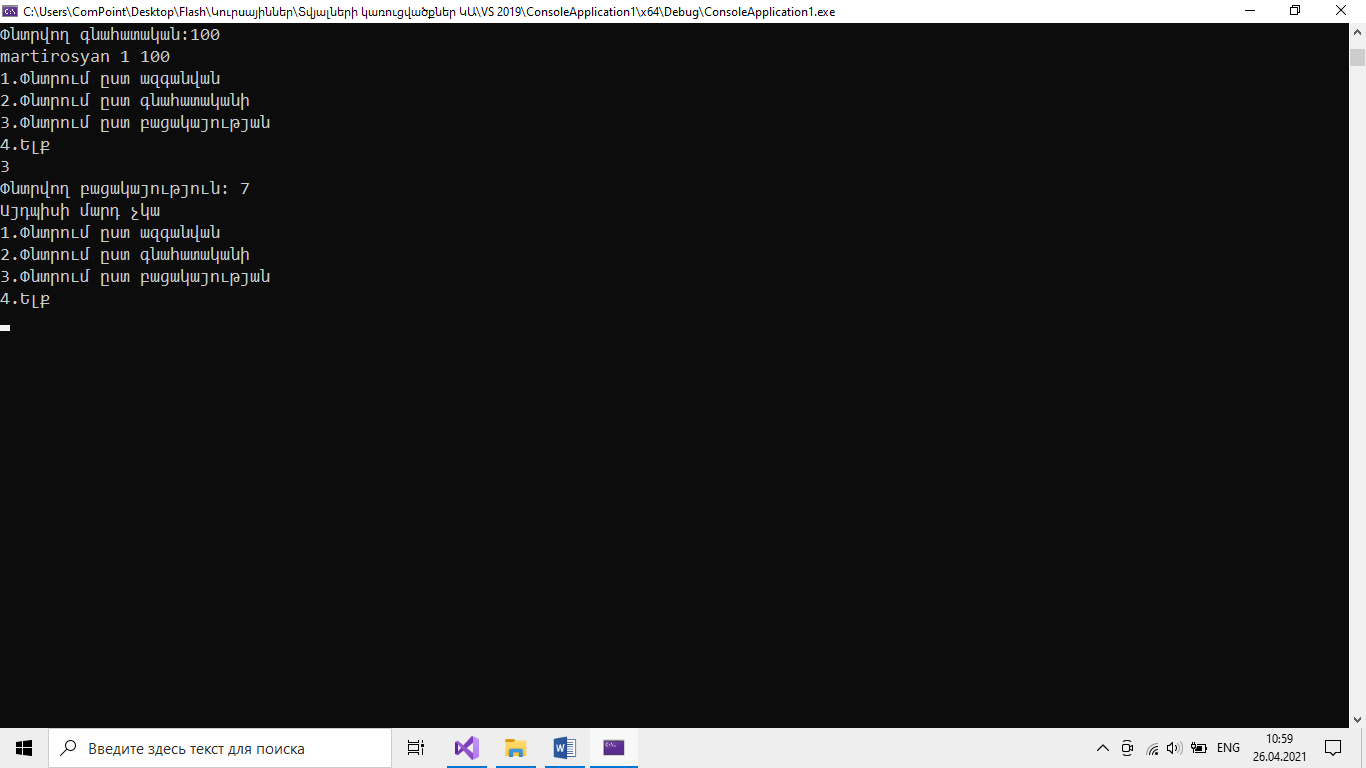


Քանի որ 65 գնահատականը ստացած ուսանող չունենք ցուցակում, տեսնում ենք այդպիսի մարդ չկա հաղորդագրությունը։ Նորից կատարենք փնտրում ըստ գնահատականի և ներմուծենք 100 գնահատականը։

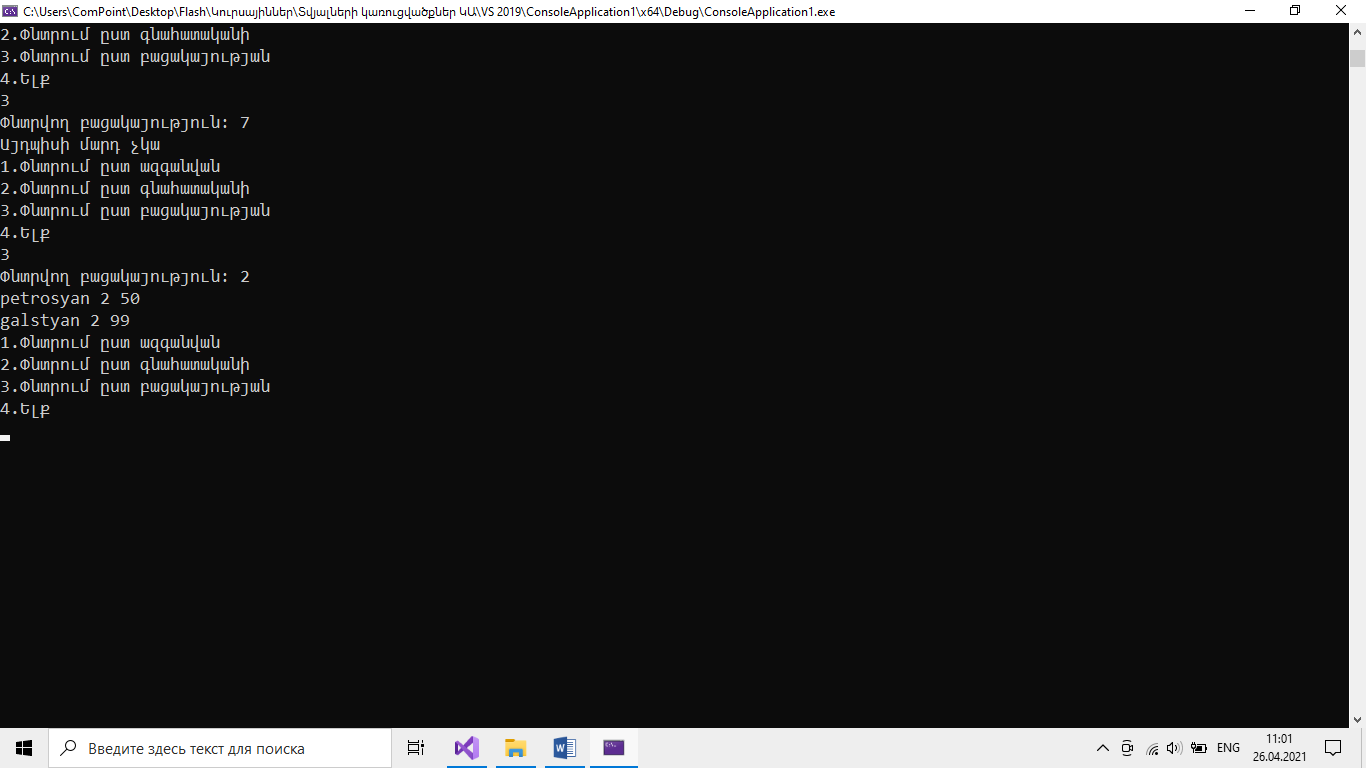


Ստացանք martirosyan ազգանունով ուսանողի տվյալները, քանի որ միայն նա էր ստացել 100 գնահատականը։

Այժմ կատարենք փնտրում ըստ բացակայությունների։ Ներմուծենք 7 թիվը։

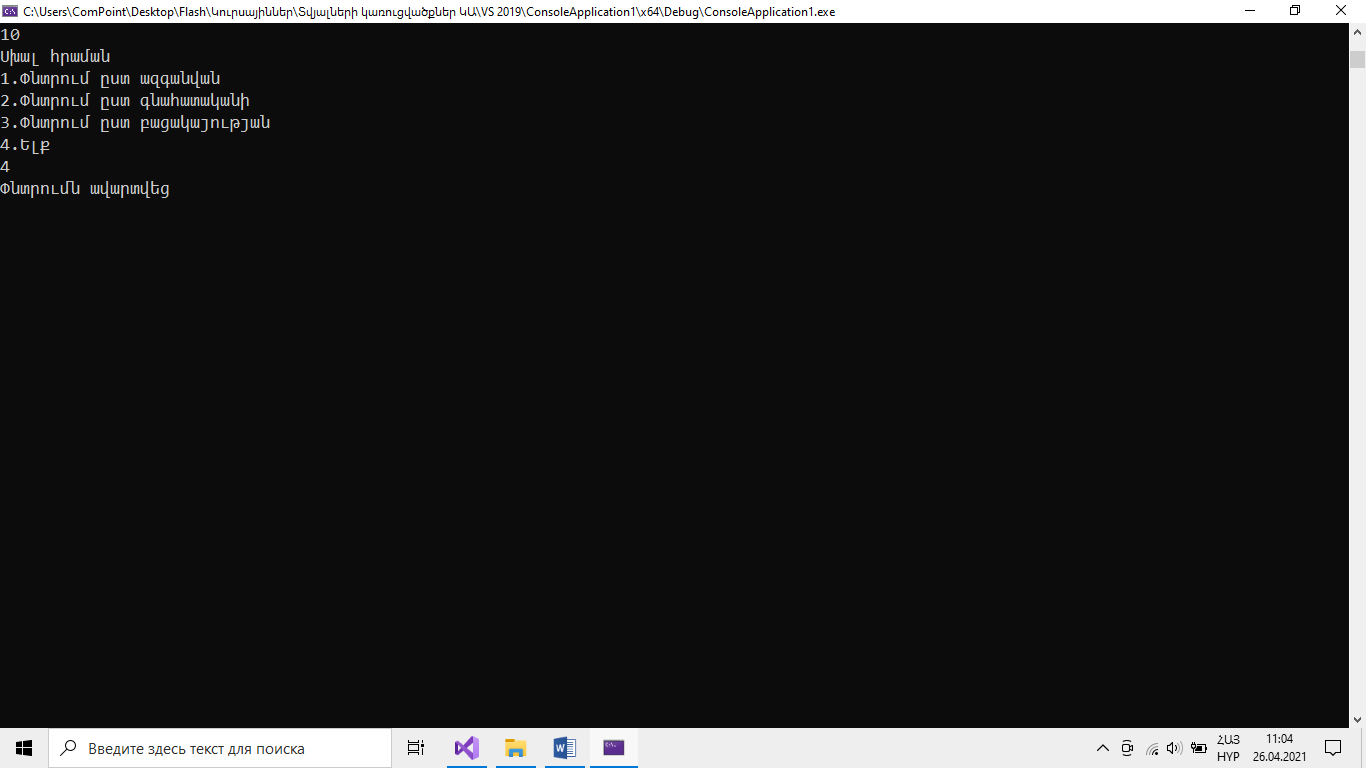


Քանի որ 7 բացակա ունեցող ուսանող չունենք, տեսնում ենք այդպիսի մարդ չկա հաղորդագրությունը։ Այժմ նորից կատարենք փնտրում ըստ բացակայությունների և ներմուծենք 2 թիվը։

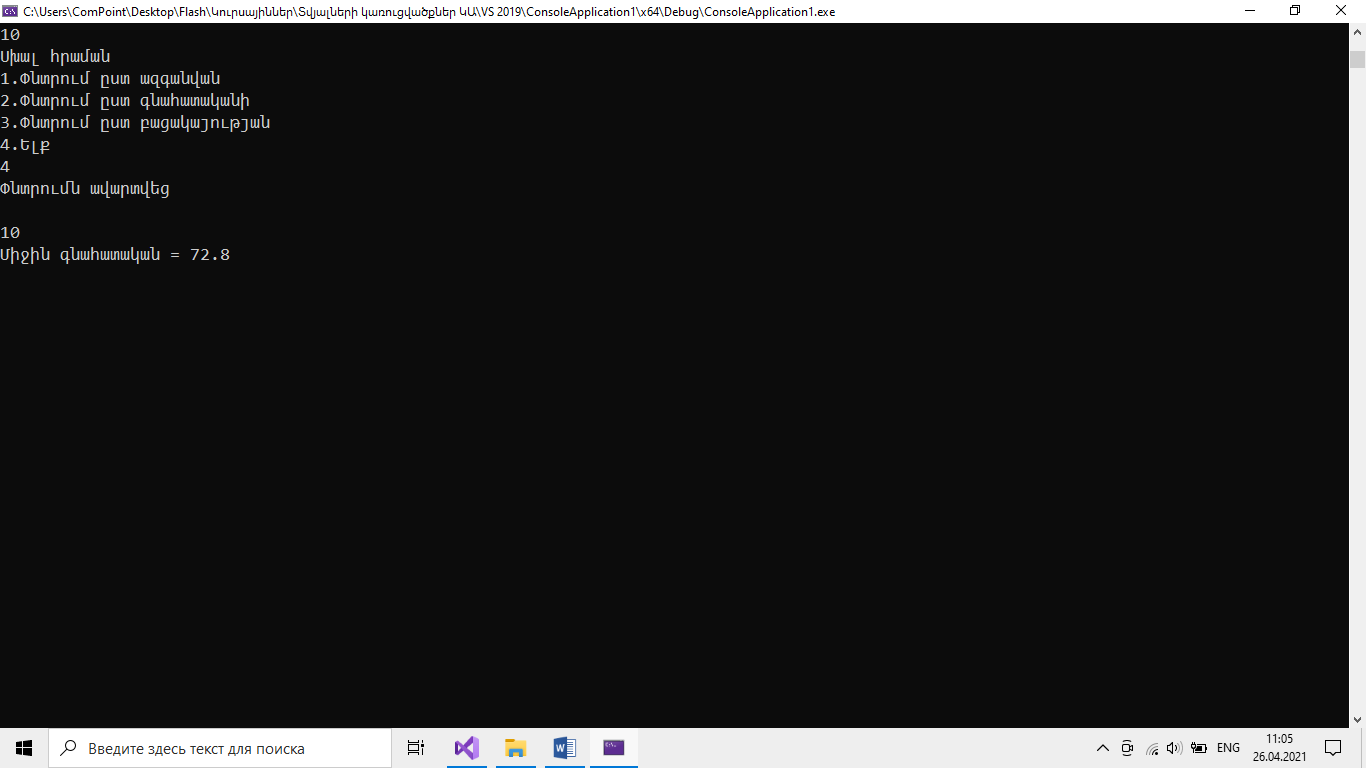


Տեսնում ենք galstyan և petrosyan ազգանունով ուսանողների տվյալները, քանի որ նրանք ունեն 2 հատ բացակայություն։

4 կետով դուրս գանք փնտրման հրամանաից։

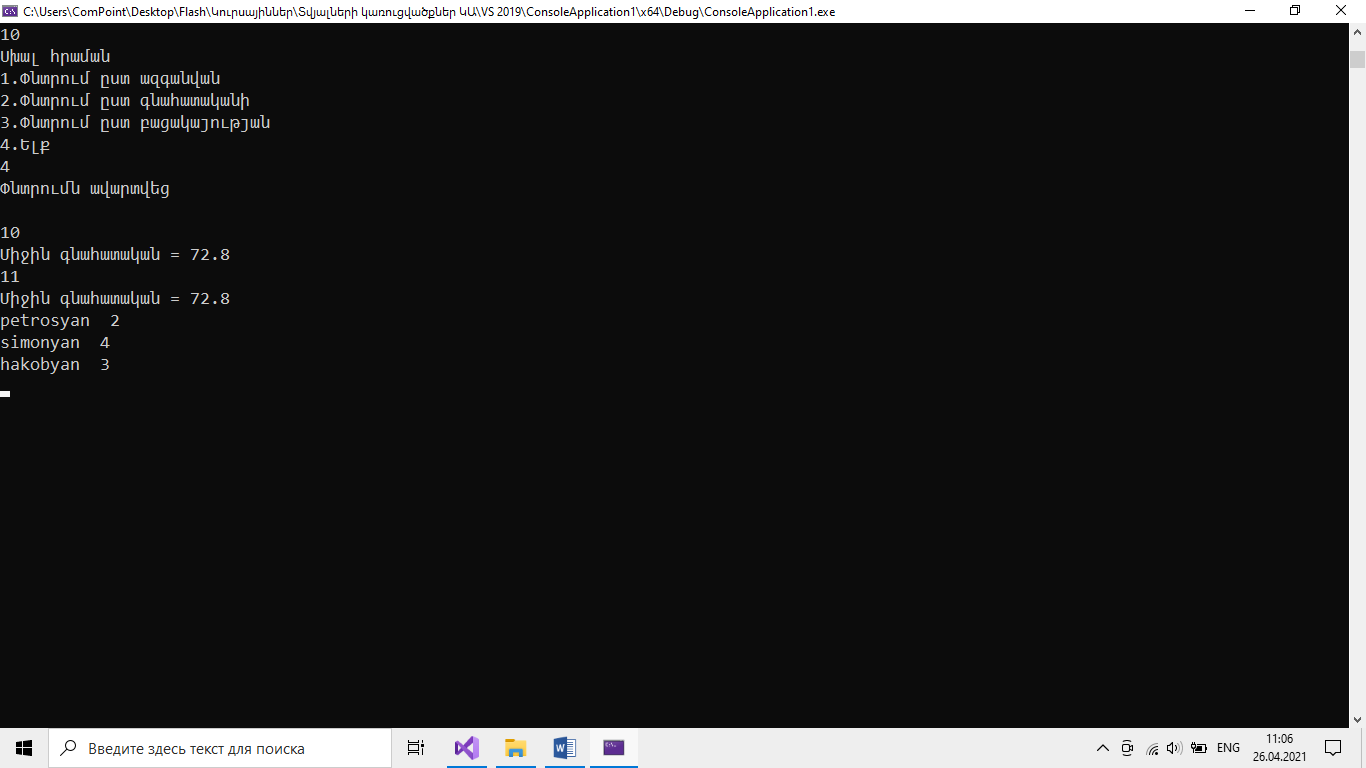


Այժմ կատարենք ուսանողների միջին գնահատականի հաշվվում։



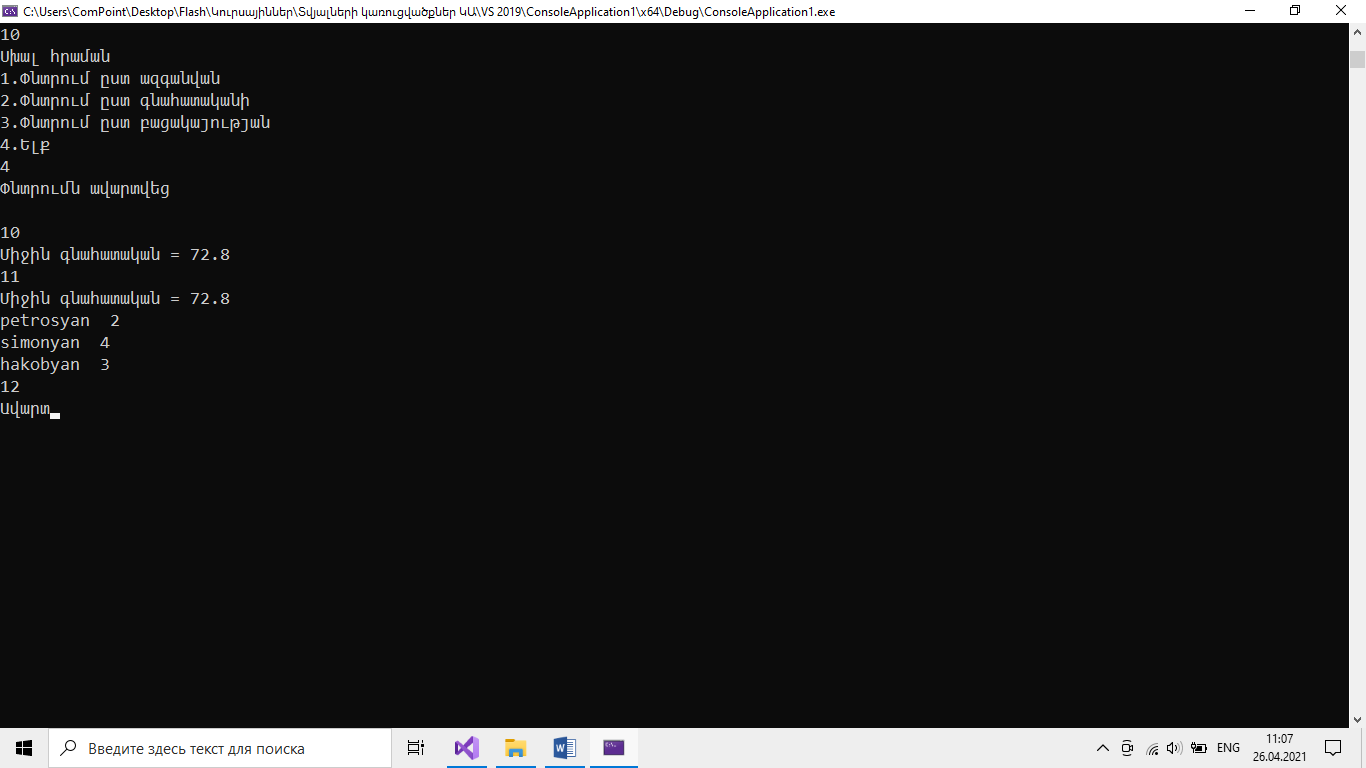
Տեսնում ենք, որ բոլոր ուսանողների միջին գնահատականը հավասար է 72.8-ի։

Կատարենք այն ուսանողների տվյալների արտածումը, ով ստացել է 72.8-ից ցածր գնահատական։



Ստացանք petrosyan,simonyan և hakobyan ազգանունով ուսանողների տվյալները, քանի որ նրանց գնահատականը փոքր է 72.8-ից։

Այժմ ավարտենք ծրագրի աշխատանքը։



# **ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ**

Այս աշխատանքը հանդիսանում է երկկապ գծային ցուցակի տեսքով ներկայացված ինֆորմացիայի մշակման օրինակ։ Աշխատանքի արդյունքում ստացանք ուսանողների տվյալների ստեղծման և մշակման ծրագիր։

Սովորեցինք, թե ինչ մեթոդներով է կարելի աշխատել ծավալուն ցուցակների հետ։

Աշխատանքի արդյունքը ցույց տվեց դինամիկ տվյալների կառուցվածքների հիմնական առանձնահատկությունները, այն են՝ դրանց չափերի և կառուցվածքների փոփոխության հնարավորությունը ծրագրի աշխատանքի ընթացքում։

Հասկացանք, որ դինամիկ տվյալների կառուցվածքների կիրառումը ինֆորմացիայի մշակումը դարձնում է ավելի ճկուն և հնարավորություն է ընձեռնում կամայական պահի կատարել անհրաժեշտ ինֆորմացիայի վերամշակում։

# **ԳՐԱԿԱՆՈՒԹՅՈՒՆ**

1. Data structures and algorithms in C++: Adam Drozdek
2. Համացանց