ლაბორატორიული მეცადინება 2: შემავალი იტერატორები

განსახილველი საკითხები:

- **ამოცანა 1** /იტერატორების დიაპაზონში ძებნის ფუნქციები/
    1. ფუნქციების კოდი >>>
    2. დამატებითი ბიბლიოთეკების გამოყენება >>>
    3. ფაილების შიგთავსი >>>
    4. პროგრამა-დრაივერი >>>
- **ამოცანა 2** /ძებნა (კლასის) ობიექტებს შორის/ >>>
    1. კლასის ფაილი >>>
    2. კლასის იპლემენტაცია >>>
- **სავარჯიშოები** >>>

ამოცანა 1. ფაილებში “ints.txt”, “reals.txt”, “strings.txt” წერია, შესაბამისად, რამდენიმე მთელი რიცხვი, რამდენიმე ნამდვილი რიცხვი და რამდენიმე სტრინგი.

იტერატორების `[first,last)` დიაპაზონში, მითითებული ობიექტის, ან გარკვეული თვისების მქონე ობიექტის იტერატორის მოძებნისა და დაბრუნებისთვის, გადატვირთეთ ტემპლიტიანი ფუნქციის ორი ვარიანტი.

ძირითად ფუნქციაში, ფაილებიდან მონაცემები ჩაწერეთ სხვადასხვა კონტეინერში, თითოეული მათგანისთვის გამოიძახეთ შესაბამისი ფუნქცია. მიღებული შედეგები დაბეჭდეთ. საქაღალდე, რომელშიც მოთავსებულია ალგორითმების ფაილი, დაამატეთ პროექტს როგორც დამატებითი ბიბლიოთეკა.

ამოხსნა:

<<< ფუნქციების კოდი

```
#pragma once
template<typename InputIterator, typename T>
InputIterator myFind(InputIterator first, InputIterator last, T b)
{
	while (first != last && *first != b)
		++first;
	return first;
}
template<typename InputIterator, typename Predicate>
InputIterator my_find_if(InputIterator first, InputIterator last, Predicate pred)
{
	while (first != last && !pred(*first))
		++first;
	return first;
}
```

<<< დამატებითი ბიბლიოთეკების გამოყენება. C++ ენის მომხმარებლებს ხშირად უწევთ დამატებითი ბიბლიოთეკების გამოყენება. მაგალითად, boost, algLIB და სხვა. Visual Studio, ისევე როგორც სხვა ინტეგრირებული გარემო, პროგრამისტს აძლევს საშუალებას რომ მან ისარგებლოს საკუთარი ბიბლიოთეკით, რომლის მისამართსაც ჩართავს პროექტში. მაგალითად, თუ ზემოთ მოყვანილ ორ ფაილს მოვათავსებთ ფაილში “`D:\muLibrary\find.h`”, მაშინ პროექტში ამ ფუნქციების გამოყენებითვის საკმარისია ჩავრთოთ ეს ფაილი შეტყობინებით: `#include"find.h"` და პროექტის თვისებებში, Configuration Properties\C/C++\Additional Include Directories -ში მივუთითოთ მისამართი D:/myLibrary/

ამის შემდეგ, ამ ალგორითმებს ხედავს პროექტი ისევე, როგორც დაინახავდა ფაილი რომ პროექტში ყოფილიყო დამატებული Project მენიუდან Add New Item -ის საშუალებით.

<<< ფაილების შიგთავსი:

"strings.txt"-ში:

```
Kote Gia SaqarTvwlo Usa Iran  home
```

"ints.txt"-ში:

```
11 3142 421 -342 105
```

"reals.txt"-ში:

```
8.12  0.1  -4.43 55.3 12.3
```

<<< პროგრამა-დრაივერი:

```
#include<iostream>
#include <string>
#include <fstream>
#include <vector>
#include <list>
using namespace std;
#include"find.h"
int main()
{
	vector<string> vec;
	{
		ifstream ifs("strings.txt");
		string s;
		while (ifs >> s) vec. emplace _back(s);
	}
	vector<string>::iterator i = myFind(vec.begin(), vec.end(), "SaqarTvwlo");
	if (i != vec.end())
	{
		cout << "In vector of strings found: " << *i << endl;
		++i;
		if (i != vec.end())
			cout << "Next is  " << *i << endl << endl;
	}
	vector<int> n;
	{
		ifstream ifs("ints.txt");
		int j;
		while (ifs >> j)  n. emplace _back(j);
	}
	auto j = my_find_if(n.begin(), n.end(), [](int n) {return n < 0; });
	if (j != n.end())
	{
		cout << "In vector of ints, the first negative is: " << *j << endl;
		++j;
		if (j != n.end())
			cout << "Next is  " << *j << endl << endl;
	}
	list<double> realList;
	{
		ifstream ifs("reals.txt");
		double x;
		while (ifs >> x) realList.emplace_front(x);
	}
	auto r = my_find_if(realList.begin(),realList.end(), [](double n){return n<0; });
	if (r != realList.end())
	{
		cout << "In list of reals, the first negative is: " << *r << endl;
		++r;
		if (r != realList.end())        cout << "Next is  " << *r << endl << endl;
	}
}
```

შედეგს აქვს სახე:

```
In vector of strings found: SaqarTvwlo
Next is  Usa

In vector of ints, the first negative is: -342
Next is  105

In list of reals, the first negative is: -4.43
Next is  0.1

Press any key to continue . . .
```

<<< **ამოცანა 2** /ძებნა (კლასის) ობიექტებს შორის/. **"volcanoes.txt"** ფაილში წერია რამდენიმე ვულკანის მონაცემები:

```
9
Etna                3340        Italy
Hekla               1491        Island
Vezuvius            1277        Italy
Kluchevskaia_sopka  4750        Kamchatka
Fujiyama            3776        Japan
Krakatau            813         Azia
Kamerun             4070        Kamerun
Stromboli           926         Italy
Hipokatepetl        5452        Mexico
```

ხოლო პირველ სტრიქონში მათი რაოდენობა (ეს გვჭირდება შეტანის გაადვილებითვის, რადგან ოპერატორების გადატვირთვას ჯერ არ ვიყენებთ). მოძებნეთ პირველივე ვულკანი, რომლის სიმაღლე 4000 მეტრზე მეტია.

მითითება: პროექტში მიუთითეთ დამატებითი ბიბლიოთეკა, შემდეგ დაამატეთ კლასი. ძირითად ფუნქციაში, მონაცემები ფაილიდან ჩაწერეთ კონტეინერში და გამოიძახეთ შესაბამისი ალგორითმი წინა ამოცანაში შექმნილი გარე ბიბლიოთეკიდან. მიღებული შედეგი დაბეჭდეთ.

შენიშვნა: კლასი და კლასის წევრები შექმენით V Studio-ს საშუალებების გამოყენებით (Class View, Add Class, Add Function და ა. შ.).

ამოხსნა: კლასის ფაილს დაპროგრამების გარემო ავტომატურად არქმევს კლასის სახელს შესაბამისი გაფართოებით - "Volcano.h" და მას შესაძლოა ჰქონდეს სახე:

<<< კლასის ფაილი

```
#pragma once
class Volcano
{
public:
	string name;
	int height;		// in meters
	string location;
	Volcano();
	~Volcano();
	Volcano(ifstream& x);
	void prnt();
	Volcano(string itsName, int itsHeight, string itsLocation);
};
```

<<< კლასის იმპლემენტაცია

მოთავსებლია "Volcano.cpp" ფაილში და მას აქვს სახე:

```
#include<iostream>
```

```
#include <string>
#include <fstream>
using namespace std;
```

```
#include "Volcano.h"
Volcano::Volcano(): height(0){}
Volcano::~Volcano() {}
Volcano::Volcano(ifstream& x)
{
	x >> name >> height >> location;
}
void Volcano::prnt()
{
	cout << name << ", height " << height << ", in " << location << endl;
}
Volcano::Volcano(string itsName, int itsHeight, string itsLocation)
{
	name = itsName;		height = itsHeight;	location = itsLocation;
}
```

კლასის ერთ-ერთ წევრს წარმოადგენს კონსტრუქტორი, რომლის პარამეტრი წარმოადგენს ნაკადს.

```
#include<iostream>
#include <string>
#include <fstream>
#include <vector>
using namespace std;
#include"find.h"
#include"volcano.h"

int main(){
	vector<Volcano> v;
	int n;
	ifstream ifs("volcanoes.txt");
	ifs >> n;
	for (int i = 0; i < n; ++i)
	{
		Volcano tmp(ifs);
		v.push_back(tmp);
	}
	auto lm = [](Volcano a) {return a.height > 4000; };
	auto i = my_find_if(v.begin(), v.end(), lm);
	if (i != v.end())
	{
		cout << "Volcano, higher or equal to 4000:" << endl << endl;
		i->prnt();
		++i;
		if (i != v.end())
			cout << "The next one is:" << endl << endl;
		i->prnt();
	}
}
```

რადგან ჩვენი კლასი და ბიბლიოთეკა დამატებულია `using namespace std;` სტრიქონის შემდეგ, ამიტომ ლინკერი მათ აკინძავს ზემოთ მითითებული ბიბლიოთეკების შემდეგ. სხვა შემთხვევაში, ჩვენს მიერ შექმნილ ფაილებში დაგვჭირდებოდა `#include` -ების გამოყენება.

<<< სავარჯიშოები.

1. პირველი ამოცანის პროგრამა დრაივერში დაამატეთ გარე ბიბლიოთეკაში მოთავსებული ერთ-ერთი ფუნქციის გამოძახება შემავალი ნაკადზე.

2. შემდეგი ცხრილებისთვის შექმენით კლასი სტრიქონში მოყვანილი მონაცემების მიხედვით. ზოგიერთი ცხრილის სტრიქონების შესაბამისი ობიექტები ჩაწერეთ ვექტორებში, სხვა ცხრილების სტრიქონების შესაბამისი ობიექტები ჩაწერეთ სიებში, და მოძებნეთ საჭირო ობიექტი გარკვეული ველის და მითითებული თვისების მიხედვით. კლასთან ერთად გაითვალისწინეთ ობიექტების შესადარებელი ბინარული პრედიკატის შექმნა.

ა. მონაცემები საქართველოს ტბების შესახებ:

ტბა	ზედაპირის ფართ., კმ^2	სიმაღლე ზღვის დონ., მ	მაქსიმ. სიღრმე, მ	ტბა	ზედაპირის ფართ	სიმაღლე	მაქსიმ. სიღრმე
ფარავანი	37.5	2073	3.3	ჯანდარი	10.6	291	7.2
პალიასტომი	18.2	-0.3	3.2	რიწა	1.49	884	101
ტაბაწყური	14.2	1997	40.2	ბაზალეთი	1.22	879	7

ბ. მონაცემები საქართველოს მდინარეების შესახებ:

მდინარე	სიგრძე (კმ)	მდინარე	სიგრძე (კმ)
მტკვარი	1515	ცხენისწყალი	184
ალაზანი	407	იორი	351
რიონი	333	სუფსა	117
ენგური	206	არაგვი	112
ხრამი	220	-	-

გ. მონაცემები კუნძულების მდებარეობის შესახებ:

კუნძულის დასახელება	მდებარეობა	კუნძულის დასახელება	მდებარეობა
გრენლანდია	ჩრდილ. ამერიკა	კუბა	ჩრდილ. ამერიკა
ახალი გვინეა	ოკეანია	ისლანდია	ევროპა
სუმატრა	აზია	ირლანდია	ევროპა
დიდი ბრიტანეთი	ევროპა	ჰოკაიდო	აზია
ჰონსიუ	აზია	ჰაიტი	ჩრდილ. ამერიკა
იავა	აზია	სახალინი	აზია
ახალი ზელანდია	ოკეანია	მადაგასკარი	აფრიკა

დ. მონაცემები მზის სისტემის პლანეტების შესახებ:

პლანეტა	დიამეტრი (კმ)	მანძილი მზემდე (მლნ კმ)	ორბიტული სიჩქარე (კმ/წმ)	პლანეტა	დიამეტრი	მანძილი მზემდე	ორბიტული სიჩქარე
მერკური	5000	58	48	სატურნი	120000	1500	7
ვენერა	12400	108	30	ურანი	51000	2900	5
დედამიწა	12700	150	24	ნეპტუნი	50000	4500	5
მარსი	6800	228	13	პლუტონი	18000	5910	4
იუპიტერი	140000	778	10	-	-	-	-