ლაბორატორიული მეცადინება 1: ფუნქციის თარგი

განსახილველი საკითხები:

- ამოცანა 1 /მაქსიმალის განსაზღვრის ტემპლიტიანი ფუნქცია/
- ამოცანა 2 ფუნქციაში მასივის გადაწოდება/ >>>
- სავარჯიშოები >>>

ამოცანა /მაქსიმალის განსაზღვრის ტემპლიტიანი ფუნქცია/. შექმენით ფუნქციის თარგი, რომელიც ვექტორში მოძებნის და დააბრუნებს მაქსიმალური ელემენტის ინდექსს. პროგრამა დრაივერში, ფუნქცია გამოიყენეთ “ints.txt”, “reals.txt”, “strings.txt” ფაილებიდან შევსებული მთელი, ნამდვილი და სტრინგების ვექტორებისთვის. შედეგები დაბეჭდეთ. სიმარტივისთვის იგულისხმეთ, რომ ვექტორები არაცარიელია.

ამოხსნა: მაქსიმალური ელემენტის ინდექსის განსაზღვრის ფუნქცია ბევრნაირად შეიძლება გაკეთდეს. ერთი შესაძლო ვარიანტი ასეთია:

```
template<typename T>
int maximal(const vector<T> &x)
{
    int ind(0);
    for(int i=1; i<x.size(); i++)
        if( x[ind] < x[i] )
            ind = i;
    return ind;
}
```

მოვათავსოთ ეს ფუნქცია ცალკე ფაილში სახელით ”myLibrary.h”, რომელიც უნდა დავამატოთ პროექტს.

ვთქვათ აგრეთვე, რომ სამ ფაილში წერია შემდეგი მონაცემები:

```
"strings.txt"-ში:
sapienti sat, veni vedi vici

"ints.txt"-ში:
11 3142 421 -342 10

"reals.txt"-ში:
12.32  65.9 -12.09 77.32  9348.1 -43.33
```

მთავარ ფუნქციაში, პირველ რიგში უნდა შევქმნათ და შევავსოთ ვექტორები, შემდეგ, - გამოვიყენოთ ჩვენი ფუნქცია.

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
using namespace std;
#include"myLibrary.h"
int main()
{
      vector<int> n;
      int j;
      {
            ifstream ifs("ints.txt");
            while (ifs >> j)
                  n.push_back(j);
      }
      cout << "In \"ints.txt\", offset of maximum equals "
            << maximal(n) << endl;

      vector<string> vec;
```

```
	string s;
	{
		ifstream ifs("strings.txt");
		while (ifs >> s)
			vec.push_back(s);
	}
	cout << "In \"strings.txt\", offset of maximum equals "
		<< maximal(vec) << endl;

	vector<double> realVect;
	double x;
	{
		ifstream ifs("reals.txt");
		while (ifs >> x)
			realVect.push_back(x);
	}
	cout << "In \"reals.txt\", offset of maximum equals "
		<< maximal(realVect) << endl;
}
```

შედეგად ეკრანზე იბეჭდება:

```
In "ints.txt", offset of maximum equals 1
In "strings.txt", offset of maximum equals 4
In "reals.txt", offset of maximum equals 4
Press any key to continue . . .
```

შევსება ხდება ლოკალურ ბლოკებში, ამიტომ შეგვიძლია ერთი და იგივე სახელი `ifs` გამოვიყენოთ ნაკადისთვის. იგივე შედეგის მიღწევა შიძლება, თუ გამოყენების შემდეგ დავხურავთ ნაკადს, შემდეგ ხელახლა გავხსნით და მივაბამთ საჭირო ფაილს:

```
int main(){
	vector<int> n;
	int j;
	ifstream ifs("ints.txt");
	while (ifs >> j)
		n.push_back(j);
	ifs.close();
	cout << "In \"ints.txt\", offset of maximum equals "
		<< maximal(n) << endl;

	vector<string> vec;
	string s;
	ifs.open("strings.txt");
	while (ifs >> s)
		vec.push_back(s);
	ifs.close();
	cout << "In \"strings.txt\", offset of maximum equals "
		<< maximal(vec) << endl;

	vector<double> realVect;
	double x;
	ifs.open("reals.txt");
	while (ifs >> x)
		realVect.push_back(x);
	ifs.close();
	cout << "In \"reals.txt\", offset of maximum equals "
		<< maximal(realVect) << endl;
}
```

პრობლემა 1: ჩვენს მიერ შექმნილი ფუნქცია წარმატებით მუშაობს ისეთ ვექტორებზე, რომლებიც სპეციალიზებულია C++ ენის მარტივ ტიპებზე. მომხმარებლის მიერ შექმნილი კლასის ობიექტებთან ამ ფუნქციას პრობლემა ექმნება **if**`( x[index] < x[i] )` სტრიქონის გამო. შესაბამისად, კლასის ობიექტებზე ან გადატვირთული უნდა იყოს "<" ოპერაცია, ან ფუნქციას უნდა დავუმატოთ მესამე არგუმენტი - ბინარული პრედიკატი (როგორიც აქვს sort-ს, მაგალითად).

პრობლემა 2: მეორე პრობლემა ისაა, რომ ვექტორის შევსება ვერ მოხდება ასეთივე ფორმით:

```
while(ifs >> a)
    v.push_back(a);
```

თუ კლასის ობიექტებზე გადატვირთული არაა შეტანის ოპერატორი ">>".

<<< ამოცანა 2 /ფუნქციაში მასივის გადაწოდება/ განვიხილოთ ასეთი საკითხი: როგორ უნდა გდავაწოდოთ მასივის სახელი ფუნქციას, რომ მან შეინარჩუნოს თავისი ტიპი (და, შესაბამისად, თავისი ატრიბუტები).

ამოხსნა: პირველ რიგში გამოვკვეთოთ პრობლემა. ვთქვათ, გვინდა რაიმე მასივის ელემენტების დაბეჭდვა. თუ ფუნქციის პარამეტრია პოინტერი და ფუნქციას გადავაწვდით მხოლოდ მასივის სახელს, ფუნქციის შიგნით, მაშინ პარამეტრი ინიციალიზაციის შემდეგ დაკარგავს მასივის ელემენტების რაოდენობას, რასაც გვიჩვენებს შემდეგი მაგალითი:

```
#include <iostream>
using namespace std;

int sz(int* p) noexcept
{
	return sizeof(p);
}

int main()
{
	int m[] = { 1,2,3,4,3,2,1 };
	cout << sz(m) << endl;
}
```

რომლის შესრულების შემდეგ დაიბეჭდება 4 ბაიტების რაოდენობა, რაც სჭირდება პოინტერს და არა მასივს.

ასეთ შემთხვევებში უაღრესად ბუნებრივი არის ფუქნციის თარგის გამოყენება. თუ ფუნქციას მივცემთ სახეს:

```
template<typename T>
int sz(T& p) noexcept
{
	return sizeof(p);
}
```

პროგრამა დაბეჭდავს უკვე 28-ს. რომ საბოლოოდ დავრწმუნდეთ შედეგში, ვნახოთ ჩვენი ამოცანის ამოხსნა (რომელიც არ იმუშავებს სტრინგების მასივისთვის, რადგან სტრიგის ზომა ცვალებადია):

```
#include <iostream>
using namespace std;

template<typename T>
void printArray(T& p) noexcept
{
	int sz = sizeof(p) / sizeof(p[0]);
	for (int i = 0; i < sz; ++i)
		cout << p[i] << '\t';
```

```
	cout << endl;
}

int main()
{
	int m[] = { 1,2,3,4,3,2,1 };
	printArray(m);
}
```

<<< სავარჯიშოები:

1. დააპროგრამეთ პრაქტიკული 1-ის ამოცანები (დავალებების ჩათვლით).
2. ამოცანა 1-ში, როგორ შეიცვლება ალგორითმის კოდი და ფუნქციის გამოძახება, თუ მოვხსნით დაშვებას იმის თაობაზე რომ ვექტორი არაცარიელია?
3. ამოცანა 1-ში, რა შეიცვლება თუ ვექტორის ნაცვლად მოვინდომებთ სიის გამოყენებას?
4. ამოცანა 1-ში, მაქსიმალი შეცვალეთ მინიმალით და შესაბამისად შეცვალეთ პროგრამები.
5. ამოცანა 1-ში, მაქსიმალურის ინდექსი მოვძებნოთ ვექტორის გარკვეული თვისების მქონე ელემენტებს შორის.
6. ამოცანა 1-ში, მაქსიმალურის ინდექსი მოვძებნოთ ვექტორის გარკვეულ ფრაგმენტში. ვთქვათ, სადაც ინდექსი იცვლება ინდექსების `[p.q)` დიაპაზონში.
7. ამოცანა 1-ში, როგორ შევავსებდით ვექტორებს კლავიატურიდან?
8. აღნიშნულ ამოცანებში მაქსიმალურის ინდექსი მოვძებნოთ რეკურსიული ალგორითმით.
9. ამოცანა 2-ში, ყველაზე მაღალის ნაცვლად, მოძებნეთ ა) ყველაზე გრძელი სახელის მქონე ვულკანი; ბ) ყველაზე დაბალი ვულკანი;
10. ბოლო ამოცანაში, კლასი შექმენით Visual Studio -ს კლასის დამატების ინსტრუმენტების გამოყენებით.
11. ფაილში "`volcanos.in`" წერია რამდენიმე ვულკანის მონაცემები. შეადგინეთ პროგრამა, რომელიც შექმნის და ფაილიდან შეავსებს ვულკანების ვექტორს, შემდეგ დათვლის
    ა. 1000 მეტრზე მაღალი ვულკანების რაოდენობას;
    ბ. იტალიაში მდებარე ვულკანების რაოდენობას.

შეგიძლიათ გამოიყენოთ და გაუმჯობესოთ მსგავსი პროექტი, რომელიც შედგება შემდეგი ფაილებისგან:

```
//"volcano.h" - კლასის ფაილი
class Volcano
{
  public:
    string name;
    int height;
    string location;
    Volcano(){}
    Volcano(ifstream & );
    void printVolcano(void);
};
Volcano :: Volcano(ifstream & x ) {
   x >> name >> height >> location;
}
void Volcano :: printVolcano(){
         cout<< name <<", its height is " << height <<" meters"<<endl
                <<"It is located in "<< location<<endl;
```

```
}
```

ფაილი "volcanoes.txt", რომელშიც წერია ვულკანების რაოდენობა და 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
```

და პროგრამა დრაივერი:

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

int main()
{
	ifstream fin("volcanoes.txt");
	vector<Volcano> vol;
	int dim;
	fin >> dim;

	for (int i = 0; i < dim; i++)
	{
		Volcano tmpVolcano(fin);
		vol.push_back(tmpVolcano);
	}
	Volcano* p(&vol[0]);
	for (int i = 1; i < vol.size(); i++)
		if (p->height < vol[i].height)
			p = &vol[i];
	cout << "Maximum height has volcano\n";
	p->printVolcano();
}
```