ლაბორატორიული მეცადინება 1: ფუნქციის თარგი

განსახილველი საკითხები:

- მაქსიმუმის განსაზღვრა
- ფუნქციაში მასივის გადაწოდება >>>
- სავარჯიშოები >>>

მაქსიმუმის განსაზღვრა. გავიხსენოთ, თუ რა საშუალებები აქვს STL ბიბლიოთეკას დიაპაზონში მაქსიმალური ელემენტის განსაზღვრისთვის.
შესაძლოა ორი შემთხვევა: თუ არსებობს ბინარული მიმართება დიაპაზონის ელემენტებისთვის და ჩვენ გვაწყობს ეს მიმართება, მაშინ გამოვიძახებთ std::max_element-ს ორი არგუმენტით: დიაპაზონის თავით და ბოლოთი. თუ არსებობს ბინარული მიმართება, მაგრამ ჩვენ გვინდა სხვანაირად შევადაროთ (მაგალითად, ყველაზე გრძელი სტრინგი მოვძებნოთ), ან საერთოდაც არაა იმართება განსაზღვრული (მაგალითად ვულკანების კლასზე), მაშინაც ჩვენ გვიწევს ბინარული პრედიკატის შექმნა და გადაცემა ფუნქციისთვის.

```
#include <iostream>
#include <string>
#include <vector>
#include <list>
#include <algorithm>
using namespace std;

int main()
{
	string s0{ "ADA" }, s1{ "FORTRAN" }, s2{ "C++" };
	vector<string> vs{ s0,s1,s2,(string)"CSPlus" };

	//მოვძებნოთ ვექტორში ყველაზე გრძელი სტრინგი
	auto lm = [](const string& a, const string& b) {return (a.length() < b.length());
};
	cout << "The lengthiest string:  " << *max_element(vs.begin(), vs.end(), lm) <<
endl;

	using
		pair = pair<double, double>;

	pair a(1.1, 0.5), b(0.5, 1.1), c(0.67, 0.75);
	list< pair> lp{ a,b,c };
	auto compPairs = [](const pair& p, const pair& q)
	{return p.second < q.second || (p.second == q.second && p.first < q.first); };
	pair ans = *max_element(lp.begin(), lp.end(), compPairs);
	cout << "The max pair:  " <<  ans.first << "  " << ans.second << endl;
}
```

პასუხი:

```
The lengthiest string:  FORTRAN
The max pair:  0.5  1.1

C:\Users\user\source\repos\tmp\Debug\tmp.exe (process 1708) exited with code 0.
Press any key to close this window . . .
```

სტრინგებზე არსებობს ლექსიკოგრაფიული მიმართება, მაგრამ ჩვენ ვეძებთ ყველაზე გრძელ სტრინგს და ამიტომ ვაკეთებთ და ვაწვდით საჭირო პრედიკატს.

წყვილებზე ვეძებთ აგრეთვე ლექსიკოგრაფიული დალაგებით, რადგან სხვა დალაგება შესაძლოა ყველგან განსაზღვრული, ანუ ჯაჭვი არ აღმოჩნდეს. ოღონდ, უპირატესობას ვაძლევთ მეორე კომპონენტს.

ყურადღება მივაქციოთ, თუ როგორ ხდება მონაცემის ტიპის სახელის შემოკლება using -ის გამოყენებით.

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

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 ფუნქციის 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. განხილულ ამოცანებში, გასინჯეთ სხვა კონტეინერებიც.
3. განხილულ ამოცანებში, მაქსიმალი შეცვალეთ მინიმალით და შესაბამისად შეცვალეთ პროგრამები.
4. განხილულ ამოცანებში, როგორ შევავსებდით ვექტორებს კლავიატურიდან?
5. 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 გააუმჯობესოთ 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;
```

```cpp
#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 << "Volcano, having the maximum height\n";
	p->printVolcano();
}
```