თემა 3. კლასის კერძო (private) და (static) სტატიკური წევრები

განხილული საკითხები:

- მარტივი კლასის მაგალითი, რომელიც ახასიათებს ტბას
- კლასი ტბა, სტატიკური მთვლელით და სტატიკური მეთოდით >>>
- სტატიკური მთვლელისთვის გამოყოფილი ადგილის გარკვევა მეხსიერებაში >>>
- სავარჯიშოები >>>

ობიექტზე ორიენტირებული პროგრამირების სტილი დიდ მნიშვნელობას ანიჭებს ინფორმაციის დაფარვას. განვიხილოთ ერთი შემთხვევა, როდესაც აუცილებელია კლასის შესახებ გარკვეული მონაცემების დაცვა ისე, რომ ობიექტებიდან ვერ მივწვდეთ მას. ასეთ შემთხვევებში დაცული ინფორმაციის ნახვა მხოლოდ ამ მიზნით შექმნილი საზიარო (public) ფუნქციის საშუალებით ხდება.

ბევრ ამოცანაში სასარგებლოა ვიცოდეთ, თუ მოცემული კლასის რამდენი ობიექტია ამჟამად მოქმედი (შექმნილია და არავის გაუუქმებია ჯერჯერობით). საქმე იმაშია, რომ მეხსიერების დინამიკური მართვის საშუალებით შესაძლებელია ახალი ობიექტის შექმნა როცა გვჭირდება და მისთვის გამოყოფილი მეხსიერების სისტემისთვის დაბრუნება, როცა იგი არაა საჭირო. თანამედროვე ენების ნაწილში (Java, C#) ამ საქმეს აკეთებს სპეციალური ფუნქცია. როდესაც ამ საკითხებზე გადავალთ, მაშინ გავარჩევთ აგრეთვე კლასის მეთოდს, რომელიც უზრუნველყოფს ობიექტის გაუქმებას - დესტრუქტორს.

განვიხილოთ ასეთი ამოცანა. უნდა შევქმნათ უმარტივესი კლასი, რომელიც შეგვაქმნევინებს ობიექტებს ტბების აღწერისთვის. ვიგულისხმოთ, რომ ტბა ხასიათდება მხოლოდ ორი მახასიათებლით: სახელი და ფართობი. ასევე სიმარტივისთვის, შევქმნათ მხოლოდ უპარამეტრო კონსტრუქტორი და ტბის ობიექტის შესახებ მონაცემების სტრინგად წარმოდგენის ფუნქცია:

მარტივი კლასის მაგალითი, რომელიც ახასიათებს ტბას

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

class Lake
{
public:
	string name;
	int area;
	Lake() {};
	string toString()
	{
		return ((string)typeid(this).name() + ":  name: " + name +", area: " +
		to_string(area));
	}
};
int main()
{
	Lake tba;
	tba.name = "Pali",
	tba.area = 1230;
	cout << tba.toString() << endl;
}
```

აქ ყველაფერი გასაგებია, შეიქმნება ერთი ობიექტი და მის ველებს მნიშვნელობები მიენიჭა კლავიატურიდან. შემდეგ იბეჭდება მის შესახებ ცნობები. `tba` ობიექტისთვის გამოყოფილია

მეხსიერების მონაკვეთი, რომელშიც მოთავსებულია მისი ორივე ველითვისთვის გამოყოფილი ადგილი. ამაში ადვილად დავრწმუნდებით, თუ დავბეჭდავთ შესაბამის მისამართებს.

ახლა ვიზრუნოთ იმაზე, რომ შევქმნათ არსებული ტბების რაოდენობის მთვლელი. ეს რამდენიმენაირად შეიძლება გაკეთდეს. ერთი ყველაზე გავრცელებული და ლამაზი გზა იმაში მდგომარეობს, რომ შევქმნათ მთვლელისთვის ცვლადი, რომელიც საერთო იქნება ყველა არსებული ტბის ობიექტისთვის და რომლის მნიშვნელობა გაიზრდება ერთით ყოველი ახალი ტბის კონსტრუირების მომენტში. გასაგებია, რომ არავითარი აუცილებლობა არაა, რომ რომელიმე ობიექტი მისწვდეს ამ ცვლადს.

აქვე შევნიშნოთ, რომ როდესაც ობიექტი გაუქმდება, ანუ წაიშლება, მისმა დესტრუქტორმა უნდა იზრუნოს ობიექტების რაოდენობის შემცირებაზე.

<<< კლასი ტბა, სტატიკური მთვლელით და სტატიკური მეთოდით

სიმარტივისთვის, მხოლოდ მთვლელი იყოს კერძო ველი (წევრი). შესაბამისად, გვჭირდება ერთი საზიარო (საერთო) მეთოდი (კლასის წევრი-ფუნქცია), რომელიც გვიჩვენებს მთვლელის მნიშვნელობას. პროგრამა დრაივერში ყოველი ობიექტის შექმნის ან წაშლის შემდეგ ვბეჭდავთ რაოდენობას. მთვლელის შემცირების სანახავად, ერთ ობიექტს შევქმნით და გავაუქმებთ დინამიკურად.

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

class Lake
{
private:
	static int Lake_count;
public:
	string name;
	int area;
	Lake()
	{
		++Lake_count;
		cout << "Lake_count=" << Lake_count << endl;
	};
	~Lake()
	{
		--Lake_count;
		cout << "Deleted: " << toString() << " Lake_count=" << Lake_count << endl;
	}
	static int get_Lake_count()
	{
		return Lake_count;
	}
	string toString()
	{
		return ((string)typeid(this).name() + ":  name: " + name + ", area: " +
			to_string(area));
	}
};

int Lake::Lake_count = 0;

int main()
{
	Lake tba;
	cout << "Number of lakes: " << Lake::get_Lake_count() << endl;
	tba.name = "Pali",
		tba.area = 1230;
```

```
	cout << tba.toString() << endl;
	Lake* p = new Lake;
	cout << "Number of lakes: " << Lake::get_Lake_count() << endl;
	p->name = "Ohoho";
	p->area = 1000;
	cout << p->toString() << endl;
	delete p;
	p = nullptr;
	cout << "Number of lakes: " << Lake::get_Lake_count() << endl;
}
```

განაცხადი

```
	int Lake::Lake_count = 0;
```

ამ ცვლადს მიანიჭებს საწყის მნიშვნელობას. კლასის შიგნით ამის გაკეთება არ ხერხდება. პროგრამა დრაივერის შესრულების მომენტში, ყოველი ახალი ობიექტის კონსტრუირების მომენტში ეს ცვლადი გაიზრდება ერთით და დაიბეჭდება მისი მიმდინარე მნიშვნელობა.

```
Lake_count=1
Number of lakes: 1
class Lake *:  name: Pali, area: 1230
Lake_count=2
Number of lakes: 2
class Lake *:  name: Ohoho, area: 1000
Deleted: class Lake *:  name: Ohoho, area: 1000 Lake_count=1
Number of lakes: 1
Deleted: class Lake *:  name: Pali, area: 1230 Lake_count=0
```

ახლა ვცადოთ შემოწმება, რომ Lake_count ცვლადი არაა მოთავსებული რომელიმე ობიექტის შიგნით. მხოლოდ ამ მიზნით ოდნავ გავაუარესოთ ჩვენი პროგრამა (მისი უსაფრთხოების თვალსაზრისით), რათა პროგრამა დრაივერიდან მივწვდეთ ამ ცვლადის მისამართს:

<<< სტატიკური მთვლელისთვის გამოყოფილი ადგილის გარკვევა მეხსიერებაში

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

class Lake
{
public:
	static int Lake_count;
	string name;
	int area;
	Lake()
	{
		++Lake_count;
		cout << "New lake created. Lake_count=" << Lake_count << endl;
	};
	~Lake()
	{
		--Lake_count;
		cout << "Deleted: " << toString() << " Lake_count=" << Lake_count << endl;
	}

	string toString()
	{
```

```
                return ((string)typeid(this).name() + ":  name: " + name + ", area: " +
                        to_string(area));
        }
};

int Lake::Lake_count = 0;

int main()
{
        Lake tba;
        tba.name = "Pali",
        tba.area = 1230;
        cout << tba.toString() << endl;

        Lake Ozero;
        tba.name = "Ladog",
        tba.area = 61230;

        cout << "&tba=" << (int)&tba << endl;
        cout << "&tba+1 =" << (int)(&tba + 1) << endl;
        cout << "&Lake::Lake_count=" << (int)&Lake::Lake_count << endl;
        cout << "&Lake::Lake()=" << (int)&Lake::Lake() << endl;
        cout << "&(tba.toString())=" << (int)&(tba.toString()) << endl;
        cout << "&(Ozero.toString())=" << (int)&(Ozero.toString()) << endl;
}
```

კონსტრუქტორი ბეჭდავს ტბების რაოდენობას ყოველი ახალი ტბის შექმნის შემდეგ. პროგრამის მუშაობის შესაძლო შედეგი არის: მესამე ტბა იქმნება ბოლო სტრიქონში ტბების

```
New lake created. Lake_count=1
class Lake *:  name: Pali, area: 1230
New lake created. Lake_count=2
&tba=1439944
&tba+1 =1439976
&Lake::Lake_count=20529892
New lake created. Lake_count=3
&Lake::Lake()=1439636
Deleted: class Lake *:  name: , area: -858993460 Lake_count=2
&(tba.toString())=1439600
&(Ozero.toString())=1439564
Deleted: class Lake *:  name: , area: -858993460 Lake_count=1
Deleted: class Lake *:  name: Ladog, area: 61230 Lake_count=0
```

კონსტრუქტორის გამოძახების შედეგად. 8-9 სტრიქონები ბეჭდავს ტბის `tba` ობიექტის დასაწყისის და დასასრულის მისამართებს. როგორც ვხედავთ, მთვლელის მისამართი ამ შუალედში არ არის. შეგვიძლია შევამოწმოთ, რომ იგი არაა არც მომდევნო ორი ტბისთვის გამოყოფილ მონაკვეთებზე.

პროგრამის მუშაობის შესაძლო შედეგი ნაჩვენებია მარცხენა ჩარჩოში.

ცხადია, კონსტრუქტორი უნდა გავათავისუფლოთ ზედმეტი დატვირთვისგან. ზედმეტია, რომ ყოველი ობიექტის კონსტრუირებისას მოხდეს ანგარიში ჩაბარება ობიექტების რაოდენობაზე. ამ მაგალითში ასე თვალსაჩინოებისთვის გვაქვს მოწყობილი.

!) ყურადღება მივაქციოთ იმ გარემოებას, რომ C/C++ ენებში სიტყვა **static** გამოიყენება რამდენიმე განსხვავებული მიზნით.

!!) თუ რამდენიმე კონსტრუქტორი გვაქვს ისეთ კლასში, რომელშიც ვაპირებთ ობიექტების მთვლელის შექმნას, აუცილებელია რომ თითოეულმა გაითვალისწინოს ობიექტების მთვლელის გაზრდა.

<<< სავარჯიშოები

1. მეცადინეობაზე განხილული ტბის კლასი გადააკეთეთ ისე, რომ მისი ველები იყოს კერძო.
2. შექმენით კლასი ზღვისთვის ველებით სახელი და სიღრმე. გაითვალისწინეთ კონსტრუქტორები, დესტრუქტორი, ობიექტის სტრინგად წარმოდგენა და ობიექტების მთვლელი. მოიყვანეთ სტატიკური წევრების გამოყენების მაგალითები.
3. შექმენით კლასი წიგნისთვის ველებით ავტორი და სახელი. გაითვალისწინეთ კონსტრუქტორები, ობიექტის სტრინგად წარმოდგენა და ობიექტების მთვლელი. მოიყვანეთ სტატიკური წევრების გამოყენების მაგალითები.
4. შექმენით კლასი ვულკანისთვის ველებით სახელი და სიმაღლე და მდებარეობა. გაითვალისწინეთ კონსტრუქტორები, ობიექტის სტრინგად წარმოდგენა და ობიექტების მთვლელი. მოიყვანეთ სტატიკური წევრების გამოყენების მაგალითები.
5. შექმენით კლასი ქვეყნისთვის ველებით სახელი და მოსახლეობის რაოდენობა. გაითვალისწინეთ კონსტრუქტორები, ობიექტის სტრინგად წარმოდგენა და ობიექტების მთვლელი. მოიყვანეთ სტატიკური წევრების გამოყენების მაგალითები.
6. დავალებებში მიტითებულ კლასებში, დაამატეთ პარამეტრიანი კონსტრუქტორები. კონსტრუქტორში და დესტრუქტორში ჩაამატეთ გარკვეული მონაცემების ბეჭდვის ფუნქცია. აგრეთვე, დინამიკურად შექმენით და გააუქმეთ ობიექტები.