Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra Informatică Aplicată

**RAPORT**

Lucrarea de laborator nr.3

Programarea Orientata pe Obiecte

A efectuat:

st. gr. C-171 D. Melniciuc

A verificat:

dr., conf.univ. M. Oșovschi

Chişinău 2018

**Tema: Supraincarcarea operatorilor**

**Scopul lucrării**: familiarizarea studenţilor cu noţiunile operatori supraîncărcaţi.

**Consideraţiile teoretice necesare**:

Atunci când creaţi o clasă, C++ vă permite să supraîncărcaţi operatorii pentru a lucra cu propriile clase.

Când supraîncărcaţi un operator, trebuie să continuaţi să utilizaţi operatorul în formatul său standard. De exemplu, dacă supraîncărcaţi operatorul plus (+), acesta trebuie să utilizeze operatorul sub forma ***operand+operand*** *.* Operatorul supraîncărcat creat se aplică numai instanţelor clasei specificate

Atunci când creaţi funcţii ***operator***membre pentru a supraîncărca funcţionarea unui operator, declaraţiile membrilor operatori vor avea forma generală prezentată mai jos:

**tip-return nume-clasa::operator #(lista-argumente)**

**{**

**// Operatii**

**}**

unde # este simbolul oricărui operator C++, exceptînd:

*.* – operator de membru al clase

\* – adresare la componenta prin pointer,

:: – operatorul de rezoluţie*,*

() ?: – operatorul condiţional*,*

operatorul *sizeof, etc.*.

Această definire se face în cadrul clasei, întocmai ca o funcţie membru.

Există două variante de definire a operatorilor:

* ca funcţie membru a clasei;
* ca funcţie prietenă a clasei.

O funcţie *operator* are aceleaşi componente pe care le are orice funcţie, include un nume, un tip returnat, argumente, corp şi, eventual, apartenenţa la o clasă.

Există trei elemente care trebuie stabilite la declararea operatorului, şi anume:

– este operator unar sau binar,

– este postfixat sau prefixat ca poziţie

– este funcţie membru sau nu – domeniu de acţiune.

***Funcţiile***operator ***membri******vor avea cu un argumen****t* ***mai puţin decît cele non-membri.***

Puteţi să supraîncărcaţi numai operatorii existenţi. C++ nu vă permite definirea unor operatori proprii.

Definirea operatorilor ca funcţii membri a unei clase prezintă o restricţie majoră: primul operand este obligatoriu să fie de tipul clasa respectiv.

În limbajul C++ supradefinirea operatorilor este supusă unui set de restricţii:

* nu este permis introducerea de noi simboluri de operatori;
* patru operatori nu pot fi redefiniţi (vezi mai sus);
* caracteristicile operatorilor nu pot fi schimbate:

1. pluralitatea (nu se poate supradefini un operator unar ca

operator binar sau invers),

2. precedenţa şi asociativitatea,

3. prioritatea lor;

* funcţia ***operator*** trebuie sa aibă ***cel******puţin***un parametru de tipul clasa căruia îi este asociat operatorul supradefinit.

Programatorul are libertatea de a alege natura operaţiei realizate de un operator, însă este recomandat ca noua operaţie să fie apropiată de semnificaţia iniţială.

Pot fi redefiniti operatorii din urmatoarea tabela:

**Tabelul 1.**

|  |  |  |
| --- | --- | --- |
| **Tipul operatorului** | **Simbolul operatorului** | **Observaţii** |
| **Binar** | **(); [] ;->;** | **Se definesc ca funcţii membri** |
| **Unar** | **+; - ;~; \*; & (tip)** |  |
| **Unar** | **++; - -** | **Nu se poate dis-tinge între pre- şi postfixare** |
| **Unar** | **new; delete** | **Poate fi supra-definit şi pentru o clasă** |
| **Binar** | **->; \*; /; %; +; -; &;|; &&; ||** |  |
| **Binar** | **<<; >> ;< ;<=; > ;>=; == ;!=** |  |
| **Binar** | **= ;+=; -= ;\*=; /=;%= ;&=; ^=;|= ;<<=; >>=** | **Se definesc ca funcţii membri** |
| **Binar** | **,** |  |

**Varianta 4:**

**4.**Să se creeze clasa Bool – variabile logice. Să se definească operatorii "+" – SAU logic, "\*" – ŞI logic, "^" – SAU EXCLUSIV, ca metode ale clasei, iar operatorii "==" şi "!=" – ca funcţii prietene. Operatorii trebuie să permită realizarea operaţiilor atît cu variabilele clasei date, cît şi cu variabilele de tip predefinit int. (Dacă numărul întreg este diferit de zero, se consideră că variabila este adevăr, altfel – fals.)

***Codul programul I:***

*#include <iostream>*

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <math.h>*

*using namespace std;*

*class Bool{*

*public:*

*int num;*

*Bool();*

*Bool(int);*

*Bool operator+(Bool obj){*

*Bool newObj;*

*newObj.num = num + obj.num;*

*return (newObj);*

*}*

*Bool operator\*(Bool obj2){*

*Bool newObj;*

*newObj.num = num \* obj2.num;*

*return (newObj);*

*}*

*Bool operator^(Bool obj3){*

*Bool newObj;*

*newObj.num = pow(num, obj3.num);*

*return (newObj);*

*}*

*};*

*Bool::Bool(){}*

*Bool::Bool(int a)*

*{num = a;}*

*int main(int argc, char const \*argv[]) {*

*int z, x;*

*cout<<"nr prestabliite (2,3)\n";*

*Bool a1(2);*

*Bool a2(3);*

*Bool a3;*

*a3 = a1 + a2;*

*cout << "\nrezultat overload pentru '+' " << a3.num << endl;*

*a3 = 0;*

*a3 = a1 \* a2;*

*cout << "rezultat overload pentru '\*' " << a3.num << endl;*

*a3 = 0;*

*a3 = a1 ^ a2;*

*cout << "rezultat overload pentru '^' " << a3.num << endl;*

*a3 = 0;*

*Bool r;*

*cout << "\nintroduce un int: ";*

*cin >> z;*

*cout << "introduce alt int: ";*

*cin >> x;*

*Bool a(z);*

*Bool b(x);*

*Bool c;*

*c = a + b;*

*cout << "\nrezultat overload pentru '+' " << c.num << endl;*

*c = 0;*

*c = a \* b;*

*cout << "rezultat overload pentru '\*' " << c.num << endl;*

*c = 0;*

*c = a ^ b;*

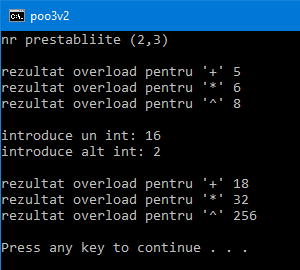
*cout << "rezultat overload pentru '^' " << c.num << endl;*

*c = 0;*

*return 0;*

*}*

***ScreenShoturi:***



***Concluzie:***

*Dupa efectuarea laborotorului am dobintid experienta in limbajul de programere C++ supraincarcind niste operatorilor, nominal ‘+’ ’\*’ ’^’. Supraincarcarea operatprului ‘^’a fost una importanta deoarece in C++ nu putem ridica la putere un numar prin operatorul dat.*