Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №5**

на тему:

“ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ”

**Виконав:**

Ст. гр. КІ-15

Залевський К.С.

**Перевірив:**

Викладач

Козак Н.Б.

Львів – 2020

**Мета роботи:** познайомитися із перевантаженням операторів.

**Теоретичні відомості:**

**Перевантаження операторів**

Кожному оператору мова С++ ставить у відповідність ім'я функції, що складається з ключового слова operator, власне оператору та аргументів відповідних типів:



Щоб використовувати операцію над об'єктами класів, ця операція повинна бути перевантажена, але є два виключення. Операції присвоювання (=) і взяття адреси (&) створюються в класі автоматично за замовчуванням, тому їх можна використовувати без явного перевантаження. За замовчуванням операція присвоювання зводиться до побітового копіювання даних-елементів класу. Проте таке побітове копіювання небезпечне для класів з елементами, що вказують на динамічно виділені області пам'яті, масиви, рядки, оскільки в цьому випадку відбувається копіювання не даних (глибоке копіювання), а лише вказівників на дані (поверхневе копіювання). Для таких класів слід явно перевантажувати операцію присвоювання і здійснювати у ній глибоке копіювання. Операція адресації також може бути використана з об'єктами будь-яких класів без перевантаження. Вона просто повертає адресу об'єкта в пам'яті. Але операцію адресації можна також і перевантажувати.

При перевантаженні операцій ( ), [], -> та = функція перевантаження операції може бути оголошена лише як метод класу. Для інших операцій функції перевантаження операцій можуть не бути методами класу.

Оператори, які можна перевантажити:



Оператори, які не можна перевантажити:

- sizeof

- . (селектор елемента структури або класу)

- \* (оператор доступу до елементу за вказівником)

- :: (оператор дозволу видимості)

- ?: (тернарний оператор)

- typeid

- const\_cast

- dynamic\_cast

- reinterpret\_cast

- static\_cast

- # і ## (символи препроцесору)

***Завдання:***

Розширити функціональність розроблених у 4 лабораторній роботі класів за допомогою операторів, що задані варіантом та оператора присвоювання. Конкретні функції операторів реалізувати на власний розсуд (крім оператора присвоювання). Організувати виведення та введення даних за допомогою класів-потоків сin, cout та перевантажених операторів вводу/виводу. Написати програму, яка демонструє роботу з об'єктами цього класу.



*Код програми:*

#include <iostream>

#include <string>

using namespace std;

class CComputer

{

private:

string cpumodel, videomodel;

double operation\_per\_seconds;//"operation per second"

int hardsize, ramsize, optime, numprogram;

public:

//Operator's

CComputer& operator= (const CComputer& other)

{

cout << endl << "Operator =" << endl;

this->cpumodel = other.cpumodel;

this->videomodel = other.videomodel;

this->operation\_per\_seconds = other.operation\_per\_seconds;

this->hardsize = other.hardsize;

this->ramsize = other.ramsize;

this->optime = other.optime;

this->numprogram = other.numprogram;

return \*this;

}

CComputer& operator \*(const int value)

{

cout << endl << "Operator \*" << endl;

this->hardsize = this->hardsize\*value;

return \*this;

}

CComputer& operator ++ ()

{

cout << endl << "Operator ++" << endl;

this->ramsize ++;

this->operation\_per\_seconds++;

return \*this;

}

bool operator ==(const CComputer& other)

{

cout << endl << "Operator ==" << (this->cpumodel == other.cpumodel && this->videomodel == other.videomodel && this->operation\_per\_seconds == other.operation\_per\_seconds && this->hardsize == other.hardsize && this->ramsize == other.ramsize && this->optime == other.optime && this->numprogram == other.numprogram) << endl;

return this->cpumodel == other.cpumodel && this->videomodel == other.videomodel && this->operation\_per\_seconds == other.operation\_per\_seconds && this->hardsize == other.hardsize && this->ramsize == other.ramsize && this->optime == other.optime && this->numprogram == other.numprogram;

}

bool operator ||(const CComputer& other)

{

cout << endl << "Operator < :: " << (this->optime == other.optime || this->numprogram == other.numprogram) << endl;

return this->optime == other.optime || this->numprogram == other.numprogram;

}

//Constrictor's | Destructor's

CComputer(string Ccpumodel, string Cvideomodel, double Coperation\_per\_seconds, int Chardsize, int Cramsize, int Coptime, int Cnumprogram) //Конструктор

{

cpumodel = Ccpumodel;

videomodel = Cvideomodel;

operation\_per\_seconds = Coperation\_per\_seconds;

hardsize = Chardsize;

ramsize = Cramsize;

optime = Coptime;

numprogram = Cnumprogram;

}

CComputer(const CComputer& other)

{

this->cpumodel = other.cpumodel;

this->videomodel = other.videomodel;

this->operation\_per\_seconds = other.operation\_per\_seconds;

this->hardsize = other.hardsize;

this->ramsize = other.ramsize;

this->optime = other.optime;

this->numprogram = other.numprogram;

}

//Setters

void Set\_cpumodel(string Scpumodel)

{

cpumodel = Scpumodel;

}

void Set\_videomodel(string Svideomodel)

{

videomodel = Svideomodel;

}

void Set\_operation\_per\_seconds(double Soperation\_per\_seconds)

{

operation\_per\_seconds = Soperation\_per\_seconds;

}

void Set\_hardsize(int Shardsize)

{

hardsize = Shardsize;

}

void Set\_ramsize(int Sramsize)

{

ramsize = Sramsize;

}

void Set\_optime(int Soptime)

{

optime = Soptime;

}

//Getters

string Get\_cpumodel()

{

return cpumodel;

}

string Get\_videomodel()

{

return videomodel;

}

double Get\_operation\_per\_seconds()

{

return operation\_per\_seconds;

}

int Get\_hardsize()

{

return hardsize;

}

int Get\_ramsize()

{

return ramsize;

}

int Get\_optime()

{

return optime;

}

int Get\_numprogram()

{

return numprogram;

}

//Install and uninstall programm

void install()

{

cout << endl << "Enter the name of the application which you want to install:";

//cin>>

numprogram++;

cout << endl << "The program is installed." << endl << "Number installed program::" << numprogram << endl;

}

void uninstall()

{

cout << endl << "Enter the name of the application which you want to uninstall:";

//cin>>

numprogram--;

cout << endl << "The program is uninstalled." << endl << "Number installed program::" << numprogram << endl;

}

void timeneed()

{

int numop;

cout << endl << "Enter the number of operations the program must perform::";

cin >> numop;

cout << endl << "Time for execution(in seconds):" << numop / operation\_per\_seconds << endl;

}

void Showdata()

{

cout << endl << "Cpu model:" << cpumodel;

cout << endl << "Video model:" << videomodel;

cout << endl << "Operation per seconds" << operation\_per\_seconds;

cout << endl << "Hard size" << hardsize;

cout << endl << "Ram size" << ramsize;

cout << endl << "Time of operation" << optime;

cout << endl << "Number of program" << numprogram;

}

};

int main()

{

string fcpuname = "i7-8400", fvname = "GTX 1660";

CComputer firstpc(fcpuname, fvname, 4000, 3000, 16, 1, 164);

string scpuname = "i7-8400K", svname = "GTX 1060 Ti";

CComputer secondpc(scpuname, svname, 3000, 2000, 12, 2, 104);

cout << "Was:" << endl << "Hadrsize: " << firstpc.Get\_hardsize() << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

firstpc \* 5;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl << "Has become:" << endl << "Hadrsize: " << firstpc.Get\_hardsize() << endl;

cout << "================================================================"<< endl;

cout << endl;

cout << endl << "Was:" << endl << "Ramsize: " << firstpc.Get\_ramsize() << endl << "OPS: " << firstpc.Get\_operation\_per\_seconds() << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

++firstpc;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl << "Has become:" << endl << "Ramsize: " << firstpc.Get\_ramsize() << endl << "OPS: " << firstpc.Get\_operation\_per\_seconds() << endl;

cout << "================================================================" << endl;

cout << endl;

cout << endl << "Was:" << endl << endl << "Optime: " << firstpc.Get\_optime() << endl << "Num programm:" << firstpc.Get\_numprogram() << endl;

cout << "Operator || :" << (secondpc || firstpc) << endl << endl<< endl;

cout << "================================================================" << endl << endl;

cout << "====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====";

cout << endl;

cout << endl << "Was before operator = :" << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

secondpc == firstpc;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "================================================================" << endl;

cout << endl;

cout << "Was:" << endl;

firstpc.Showdata();

cout << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

firstpc = secondpc;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << "Has become:" << endl;

firstpc.Showdata();

cout << endl;

cout << "================================================================" << endl;

cout << endl;

cout << endl << "has become after operator = :" << endl;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

secondpc == firstpc;

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====\*\*\*\*====" << endl;

return 0;

}

*Вікно результату:*



*Висновок:* я познайомився з таким «явищем», як перевантаження операторів. Також пропрактикувався в їх написанні.