МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕХНІЧНИЙ КОЛЕДЖ ТНТУ ІМЕНІ ІВАНА ПУЛЮЯ

ВІДДІЛЕННЯ ЕЛЕКТРОННИХ АПАРАТІВ

Циклова комісія програмних систем і комплексів

**ЗВІТ**

про виконання лабораторних робітз дисципліни:

**«ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»**

Студента  3  курсу групи КН-321 спеціальності 122 «Комп’ютерні науки»

Слюсарчин О.В

(прізвище та ініціали)

Перевірив: Р.О. Слободян

(підпис)

Тернопіль – 2020

**Лабораторна робота №3**

**Тема:**  Перевантаження операцій класу

**Мета:** Ознайомитись зі способами перевантаження операцій та навчитись використовувати їх при роботі з об`єктами.

**Завдання 1** В класі Int, який розроблений в завданні №1 лабораторної роботи №1, перевизначте чотири цілочисельні арифметичні операції («+», «-», «\*» , «/») так, щоб їх можна було використовувати для операцій з об'єктами класу Int. Якщо результат будь-якої з операцій виходить за межі типу int (в 32-бітній системі), що може мати значення від 2 147 483 648 до -2 147 483 648, то операція повинна послати повідомлення про помилку і завершити програму. Такі типи даних корисні там, де помилки можуть бути викликані арифметичним переповненням, яке неприпустимо. Напишіть програму для перевірки цього класу.

Підказка: для полегшення перевірки переповнення виконуйте обчислення з використанням типу long double.

**КОД ПРОГРАМИ**

#include <iostream>

#include <math.h>

using namespace std;

class **Int**

{

int number;

public:

**Int**()

{

number = 0;

}

Int operator + (Int number) {

int sum = this->number + number.GetNumber();

if(sum > INT\_MAX) {

cout << "Your sum is bigger than Max Integer value" << endl;

} else {

Int result = Int();

result.SetNumber(sum);

return result;

}

}

Int operator \* (Int number) {

int multiply = this->number \* number.GetNumber();

if(multiply > INT\_MAX || multiply < INT\_MIN) {

cout << "Your sum is bigger than Max or less than Min Integer value" << endl;

} else {

Int result = Int();

result.SetNumber(multiply);

return result;

}

}

Int operator / (Int number) {

int devide = this->number / number.GetNumber();

if(devide > INT\_MAX || devide < INT\_MIN) {

cout << "Your sum is bigger than Max or less than Min Integer value" << endl;

} else {

Int result = Int();

result.SetNumber(devide);

return result;

}

}

Int operator - (Int number) {

int minus = this->number - number.GetNumber();

if(minus > INT\_MAX || minus < INT\_MIN) {

cout << "Your sum is bigger than Max or less than Min Integer value" << endl;

} else {

Int result = Int();

result.SetNumber(minus);

return result;

}

}

int **GetNumber**() {

return number;

}

void **SetNumber**(int number) {

this->number = number;

}

};

int **main**()

{

Int value1 = Int();

value1.SetNumber(15);

Int value2 = Int();

value2.SetNumber(16);

Int sumResult = value1 + value2;

cout << "Sum is:" << endl;

cout << sumResult.GetNumber() << endl;

Int multiplyResult = value1 \* value2;

cout << "Multiply is:" << endl;

cout << multiplyResult.GetNumber() << endl;

Int devideResult = value1 / value2;

cout << "Devide is:" << endl;

cout << devideResult.GetNumber() << endl;

Int minusResult = value1 - value2;

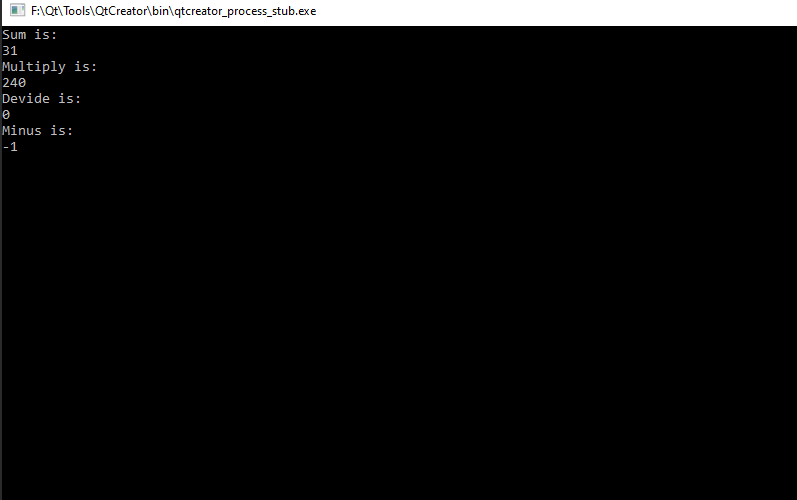
cout << "Minus is:" << endl;

cout << minusResult.GetNumber() << endl;

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ**

****

**Завдання 2** Для класу, який розробленого згідно індивідуального завдання лабораторної роботи № 2, визначити операції: - зчитування з потоку вводу sdt::cin; - виводу у потік std::cout. Перевірити роботу перевизначених функцій у функції main() за допомогою коду: НазваКласуЗгідноВаріанта myObject; std::cin >> myObject; std::cout << myObject;

**КОД ПРОГРАМИ**

#include <iostream>

using namespace std;

const int TOYS\_ARRAY\_SIZE = 3;

class **Toy** {

private: char Owner;

private: char \*OwnerPointer;

private: int ToyOld;

public:

//constructors

**Toy**(){};

**Toy**(char \*OwnerPointer, int ToyOld) {

this->Owner = \*OwnerPointer;

this->ToyOld = ToyOld;

};

friend std::ostream& operator<< (std::ostream &out, const Toy &point);

std::ostream& operator<< (std::ostream &out, const Toy &point) {

cout << "Toy(" << Toy.Owner << ", " << Toy.OwnerPointer << ", " << Toy.Old << ")";

}

//getters and setters

char **GetOwner**() {

return Owner;

};

void **SetOwner**(char \*OwnerPointer) {

this->Owner = \*OwnerPointer;

};

int **GetToyOld**() {

return this->ToyOld;

};

void **SetToyOld**(int \*ToyOld){

this->ToyOld = \*ToyOld;

};

Toy **Input**() {

char ToyName;

int ToyOld;

cout << "Enter toy's name:" << endl;

cin >> ToyName;

cout << "Enter toy's age:" << endl;

cin >> ToyOld;

cout << endl;

Toy myToy = Toy(&ToyName, ToyOld);

return myToy;

};

void **Print**() {

cout << this->Owner << endl;

cout << this->ToyOld << endl;

};

//destructor

~**Toy**(){};

};

int **main**()

{

char ToyName1 = 'A';

int ToyOld1 = 1;

char ToyName2 = 'B';

int ToyOld2 = 2;

Toy myToy1 = Toy(&ToyName1, ToyOld1);

Toy myToy2 = Toy();

myToy2.SetOwner(&ToyName2);

myToy2.SetToyOld(&ToyOld2);

//enter Toy object from keyboard

Toy myToy3 = Toy().Input();

Toy ToysArray[TOYS\_ARRAY\_SIZE] = {

myToy1, myToy2, myToy3

};

for(int i = 0; i < TOYS\_ARRAY\_SIZE; i++) {

ToysArray[i].Print();

}

}

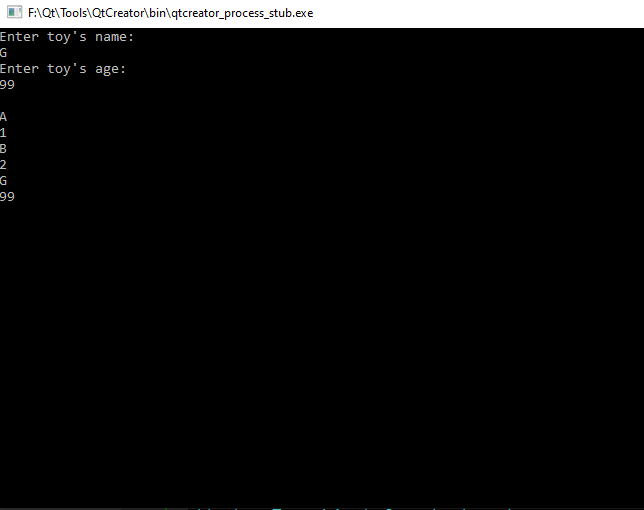
std::cout << ToyName3;

std::cout << ToyOld3;

return 0;

}r

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ**



**Завдання 3** Для заданого варіанта індивідуального завдання виконати перевантаження операцій для зручності роботи з об’єктами. При необхідності оголосіть певні операторні функції друзями класу.

Створити клас – вектор, який має у закритій частині вказівник на дані цілого типу та кількість елементів. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення елементів вектора, операції + , – , \* , = та [ ] .

**КОД ПРОГРАМИ**

#include <iostream>

using namespace std;

class **Vector**{

int \*dataPointer;

int itemCount;

int items[5];

public:

**Vector**()

{

\*dataPointer = 0;

int items[10];

}

int operator [] (int i) {

return items [i];

}

Vector operator + (int \*data) {

int sum = \*dataPointer + \*data;

Vector result = Vector();

result.SetDataPointer(&sum);

return result;

}

Vector operator \* (int \*data) {

int sum = \*dataPointer \* \*data;

Vector result = Vector();

result.SetDataPointer(&sum);

return result;

}

Vector operator - (int \*data) {

int sum = \*dataPointer - \*data;

Vector result = Vector();

result.SetDataPointer(&sum);

return result;

}

Vector operator = (int \*data) {

int sum = \*dataPointer = \*data;

Vector result = Vector();

result.SetDataPointer(&sum);

return result;

}

Vector operator / (int \*data) {

int sum = \*dataPointer / \*data;

Vector result = Vector();

result.SetDataPointer(&sum);

return result;

}

int **GetDataPointer** () {

return \*dataPointer;

}

void **SetDataPointer**(int \*data) {

\*dataPointer = \*data;

}

};

int **main**()

{

int val1 = 6;

int val2 = 9;

Vector value1 = Vector();

value1.SetDataPointer(&val1);

Vector value2 = Vector();

value2.SetDataPointer(&val2);

Vector sumResult = value1 + value2 ;

cout << "Sum is:" << endl;

cout << sumResult.GetDataPointer() << endl;

Vector multiplyResult = value1 \* value2;

cout << "Multiply is:" << endl;

cout << multiplyResult.GetDataPointer() << endl;

Vector devideResult = value1 / value2;

cout << "Devide is:" << endl;

cout << devideResult.GetDataPointer() << endl;

Vector minusResult = value1 - value2;

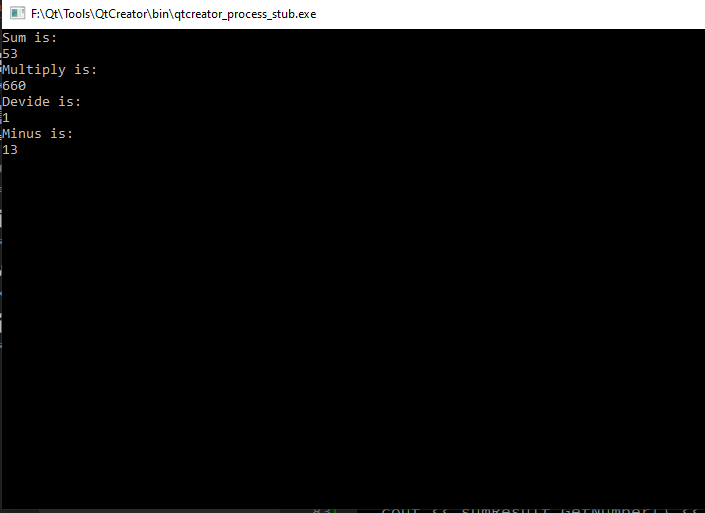
cout << "Minus is:" << endl;

cout << minusResult.GetDataPointer() << endl;

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ**



**Висновок:** Ознайомився зі способами перевантаження операцій та навчитись використовувати їх при роботі з об`єктами.