ЗВІТ

про виконання лабораторної роботи № 2.2

« Перевантаження операцій »

з дисципліни

«Об’єктно-орієнтоване програмування»

студента групи ІТ-12

М’ягких Владислава Юрійовича

**Мета роботи:**

Освоїти використання конструкторів та перевантаження операцій.

**Умова завдання:**

В кожній лабораторній роботі цієї теми потрібно реалізувати в тому або іншому вигляді визначення нового класу. У всіх завданнях необхідно реалізувати:

• методи доступу (константні методи зчитування та методи запису) значення кожного поля;

• конструктор ініціалізації (один або декілька),

• конструктор без аргументів і

• конструктор копіювання.

Вказані в завданні операції реалізуються за допомогою перевантаження підходящих операцій.

У всіх завданнях обов’язково мають бути реалізовані відповідні операції:

• присвоєння,

• введення з клавіатури,

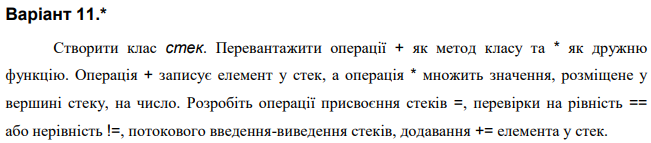
• виводу на екран,

• приведення типу – перетворення у літерний рядок.

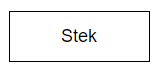
Перевантаження операцій виконується таким чином: підходящі операції реалізуються як методи класу, а інші – як зовнішні дружні функції.

Для демонстрації роботи з об’єктами нового типу у всіх завданнях потрібно написати головну функцію. У програмі обов’язково мають бути продемонстровані різні способи створення об’єктів і масивів об’єктів. Програма має демонструвати використання всіх функцій і методів. Вона має виводити на екран розмір класу в режимі #pragma pack(1) і без нього.

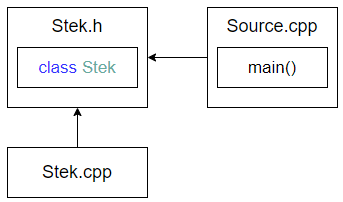
Визначення класу та реалізацію його методів слід розмістити в окремих модулях.



**UML-діаграма класів:**



**Структурна схема програми:**



**Текст програми:**

//////////////

// Source.cpp

//////////////

#include <iostream>

#include "Stek.h"

using namespace std;

int main()

{

Stek a(0), b(1);

Stek c;

int q;

cout << "Rozmir stekyA = "; cin >> q;

for (int i = 0; i < q; i++)

{

cout << "Stek A" << endl;

cout << "b = "; cin >> b;

c = a + b;

cout << endl;

}

cout << "Rozmir stekyB = "; cin >> q;

for (int i = 0; i < q; i++)

{

cout << "Stek B" << endl;

cout << "a = "; cin >> a;

c = b + a;

cout << endl;

}

cout << "Stek A = " << a;

cout << "Stek B = " << b << endl;

cout << "Chislo = "; cin >> q;

c = a \* q;

c = b \* q;

cout << "Stek A = " << a;

cout << "Stek B = " << b << endl;;

bool w = a == b;

cout << "Rivnist chisla = " << w << endl << endl;

cout << "Chislo = "; cin >> q;

a += q;

b += q;

cout << "Stek A = " << a;

cout << "Stek B = " << b;

a.Pop();

b.Pop();

}

//////////////

// Stek.h

//////////////

#pragma once

#include<iostream>

#include<iomanip>

#include<string>

#include<sstream>

using namespace std;

class Stek

{

private:

int x;

struct stek

{

stek\* link;

int info;

stek(stek\* n, int x) : info(x), link(n) {}

};

stek\* top;

public:

Stek();

Stek(int x);

int getX() const { return x; }

void setX(int value) { x = value; }

Stek operator + (Stek& a);

Stek& operator = (const Stek&);

friend Stek operator \* (Stek r, int x);

friend ostream& operator << (ostream& out, const Stek& a);

friend istream& operator >> (istream& in, Stek& a);

friend bool operator == (Stek а, Stek b);

friend bool operator != (Stek а, Stek b);

friend Stek operator +=(const Stek& a, int b);

int Pop();

};

//////////////

// Stek.cpp

//////////////

#include "Stek.h"

Stek::Stek()

: top(0), x(1)

{}

Stek::Stek(int q)

{

setX(q);

}

Stek operator += (const Stek& a, int b)

{

Stek::stek\* L = a.top;

while (L != NULL)

{

L->info += b;

L = L->link;

}

return a;

}

bool operator != (Stek a, Stek b)

{

return !(a == b);

}

bool operator == (Stek a, Stek b)

{

Stek::stek\* L = a.top;

Stek::stek\* K = b.top;

while (L != NULL)

{

while (K != NULL)

{

if (L->info != K->info)

return false;

K = K->link;

if (((K->link == NULL) && (L->link != NULL)) || ((K->link != NULL) && (L->link == NULL)));

return false;

}

L = L->link;

}

return true;

}

Stek& Stek::operator = (const Stek& r)

{

x = r.x;

return \*this;

}

Stek operator \* (Stek r, int x)

{

Stek::stek\* L = r.top;

/\*while (L->link != NULL)

{

L = L->link;

}\*/

L->info = L->info \* x;

return r;

}

Stek Stek:: operator + (Stek& a)

{

//Stek t;

//t.x =this->x + a.x;

top = new stek(top, a.x);

return a;

}

istream& operator >> (istream& in, Stek& a)

{

in >> a.x;

return in;

}

ostream& operator << (ostream& out, const Stek& a)

{

Stek::stek\* L = a.top;

while (L != NULL)

{

out << L->info << setw(4);

L = L->link;

}

cout << endl;

return out;

}

int Stek::Pop()

{

int top1 = 0;

while (top != NULL)

{

top1 = top->info;

Stek::stek\* old = top;

top = top->link;

delete old;

}

return top1;

}

**Посилання на git-репозиторій з проектом:**

**Висновки:**

На даній лабораторній роботі, я навчився використовувати конструктори та перевантаження операцій.