Міністерство освіти і науки України  
Національний технічний університет України «Київський політехнічний  
інститут імені Ігоря Сікорського"  
Факультет інформатики та обчислювальної техніки  
Кафедра інформатики та програмної інженерії

Звіт  
з лабораторної роботи № 4 з дисципліни  
«Основи програмування 2. Методології програмування»  
«Перевантаження операторів»

Варіант\_\_16\_\_

Виконав студент \_\_ІП-15,\_Куманецька\_Ірина\_Вікторівна\_\_\_  
Перевірив \_\_\_Вечерковська Анастасія Сергіївна\_\_\_\_\_\_

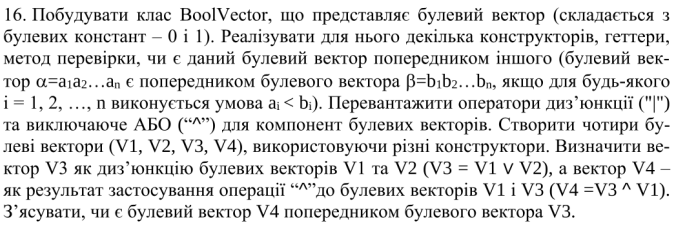
Київ 2022

**Лабораторна робота 4**

**Мета** – вивчити механізми створення класів з використанням перевантажених операторів(операцій).

**Індивідуальне завдання**

**Варіант 16**



**Виконання на С++**

main.cpp

#include <iostream>  
#include "func.h"  
#include "BoolVector.h"  
  
using namespace std;  
  
int main() {  
 BoolVector vec1(input\_vec()), vec2(input\_vec());  
 BoolVector vec3 = vec1|vec2;  
 cout << "Vector 3 (disjunction v1|v2):" << endl;  
 print\_vec(vec3);  
 BoolVector vec4 = vec1^vec3;  
 cout << "Vector 4 (exclusive or v1^v3):" << endl;  
 print\_vec(vec4);  
 cout << "V4 is predecessor of V3: ";  
 if (vec4.is\_prev(vec3)) {cout << "True" << endl;}  
 else {cout << "False" << endl;}  
}

BoolVector.cpp

#include "BoolVector.h"  
  
using namespace std;  
  
BoolVector::BoolVector(const BoolVector &obj){  
 size = obj.size;  
 vec = new bool [size];  
 for (int i = 0; i < size; ++i) {  
 vec[i] = obj.vec[i];  
 }  
}  
  
BoolVector::BoolVector(int n) {  
 size = n;  
 vec = new bool [n];  
}  
  
BoolVector::BoolVector(string line) {  
 size = line.length();  
 vec = new bool [size];  
 for (int i = 0; i < size; i++) {  
 vec[i] = (bool)(line[i]-48);  
 }  
}  
  
BoolVector::BoolVector(const bool\* nums, int len) {  
 size = len;  
 vec = new bool [len];  
 for (int i = 0; i < len; i++) {vec[i] = nums[i];}  
}  
  
BoolVector BoolVector::operator|(const BoolVector& sec) {  
 bool\*\* vecs = get\_vecs\_same\_size(sec);  
 bool\* f\_vec = vecs[0];  
 bool\* s\_vec = vecs[1];  
 int length = max(size, sec.size);  
 bool\* res = new bool [length];  
 for (int i = 0; i < length; ++i) {  
 res[i] = f\_vec[i] | s\_vec[i];  
 }  
 return BoolVector(res, length);  
}  
  
BoolVector BoolVector::operator^(const BoolVector& sec) {  
 bool\*\* vecs = get\_vecs\_same\_size(sec);  
 bool\* f\_vec = vecs[0];  
 bool\* s\_vec = vecs[1];  
 int len = max(size, sec.size);  
 bool\* res = new bool [len];  
 for (int i = 0; i < len; ++i) {  
 res[i] = f\_vec[i]^s\_vec[i];  
 }  
 return BoolVector(res, len);  
}  
  
bool BoolVector::is\_prev(const BoolVector& sec) {  
 bool\*\* res = get\_vecs\_same\_size(sec);  
 int len = max(size, sec.size);  
 for (int i = 0; i < len; ++i) {  
 if (res[0][i] && !res[1][i]) return false;  
 }  
 return true;  
}  
  
bool\*\* BoolVector::get\_vecs\_same\_size(const BoolVector& sec){  
 bool\*\* vecs = new bool\* [2];  
 if (size > sec.size){  
 bool\* s\_vec = new bool [size];  
 for (int i = 0; i < size-sec.size; i++){s\_vec[i] = false;}  
 for (int i = size-sec.size; i < size ; i++) { s\_vec[i]=sec.vec[i-size+sec.size];}  
 vecs[0] = vec;  
 vecs[1] = s\_vec;  
 }  
 else if (sec.size > size){  
 bool\* f\_vec = new bool [sec.size];  
 for (int i = 0; i < sec.size-size; i++){f\_vec[i] = false;}  
 for (int i = sec.size-size; i < sec.size ; i++) { f\_vec[i]=vec[i-sec.size+size];}  
 vecs[0] = f\_vec;  
 vecs[1] = sec.vec;  
 }  
 else {  
 vecs[0] = vec;  
 vecs[1] = sec.vec;  
 }  
 return vecs;  
}

BoolVector.h

#pragma once  
#include <iostream>  
#include <vector>  
  
using namespace std;  
  
class BoolVector {  
 int size;  
 bool\* vec;  
public:  
 explicit BoolVector(int n);  
 explicit BoolVector(string line);  
 explicit BoolVector(const bool\* nums, int len);  
 BoolVector(const BoolVector &obj);  
 ~BoolVector(){delete[] vec;}  
 bool\* get\_vec(){return vec;}  
 int get\_size() const{return size;}  
 bool\*\* get\_vecs\_same\_size(const BoolVector& sec);  
 BoolVector operator|(const BoolVector& sec);  
 BoolVector operator^(const BoolVector& sec);  
 bool is\_prev(const BoolVector& sec);  
};

func.cpp

#include "func.h"  
  
bool is\_bool(string line){  
 for (char c : line) {  
 if (c != '0' and c != '1'){return false;}  
 }  
 return true;  
}  
  
string input\_vec(){  
 string vec;  
 cout << "Enter the vector:" << endl;  
 cin >> vec;  
 cin.ignore();  
 while (!is\_bool(vec)){  
 cout << "All symbols have to be 0 or 1. Try again:" << endl;  
 cin >> vec;  
 cin.ignore();  
 }  
 cout << "Recorded" << endl;  
 return vec;  
}  
  
void print\_vec(BoolVector vec){  
 bool\* num = vec.get\_vec();  
 for (int i = 0; i < vec.get\_size(); ++i) {cout << num[i];}  
 cout << endl;  
}

func.h

#pragma once  
#include <iostream>  
#include "BoolVector.h"  
  
using namespace std;  
  
string input\_vec();  
bool is\_bool(string line);  
void print\_vec(BoolVector vec);

**Висновки**

Протягом лабораторної робити було розглянуто роботу з класами та перевантаженими операторами. Також були використані різні види конструкторів: з одним та двома параметрами, конструктор копіювання.