Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 4

з дисципліни

“ООП”

Тема: «Операторні і дружні функції С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко  Максим  Миколайович |  | студент 2-го курсу  гр. ІП-51 ФІОТ  Субцельний Олександр Володимирович |

Київ – 2016

**ЗМІСТ:**

[1 Мета роботи 3](#_Toc463896091)

[2 Постановка задачі (варіант 20, рівень В) 4](#_Toc463896092)

3 ПОКРОКОВИЙ АЛГОРИТМ………………………………………………...5

[3 Діаграма класів 5](#_Toc463896093)

[4 Код програми 6](#_Toc463896094)

[5 Приклади виконання програми 10](#_Toc463896095)

[6 Висновок 11](#_Toc463896096)

# Мета роботи

Цель работы – изучить особенности операторных и дружественных функций. Освоить принципы написания: функций преобразования типов объектов, перегрузки операций и дружественных функций.

# Постановка задачі (варіант 20, рівень В)

Спроектировать класс «Multi\_set», который содержит множество упорядоченных символьных элементов. Для него определить: операцию объединения «+», ее же в сокращенной форме, операцию пересечения «\*», ее же в сокращенной форме, операцию разности «-», ее же в сокращенной форме. Определить операции поэлементного добавления и удаления: «<<» и «>>» соответственно. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию. Множество вводится пользователем в виде строки с элементами разделенными запятыми, и преобразовывается в корректное множество. Множество не должно содержать строки.

# ПОКРОКОВИЙ АЛГОРИТМ

-Основний алгоритм

1. Початок
2. Введення рядка елементів першої множини
3. Запис рядка у множину А
4. Введення рядка елементів для другої множини
5. Запис рядка у множину В
6. Операція Обєднання
7. Операція Віднімання
8. Операція пересічення
9. Операція обєднання в короткій формі
10. Операція пересічення в короткіій формі
11. Операція пересічення в короткій формі
12. Операція додавання елемента у множину
13. Операція видалення елемента з множини
14. Кінець

-операція обєднання

1. Початок
2. Створити вихідну множину
3. Записати у вихідну множину всі елементи другої множини
4. Цикл по всім елементам першої множини
   1. Цикл по всім елементам другої множини
      1. Якщо елемент першої множини не входить в другу множину записати елемент першої множини у вихідну множину
5. Кінець

-операція віднімання

1. Початок
2. Створити вихідну множину
3. Записати у вихідну множину всі елементи першої множини
4. Цикл по всім елементам вихідної множини
   1. Цикл по всім елементам другої мгножини
      1. Якщо елемент другої множини входить в елемент вихадної видалити даний елемент з вихідної множини
5. Кінець

-операція пересічення

1. Початок
2. Створення вихідної множини
3. Цикл по всім елементам першої множини
   1. Цикл по всім елементам другої множини
      1. Якщо Елемент першої множини входить в другу множину записати даний елемент в вихідну множину
4. Кінець

-операція додавання елемента у множну

1. Початок
2. Якщо елемент , який вводиться існує у множині ввести елемент заново
3. Інакше додати в множину елемент
4. Відсортувати множину
5. Кінець

-операція видалення елемента з множини

1. Початок
2. Якщо елемент існує у множині видалити його, циклом проходи всі елементи множини починаючи з даного:
   1. Перезаписати на місце даного елемента наступний
3. Кінець

# Діаграма класів

Діаграма класів наведена на рисунку 4.1:

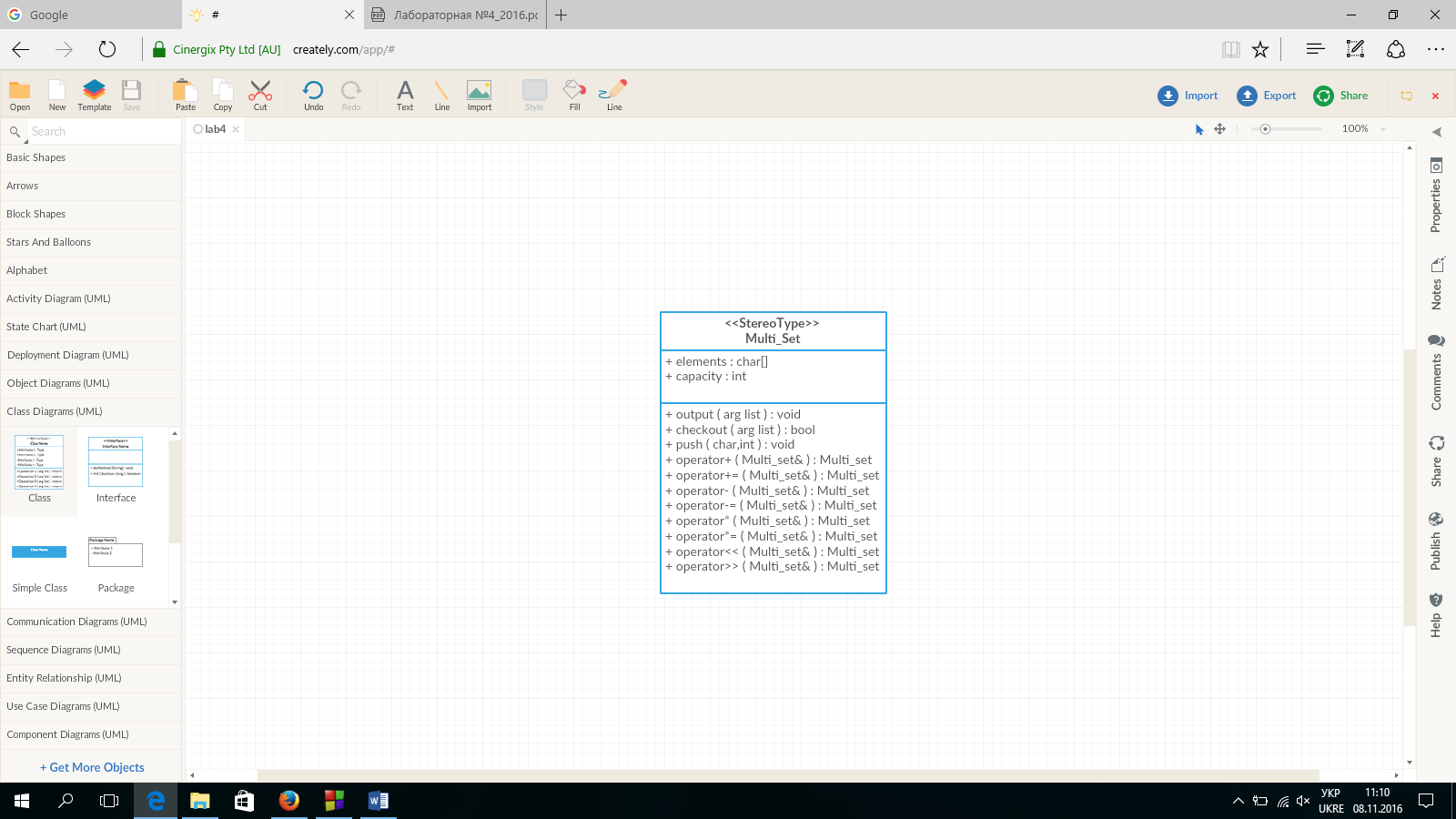


Рисунок 4.1 – Діаграма класів

# Код програми

“main.cpp”

#include <iostream>

#include "Multi\_Set.h"

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

int main() {

printf("OOP lab work #4\nVariant - 20, Level - B\nTask: Write Multi\_Set of charecters\nWriter: Alexander Subtselnyi\n\n");

printf("Menu:\n1. Start program\n2. Exit\n\nMake your choice ");

int t;

scanf("%d", &t);

switch (t) {

case 1: {

char str[20];

cout<<"Enter set 1 through ','"<<endl;

cin>>str;

int len = strlen(str);

Multi\_Set A(str,len);

str[0]='/0';

cout<<"Enter set 2 through ','"<<endl;

cin>>str;

len = strlen(str);

Multi\_Set B(str,len);

str[0]='/0';

printf("\nSet A: "); A.outputSet();

printf("\nSet B: "); B.outputSet();

printf("\n\nUnion of A and B: "); (A + B).outputSet();

printf("\nDifference of A and B: "); (A - B).outputSet();

printf("\nDifference of B and A: "); (B - A).outputSet();

printf("\nIntersection of A \* B: "); (A \* B).outputSet();

printf("\n\nUnion of A and B: "); (A += B).outputSet();

printf("\nSet A: "); A.outputSet();

printf("\nDifference of A and B: "); (A -= B).outputSet();

printf("\nSet A: "); A.outputSet();

printf("\nIntersection of A \* B: "); (A \*= B).outputSet();

printf("\nSet A: "); A.outputSet();

if (!A.checkout()){

cout<<"Enter set 1 again, because its empty"<<endl;

cin>>str;

len = strlen(str);

A.push(str,len);

}

cout<<"Print symbol to add to A"<<endl;

char c;

cin>>c;

A << c;

cout<<"Adding symbol "<<c<<" to A"<<endl;

A.outputSet();

cout<<"Print symbol to delete from A"<<endl;

cin>>c;

A >> c;

cout<<"Deleting symbol "<<c<<" from A"<<endl;

A.outputSet();

printf("\n");

cout<<"OK";

}

case 2: return 0;

}

cout<<"ol";

return 0;

}

//Ââåäåííÿ ñòðîêè

“Multi\_set.h”

#ifndef MULTI\_SET\_H

#define MULTI\_SET\_H

#pragma once

#define countOfNegativeElemnts 128

#include <ctype.h>

#include <cstdio>

#define num 5

#define number 10

#include <cstdlib>

#include <cstring>

#include <iostream>

using namespace std;

class Multi\_Set {

private:

char elements[20]; // CharSet

int capacity=0;

public:

Multi\_Set(void); // Default constructor

Multi\_Set(char\*,int); // Constructor

void outputSet(void) const; // Output all CharSet on screen

bool checkout();

void push(char\*,int);

// ---- OPERATIONS ----

//Union

const Multi\_Set operator+ (const Multi\_Set &obj) const;

Multi\_Set& operator+= (const Multi\_Set &obj);

//subtraction

const Multi\_Set operator- (const Multi\_Set &obj) const;

Multi\_Set& operator-= (const Multi\_Set &obj);

//Intersection

const Multi\_Set operator\* (const Multi\_Set &obj) const;

Multi\_Set& operator\*= (const Multi\_Set &obj);

// Adding and removing

Multi\_Set& operator<< (const char& c);

Multi\_Set& operator>> (const char& c);

};

#endif // MULTI\_SET\_H

“set.cpp”

#include "set.h"

#include "Multi\_Set.h"

// Default constructor

Multi\_Set::Multi\_Set(void) {

for (int i = 0; i < num; i++) elements[i] = '0';

}

// Constructor

Multi\_Set::Multi\_Set(char\* str,int len) {

for (int i = 0; i < len+1; i++) elements[i] = '/0';

int i=0;

char \* pch = strtok (str,",");

while (pch != NULL)

{

strcpy(&elements[i++],pch);

capacity++;

pch = strtok (NULL, ",");

}

}

// Output all CharSet on screen

void Multi\_Set::outputSet(void) const {

cout<<"Number of elements in a set: "<<capacity<<" ";

for (int i=0;i<capacity;i++){

cout<<elements[i]<<" ";

}

cout<<endl;

}

bool Multi\_Set::checkout(){

if ((elements[0]=='/0')||capacity==0)

return false;

return true;

}

void Multi\_Set::push(char\* str,int len){

for (int i = 0; i < len+1; i++) elements[i] = '/0';

int i=0;

char \* pch = strtok (str,",");

while (pch != NULL)

{

strcpy(&elements[i++],pch);

capacity++;

pch = strtok (NULL, ",");

}

}

// ---- OPERATIONS ----

//Union

const Multi\_Set Multi\_Set::operator+ (const Multi\_Set &obj) const {

Multi\_Set tmp = (\*this);

Multi\_Set result;

result.capacity=tmp.capacity;

for (int i=0; i<tmp.capacity; i++)

result.elements[i]=tmp.elements[i];

for (int i = 0; i < obj.capacity; i++){

bool flag=true;

for (int j=0;j<tmp.capacity;j++){

if (tmp.elements[j]==obj.elements[i])

flag=false;

}

if (flag) result.elements[result.capacity++]=obj.elements[i];

}

result.elements[result.capacity]='/0';

return result;

}

Multi\_Set& Multi\_Set::operator+= (const Multi\_Set &obj) {

for (int i = 0; i <obj.capacity; i++){

bool flag=true;

for (int j=0;j<this->capacity; j++){

if (this->elements[j]==obj.elements[i])

flag=false;

}

if (flag) this->elements[this->capacity++] = obj.elements[i];

}

this->elements[this->capacity]='/0';

return (\*this);

}

//Difference

const Multi\_Set Multi\_Set::operator- (const Multi\_Set &obj) const {

Multi\_Set tmp=(\*this);

Multi\_Set result;

result.capacity=tmp.capacity;

for (int i=0; i<tmp.capacity;i++)

result.elements[i]=tmp.elements[i];

for (int i = 0; i < result.capacity; i++){

for (int j=0;j<obj.capacity;j++){

if (result.elements[i]==obj.elements[j]){

int ccount = i;

for (;ccount<num-1;ccount++)

result.elements[ccount]=result.elements[ccount+1];

result.elements[ccount]='0';

i--;

result.capacity--;

}

}

}

return result;

}

Multi\_Set& Multi\_Set::operator-= (const Multi\_Set &obj) {

int ccount=0;

for (int i = 0; i < this->capacity; i++)

for (int j=0; j<obj.capacity;j++)

if (this->elements[i]==obj.elements[j]){

for ( ccount = i;ccount<this->capacity;ccount++)

this->elements[ccount]=this->elements[ccount+1];

this->elements[ccount]='0';

this->capacity--;

}

return (\*this);

}

//Intersection

const Multi\_Set Multi\_Set::operator\* (const Multi\_Set &obj) const {

Multi\_Set tmp = (\*this);

Multi\_Set result;

int ccount=0;

for (int i = 0; i < tmp.capacity; i++)

for (int j=0;j<obj.capacity;j++)

if (tmp.elements[i]==obj.elements[j]){

result.elements[ccount++]=tmp.elements[i];

result.capacity++;

}

result.elements[ccount]='0';

return result;

}

Multi\_Set& Multi\_Set::operator\*= (const Multi\_Set &obj) {

for (int i = 0; i < this->capacity; i++){

if (this->elements[0]=='/0')

break;

bool flag=false;

for (int j=0;j<obj.capacity;j++)

if (this->elements[i]==obj.elements[j])

flag=true;

if(!flag){

int ccount = i;

for (;ccount<num-1;ccount++)

this->elements[ccount]=this->elements[ccount+1];

this->elements[ccount]='0';

i--;

this->capacity--;

}

}

return (\*this);

}

// Adding and remove

Multi\_Set& Multi\_Set::operator<< (const char& c) {

this->elements[(this->capacity)] = c;

this->capacity++;

return (\*this);

}

Multi\_Set& Multi\_Set::operator>> (const char& c) {

for (int i=0;i<(this->capacity);i++)

if (this->elements[i]==c){

int ccount=i;

for (;ccount<this->capacity-1;ccount++)

this->elements[ccount]=this->elements[ccount+1];

this->capacity--;

}

return (\*this);

}}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 6.1:

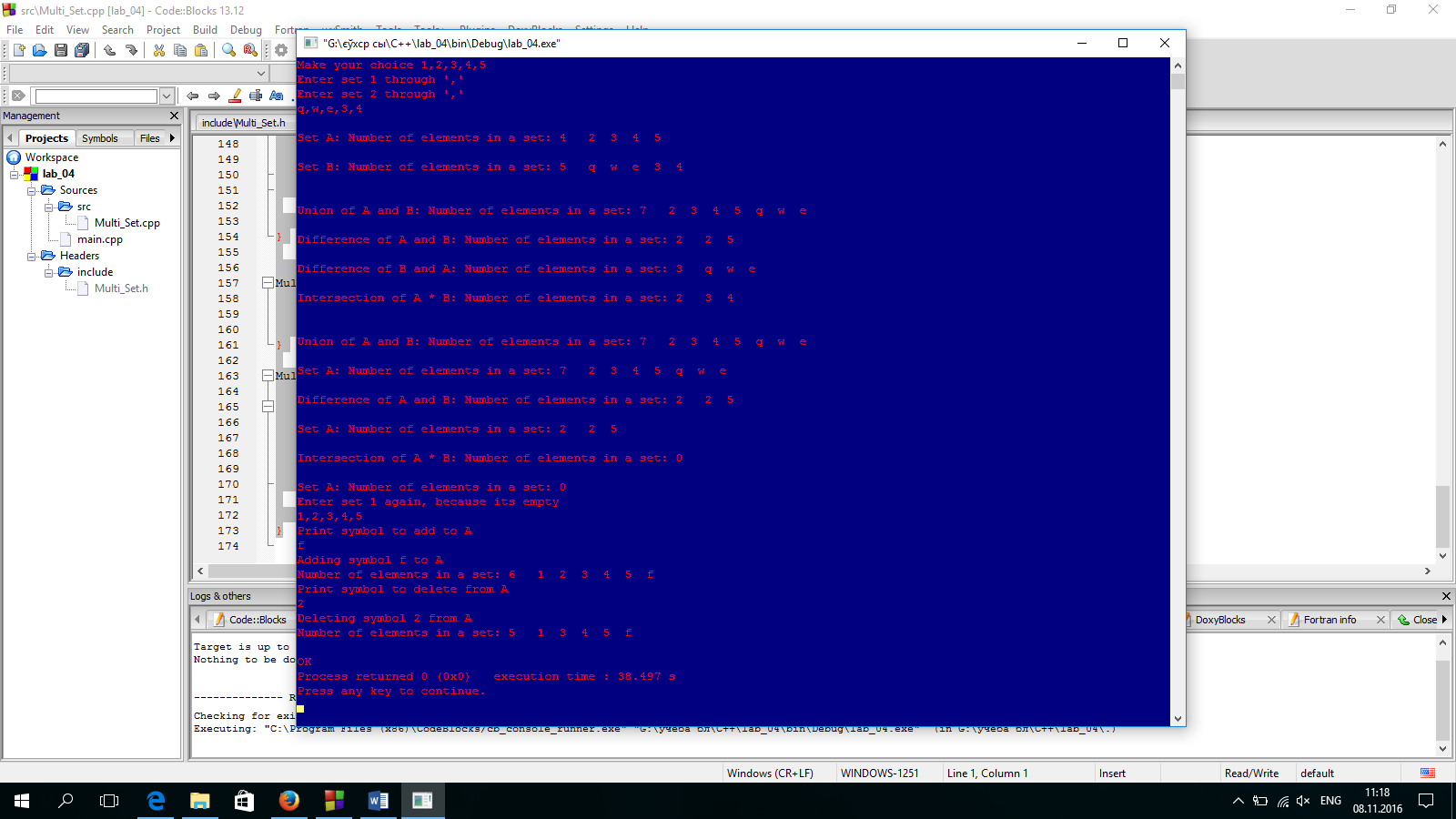


Рисунок 6.1 – Приклад виконання програми

# Висновок

Отже, дана програма створює множину символів коду ASCII та реалізує базові операції над нею (об’єднання, перетин, різниця, додавання символа до множини та видалення заданого символу).Розглянемо 2 множини «1,2,3,4,5» та «q,w,e,3,4».Першими проходять операції обєднання. Після даної операції ми отримуємо множину «1,2,3,4,5,q,w,e». Після операції віднімання від 1 множини другої отримаємо «1,2,5». Після операції перетину отримуємо «3,4». Після додавнання до множини символу та видалення в множину успішно записується символта видаляється.

Отже програма працює правльно.