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: «Обробка виключних ситуацій С++»

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

Київ – 2016

**ЗМІСТ:**

[1 Мета роботи 3](#_Toc463896091)

[2 Постановка задачі (варіант 7, рівень Б) 4](#_Toc463896092)

[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 7, рівень В)

Спроектировать класс «Set», который содержит множество символьных элементов, обязательно уникальных и упорядоченных. 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.

Добавити обробники виключних ситуацій.

# Діаграма класів

Діаграма класів наведена на рисунку 3.1:

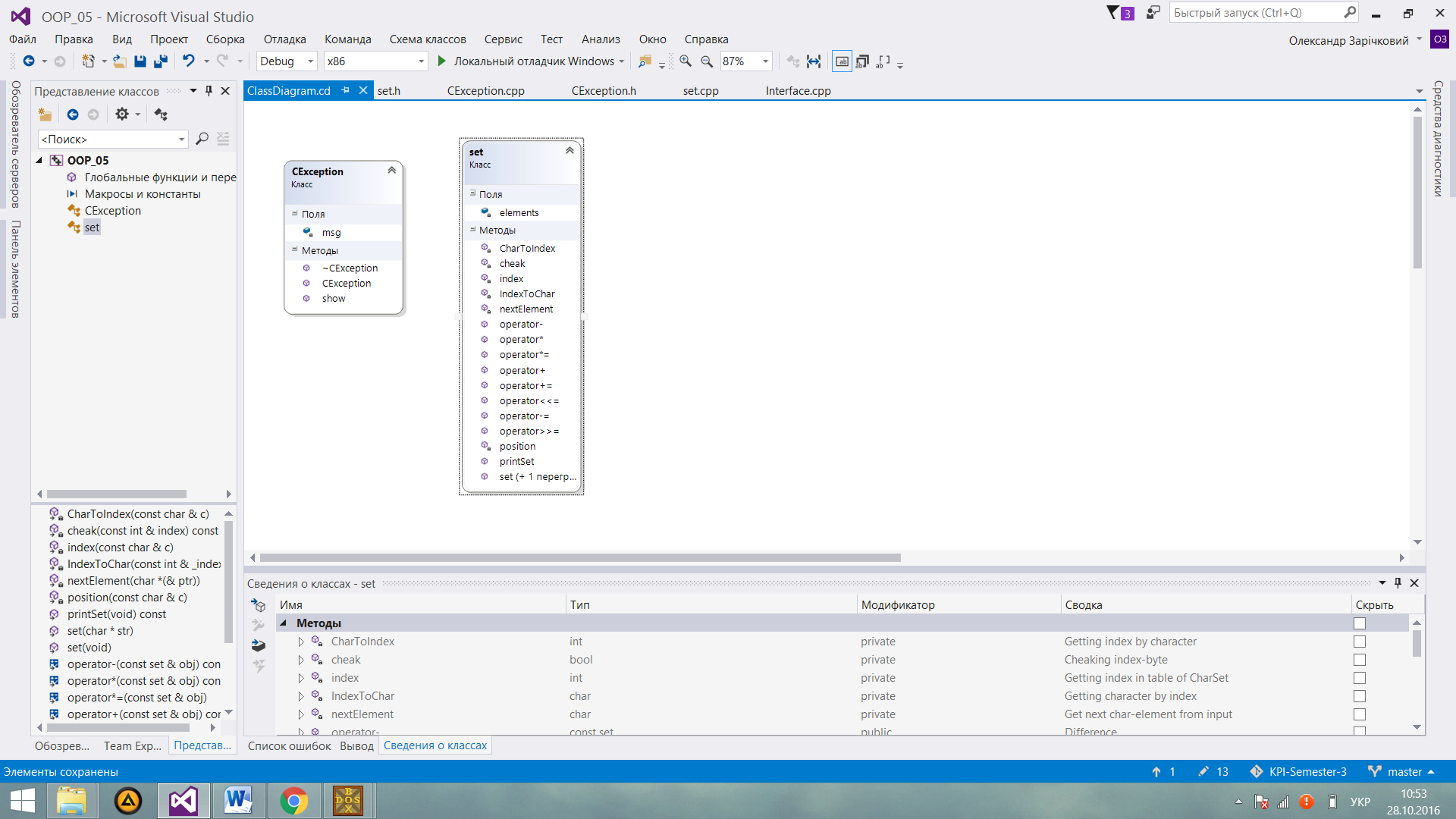


Рисунок 3.1 – Діаграма класів

# Код програми

“main.cpp”

#include "Interface.h"

int main(void) {

while (outputMenu());

}

“Interface.h”

#pragma once

#define size 1024

bool outputMenu(void); // Виведення меню

char\* inputStr(char c); // Введення строки

“Interface.cpp”

#include "Interface.h"

#include "set.h"

#include "CException.h"

#include <cstdio>

#include <cstdlib>

#include <cstring>

// Виведення меню

bool outputMenu(void) {

system("cls");

printf("OOP lab work #4\nVariant - 7, Level - C\nTask: Write set of charecters\nWriter: Alexander ALEXKIRNAS Zarichkovyi\n\n");

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

try {

char str[size];

gets\_s(str, size);

if (strlen(str) != 1) throw CException("Wrong input format!!!");

switch (str[0]) {

case '1': {

char\* str = inputStr('A');

set A(str);

delete[] str;

str = inputStr('B');

set B(str);

delete[] str;

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

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

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

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

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

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

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

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

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

for (char i = 'A'; i <= 'E'; i++) A <<= i;

printf("\n\nAdding to A symbols 'A' - 'E': "); A.printSet();

for (char i = 'C'; i <= 'F'; i++) A >>= i;

printf("\nDeleting from A symbols 'C' - 'F': "); A.printSet();

printf("\n");

system("pause");

break;

}

case '2': return false;

default: throw CException("Unexpected input!");

}

return true;

}

catch (CException& excection) {

excection.show();

system("pause");

return true;

}

}

//Введення строки

char\* inputStr(char c) {

char\* str = new char[size];

printf("Input %c set: ", c);

gets\_s(str, size);

return str;

}

“set.h”

#pragma once

#define countOfNegativeElemnts 128

#define baseElement unsigned \_\_int64

#define capacity 4

class set {

private:

baseElement elements[capacity]; // CharSet

static char nextElement(char\* (&ptr)); // Get next char-element from input

static int index(const char &c); // Getting index in table of CharSet

static int position(const char &c); // Getting position character in table cell

static int CharToIndex(const char& c); // Getting index by character

static char IndexToChar(const int& \_index); // Getting character by index

bool cheak(const int& index) const; // Cheaking index-byte

public:

set(void); // Default constructor

set(char\* str); // Constructor

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

// ---- OPERATIONS ----

//Union

const set operator+ (const set &obj) const;

set& operator+= (const set &obj);

//Difference

const set operator- (const set &obj) const;

set& operator-= (const set &obj);

//Intersection

const set operator\* (const set &obj) const;

set& operator\*= (const set &obj);

// Adding and remove

set& operator<<= (const char& c);

set& operator>>= (const char& c);

};

“set.cpp”

#include "set.h"

#include "CException.h"

#include <ctype.h>

#include <cstdio>

#include <cstdlib>

#include <string.h>

// Get next char-element from input

char set::nextElement(char\* (&ptr)) {

while (isspace(\*ptr)) ptr++; // Eating space elements before symbol

char symbol = \*ptr;

ptr++;

while (isspace(\*ptr)) ptr++; // Eateing space elents after symbol

if (\*ptr != ',' && \*ptr) {

char msg[sizeT];

msg[0] = 0;

strcpy(msg, "Error! Wrong input format!!! Expected \",\" but found:\n");

strcat(msg, ptr);

throw CException(msg);

system("pause");

}

if(\*ptr) ptr++;

return symbol;

}

// Getting index in table of CharSet

inline int set::index(const char &c) {

return CharToIndex(c) / (sizeof(baseElement) \* 8);

}

// Getting position character in table cell

inline int set::position(const char &c) {

return CharToIndex(c) % (sizeof(baseElement) \* 8);

}

// Getting character index

inline int set::CharToIndex(const char& c) {

return static\_cast<int> (c) + countOfNegativeElemnts;

}

// Getting character by index

inline char set::IndexToChar(const int& index) {

return static\_cast<char> (index - countOfNegativeElemnts);

}

// Cheaking index-byte

bool set::cheak(const int& \_index) const{

int base = (sizeof(baseElement) \* 8);

return elements[\_index / base] & (1ll << position(\_index % base));

}

// Default constructor

set::set(void) {

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

}

// Constructor

set::set(char\* str) {

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

while (\*str){

char symbol = nextElement(str);

(\*this) <<= symbol;

}

}

// Output all CharSet on screen

void set::printSet(void) const {

bool flag = false;

bool isFirst = true;

for (int i = 0, size = sizeof(baseElement) \* capacity \* 8; i < size; i++)

if (cheak(i)) {

if (!isFirst) printf(", ");

printf("%c", IndexToChar(i));

flag = true;

isFirst = false;

}

if (!flag) printf("CharSet is Empty!");

}

// ---- OPERATIONS ----

//Union

const set set::operator+ (const set &obj) const {

set tmp = (\*this);

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

tmp.elements[i] |= obj.elements[i];

return tmp;

}

set& set::operator+= (const set &obj) {

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

this->elements[i] |= obj.elements[i];

return (\*this);

}

//Difference

const set set::operator- (const set &obj) const {

set tmp(\*this);

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

tmp.elements[i] ^= this->elements[i] & obj.elements[i];

return tmp;

}

set& set::operator-= (const set &obj) {

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

this->elements[i] ^= this->elements[i] & obj.elements[i];

return (\*this);

}

//Intersection

const set set::operator\* (const set &obj) const {

set tmp;

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

tmp.elements[i] = this->elements[i] & obj.elements[i];

return tmp;

}

set& set::operator\*= (const set &obj) {

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

this->elements[i] &= obj.elements[i];

return (\*this);

}

// Adding and remove

set& set::operator<<= (const char& c) {

elements[index(c)] |= (1ll << position(c));

return (\*this);

}

set& set::operator>>= (const char& c) {

if(elements[index(c)] & (1ll << position(c)))

elements[index(c)] ^= (1ll << position(c));

return (\*this);

}

“CException.h”

#pragma once

class CException {

private:

char\* msg;

public:

CException(char\* message);

~CException();

void show(void);

};

“ CException.cpp”

#include "CException.h"

#include <cstdio>

#include <string.h>

CException::CException(char\* message) {

msg = new char[strlen(message) + 1];

strcpy(msg, message);

}

CException::~CException() {

delete[] msg;

msg = nullptr;

}

void CException::show(void) {

printf("%s\n", msg);

}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 5.1:

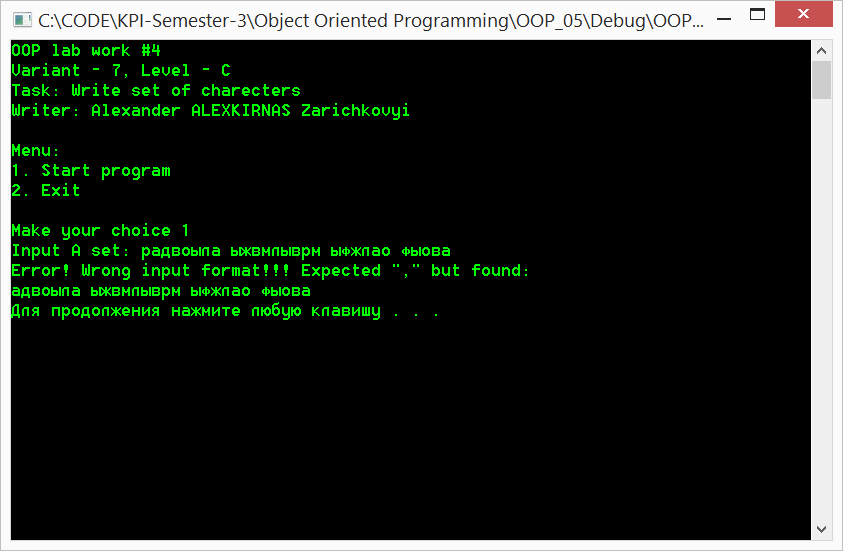


Рисунок 5.1 – Приклад виконання програми

# Висновок

Отже, дана програма створює множину символів коду ASCII та реалізує базові операції над нею (об’єднання, перетин, різниця). Добавлено обробники виключних ситуацій у всіх потенційно небезпечних місяцях програми. Оскільки дана програма реалізована за модульним принципом і кожен модуль програми створювався й налагоджувався автономно, то дана програма буде видавати очікуваний результат на всіх наборах вхідних даних.