Минобрнауки России

Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный

электротехнический университет «ЛЭТИ»

им В. И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Лабораторная работа № 2**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Множество как объект»**

Выполнили студенты группы 4311: Михаил Левицкий , Фролов Никита

Принял: Аббас С. А.

Санкт-Петербург

2025

# **1. Цель работы**

Сравнение процедурного и объектно-ориентированного подходов на примере задачи обработки множеств

# **2. Задание 27**

Универсум: Прописные латинские буквы

Что надо вычислить: Множество, содержащее все буквы из A, не являющиеся общими для B, C и D

**3.** **Результаты эксперимента с четырьмя структурами данных на основе классов (рисунки с тестами и таблица результатов измерения времени).**

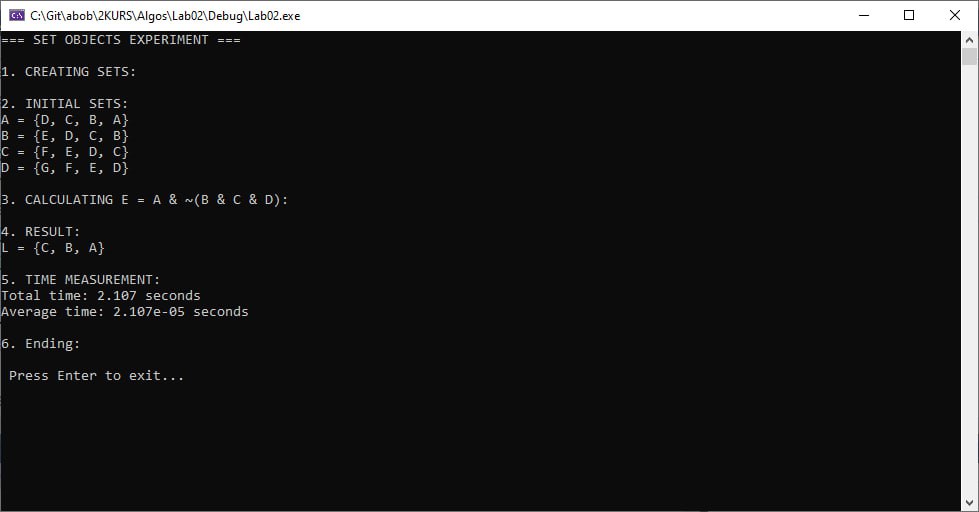
Рис. 1 - LIST  


Рис. 2 - ARRAY.

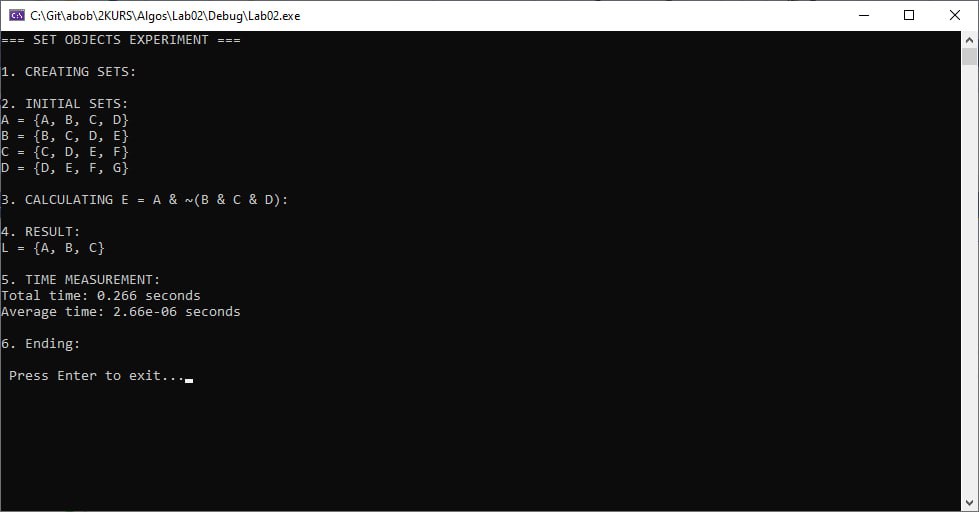
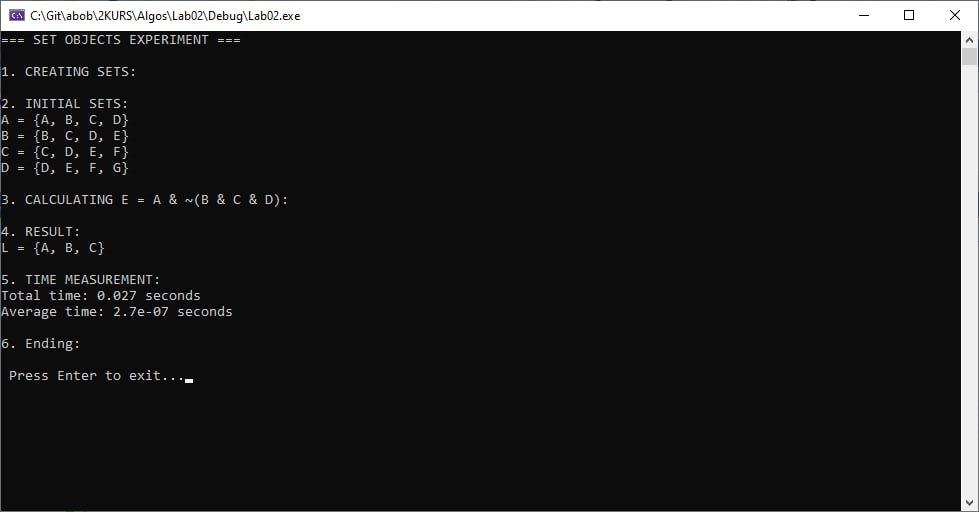
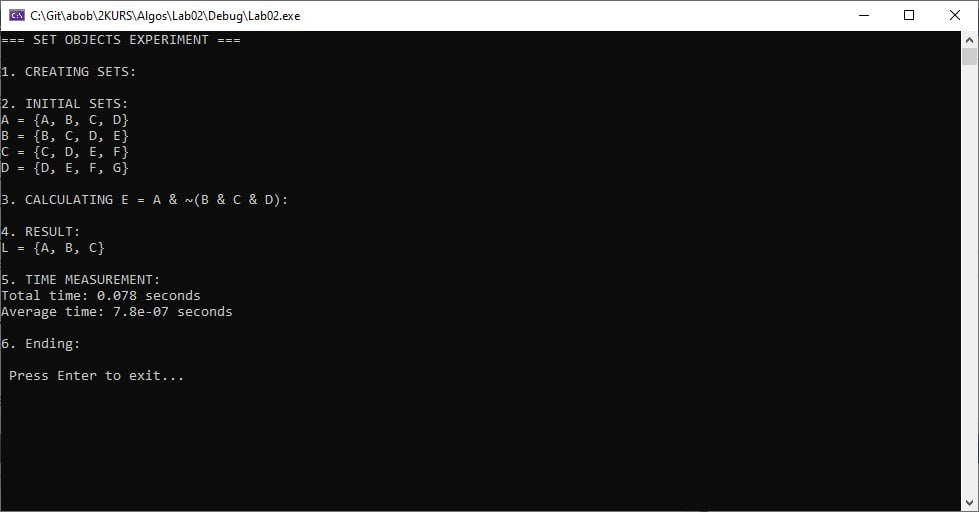
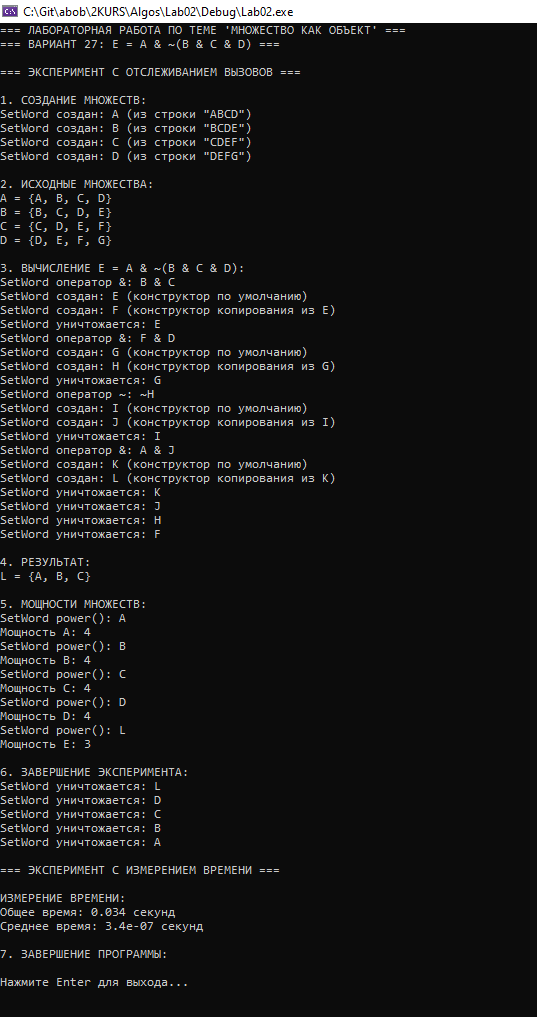
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Рис. 3 - Машинное слово  


Рис. 4 - SETBOOL  
 **3.1. Таблица результатов измерения времени**

|  |  |
| --- | --- |
| **Реализация** | **Реузльтат измерения времени** |
| **SetArray** | **0.266** |
| **SetWord** | **0.027** |
| **SetList** | **2.107** |
| **SetBool** | **0.078** |

Исходя из таблицы мы видим ,что самая быстрая реализация это машинное слово , а самая медленная это обход списка.

4. **Результат эксперимента с отслеживанием вызовов функций-членов**.



5**. Выводы**

В ходе работы были изучены и получены практические навыки сравнения процедурного и объектно-ориентированного подходов на примере задачи обработки множеств. Был проведён эксперимент с четырьмя структурами данных, реализованными на основе классов.

Установлено, что наиболее медленной реализацией является связный список (SetList), тогда как наибольшую скорость показало представление множества в виде машинного слова (SetWord).

Также было выполнено качественное исследование количества вызовов функций-членов, что позволило оценить внутренние издержки каждой реализации.

**6. Список используемых источников**

1. Колинько, П.Г. Пользовательские структуры данных : методические указания по дисциплине «Алгоритмы и структуры данных, часть 1».СПбГЭТУ «ЛЭТИ», 2025

2. Множества в памяти ЭВМ // Алгоритмы и структуры данных. Лекционнные материалы.

7**. Приложение. Текст программы**  
**Основная программа:**#include <ctime>

#include <iostream>

// Выберите представление (раскомментируйте одну строку):

#define USE\_ARRAY

// #define USE\_LIST

// #define USE\_WORD

// #define USE\_BOOL

#ifdef USE\_ARRAY

#include "SetArray.h"

typedef SetArray SetType;

#elif defined(USE\_LIST)

#include "SetList.h"

typedef SetList SetType;

#elif defined(USE\_WORD)

#include "SetWord.h"

typedef SetWord SetType;

#elif defined(USE\_BOOL)

#include "SetBool.h"

typedef SetBool SetType;

#endif

int main() {

{

std::cout << "=== SET OBJECTS EXPERIMENT ===" << std::endl;

// Создание множеств

std::cout << "\n1. CREATING SETS:" << std::endl;

SetType A("ABCD");

SetType B("BCDE");

SetType C("CDEF");

SetType D("DEFG");

// Показ исходных данных

std::cout << "\n2. INITIAL SETS:" << std::endl;

A.Show();

B.Show();

C.Show();

D.Show();

// Вычисление E по формуле из первой лабы: E = A & ~(B & C & D)

std::cout << "\n3. CALCULATING E = A & ~(B & C & D):" << std::endl;

SetType E = A & ~(B & C & D);

// Результат

std::cout << "\n4. RESULT:" << std::endl;

E.Show();

// Измерение времени

std::cout << "\n5. TIME MEASUREMENT:" << std::endl;

const int REPEAT\_COUNT = 100000;

clock\_t start = clock();

// Сброс счетчика перед циклом измерения времени

SetType::ResetCounter();

for (int i = 0; i < REPEAT\_COUNT; i++) {

SetType temp = A & ~(B & C & D);

}

clock\_t end = clock();

double time = double(end - start) / CLOCKS\_PER\_SEC;

std::cout << "Total time: " << time << " секунд" << std::endl;

std::cout << "Average time: " << time / REPEAT\_COUNT << " секунд" << std::endl;

std::cout << "\n6. Ending:" << std::endl;

}

std::cout << "\n Press Enter to exit...";

std::cin.get();

return 0;

}  
  
**Первый файл:SetArray.h**#pragma once

#include <iostream>

class SetArray {

private:

static int N, cnt;

int n;

char S;

char\* A;

public:

// Конструкторы

SetArray() : n(0), S('A' + cnt++), A(new char[N + 1]) {

A[0] = 0;

}

SetArray(const SetArray& other) : n(other.n), S('A' + cnt++), A(new char[N + 1]) {

for (int i = 0; i <= other.n; i++) {

A[i] = other.A[i];

}

}

SetArray(const char\* str) : n(0), S('A' + cnt++), A(new char[N + 1]) {

A[0] = 0;

if (str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

bool found = false;

for (int j = 0; j < n; j++) {

if (A[j] == str[i]) {

found = true;

break;

}

}

if (!found) A[n++] = str[i];

}

}

A[n] = 0;

}

}

// Деструктор

~SetArray() {

delete[] A;

}

// Оператор присваивания

SetArray& operator=(const SetArray& other) {

if (this != &other) {

n = other.n;

for (int i = 0; i <= other.n; i++) {

A[i] = other.A[i];

}

}

return \*this;

}

// Операции с множествами

SetArray operator&(const SetArray& other) const {

SetArray result;

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

bool found = false;

for (int j = 0; j < other.n; j++) {

if (A[i] == other.A[j]) {

found = true;

break;

}

}

if (found) result.A[result.n++] = A[i];

}

result.A[result.n] = 0;

return result;

}

SetArray operator|(const SetArray& other) const {

SetArray result = \*this;

for (int i = 0; i < other.n; i++) {

bool found = false;

for (int j = 0; j < result.n; j++) {

if (result.A[j] == other.A[i]) {

found = true;

break;

}

}

if (!found) result.A[result.n++] = other.A[i];

}

result.A[result.n] = 0;

return result;

}

SetArray operator~() const {

SetArray result;

for (char c = 'A'; c <= 'Z'; c++) {

bool found = false;

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

if (A[i] == c) {

found = true;

break;

}

}

if (!found) result.A[result.n++] = c;

}

result.A[result.n] = 0;

return result;

}

// Методы

void Show() const {

std::cout << S << " = {";

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

std::cout << A[i];

if (i < n - 1) std::cout << ", ";

}

std::cout << "}" << std::endl;

}

int power() const { return n; }

// Статический метод для сброса счетчика

static void ResetCounter() { cnt = 0; }

};

int SetArray::N = 26;

int SetArray::cnt = 0;  
  
**Второй файл:SetBool.h**#pragma once

#include <iostream>

class SetBool {

private:

static int N, cnt;

char S;

bool elements[26];

public:

SetBool() : S('A' + cnt++) {

for (int i = 0; i < 26; i++) {

elements[i] = false;

}

}

SetBool(const SetBool& other) : S('A' + cnt++) {

for (int i = 0; i < 26; i++) {

elements[i] = other.elements[i];

}

}

SetBool(const char\* str) : S('A' + cnt++) {

for (int i = 0; i < 26; i++) {

elements[i] = false;

}

if (str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

elements[str[i] - 'A'] = true;

}

}

}

}

~SetBool() {}

SetBool& operator=(const SetBool& other) {

if (this != &other) {

for (int i = 0; i < 26; i++) {

elements[i] = other.elements[i];

}

}

return \*this;

}

SetBool operator&(const SetBool& other) const {

SetBool result;

for (int i = 0; i < 26; i++) {

result.elements[i] = elements[i] && other.elements[i];

}

return result;

}

SetBool operator|(const SetBool& other) const {

SetBool result;

for (int i = 0; i < 26; i++) {

result.elements[i] = elements[i] || other.elements[i];

}

return result;

}

SetBool operator~() const {

SetBool result;

for (int i = 0; i < 26; i++) {

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

}

return result;

}

void Show() const {

std::cout << S << " = {";

bool first = true;

for (int i = 0; i < 26; i++) {

if (elements[i]) {

if (!first) std::cout << ", ";

std::cout << char('A' + i);

first = false;

}

}

std::cout << "}" << std::endl;

}

int power() const {

int count = 0;

for (int i = 0; i < 26; i++) {

if (elements[i]) count++;

}

return count;

}

// Статический метод для сброса счетчика

static void ResetCounter() { cnt = 0; }

};

int SetBool::N = 26;

int SetBool::cnt = 0;  
  
**Третий файл:SetList.h**#pragma once

#include <iostream>

struct ListNode {

char data;

ListNode\* next;

ListNode(char d, ListNode\* n = nullptr) : data(d), next(n) {}

};

class SetList {

private:

static int N, cnt;

int n;

char S;

ListNode\* head;

public:

SetList() : n(0), S('A' + cnt++), head(nullptr) {}

SetList(const SetList& other) : n(0), S('A' + cnt++), head(nullptr) {

ListNode\* current = other.head;

while (current) {

add(current->data);

current = current->next;

}

}

SetList(const char\* str) : n(0), S('A' + cnt++), head(nullptr) {

if (str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

add(str[i]);

}

}

}

}

~SetList() {

clear();

}

SetList& operator=(const SetList& other) {

if (this != &other) {

clear();

ListNode\* current = other.head;

while (current) {

add(current->data);

current = current->next;

}

}

return \*this;

}

SetList operator&(const SetList& other) const {

SetList result;

ListNode\* current = head;

while (current) {

if (other.contains(current->data)) {

result.add(current->data);

}

current = current->next;

}

return result;

}

SetList operator|(const SetList& other) const {

SetList result = \*this;

ListNode\* current = other.head;

while (current) {

if (!result.contains(current->data)) {

result.add(current->data);

}

current = current->next;

}

return result;

}

SetList operator~() const {

SetList result;

for (char c = 'A'; c <= 'Z'; c++) {

if (!contains(c)) {

result.add(c);

}

}

return result;

}

void Show() const {

std::cout << S << " = {";

ListNode\* current = head;

while (current) {

std::cout << current->data;

if (current->next) std::cout << ", ";

current = current->next;

}

std::cout << "}" << std::endl;

}

int power() const { return n; }

// Статический метод для сброса счетчика

static void ResetCounter() { cnt = 0; }

private:

void add(char c) {

if (contains(c)) return;

head = new ListNode(c, head);

n++;

}

bool contains(char c) const {

ListNode\* current = head;

while (current) {

if (current->data == c) return true;

current = current->next;

}

return false;

}

void clear() {

while (head) {

ListNode\* temp = head;

head = head->next;

delete temp;

}

n = 0;

}

};

int SetList::N = 26;

int SetList::cnt = 0;  
  
**Четвертый файл:SetWord.h**#pragma once

#include <iostream>

class SetWord {

private:

static int N, cnt;

char S;

unsigned int word;

public:

SetWord() : S('A' + cnt++), word(0) {}

SetWord(const SetWord& other) : S('A' + cnt++), word(other.word) {}

SetWord(const char\* str) : S('A' + cnt++), word(0) {

if (str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

word |= (1 << (str[i] - 'A'));

}

}

}

}

~SetWord() {}

SetWord& operator=(const SetWord& other) {

if (this != &other) {

word = other.word;

}

return \*this;

}

SetWord operator&(const SetWord& other) const {

SetWord result;

result.word = word & other.word;

return result;

}

SetWord operator|(const SetWord& other) const {

SetWord result;

result.word = word | other.word;

return result;

}

SetWord operator~() const {

SetWord result;

result.word = ~word & 0x03FFFFFF;

return result;

}

void Show() const {

std::cout << S << " = {";

bool first = true;

for (int i = 0; i < 26; i++) {

if (word & (1 << i)) {

if (!first) std::cout << ", ";

std::cout << char('A' + i);

first = false;

}

}

std::cout << "}" << std::endl;

}

int power() const {

int count = 0;

for (int i = 0; i < 26; i++) {

if (word & (1 << i)) count++;

}

return count;

}

// Статический метод для сброса счетчика

static void ResetCounter() { cnt = 0; }

};

int SetWord::N = 26;

int SetWord::cnt = 0;