Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)
Факультет компьютерных технологий и информатики

Кафедра ВТ

Отчёт
по лабораторной работе № 2
на тему: “Множества”
по дисциплине “Алгоритмы и структуры данных”
Вариант 25

Выполнили: Студенты группы 6306
Гордиенко М. Е.
Пустовойтова А. А.
Принял: Колинько П. Г.

Санкт-Петербург
2017

Цель

Исследовать 4 способа хранения множеств в памяти ЭВМ, используя классы.

Задание

Универсум — десятичные цифры.

Вычислить множество, содержащие цифры, имеющиеся в каждом из множеств A, B, C, D.

E = A^B^C^D.

Способы задания множества

1. Массив символов
2. Список
3. Массив битов
4. Машинное слово

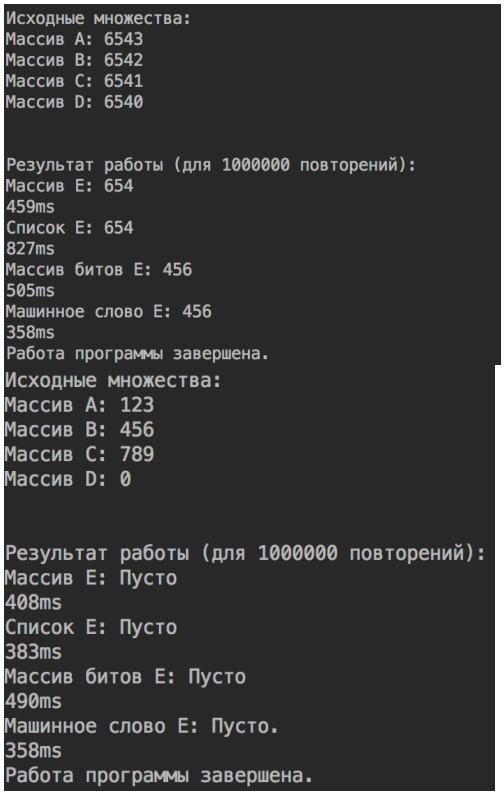
Результаты эксперимента с использованием классов

Примеры представлены в таблице 1.

Таблица. 1. Примеры

№	Исходные множества				Результат
	A	B	C	D	E
1	{6543}	{6542}	{6541}	{6540}	{654}
2	{123}	{456}	{789}	{0}	{ }

1. Демонстрация работы программы с контрольными примерами 1 и 2 из таблицы 1.



Временная сложность

Временная сложность представлена в таблице 2.

Таблица. 2. Временная сложность

Способ представления	Ожидаемая	Фактическая
Массив	O(n^2)	O(n^2)
Список	O(n^2)	O(n^2)
Машинное слово	O(1)	O(1)
Массив битов	O(1)	O(1)

Результаты измерения времени обработки

Результаты измерения времени обработки представлены в таблице 3.

Таблица. 3. Результаты измерения времени обработки

Способ представления	Количество миллисекунд	Количество повторов цикла
Массив	~359-892	1000000
Список	~351-2298	
Машинное слово	~490-543	
Массив битов	~358-364	

Вывод: Представление множества в виде машинного слова самый быстрый из способов обработки данных для рассмотренной задачи, т.к. оно не зависит от количества элементов в множестве. Классы оказались удобными в использовании. Дают возможность защиты переменных от несанкционированного изменения. Облегчают понимание программы.

Список используемых источников

- Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Федеральный образовательный стандарт / сост.: П.Г. Колинько. - СПб.: Изд-во СПБГЭТУ "ЛЭТИ", 2017. - 64 с.

Array.cpp

```
#include <cmath>
#include "Array.h"

Array& Array::operator = (const Array &arr) {
    if (&arr != this) {
        short i = 0;
        do {
            this->elements[i] = arr.elements[i];
            i++;
        } while (arr.elements[i - 1] != '\0');
        this->size = arr.size;
    }
    return *this;
}

void Array::addElem(char elem) {
    if (size == max_size) {
        cout << "Ошибка. Массив полон.\n";
    }
    else {
        this->elements[size++] = elem;
        this->elements[size] = '\0';
    }
}

char* Array::getElements() {
    return elements;
}

void Array::init() {
    cout << "Введите массив " << name << ": ";
    char ch;
    do {
        ch = (char)cin.get();
        if (isdigit(ch) && !Helper::elemInArr(ch, this->elements)) {
            this->elements[size] = ch;
            size++;
            this->elements[size] = '\0';
        }
    } while (ch != '\n');
}

void Array::print() {
    cout << "Массив " << name << ": ";
    if (size == 0) {
        cout << "Пусто\n";
    }
    else {
        cout << elements << endl;
    }
}

Array& Array::conj(const Array &first_arr, const Array &sec_arr, const Array &third_arr, const Array &fourth_arr){
    if (size > 0) {
        elements[0] = '\0';
```

```
        size = 0;
    }
    for (short i = 0; i <= first_arr.size; i++) {
        if (Helper::elemInArr(first_arr.elements[i], sec_arr.elements) &&
            Helper::elemInArr(first_arr.elements[i], third_arr.elements) &&
            Helper::elemInArr(first_arr.elements[i], fourth_arr.elements)) {
            this->addElem(first_arr.elements[i]);
        }
    }
    return *this;
}

void Array::randInit() {
    unsigned short num = (unsigned short)(rand() % (int)pow(2, max_size));
    unsigned short i = 0, j = 0;
    while (num > 0) {
        if (num % 2 > 0) elements[j++] = (char)(i + '0');
        num = num >> 1;
        i++;
    }
    size = j - 1;
    elements[j] = '\0';
}
```

Array.h

```
#ifndef INC_2_ARRAY_H
#define INC_2_ARRAY_H

#include <iostream>
#include "Helper.h"
using namespace std;

class Array {
public:
    Array() {
        elements = new char[max_size + 1];
        elements[0] = '\0';
        name = (char)arr_count + 'A';
        arr_count++;
        size = 0;
    }

    Array(const Array &arr) {
        elements = new char[max_size + 1];
        short i = 0;
        do {
            elements[i] = arr.elements[i];
            i++;
        } while (elements[i-1] != '\0');
        name = arr.name;
        size = arr.size;
    }

    ~Array() {
        delete elements;
    }

    Array& operator = (const Array &arr);

    void addElem(char elem);

    void init();

    void randInit();

    void print();

    char* getElements();
```

```
    Array& conj(const Array &first_arr, const Array &sec_arr, const Array &third_arr, const Array &fourth_arr);

private:
    char* elements = nullptr;

    int size;

    static const int max_size = 10;

    static unsigned short arr_count;

    char name;
};


#endif //INC_2_ARRAY_H
```

Bit.cpp

```
#include <iostream>
#include <cmath>
#include "Bit.h"

Bit& Bit::operator = (const Bit &obj) {
    if (&obj != this) {
        this->count = obj.count;
    }
    return *this;
}

Bit& Bit::operator = (const unsigned short count) {
    this->count = count;
    return *this;
}

Bit& Bit::operator = (const char* arr) {
    for (unsigned short i = 0; arr[i] != '\0'; i++) {
        this->count += (pow(2, arr[i] - '0'));
    }
    return *this;
}

Bit Bit::conj(const Bit &first_bit, const Bit &sec_bit, const Bit &third_bit, const Bit &fourth_bit) {
    this->count = first_bit.count & sec_bit.count & third_bit.count & fourth_bit.count;
    return *this;
}

void Bit::print() {
    using namespace std;
    cout << "Машинное слово " << name << ": ";
    if (this->count == 0) {
        cout << "Пусто.\n";
    }
    else {
        unsigned short i = 0, num = this->count;
        while (num > 0) {
            if (num % 2 > 0) cout << i;
            num = num >> 1;
            i++;
        }
        cout << endl;
    }
}
```

Bit.h

```
#ifndef INC_2_BIT_H
#define INC_2_BIT_H


class Bit {
public:
    Bit() {
        count = 0;
```

```
        name = char(bit_count + 'A');
        bit_count++;
    }

    Bit(const Bit &obj) {
        this->count = obj.count;
    }

    Bit(unsigned short num) {
        count = num;
    }

    ~Bit() {}

    Bit& operator = (const Bit &obj);

    Bit& operator = (const unsigned short count);

    Bit& operator = (const char* arr);

    Bit conj(const Bit &first_bit, const Bit &sec_bit, const Bit &third_bit, const Bit &fourth_bit);

    void print();
private:
    char name;

    unsigned short count;

    static unsigned short bit_count;
};


#endif //INC_2_BIT_H
```

Bit_Arr.cpp

```
#include <iostream>
#include "Bit_Arr.h"

Bit_Arr& Bit_Arr::operator = (const Bit_Arr &obj) {
    for (short i = 0; i < size; i++) {
        this->elements[i] = obj.elements[i];
    }
    return *this;
}

Bit_Arr& Bit_Arr::operator = (char* arr) {
    for (short i = 0; i < size; i++) elements[i] = false;
    for (short i = 0; arr[i] != '\0'; i++) this->elements[arr[i] - '0'] = true;
    return *this;
}

void Bit_Arr::print() {
    std::cout << "Массив битов " << name << ": ";
    if (this->isEmpty()) {
        std::cout << "Пусто";
    }
    else {
        for (short i = 0; i < size; i++) {
            if (elements[i]) std::cout << i;
        }
    }
    std::cout << std::endl;
}

Bit_Arr Bit_Arr::conj(const Bit_Arr &arr_1, const Bit_Arr &arr_2, const Bit_Arr &arr_3, const Bit_Arr &arr_4) {
    for (short i = 0; i < size; i++)
        this->elements[i] = arr_1.elements[i] && arr_2.elements[i]
                            && arr_3.elements[i] && arr_4.elements[i];
    return *this;
}

bool Bit_Arr::isEmpty() {
    bool result = true;
    for (short i = 0; i < size; i++) {
        if (elements[i]) result = false;
```

```
    }
    return result;
}
```

Bit_Arr.h

```
#ifndef INC_2_BIT_ARR_H
#define INC_2_BIT_ARR_H


class Bit_Arr {
public:
    Bit_Arr() {
        elements = new bool[size];
        for (short i = 0; i < size; i++) elements[i] = false;
        name = (char)(bit_arr_count++) + 'A';
    }

    Bit_Arr(Bit_Arr &bit_arr) {
        elements = new bool[size];
        for (short i = 0; i < size; i++) elements[i] = bit_arr.elements[i];
        name = bit_arr.name;
    }

    ~Bit_Arr() {
        delete elements;
    }

    Bit_Arr& operator = (const Bit_Arr &obj);

    Bit_Arr& operator = (char* arr);

    void print();

    Bit_Arr conj(const Bit_Arr &arr_1, const Bit_Arr &arr_2, const Bit_Arr &arr_3, const Bit_Arr &arr_4);

    bool isEmpty();
private:
    bool* elements;
    const short size = 10;
    char name;
    static unsigned short bit_arr_count;
};


#endif
```

Helper.cpp

```
#include <cmath>
#include <iostream>
#include "Helper.h"

bool Helper::elemInArr(char elem, char* array){
    for (short i = 0; array[i] != '\0'; i++) {
        if (elem == array[i]) return true;
    }
    return false;
}

unsigned short Helper::bitFromArr(char* array) {
    unsigned short result = 0;
    for (short i = 0; array[i] != '\0'; i++) {
        result += pow(2, array[i] - '0');
    }
    return result;
}

unsigned short Helper::showMenu() {
    using namespace std;
    cout << "Выберете соответствующий пункт меню:\n" <<
         "(1) - Ввести множества вручную.\n" <<
         "(2) - Обработать случайные множества.\n" <<
         "(3) - Тест времени обработки.\n" <<
```

```
        "(0) - Выход из программы.\n" <<
        "Ваш выбор: ";
    return 3; // Максимальный номер пункта меню.
}

unsigned short Helper::Choise (unsigned short menu_count) {
    char ch;
    unsigned short num = 0;
    do {
        ch = (char) getc(stdin);
        std::cin.ignore(INT64_MAX, '\n');
        if (isdigit(ch)) {
            num = (unsigned short) ch - '0';
        }
        if (num <= menu_count) {
            return num;
        }
        else std::cout << "Ошибка. Такого пункта нет.\n Введите заново: ";
    } while (true);
}
```

Helper.h

```
#ifndef INC_2_HELPER_H
#define INC_2_HELPER_H


class Helper {
public:
    static bool elemInArr(char elem, char* array);
    static unsigned short bitFromArr(char* array);
    static unsigned short showMenu();
    static unsigned short Choise (unsigned short menu_count);
};


#endif
```

List.cpp

```
#include <iostream>
#include "List.h"

List& List::operator = (const List &list) {
    Elem *this_elem, *list_elem = list.head;
    if (this->head != nullptr) this->clear();
    while (list_elem != nullptr) {
        this_elem = new Elem;
        if (this->head == nullptr) this->head = this_elem;
        this_elem->count = list_elem->count;
        list_elem = list_elem->next;
    }
}

List& List::operator = (char* arr) {
    if (this->head != nullptr) this->clear();
    unsigned short i = 0;
    Elem* this_elem, *prev_elem = nullptr;
    while (arr[i] != '\0') {
        this_elem = new Elem;
        if (this->head == nullptr) this->head = this_elem;
        this_elem->count = arr[i++];
        if (prev_elem != nullptr) prev_elem->next = this_elem;
        prev_elem = this_elem;
    }
}

List& List::conj(List &list_1, List &list_2, List &list_3, List &list_4) {
    if (this->head != nullptr) this->clear();
    Elem *this_elem = list_1.head;
    while (this_elem != nullptr) {
        if (list_2.elemInList(this_elem->count) &&
                list_3.elemInList(this_elem->count) &&
                list_4.elemInList(this_elem->count))
```

```
            this->addElem(this_elem->count);
        this_elem = this_elem->next;
    }
    return *this;
}

bool List::elemInList(char elem) {
    Elem *this_elem = this->head;
    while (this_elem != nullptr) {
        if (this_elem->count == elem) return true;
        else this_elem = this_elem->next;
    }
    return false;
}

void List::print() {
    std::cout << "Список " << name << ": ";
    if (head == nullptr) std::cout << "Пусто";
    else {
        Elem *this_elem = this->head;
        while (this_elem != nullptr) {
            std::cout << this_elem->count;
            this_elem = this_elem->next;
        }
    }
    std::cout << "\n";
}

void List::clear() {
    Elem *this_elem, *prev_elem = this->head;
    while (prev_elem != nullptr) {
        this_elem = prev_elem->next;
        delete prev_elem;
        prev_elem = this_elem;
    }
    this->head = nullptr;
}

void List::addElem(char count) {
    if (head == nullptr) {
        head = new Elem;
        head->count = count;
    }
    else {
        Elem *this_elem = head;
        while (this_elem->next != nullptr) {
            this_elem = this_elem->next;
        }
        this_elem->next = new Elem;
        this_elem->next->count = count;
    }
}
```

List.h

```
#ifndef INC_2_LIST_H
#define INC_2_LIST_H


class List {
public:
    List() {
        head = nullptr;
        name = list_count + 'A';
        list_count++;
    }

    List(const List &list) {
        if (list.head == nullptr) {
            this->head = nullptr;
        }
        else {
            Elem *this_elem = new Elem;
            Elem *list_elem = list.head;
            this->head = this_elem;
            this_elem->count = list.head->count;
```

```
            while (list_elem->next != nullptr) {
                list_elem = list_elem->next;
                this_elem = this_elem->next = new Elem;
                this_elem->count = list_elem->count;
            }
        }
    }

    ~List() {
        Elem* elem = this->head;
        while (elem != nullptr) {
            elem = elem->next;
            delete this->head;
            this->head = elem;
        }
    }

    List& operator = (const List &list);

    List& operator = (char* arr);

    List& conj(List &list_1, List &list_2, List &list_3, List &list_4);

    void clear();

    void addElem (char count);

    void print();
private:
    bool elemInList(char elem);

    char name;

    static unsigned short list_count;

    struct Elem {
        char count;
        Elem* next = nullptr;
    };
    Elem* head;
};


#endif //INC_2_LIST_H
```

---

main.cpp

```
#include "Array.h"
#include "Bit.h"
#include "Bit_Arr.h"
#include "Tester.h"
#include "List.h"

unsigned short Array::arr_count = 0,
        Bit::bit_count = 0,
        Bit_Arr::bit_arr_count = 0,
        List::list_count = 0;

int main() {
    setlocale(0, " ");
    srand((unsigned)time(nullptr));
    Array arr_1, arr_2, arr_3, arr_4;
    int choise;
    choise = Helper::Choise(Helper::showMenu());
    switch (choise) {
        case 1:
        case 3:{
            arr_1.init();
            arr_2.init();
            arr_3.init();
            arr_4.init();
            break;
        }
        case 2: {
            arr_1.randInit();
            arr_2.randInit();
```

```
            arr_3.randInit();
            arr_4.randInit();
            break;
        }
        case 0: {
            cout << "Выход из программы..\n";
            break;
        }
        default: {}
    }
    if (choise == 3) {
        Tester::timeTest(arr_1, arr_2, arr_3, arr_4);
    }
    else if (choise){
        Tester::run(arr_1, arr_2, arr_3, arr_4);
    }
    cout << "Работа программы завершена.";
    return 0;
}
```

Tester.cpp

```
#include "Tester.h"
#include "Bit_Arr.h"
#include "Bit.h"
#include "List.h"

void Tester::run(Array &arr_1, Array &arr_2, Array &arr_3, Array &arr_4) {
    std::cout << "\nИсходные множества:\n";
    arr_1.print();
    arr_2.print();
    arr_3.print();
    arr_4.print();
    Array arr_ans;
    arr_ans.conj(arr_1, arr_2, arr_3, arr_4);
    cout << "\nРезультат работы:\n";
    arr_ans.print();
    List list_1, list_2, list_3, list_4, list_ans;
    list_1 = arr_1.getElements();
    list_2 = arr_2.getElements();
    list_3 = arr_3.getElements();
    list_4 = arr_4.getElements();
    list_ans.conj(list_1, list_2, list_3, list_4);
    list_ans.print();
    Bit_Arr bit_arr_1, bit_arr_2, bit_arr_3, bit_arr_4, bit_arr_ans;
    bit_arr_1 = arr_1.getElements();
    bit_arr_2 = arr_2.getElements();
    bit_arr_3 = arr_3.getElements();
    bit_arr_4 = arr_4.getElements();
    bit_arr_ans.conj(bit_arr_1, bit_arr_2, bit_arr_3, bit_arr_4);
    bit_arr_ans.print();
    Bit bit_1, bit_2, bit_3, bit_4, bit_ans;
    bit_1 = arr_1.getElements();
    bit_2 = arr_2.getElements();
    bit_3 = arr_3.getElements();
    bit_4 = arr_4.getElements();
    bit_ans.conj(bit_1, bit_2, bit_3, bit_4);
    bit_ans.print();
}

void Tester::timeTest(Array arr_1, Array arr_2, Array arr_3, Array arr_4) {
    std::cout << "\nИсходные множества:\n";
    arr_1.print();
    arr_2.print();
    arr_3.print();
    arr_4.print();
    Array arr_ans;
    List list_1, list_2, list_3, list_4, list_ans;
    list_1 = arr_1.getElements();
    list_2 = arr_2.getElements();
    list_3 = arr_3.getElements();
    list_4 = arr_4.getElements();
    Bit_Arr bit_arr_1, bit_arr_2, bit_arr_3, bit_arr_4, bit_arr_ans;
    bit_arr_1 = arr_1.getElements();
    bit_arr_2 = arr_2.getElements();
    bit_arr_3 = arr_3.getElements();
```

```
    bit_arr_4 = arr_4.getElements();
    Bit bit_1, bit_2, bit_3, bit_4, bit_ans;
    bit_1 = arr_1.getElements();
    bit_2 = arr_2.getElements();
    bit_3 = arr_3.getElements();
    bit_4 = arr_4.getElements();
    unsigned long arr_start_time, arr_finish_time, arr_time = 0,
    list_start_time, list_finish_time, list_time = 0,
    bit_arr_start_time, bit_arr_finish_time, bit_arr_time = 0,
    bit_start_time, bit_finish_time, bit_time = 0;
    unsigned int cicle_times = 1000000;
    for (unsigned int i = 0; i < cicle_times; i++) {
        arr_start_time = (unsigned long)clock();
        arr_ans.conj(arr_1, arr_2, arr_3, arr_4);
        arr_finish_time = (unsigned long)clock();
        arr_time += (arr_finish_time - arr_start_time);

        list_start_time = (unsigned long)clock();
        list_ans.conj(list_1, list_2, list_3, list_4);
        list_finish_time = (unsigned long)clock();
        list_time += (list_finish_time - list_start_time);

        bit_arr_start_time = (unsigned long)clock();
        bit_arr_ans.conj(bit_arr_1, bit_arr_2, bit_arr_3, bit_arr_4);
        bit_arr_finish_time = (unsigned long)clock();
        bit_arr_time += (bit_arr_finish_time - bit_arr_start_time);

        bit_start_time = (unsigned long)clock();
        bit_ans.conj(bit_1, bit_2, bit_3, bit_4);
        bit_finish_time = (unsigned long)clock();
        bit_time += (bit_finish_time - bit_start_time);
    }
    cout << "\nРезультат работы (для " << cicle_times << " повторений):\n";
    arr_ans.print();
    cout << arr_time * 1000 / CLOCKS_PER_SEC << "ms\n";
    list_ans.print();
    cout << list_time * 1000 / CLOCKS_PER_SEC << "ms\n";
    bit_arr_ans.print();
    cout << bit_arr_time * 1000 / CLOCKS_PER_SEC << "ms\n";
    bit_ans.print();
    cout << bit_time * 1000 / CLOCKS_PER_SEC << "ms\n";
}
```

Tester.h

```
#ifndef INC_2_TESTER_H
#define INC_2_TESTER_H


#include "Array.h"

class Tester {
public:
    static void run(Array &arr_1, Array &arr_2, Array &arr_3, Array &arr_4);
    static void timeTest(Array arr_1, Array arr_2, Array arr_3, Array arr_4);
};


#endif //INC_2_TESTER_H
```