Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ» имени В.И. Ульянова (Ленина)

Факультет компьютерных технологий и информатики
Кафедра вычислительной техники

Зачётная работа №2
по дисциплине «Алгоритмы и структуры данных»
«Множества как объект»

Вариант№21

Выполнил: студент группы 5005
Зинатуллин Азат Салаватович

Проверил: старший преподаватель
Колинько Павел Георгиевич

Цель работы

Научиться представлять множества разными способами, используя механизм классов, подсчитать время выполнения для каждого способа, обеспечить отслеживание вызовов функций-членов при вычислении пятого множества.

1. Задание

Необходимо найти множество, содержащее цифры, имеющиеся в A или B или являющиеся общими для C и D.
Формализация задания: E=A|B|(C&D)
Необходимо решить задание четырьмя способами представления множеств:
— массив
— список
—машинное слово
— массив битов
Преобразовать программу, созданную в 1 лабораторной работе так, чтобы множества были объектами некоторого класса, а операции над ними – функциями-членами этого класса. Добиться, чтобы функция main() во всех вариантах была одинакова, менялось только определение классов. Для одного из вариантов обеспечить отслеживания вызовов функций-членов при вычислении пятого множества по четырем исходным.

2. Контрольные тесты

```
A = [1256]
B = [01678]
C = [59]
D = [02679]
E = [12560789] Middle power=4 Time=218089 / 100000
```

Тест 1. Представление исходных данных в виде массива символов.

```
A = 98430
B = 810
C = 873210
D = 98710

E = [1034897] Middle power=5 Time=1358415 / 100000
```

Тест 2. Представление исходных данных в виде списка.

```
A = [01234689]
B = [0145789]
C = [0345689]
D = [01]
E = [0123456789] Middle power=4 Time=12952 / 100000
```

Тест 3. Представление исходных данных в виде машинного слова.

```
A = [--2345--8-]
B = [---3-5-7--]
C = [--2-4---89]
D = [0-2-4-6-8-]
E = [--2345-78-] Middle power=3 Time=54572 / 100000
```

Тест 4. Представление исходных данных в виде массива битов.

Время обработки множеств

Тип представления	Массивы символов	Список	Машинное слово	Вектор битов
Время обработки	4361 мкс	27168 мкс	259 мкс	1091 мкс

3. Результат эксперимента с отслеживанием вызова функций-членов.

```
Конструктор множества
A = [0289]
Конструктор множества
B = [0578]
Конструктор множества
C = [013478]
Конструктор множества
D = [1368]
Конструктор по умолчанию
Операция объединения
Конструктор по умолчанию
Операция пересечения
Конструктор по умолчанию
Операция объединения
Конструктор по умолчанию
Оператор присваивания
Конструктор копии
Деструктор
Деструктор
Деструктор
Деструктор
Функция вывода множества на экран
E = [02895713] Middle power=5 Time=112 / 100000
Деструктор
Деструктор
Деструктор
Деструктор
Деструктор
```

Тест. Отслеживание вызова функций-членов для представления множества массивом символов

4. Выводы

Эта задача является той же задачей, которая была решена в работе №1, значит данные лучше всего представлять теми же способами, которые давали наилучший результат в работе №1.

Использование классов упростило работу с множествами: теперь все операции определены внутри класса и тому, кто использует функции не нужно думать о внутреннем представлении данных. Также это упростило синтаксис основной функции. С другой стороны, производятся дополнительные действия по обработке лишних, в рамках этой задачи, данных и механизмов, что влечет за собой увеличение времени исполнения алгоритма.

5. Список использованных источников

1. Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Ч. 1. Вып. 1701 (для заочников) / сост.: П. Г. Колинько. — СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2017. — 64 с.

2. Курс видео уроков по C++. https://www.youtube.com/playlist?list=PLbmlzoDQrXVFC13GjpPrJxl6mzTiX65gs

3. Стефан Р. Дэвис. C++ для чайников, 4-е издание.

4. Использовал, как справочник по некоторым функциям. http://cppstudio.com/cat/274/

6. Приложение

main.cpp - Файл содержит главную функцию, создание объектов множеств, вычесление конечного множества вызовом соответсвующих функций из классов.
array.h - Способ представления массивом символов.
machineword.h - Способ представления машинным словом.
arraybit.h - Способ представления массивом битов.
spisok.h - Способ представления списком.

```
// Задание выполнил студент группы 5005 Зинатуллин Азат.
```

```cpp
// Вариант №21. Универсум десятичные числа.
// Множество, содержащее цифры, имеющиеся в A или B или являющиеся общими для C и D.
// Формализация задания: E=A|B|(C&D)

#include <iostream>
#include <time.h>
#include "array.h"

using namespace std;

 // определение статического члена класса
const long q0 = 100000; // количество повторений цикла времени

int main()
{   //    srand(static_cast<unsigned long>(5));
        srand(time(nullptr));
        Set A('A'), B('B'), C('C'), D('D'), E;

        clock_t begin = clock();

        for(long q =0; q < q0; q++)
        {
            E = (A|B|(C&D));
        }

        clock_t end = clock();
        E.Show();

        cout << " Middle power=" <<
        (A.power() + B.power() + C.power() + D.power() + E.power())/5 <<
        " Time=" << (end-begin) << " / " << q0 << endl;

    return 0;
}
```

```cpp
//  array.h (способ представления массивом символов)
#pragma once
#ifndef Mnogestva_2_array_h
#define Mnogestva_2_array_h

#include <cstring>
#include <iostream>

using namespace std;


class Set
{
    private:            // Закрытая часть класса – данные

        static int N;   // Мощность универсума
        int n;          // Мощность множества
        char S, *A;     // Символ тег и память для множества

    public:             // Открытая часть – функции для работы с множеством

        Set operator | (const Set&) const;  // Объединение
        Set operator & (const Set&) const;  // Пересечние
        void Show();                        // Вывод множества на экран
        int power() {return n;}             // Получение мощности множества
        Set(char);                          // Конструктор множества
        Set();                              // Конструктор по умолчанию
        Set(const Set&);                    // Конструктор копии
        Set operator = (const Set&);        // Оператор присваивания
        ~Set(){cout << "Деструктор\n"; delete []A;}                 // деструктор
};

    int Set:: N = 10;

    Set::Set(): n(0), S('0'), A(new char[N+1])
    {
        cout << "Конструктор по умолчанию\n";
        A[0]= 0;
    }

    Set::Set(char s): n(0), S(s), A(new char[N+1])
    {
        cout << "Конструктор множества";
        for(int i = 0; i < N; i++)
            if(rand()%2)
                A[n++] = i+'0';
                A[n] = 0;
        cout << '\n' << S << " = [" << A << "]" << endl;
    }
```

```
    Set Set::operator & (const Set& B) const
    {
        cout << "Оператор пересечения" << endl;
        Set C;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < B.n; j++)
                if (A[i] == B.A[j]) C.A[C.n++] = A[i];
        }
        C.A[C.n] = 0;
        return C;
    }

    Set Set:: operator | (const Set &B) const
    {
        cout << "Оператор объединения" << endl;
        Set C;
        memcpy(C.A, A, n);
        C.n = n;
        for(int i = 0; i <B.n; i++)
        {
            bool f = true;
            for(int j = 0; j < n; j++)
                if (B.A[i] == A[j])
                    f = false;
                if (f) C.A[C.n++] = B.A[i];
        }
        C.A[C.n] = 0;
        return C;
    }

    Set::Set(const Set & B) : S(B.S), n(B.n), A(new char[N+1])
    {
        cout << "Конструктор копии" << endl;
        memcpy(A, B.A, N+1);
    }

    Set  Set::operator = (const Set& B)
    {
        cout << "Оператор присваивания" << endl;
        if (this != &B)
        {
            n = B.n;
            memcpy(A,B.A,N+1);
            S = 'E';
        }
    return *this;
    }

    void Set::Show()
    {
        cout << "Функция вывода множества на экран";
        cout << '\n' << S << " = [" << A << "]";
    }

#endif
```

```
//  machineword.h (способ представления машинным словом)
#ifndef Mnogestva_2_machineword_h
#define Mnogestva_2_machineword_h

#include <cstring>
#include <iostream>

using namespace std;


class Set
{
private:            // Закрытая часть класса – данные

    static int N;   // Мощность универсума
    int n;          // Мощность множества
    int mw;         // Память для машинных слов
    char S;         // Символ тег

public:             // Открытая часть – функции для работы с множеством

    Set operator | (const Set&) const;  // Объединение
    Set operator & (const Set&) const;  // Пересечние
    void Show();                        // Вывод множества на экран
    int power() {return n;}             // Получение мощности множества
    Set(char);                          // Конструктор множества
```

```
    Set();                                  // Конструктор по умолчанию
    Set(const Set&);                        // Конструктор копии
    Set operator = (const Set&);            // Оператор присваивания

};

int Set:: N = 10;

    Set::Set(): n(0), S('0')
        {mw= 0;}

    Set::Set(char s): n(0), S(s)
    {
        mw = rand();

        cout << S << " = [";
        for(int i = 0; i < N; i++)
        {
            if((mw >> i) & 1){
                cout << char(i + '0');
                n++;}
        }
        cout << "]" << endl;

    }

    Set Set:: operator | (const Set& B) const
    {
        Set C;
        C.mw = B.mw | mw;
        return C;
    }

    Set Set:: operator & (const Set& B) const
    {
        Set C;
        C.mw = B.mw & mw;
        return C;
    }

    Set::Set(const Set&B)
    {
        S=B.S;
        n=B.n;
        mw=B.mw;
    }

    Set Set :: operator = (const Set&B)
    {
        if(this != &B)
        {
            n=B.n;
            mw = B.mw;
            S = 'E';
        }
        return *this;
    }
    void Set :: Show()
    {
        cout << S << " = [";
        for(int i = 0; i < N; i++)
              if((mw >> i) & 1)
                  cout << char(i + '0');
        cout << "]";
    }

#endif



//  arraybit.h (способ представления массивом битов)
#ifndef Mnogestva_2_arraybit_h
#define Mnogestva_2_arraybit_h

#include <cstring>
#include <iostream>


using namespace std;


class Set
{
private:            // Закрытая часть класса – данные

    static int N;   // Мощность универсума
    int n;          // Мощность множества
    char bA[10];    // Память под массив битов
```

```
    char S;         // Символ тег

public:             // Открытая часть – функции для работы с множеством

    Set operator | (const Set&) const;  // Объединение
    Set operator & (const Set&) const;  // Пересечние
    void Show();                        // Вывод множества на экран
    int power() {return n;}             // Получение мощности множества
    Set(char);                          // Конструктор множества
    Set();                              // Конструктор по умолчанию
    Set(const Set&);                    // Конструктор копии
    Set operator = (const Set&);        // Оператор присваивания


};

int Set:: N = 10;

    Set::Set(): n(0), S('0')
    {
        bA[0]= 0;
    }

    Set::Set(char s): n(0), S(s)
    {

        cout << S << " = [";
        for(int i = 0; i < N; i++)
        {

            bA[i]=rand()%2;

            if(bA[i]) {cout << char(i + '0');
                n++;
            }
            else cout << "-";
            }
        cout << "]" << endl;
    }

    Set Set::operator & (const Set& B) const
    {
        Set C;
        for (int i = 0; i<N; ++i)
            C.bA[i] = B.bA[i]&bA[i];
        return C;
    }

    Set Set:: operator | (const Set& B) const
    {
        Set C;
        for (int i = 0; i<N; ++i)
            C.bA[i] = B.bA[i]|bA[i];
        return C;
    }

    Set::Set(const Set&B) :  S(B.S), n(B.n)
    {
        memcpy(bA, B.bA, N);
    }

    Set Set :: operator = (const Set& B)
    {
        if(this != &B)
        {
            n=B.n;
            memcpy(bA,B.bA,N);
            S = 'E';
        }
        return *this;
    }

    void Set :: Show()
    {
        cout << S << " = [";
        for(int i = 0; i < N; ++i)
            if(bA[i]) cout << char(i + '0');
            else cout << "-";
        cout << "]";
    }

#endif



//  spisok.h (способ представления списком)
#ifndef Mnogestva_2_spisok_h
```

```
#define Mnogestva_2_spisok_h

#include <cstring>
#include <iostream>

using namespace std;

struct EL {
    char d;
    EL * next;
    EL(char d, EL * next = nullptr) : d(d), next(next) {}
    ~EL() {if (next) delete next;}
};

class Set
{
private:            // Закрытая часть класса – данные

    static int N;   // Мощность универсума
    int n;          // Мощность множества
    char S;         // Символ тег и память для множества
    EL *La;         // Указатель на список


public:             // Открытая часть – функции для работы с множеством

    Set operator | (const Set&) const;  // Объединение
    Set operator & (const Set&) const;  // Пересечние
    void Show();                        // Вывод множества на экран
    int power() {return n;}             // Получение мощности множества
    Set(char);                          // Конструктор множества
    Set();                              // Конструктор по умолчанию
    Set(const Set&);                    // Конструктор копии
    Set operator = (const Set&);        // Оператор присваивания
    ~Set(){delete La;}                  // деструктор
};

int Set:: N = 10;


    Set::Set(): n(0), S('0')
    {
        La = nullptr;
    }


    Set::Set(char s): n(0), S(s)
    {
        La = nullptr;
        for(int i=0; i<N; ++i)
        {
            char A;
            if(rand()%2)
            {
                A=i+'0';
                EL* e = new EL(A, La);
                La = e;
                A=0;
                n++;
            }
        }
        cout << S << " = ";
        for (EL *v = La; v; v = v->next)
            cout << v->d;
        cout << endl;
    }

    Set Set::operator & (const Set& B) const
    {
        Set C;
        for (EL* u = La; u; u = u->next)
        {
            bool f=true;
            for (EL* v = B.La; v&&f; v = v->next)
                if (u->d == v->d)
                {
                    EL* e = new EL(u->d, C.La);
                    C.La = e;
                    f=false;
                }
        }
        return C;
    }

    Set Set:: operator | (const Set &B) const
    {
        Set C;
        for (EL* u = La; u; u = u->next)
```

```
        {
            EL* e = new EL(u->d, C.La);
            C.La = e; ++C.n;
        }
        for (EL* u = B.La; u; u = u->next)
        {
            bool f = true;
            for (EL* v = C.La; v; v = v->next)
                if (u->d == v->d) f=false;
            if(f)
            {
                EL* e = new EL(u->d, C.La);
                C.La = e;
            }
        }

        return C;
    }

    Set::Set(const Set & B) : S(B.S), n(B.n)
    {
        La = nullptr;
        for (EL* u = B.La; u; u = u->next)
        {
            EL* e = new EL(u->d, La);
            La = e;
        }
    }

    Set Set::operator = (const Set& B)
    {
        if (this != &B)
        {
            n=B.n;
            La = nullptr;
            for (EL* u = B.La; u; u = u->next)
            {
                EL* e = new EL(u->d, La);
                La = e;
            }
            S = 'E';

        }
        return *this;
    }

    void Set::Show()
    {
        cout << '\n' << S << " = [";
        for (EL *v = La; v; v = v->next)
            cout << v->d ;
            cout << "]";
    }


#endif
```