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 вычислительной техники

**Отчёт по заданию № 2**

**на тему: “Множество как объект”**

**по дисциплине**

**“Алгоритмы и структуры данных”**

**Вариант 26**

|  |  |
| --- | --- |
| Выполнили студенты гр.: | Яловега Н.В. |
| Проверил: | Колинько П.Г. |

Санкт-Петербург, 2020 г.

Оглавление

[Введение 3](#_Toc53603017)

[1. Задание 3](#_Toc53603018)

[2. Формализация задания 3](#_Toc53603019)

[3. Контрольные примеры 4](#_Toc53603020)

[4. Результаты эксперимента 5](#_Toc53603021)

[Вывод 8](#_Toc53603022)

[Список используемых источников 9](#_Toc53603023)

[Приложение 1 (Исходный текст программы) 10](#_Toc53603024)

## Введение

Целью задания является исследование эффекта от использования классов.

## 1. Задание

Множество содержащие все буквы из A, не являющиеся общими для B, C и D.

Универсум — прописные латинские буквы.

## 2. Формализация задания

Условие задачи можно записать в виде формулы:

## 3. Контрольные примеры

Таблица. 1. Контрольные примеры

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Исходные множества** | | | | **Результат** |
| **A** | **B** | **C** | **D** | **R** |
| 1 | AHMNPXYZ | AFGIKLMOUVWY | ABCDFIJSTZ | AEHMPQSUW | HMNPXYZ |
| 2 | FHJKORSVWX | ACDGOQRX | BCEFHJKMNOPQVY | BCEFGHIJOPVY | FHKRSVWX |
| 3 | ADEFGHIJNOPRSUVXZ | AFGHIJMPQRUWXYZ | FJKLMNQTVWXZ | BCFGJKLMORTWXZ | ADEGHINOPRSUV |
| 4 | ACDFJKMPSV | BDEGHILMOPQSTUW | ADEFIJKLMNOPRTUVXY | ADFGMNOPQTUXYZ | ACFJKSV |
| 5 | ABC | ABC | ABC | ABC | Пустое множество |

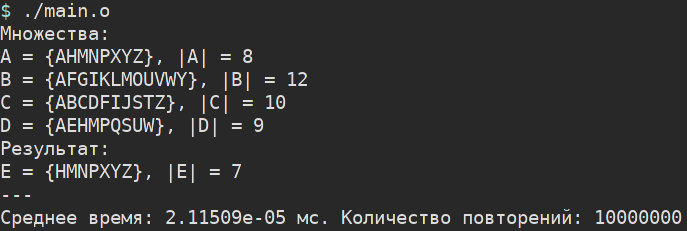
## 4. **Результаты измерения времени обработки**

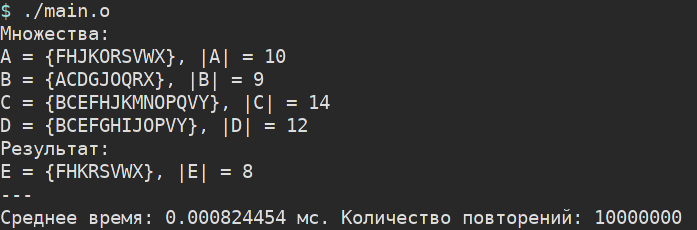
Результаты измерения времени обработкипредставлены в таблице 2*.*

Таблица. 2. Результаты измерения времени обработки

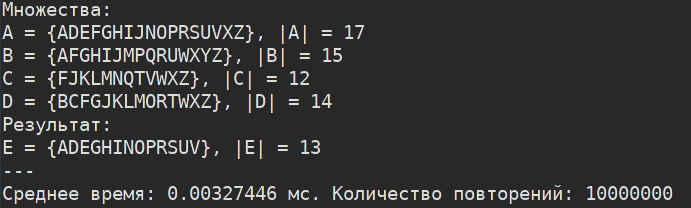
|  |  |  |  |
| --- | --- | --- | --- |
| **Способ представления** | **Время (мс)** | **Время в прошлой программе (мс)** | **Количество повторов цикла** |
| Массив символов | 0.00208 - 0.00327 | 0.0001642 - 0.0022 | 10000000 |
| Список | 0.003456 -  0.00508 | 0.0003535 -  0.00322 |
| Массив битов | 0.00072 -  0.00082 | 0.000087 -  0.000108 |
| Машинное слово | 0.000019 -  0.000021 | 0.0000016 -  0.000002 |

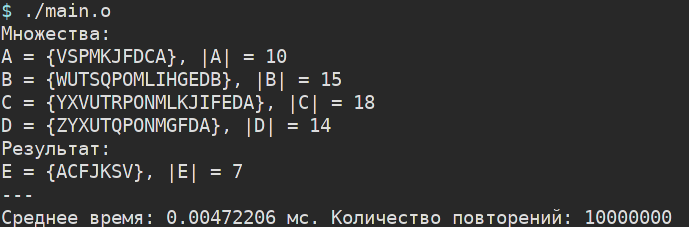
На скриншотах ниже приведены результаты выполнения программы со структурами, в том числе с отслеживанием вызова функций:

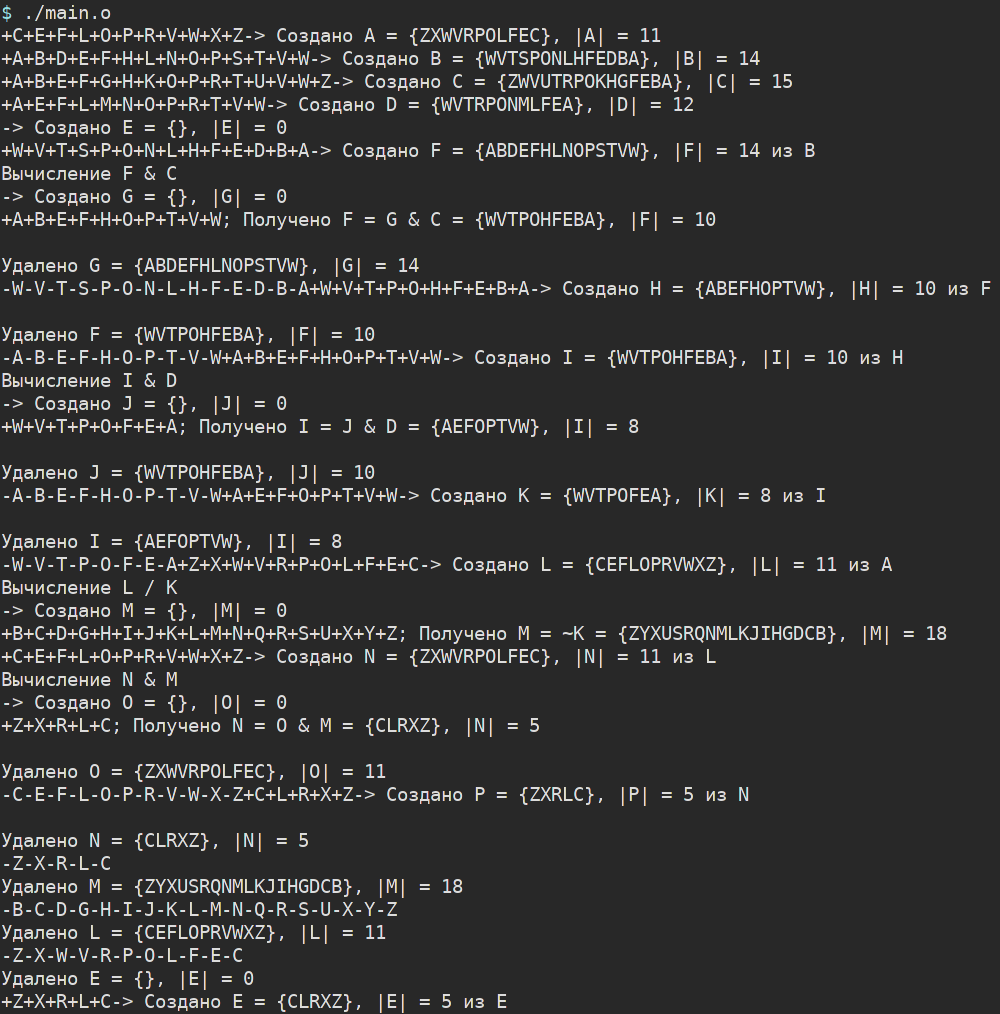
Рисунок 1: Результат работы программы с машинным словом

Рисунок 2: Результат работы программы с битовым массивом

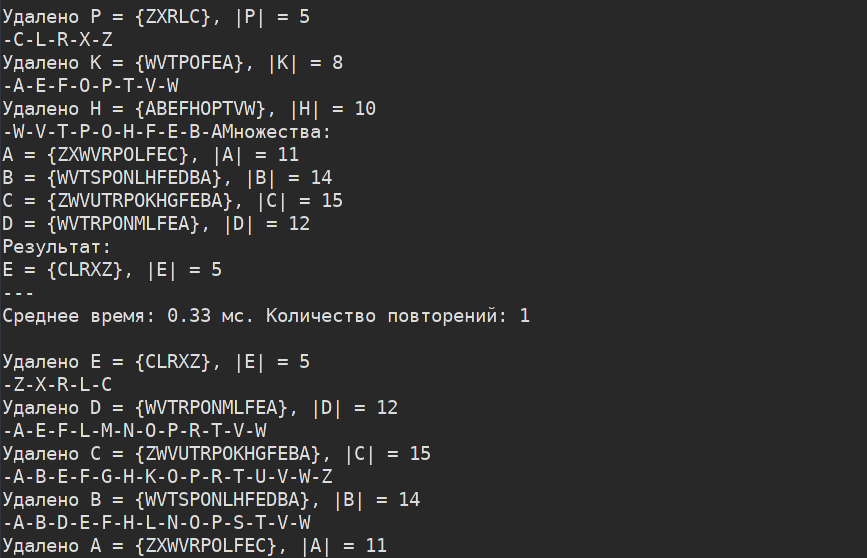
*.*

Рисунок 3: Результат работы программы с массивом символов

Рисунок 4: Результат работы программы со списком

Рисунок 5: Результат работы программы со списком с отслеживанием вызовов

Как видно из таблицы 2, при реализации множеств при помощи классов, время выполнения программы немного увеличивается.

Рисунок 6: Результат работы программы со списком с отслеживанием вызовов

## Вывод

От любого метода программирования мы ждем, что он поможет нам в решении наших проблем. Но одной из самых значительных проблем в программировании является сложность. Чем больше и сложнее программа, тем важнее становится разбить ее на небольшие, четко очерченные части. Чтобы побороть сложность, мы должны абстрагироваться от мелких деталей. В этом смысле классы представляют собой весьма удобный инструмент.

* Классы позволяют проводить конструирование из полезных компонент, обладающих простыми инструментами, что дает возможность абстрагироваться от деталей реализации.
* Данные и операции вместе образуют определенную сущность и они не «размазываются» по всей программе, как это нередко бывает в случае процедурного программирования.
* Локализация кода и данных улучшает наглядность и удобство сопровождения программного обеспечения.
* Инкапсуляция информации защищает наиболее критичные данные от несанкционированного доступа.

В ходе выполнения опытов с программой удалось выяснить, что при работе с классами время выполнения программы немного увеличивается.

## Список используемых источников

1. Алгоритмы и структуры данных – Лекция от 25.09.2015.
2. Алгоритмы и структуры данных – Лекция от 02.10.2015.
3. Колинько П.Г. Пользовательские структуры данных / Методические указания по дисциплине «Алгоритмы и структуры данных» - Санкт-Петербург: СПбГЭТУ «ЛЭТИ», 2020.

## Приложение 1 (Исходный текст программы)

**1) main.cpp**

#include <iostream>

#include <ctime>

using namespace std;

// #include "char\_array.h"

#include "list.h"

// #include "bool.h"

// #include "word.h"

int Set::U = 26;

int Set::count = 0;

const long ROLLS = 10000000;

int main()

{

List::clear();

srand(time(NULL));

Set A('A'), B('B'), C('C'), D('D'), E;

List::mark();

clock\_t start = clock();

for (long rolls = 0; rolls < ROLLS; rolls++)

{

List::release();

E = A/(B&C&D);

}

clock\_t end = clock();

cout << "Множества: \n";

A.print();

B.print();

C.print();

D.print();

cout << "Результат: \n";

E.print();

cout << "--- \n";

cout << "Среднее время: " << 1000\*((((double)(end - start))/CLOCKS\_PER\_SEC)/ROLLS) <<

" мс. Количество повторений: " << ROLLS << endl;

return 0;

}

**2) bool.h**

#ifndef BOOL\_H

#define BOOL\_H

class Set

{

private:

static int U, count; // мощность юниверсума и количество множеств

char name; // название множества

bool \*S; // память для множества

public:

Set(char); // конструктор класса

Set(); // конструктор по умолчанию

Set(const Set&); // конструтор копирования

~Set(){delete [] S;} // деструктор класса

// операции над множествами

Set& operator = (const Set&);

Set operator & (const Set&) const;

Set operator | (const Set&) const;

Set operator / (const Set&) const;

Set operator ~ () const;

// интерфейс

inline char get\_name() {return name;}

int power();

void print();

};

#endif

**3) bool.cpp**

#include "bool.h"

#include <iostream>

// конструторы

Set::Set(): name('A' + count++), S(new bool[U])

{

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

S[i] = 0;

}

Set::Set(char tag): name(tag), S(new bool[U])

{

count++;

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

S[i] = rand()%2;

}

Set::Set(const Set& other): name('A' + count++), S(new bool[U])

{

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

S[i] = other.S[i];

}

// операции

Set & Set::operator = (const Set& other)

{

if (this != &other)

{

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

S[i] = other.S[i];

}

return \*this;

}

Set Set::operator & (const Set& other) const

{

Set C;

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

C.S[i] = S[i] && other.S[i];

return C;

}

Set Set::operator | (const Set& other) const

{

Set C;

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

C.S[i] = S[i] || other.S[i];

return C;

}

Set Set::operator / (const Set& other) const

{

Set C;

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

C.S[i] = S[i] && !other.S[i];

return C;

}

Set Set::operator ~ () const

{

Set C;

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

C.S[i] = !S[i];

return C;

}

// интерфейс

int Set::power()

{

int counter = 0;

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

if (S[i])

counter++;

return counter;

}

void Set::print()

{

char\* result = new char[U+1];

int i,j;

for(i = 0, j = 0; i < U; ++i)

if(S[i])

result[j++] = i + 'A';

result[j] = '\0';

std::cout << name << " = {" << result << "}, |" << name << "| = " << this->power() << std::endl;

delete [] result;

}

**4) char\_array.h**

#ifndef CHAR\_H

#define CHAR\_H

class Set

{

private:

static int U, count; // мощность юниверсума и количество множеств

char name; // название множества

int pow; // мощность множества

char \*S; // память для множества

public:

Set(char); // конструктор класса

Set(); // конструктор по умолчанию

Set(const Set&); // конструтор копирования

~Set(){delete [] S;} // деструктор класса

// операции над множествами

Set& operator= (const Set&);

Set& operator &= (const Set&);

Set operator & (const Set&) const;

Set& operator |= (const Set&);

Set operator | (const Set&) const;

Set operator / (const Set&) const;

Set operator ~ () const;

// интерфейс

inline char get\_name() {return name;}

inline int power() {return pow;}

void print();

};

#endif

**5) char\_array.cpp**

#include "char\_array.h"

#include <iostream>

// конструторы

Set::Set(): name('A' + count++), pow(0), S(new char[U+1])

{

S[0] = '\0';

}

Set::Set(char tag): name(tag), pow(0), S(new char[U+1])

{

count++;

auto w = rand();

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

if((w >> i) & 1)

S[pow++] = i + 'A';

S[pow] = '\0';

}

Set::Set(const Set& other): name('A' + count++), pow(other.pow), S(new char[U+1])

{

for (int i = 0; other.S[i]; ++i)

S[i] = other.S[i];

S[pow] = '\0';

}

// операции

Set & Set::operator = (const Set& other)

{

if (this != &other)

{

pow = other.pow;

for(int i = 0; other.S[i]; i++)

{

S[i] = other.S[i];

}

S[pow] = '\0';

}

return \*this;

}

Set & Set::operator &= (const Set& other)

{

Set C(\*this);

pow = 0;

for (int i = 0; i < C.pow; ++i)

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

if(C.S[i] == other.S[j])

S[pow++] = C.S[i];

S[pow] = '\0';

return \*this;

}

Set Set::operator & (const Set& other) const

{

Set C(\*this);

return (C &= other);

}

Set & Set::operator |= (const Set& other)

{

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

{

bool f = true;

for(int j = 0; j < pow; j++)

if(other.S[i] == S[j])

f = false;

if (f)

S[pow++] = other.S[i];

}

S[pow] = '\0';

return \*this;

}

Set Set::operator | (const Set& other) const

{

Set C(\*this);

return (C |= other);

}

Set Set::operator / (const Set& other) const

{

Set C(\*this);

return C & (~other);

}

Set Set::operator ~ () const

{

Set result;

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

{

bool f = true;

for (int j = 0; j < pow; ++j)

if (c == S[j])

{

f = false;

break;

}

if (f)

result.S[result.pow++] = c;

}

result.S[result.pow] = 0;

return result;

}

// интерфейс

void Set::print()

{

std::cout << this->get\_name() << " = {" << S << "}, |" << this->get\_name() << "| = " << this->power() << std::endl;

}

**6) list.h**

#ifndef LIST\_H

#define LIST\_H

#include <iostream>

class List{

char e;

List \*next; // указатель на следующий элемент списка

static const int maxmup = 200; // максимальное количество элементов

static List mem[maxmup]; // память для элементов списков

static int mup, mup0;

public:

List(): e('!'), next(nullptr){} // конструктор по умолчанию

List(char e, List \*n = nullptr): e(e), next(n) { /\* std::cout << "+" << e; \*/ }

~List(){

if(this)

{//Прямой вызов деструктора требует такой проверки!!!

delete next;

// std::cout << "-" << e;

e = '-';

}

else std::cout << "<Empty!>";

}

static void\* operator new(size\_t) { //Перегрузка new и delete - только функциями static!!!

return (mup < maxmup? &mem[mup++] : nullptr); }

static void operator delete(void \*, size\_t) { }

static void mark(){ mup0 = mup;}

static void release() { mup = mup0; }

static void clear(){ mup = 0;}

friend class Set;

friend std::ostream & operator << ( std::ostream & o, List & S); //Перегрузка << для вывода

};

class Set

{

private:

static int U, count; // мощность юниверсума и количество множеств

char name; // название множества

int pow; // мощность множества

List\* S; // память для множества

public:

Set(); // конструктор по умолчанию

Set(char); // конструктор класса

Set(const Set&); // конструтор копирования

~Set()

{

std::cout << "\nУдалено " << name << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

S->List::~List(); //Здесь нужен явный вызов деструктора

}

// операции над множествами

Set& operator = (const Set&);

Set & operator |= (const Set&);

Set & operator &= (const Set&);

Set operator & (const Set&) const;

Set operator | (const Set&) const;

Set operator / (const Set&) const;

Set operator ~ () const;

void swap(Set & other) {std::swap(pow, other.pow); std::swap(S, other.S);}

// интерфейс

char get\_name() {return name;}

int power();

void print();

};

#endif

**7) list.cpp**

#include "list.h"

std::ostream & operator << (std::ostream & o, List & S)

{

for (auto p = &S; p; p = p->next)

o << p->e;

return o;

}

List List::mem[List::maxmup]; //"Свободная память"

int List::mup = 0, List::mup0 = 0; //Рабочий и резервный указатели на свободное место

Set::Set() : name('A' + count++), pow(0), S(nullptr)

{

// std::cout << "-> Создано " << name << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

}

Set::Set(char n) : name(n), pow(0), S(nullptr)

{

count++;

auto w = rand();

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

if ((w >> i) & 1)

S = new List(i + 'A', S), ++pow;

// std::cout << "-> Создано " << name << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

}

Set::Set(const Set & other) : name('A'+count++), pow(other.pow), S(nullptr)

{

for(auto p = other.S; p; p = p->next)

S = new List(p->e, S);

// std::cout << "-> Создано " << name << " = {" << \*S << "}, |" << name << "| = " << pow

// << " из " << other.name << std::endl;

}

Set & Set::operator &= (const Set& B)

{

Set C;

for (auto i = S; i; i = i->next)

{

for (auto j = B.S; j; j = j->next)

if (i->e == j->e)

C.S = new List(i->e, C.S), ++C.pow;

}

swap(C);

// std::cout << "; Получено " << name << " = " << C.name << " & " << B.name

// << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

return \*this;

}

Set Set::operator & (const Set& B) const

{

Set C(\*this);

// std::cout << "Вычисление " << C.name << " & " << B.name << std::endl;

return C &= B;

}

Set & Set::operator |= (const Set & B)

{

Set C(\*this);

for (auto i = B.S; i; i = i->next)

{

bool f = true;

for (auto j = S; f && j; j = j->next)

f = f && (i->e != j->e);

if (f)

C.S = new List(i->e, C.S), ++C.pow;

}

swap(C);

// std::cout << "; Получено " << name << " = " << C.name << " | " << B.name

// << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

return \*this;

}

Set Set::operator | (const Set& B) const

{

Set C(\*this);

// std::cout << "Вычисление " << C.name << " | " << B.name << std::endl;

return C |= B;

}

Set Set::operator / (const Set& B) const

{

Set C(\*this);

// std::cout << "Вычисление " << C.name << " / " << B.name << std::endl;

return C & ~B;

}

Set Set::operator ~ () const

{

Set C;

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

{

bool f = true;

for (auto j = S; j && f; j = j->next)

if(c == j->e)

f = false;

if(f)

C.S = new List(c, C.S) , ++C.pow;

}

// std::cout << "; Получено " << C.name << " = ~" << name

// << " = {" << \*C.S << "}, |" << C.name << "| = " << C.pow << std::endl;

return C;

}

Set& Set::operator = (const Set & B)

{

if (this != &B)

{

// std::cout << "\nУдалено " << name << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

delete S;

S = nullptr;

pow = 0;

for(List \* p = B.S; p; p = p->next)

S = new List(p->e, S), ++pow;

}

// std::cout << "-> Создано " << name << " = {" << \*S << "}, |" << name << "| = " << pow

// << " из " << name << std::endl;

return \*this;

}

void Set::print()

{

std::cout << name << " = {" << \*S << "}, |" << name << "| = " << pow << std::endl;

}

**8) word.h**

#ifndef WORD\_H

#define WORD\_H

class Set

{

private:

static int U, count; // мощность юниверсума и количество множеств

char name; // название множества

unsigned int S; // память для множества

public:

Set(char); // конструктор класса

Set(): name('A' + count++), S(0){} // конструктор по умолчанию

Set(const Set& other): name('A' + count++), S(other.S){} // конструтор копирования

// операции над множествами

Set& operator= (const Set&);

Set operator & (const Set&) const;

Set operator | (const Set&) const;

Set operator / (const Set&) const;

Set operator ~ () const;

// интерфейс

char get\_name() {return name;}

int power();

void print();

};

#endif

9) word.cpp

#include "word.h"

#include <iostream>

// конструторы

Set::Set(char tag): name(tag), S(rand()%0x3ffffff)

{

count++;

}

// операции

Set & Set::operator = (const Set& other)

{

if (this != &other)

S = other.S;

return \*this;

}

Set Set::operator & (const Set& other) const

{

Set C;

C.S = S & other.S;

return C;

}

Set Set::operator | (const Set& other) const

{

Set C;

C.S = S | other.S;

return C;

}

Set Set::operator / (const Set& other) const

{

Set C;

C.S = S & ~other.S;

return C;

}

Set Set::operator ~ () const

{

Set C;

C.S = ~S & 0x3ffffff;

return C;

}

// интерфейс

int Set::power()

{

int pow = 0;

unsigned int n = S;

while(n != 0)

{

pow++;

n &= (n-1);

}

return pow;

}

void Set::print()

{

char\* result = new char[U+1];

int i,j;

for(i = 0, j = 0; i < U; ++i)

if(1 << i & S)

result[j++] = i + 'A';

result[j] = '\0';

std::cout << name << " = {" << result << "}, |" << name << "| = " << this->power() << std::endl;

delete [] result;

}