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 34**

|  |  |
| --- | --- |
| Выполнили студенты гр.9308: | Дементьев Д.П., Кашапова О.С. |
| 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)

[Приложение 2 (Реализация структур) 11](#_Приложение 2 (Реализация структур))

## Введение

Целью задания является исследование эффекта от использования классов.

## 1. Задание

Получить множество, содержащее буквы множества A, которых нет во множествах B, C или D. Универсум — строчные латинские буквы

## 2. Формализация задания

Условие задачи можно записать в виде формулы:  
((A & ~B) & ~C) & ~D, где A, B, C, D являются классами.

## 3. Контрольные примеры

Контрольные примеры представлены в таблице 1*.*

Таблица. 1. Контрольные примеры

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Исходные множества** | | | | **Результат** |
| **A** | **B** | **C** | **D** | **E** |
| 1 | abcdef | bnm | tyu | q | acdef |
| 2 | abcde | bc | ed | a | Пустое множество |
| 3 | bcdeghijsw | acdejnpuwxz | acjkmorstuvy | abefghjmnoqrstvy | i |
| 4 | qwertyuiopasdfghjklzxcvbnm | zxcvbnm | asdfghjkl | tyuiop | qwer |
| 5 | m | c | d | e | m |

## 4. Результаты эксперимента

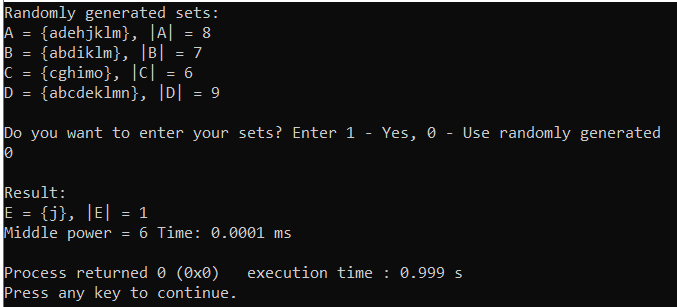
Результаты измерения времени обработкипредставлены в таблице 2*.*

Таблица. 2. Результаты измерения времени обработки

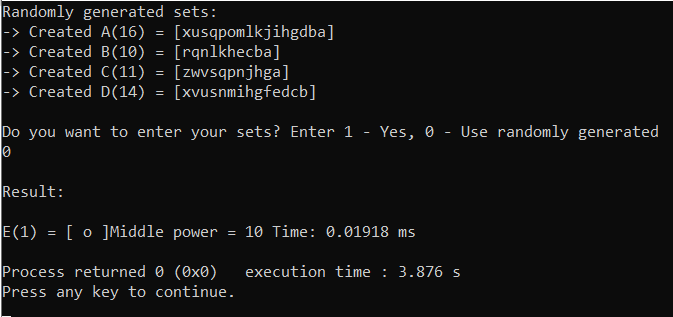
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Способ представления** | **Время(с)** | **Время в прошлой программе(с)** | **Количество повторов цикла** | **Зависимость от количества в множестве** |
| Массив символов | 0.00461 - 0.00771 | 0.000191 - 0.002 | 100000 | Есть |
| Список | 0.0146 - 0.02 | 0.001186 - 0.005 | Есть |
| Машинное слово | 0.0001 | 2\*10-6 | Нет |
| Массив битов | 0.00264 - 0.003 | 3.6\*10-5 | Да |

Основываясь на полученных данных, которые приведены в таблице, можно заметить, что при работе с классами все способы работают гораздо медленнее, чем при линейном программировании.

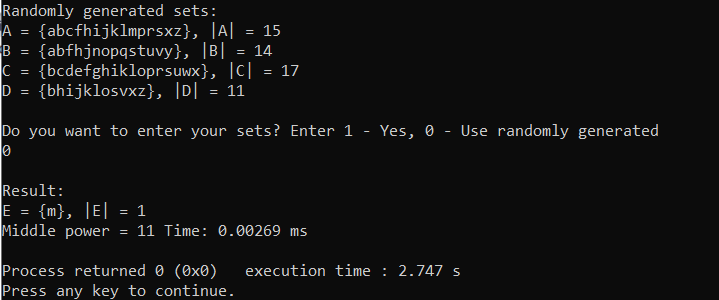
На скриншотах ниже приведены результаты выполнения программы со структурами, а также списка с отслеживанием:



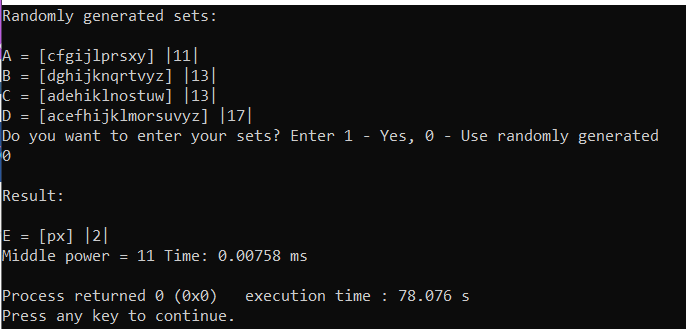
*Рисунок 1. Машинное слово.*



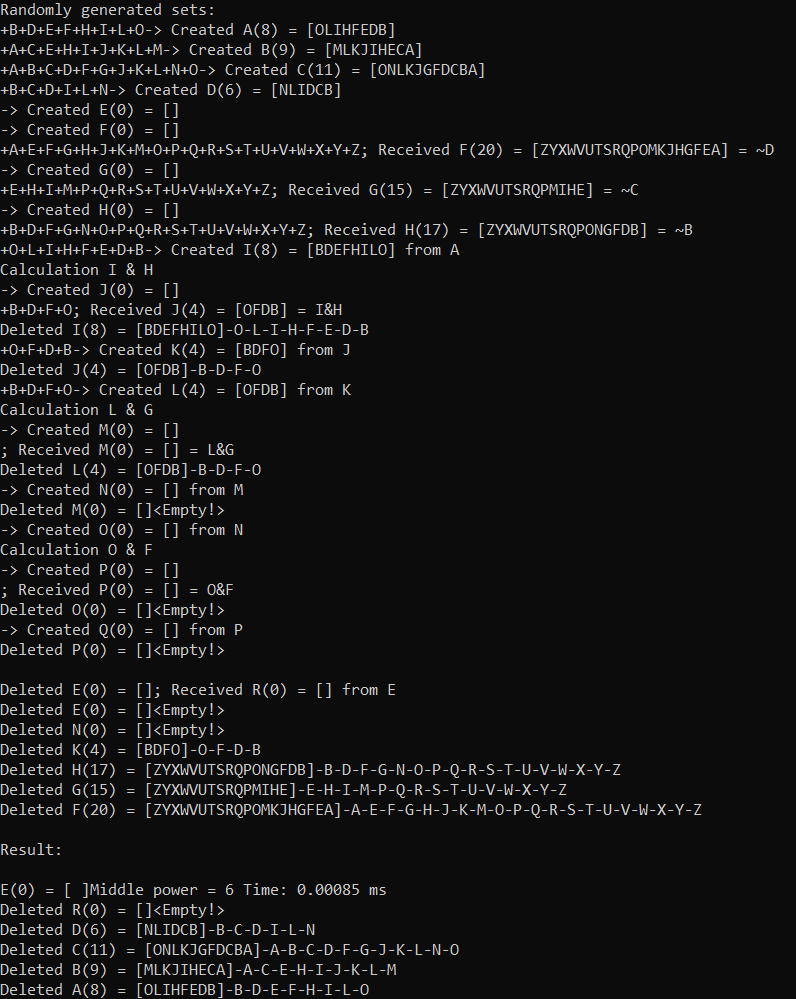
*Рисунок 2. Протокол выполнения программы со списками.*



*Рисунок 3. Протокол выполнения программы с массивом битов  
  
.*



*Рисунок 4. Протокол выполнения программы со статическим массивом*

 *Рисунок 5. Список с отслеживанием*

## Вывод

При процедурном программировании функ­ции акку­рат­но вло­же­ны друг в дру­га, вза­и­мо­дей­ству­ют друг с дру­гом, мож­но пере­дать дан­ные из одной функ­ции в дру­гую. Но при большом их количестве изменение в одной из них влечет изменения всех связанных с ней функций. Появляется сложность написания, чтения, отладки кода.

Объектно-ориентированное программирование помогает исправить эту проблему за счёт трёх принципов ООП.

* **Инкап­су­ля­ция — объ­ект неза­ви­сим:** каж­дый объ­ект устро­ен так, что нуж­ные для него дан­ные живут внут­ри это­го объ­ек­та, а не где-то сна­ру­жи в про­грам­ме.Данные и операции вместе образуют определенную сущность и они не «размазываются» по всей программе, как это нередко бывает в случае процедурного программирования.
* **Абстрак­ция — у объ­ек­та есть «интер­фейс»:** у объ­ек­та есть мето­ды и свой­ства, к кото­рым мы можем обра­тить­ся извне это­го объ­ек­та.
* **Насле­до­ва­ние — спо­соб­ность к копи­ро­ва­нию.** ООП поз­во­ля­ет созда­вать мно­го объ­ек­тов по обра­зу и подо­бию дру­го­го объ­ек­та.

Если использовать конструктор копии класса, то можно заметить, что адрес в памяти у класса и его копии один и тот же. Именно поэтому мы перегружаем оператор присваивания, чтобы адреса у копии и оригинала были разные. Если этого не сделать, то возможна ошибка при вызове деструктора, который будет пытаться два раза освободить одну и ту же область в памяти.

После реализации всех структур данных оказалось, что все при реализации ОО способом все из них оказались гораздо медленнее.

## Список используемых источников

1. Колинько П.Г. Пользовательские структуры данных / Методические указания по дисциплине «Алгоритмы и структуры данных» - Санкт-Петербург: СПбГЭТУ «ЛЭТИ», 2020.
2. Операции над множествами. <https://www.cyberforum.ru/cpp-beginners/thread2191195.html>
3. Перегрузка операторов пример. ООП. Перегрузка оператора присваивания. C++ Для начинающих. Урок#83. <https://www.youtube.com/watch?v=nMM98LVJn-U>
4. Урок №114. Спецификаторы доступа public и private. <https://ravesli.com/urok-114-spetsifikatory-dostupa-public-i-private/>

## Приложение 1 (Исходный текст программы)

#include <iostream>

#include <clocale>

#include <cstdlib>

#include <ctime>

using namespace std;

const int NMAX=26;

#include "Set.h" //Реализация структур данных

int Set :: N = NMAX, Set :: cnt = 0; // Определение статических членов класса

const long rolls = 100000; //Количество итераций

int main(){

srand(time(nullptr));

char Q;

char A\_0[NMAX+1], B\_0[NMAX+1], C\_0[NMAX+1], D\_0[NMAX+1];

cout << "Randomly generated sets:" << endl;

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

cout << endl << "Do you want to enter your sets? Enter 1 - Yes, 0 - Use randomly generated" << endl;

cin >> Q;

if (Q=='1')

{

cout << "A: "; cin >> A\_0; cout << "B: "; cin >> B\_0; cout << "C: "; cin >> C\_0; cout << "D: "; cin >> D\_0;

A.rewrite(A\_0); B.rewrite(B\_0); C.rewrite(C\_0); D.rewrite(D\_0);

}

clock\_t start = clock();

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

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

}

clock\_t ticks = clock() - start;

cout << "\nResult:\n";

E.Show('E');

cout << " Middle power = " << (A.power()+B.power()+C.power()+D.power()+E.power())/5 <<

" Time: " << 1000 \* ((((double) ticks) / CLOCKS\_PER\_SEC) / rolls) << " ms" << endl;

return 0;}

## Приложение 2 (Реализация структур)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Array\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

static int N, cnt; // мощность универсума

int n; // мощность множества

char S, \* A; // тег и память для множестваpublic: // Открытая часть — функции для работы с множеством

//Конструкторы и деструктор

Set(char); // конструктор множества

Set(); // конструктор по умолчанию

Set(const Set&); // конструктор копии

Set& operator = (const Set&); // перегрузка оператора присваивания

~Set() { delete[] A; } // деструктор

//Операции над множеством

Set operator | (const Set&) const;

Set operator & (const Set&) const;

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

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

Set operator ~ () const;

//Интерфейс

void Show(); //Вывод множества на экран

void Show(char c); //Вывод с фиксированным тэгом

int power( ) { return n; } //Мощность

void rewrite(char str[]); //Перезаписать множество};

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

Set C(\*this);

n = 0;

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

{

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

if (C.A[i] == B.A[j]) A[n++] = C.A[i];

}

A[n] = 0;

return \*this;}

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

Set C(\*this);

return (C &= B);}

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

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) A[n++] = B.A[i];

}

A[n] = 0;

return \*this;}

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

Set C = \*this;

return (C |= B);}

Set Set :: operator ~ () const{

Set C;

for (char l = 'a'; l <= 'z'; ++l)

{

bool f = true;

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

if (l == A[j]) { f = false; break; }

if (f) C.A[C.n++] = l;

}

C.A[C.n] = 0;

return C;}

Set::Set() : n(0), S('A' + cnt++), A(new char[N + 1]) // Конструктор пустого множества{

A[0] = '\0';}

Set::Set(char s) : S('A' + cnt++), n(0), A(new char[N + 1]) // Конструктор случайного множества{

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

if (rand() % 2) A[n++] = i + 'a';

A[n] = '\0';

cout << '\n' << S << " = [" << A << "] " << "|" << this->power() << "|";}

Set :: Set(const Set & B): S('A'+ cnt++), n(B.n), A(new char[N+1]) // Конструктор копирования{

char \*dst(A), \*src(B.A);

while (\*dst++ = \*src++);}

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

if (this != &B)

{

char \*dst(A), \*src(B.A);

n = B.n;

while (\*dst++ = \*src++); S = 'A'+cnt++;

}

return \*this;}

void Set::rewrite(char str[]){

n = 0;

for (int i = 0; i < strlen(str); i++)

A[n++] = str[i];

A[n] = '\0';

Show();}

void Set::Show(){

cout << '\n' << S << " = [" << A << "] " << "|" << power() << "|" << endl;}

void Set::Show(char c){

cout << '\n' << c << " = [" << A << "] " << "|" << power() << "|" << endl;}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Array\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Machine Word\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Set{private:

static int N, cnt; // мощность юниверсума и количество множеств

char name; // название множества

unsigned int S; // память для множестваpublic:

Set(char); // конструктор класса

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

Set(const Set& other): name('A' + cnt++), 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

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

int power();

void rewrite(char str[]); //Перезаписать множество

void Show();

void Show(char);};

// конструтор со случайным заполнениемSet::Set(char tag): name(tag), S(rand()%0x3ffffff){

cnt++;

Show();}

// 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;}

// Интерфейсint Set::power(){

int pow = 0;

unsigned int n = S;

while(n != 0)

{

pow++;

n &= (n-1);

}

return pow;}

void Set::rewrite(char str[]){

S = 0;

for (int i = 0; i < strlen(str); i++)

S = S | 1 << str[i] - 'a';

Show();}

void Set::Show(){

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

int i,j;

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

if(1 << i & S)

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

result[j] = '\0';

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

delete [] result;}

void Set::Show(char c){

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

int i,j;

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

if(1 << i & S)

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

result[j] = '\0';

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

delete [] result;}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Machine Word\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Boolean\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Set{private:

static int N, cnt; // мощность юниверсума и количество множеств

char name; // название множества

bool \*S; // память для множестваpublic:

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 rewrite(char str[]); //Перезаписать множество

void Show();

void Show(char c);};

Set::Set(): name('A' + cnt++), S(new bool[N]) // конструтор пустого множества{

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

S[i] = 0;}

Set::Set(char tag): name(tag), S(new bool[N]) // конструтор случайного множества{

cnt++;

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

S[i] = rand()%2;

Show();}

Set::Set(const Set& other): name('A' + cnt++), S(new bool[N]) // конструтор копирования{

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

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

// операцииSet & Set::operator = (const Set& other){

if (this != &other)

{

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

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

}

return \*this;}

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

Set C;

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

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

return C;}

Set Set::operator ~ () const{

Set C;

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

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

return C;}

// интерфейсint Set::power(){

int counter = 0;

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

if (S[i])

counter++;

return counter;}

void Set::rewrite(char str[]){

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

S[i] = false;

for (int i = 0; i < strlen(str); i++)

S[str[i]-'a'] = true;

Show();}

void Set::Show(){

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

int i,j;

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

if(S[i])

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

result[j] = '\0';

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

delete [] result;}

void Set::Show(char c){

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

int i,j;

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

if(S[i])

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

result[j] = '\0';

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

delete [] result;}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Boolean\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class El{ //Элемент множества(узел)

char e;

El \*next;public:

El(): e('!'), next(nullptr){ }

El(char e, El \*n = nullptr): e(e), next(n) { }

~El(){ if(this) delete next; } //Деструктор

friend class Set;

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

for (El\* p = &S; p; p = p->next) o << p->e;

return o;}

class Set{private:

static int N; //Мощность универсума

static int cnt; //Порядковый номер множества

int n; //Мощность множества

char S; //Тег

El \*A; //Список элементов

public:

//Конструкторы и деструктор

Set(); //Пустое множество

Set(char); //Случайное произвольной мощности (аргумент игнорируется)

Set(char, char[]); //Конструктор для множества вводимого с клавиатуры

Set(const Set &); //Перегрузка копирования

Set(Set &&); //Копирование с переносом

~Set() { A->El::~El(); } //Деструктор

//Операции

Set & operator = (const Set&); //Перегрузка присваивания

Set & operator = (Set &&); //Присваивание с переносом

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

Set & operator |= (const Set&);

Set & operator &= (const Set&);

Set operator | (const Set&) const;

Set operator & (const Set&) const;

Set operator ~ () const;

//Интерфейс

void Show();

void Show(char);

int power() { return n; }

void rewrite(char str[]); //Перезаписать множество};

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

Set::Set(char) : S('A'+cnt++), n(0){

A = nullptr;

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

if (rand()%2) A = new El(i + 'a', A), ++n;

std::cout << "-> Created " << S << "(" << n << ") = [" << \*A << "] \n";}

Set::Set(char c, char str[]) : S('A'+cnt++), n(0){

A = nullptr;

N = strlen(str);

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

A = new El(str[i], A), ++n;

std::cout << "-> Created " << S << "(" << n << ") = [" << \*A << "] \n";}

Set::Set(const Set & B) : n(B.n), S('A'+cnt++), A(nullptr){

for(El \* p = B.A; p; p = p->next) A = new El(p->e, A);}

Set::Set( Set && B) : n(B.n), S('A'+cnt++), A(B.A){

B.A = nullptr;}

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

Set C;

for (El \* i = A; i; i = i->next)

{

for (El \* j = B.A; j; j = j->next)

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

C.A = new El(i->e, C.A), ++C.n;

}

swap(C);

return \*this;}Set Set::operator & (const Set& B) const{

Set C(\*this);

return C &= B;}

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

Set C(\*this);

for (El \* i = B.A; i; i = i->next)

{

bool f = true;

for (El \* j = A; f && j; j = j->next)

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

if (f)

C.A = new El(i->e, C.A), ++C.n;

}

swap(C);

return \*this;}Set Set::operator | (const Set& B) const{

Set C(\*this);

return C |= B;}Set Set::operator ~ ()const{

Set C;

for (char c = 'a'; c <= 'z'; ++c)

{

bool f = true;

for (El \* j = A; j && f; j = j->next)

if(c == j->e) f = false;

if (f)

C.A = new El(c, C.A) , ++C.n;

}

return C;}

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

if (this != &B)

{

delete A;

A = nullptr;

n = 0;

for(El \* p = B.A; p; p = p->next)

A = new El(p->e, A), ++n;

S = 'A'+cnt++;

}

return \*this;}

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

swap(B);

delete B.A; B.A = nullptr;

S = 'A'+cnt++;

return \*this;}

void Set::rewrite(char str[]){

delete A;

A = nullptr;

n=0;

N = strlen(str);

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

A = new El(str[i], A), ++n;

std::cout << "-> Created " << S << "(" << n << ") = [" << \*A << "] \n";}

void Set::Show(){

std::cout<<'\n'<< S << "(" << n << ") = [ ";

for(El \* p = A; p; p = p->next) std:: cout << p->e << " ";

std::cout << "]";}

void Set::Show(char c){

std::cout<<'\n'<< c << "(" << n << ") = [ ";

for(El \* p = A; p; p = p->next) std:: cout << p->e << " ";

std::cout << "]";}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*