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 | ABCDFGHIJ | GHIJ | ABHIJ | ABFGHI | ABCDFGJ |
| 2 | ACDEKMOPRSUVWXZ | ABCDHIJLMNPQRWYZ | ACDFJKMNOSUVWYZ | BCDEGHILNPQRUVY | AEKMOPRSUVWXZ |
| 3 | ABCDFGHIJLMNQRSTUVZ | ABGJLMOQRSUY | ACEGHIJLMNOQTWZ | AFGJKLMPVWYZ | BCDFHINQRSTUVZ |
| 4 | ABC | DEF | GHI | XYZ | ABC |
| 5 | ABC | ABC | ABC | ABC | Пустое множество |

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

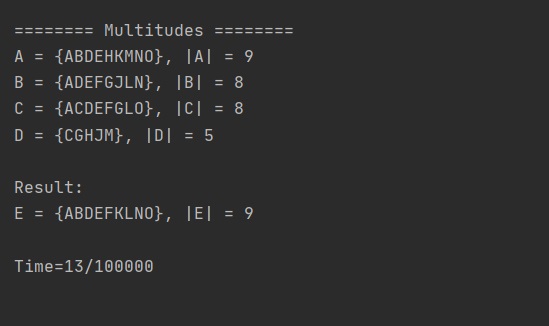
Результаты измерения времени обработкипредставлены в таблице 2*.*

Таблица. 2. Результаты измерения времени обработки

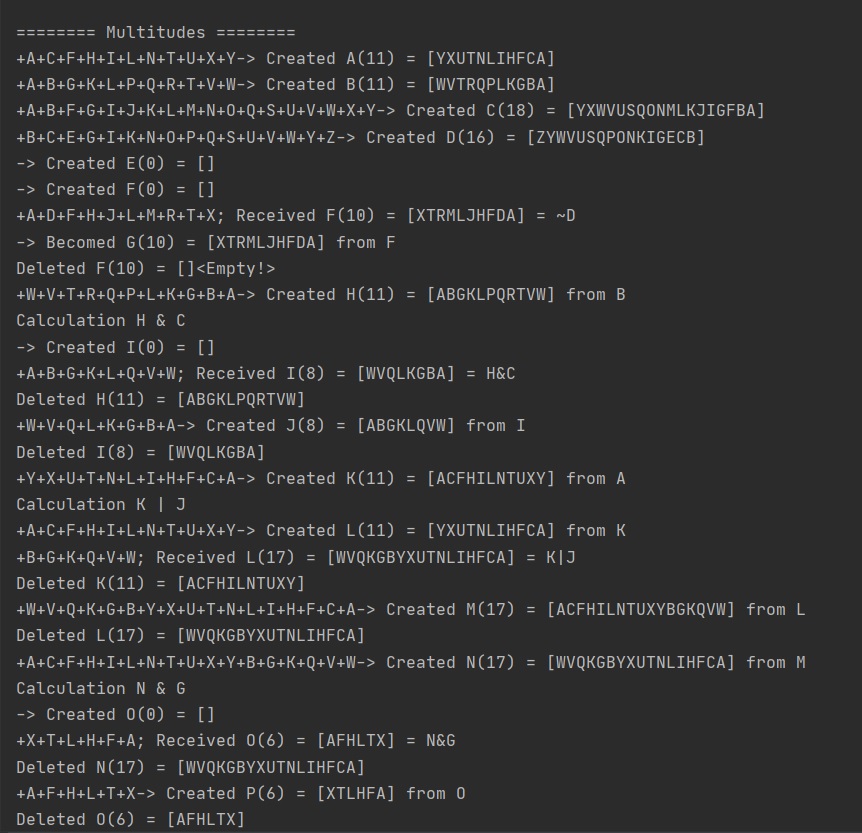
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Способ представления** | **Время (мс)** | **Время в прошлой программе (мс)** | **Количество повторов цикла** | **Зависимость от количества в множестве** |
| Массив символов |  | 0.0001642 - 0.0022 | 10000000 | Есть |
| Список |  | 0.0003535 -  0.00322 | Есть |
| Массив битов | 0.0007421 -  0.0007877 | 0.000087 -  0.000108 | Нет |
| Машинное слово |  | 0.0000016 -  0.000002 | Нет |

Основываясь на полученных данных, которые приведены в таблице, можно заметить, что при работе с классами списки работают гораздо быстрее, чем при линейном программировании. Другие же способы наоборот работают немного медленнее.

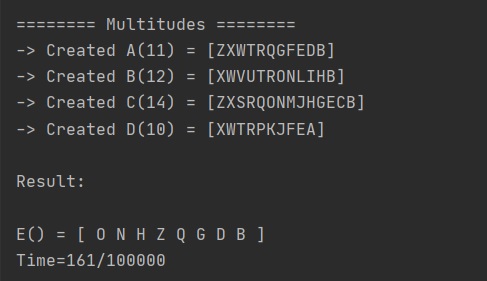
На скриншотах ниже приведены результаты выполнения программы со структурами, в том числе с отслеживанием вызова функций:



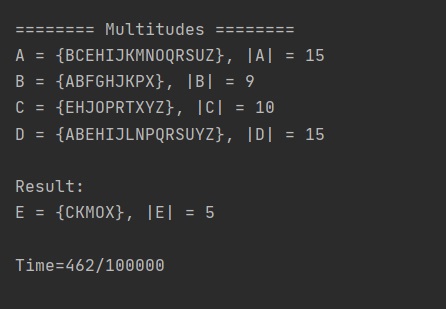
*Рисунок 1. Протокол выполнения программы с машинным словом.*

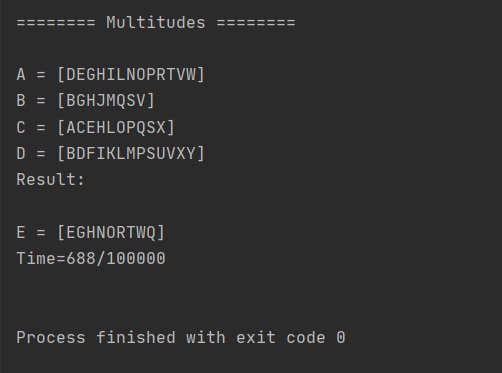


*Рисунок 2. Протокол выполнения программы со списками с отслеживанием вызова функций.*



*Рисунок 3. Протокол выполнения программы со списками.*



*Рисунок 4. Протокол выполнения программы с массивом битов**.*

*Рисунок 5. Протокол выполнения программы со статическим массивом*

## 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.

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.

В ходе выполнения опытов с программой удалось выяснить, что при работе с классами списки работают гораздо быстрее, чем при линейном программировании. Остальные же способы стали работать медленнее.

## 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.
4. Операции над множествами. https://www.cyberforum.ru/cpp-beginners/thread2191195.html

## Приложение 1 (Исходный текст программы)

1. #include <iostream>  
   #include <clocale>  
   #include <cstdlib>  
   #include <ctime>  
     
   using namespace std;  
     
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MWORD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
     
   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  
    inline char get\_name() {return name;}  
    int power();  
     
    void Show();  
   };// конструторы  
   int Set :: U = 26;  
   int Set :: count = 0;  
     
   Set::Set(char tag): name(tag), S(rand()%0x3ffffff)  
   {  
    count++;  
    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;  
   }  
     
   // интерфейс  
     
   int Set::power()  
   {  
    int pow = 0;  
    unsigned int n = S;  
    while(n != 0)  
    {  
    pow++;  
    n &= (n-1);  
    }  
    return pow;  
   }  
     
     
   void Set::Show()  
   {  
    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;  
   }  
     
     
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MWORD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
     
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bool\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   /\*  
   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 Show();  
   };  
   // конструторы  
   int Set :: U = 26;  
   int Set :: count = 0;  
     
   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;  
    Show();  
   }  
     
   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::Show()  
   {  
    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;  
   }  
   \*/  
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bool\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
   /\*  
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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 множества  
   public: // Открытая часть — функции для работы с множеством  
    Set operator | (const Set&) const; // объединение  
    Set operator & (const Set&) const; // пересечение  
    Set & operator &= (const Set& B);  
    Set & operator |= (const Set& B);  
    Set operator ~ () const; // дополнение до универсума  
    void Show(); // вывод множества на экран  
    //int power( ) { return n; } // получение мощности  
    Set(char); // конструктор множества  
    Set(); // ещё конструктор — по умолчанию  
    Set(const Set&); // конструктор копии  
    Set& operator = (const Set&); // оператор присваивания  
    ~Set() { delete[] A; } // деструктор  
   };  
     
   int Set::N = 26;  
   int Set::cnt = 0;  
     
   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(char s) : S('A' + cnt++), n(0), A(new char[N + 1])  
   {  
    for (int i = 0; i < N; i++)  
    if (rand() % 2) A[n++] = i + 'A';  
    A[n] = '\0';  
    cout << '\n' << S << " = [" << A << "]";  
     
   }  
     
   Set :: Set(const Set & B): S('A'+ cnt++), n(B.n), A(new char[N+1])  
   {  
    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++;  
    }  
    //cout << '\n' << 'E' << " = [" << A << "]";  
    return \*this;  
   }  
     
   void Set::Show()  
   {  
    cout << '\n' <<"E" << " = [" << A << "]";  
   }  
     
   Set::Set() : n(0), S('A' + cnt++), A(new char[N + 1])  
   {  
    A[0] = 0;  
   }  
   \*/  
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Array\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
   /\*  
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
   class El{  
    char e;  
    El \*next;  
    static const int maxmup = 200;  
    static El mem[maxmup]; //Свободная память для элементов списков  
    static int mup, mup0;  
   public:  
    El(): e('!'), next(nullptr){ }  
    El(char e, El \*n = nullptr): e(e), next(n) { }  
    ~El(){ if(this) {//Прямой вызов деструктора требует такой проверки!!!  
    delete next;  
    }  
    //else 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;} //Фиксировать состояние памяти  
    static void release() { mup = mup0; } //Сбросить до фиксированного  
    static void clear(){ mup = 0; } //Очистить память полностью  
    friend class Set;  
    friend std::ostream & operator << ( std::ostream & o, El & S); //Перегрузка << для вывода  
    friend static void memOut(); //Вспомогательная функция для вывода содержимого памяти  
   };  
   std::ostream & operator << (std::ostream & o, El & S)  
   {  
    for (El\* p = &S; p; p = p->next) o << p->e;  
    return o;  
   }  
     
   El El::mem[El::maxmup]; //"Свободная память"  
   int El::mup = 0, El::mup0 = 0; //Рабочий и резервный указатели на свободное место  
     
   void memOut( ) //Вывод использованного содержимого "свободной памяти"  
   {  
    std::cout << "\n\nMemory for list elements (total - " << std::dec << El::mup << ")\n";  
    for(int i = 0; i < El::maxmup; ++i) cout << El::mem[i].e;  
   }  
     
   //2. Объявление класса «множество-список» с автоматической нумерацией вновь создаваемых множеств  
   class Set  
   {  
   private:  
    static const int N; //Мощность универсума  
    static int num; //Порядковый номер множества  
    int n; //Мощность множества  
    char S; //Тег  
    El \*A; //Список элементов  
     
   public:  
    Set(); //Пустое множество  
    Set(char); //Случайное произвольной мощности (аргумент игнорируется)  
    Set(const Set &);  
    Set(Set &&);  
    Set & operator = (const Set&);  
    Set & operator = (Set &&);  
    ~Set() {  
    //std::cout << "Deleted " << S <<"(" << std::dec << n << ") = [" << \*A << "]";  
    A->El::~El();} //Здесь нужен явный вызов деструктора  
     
    void Show();  
    int power() { return n; }  
    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;  
   };  
     
   Set::Set() : n(0), S('A'+num++), A(nullptr)  
   {  
    //std::cout << "-> Created " << S << "(" << n << ") = [" << \*A << "] \n";  
   }  
     
   Set::Set(char) : S('A'+num++), n(0)  
   {  
    A = nullptr; //auto w = rand();  
    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(const Set & B) : n(B.n), S('A'+num++), A(nullptr)  
   {  
    for(El \* p = B.A; p; p = p->next) A = new El(p->e, A);  
    //std::cout << "-> Created " << S << "(" << n << ") = ["  
    // << \*A << "] from " << B.S << std::endl;  
   }  
     
   Set::Set( Set && B) : n(B.n), S('A'+num++), A(B.A)  
   {  
    B.A = nullptr;  
    //std::cout << "-> Becomed " << S << "(" << n << ") = ["  
    // << \*A << "] from " << B.S << std::endl;  
   }  
     
   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);  
    // std::cout << "; Received " << S << "(" << n << ") = ["  
    // << \*A << "] = " << C.S << "&" << B.S << std::endl;  
    return \*this;  
   }  
   Set Set::operator & (const Set& B) const  
   {  
    Set C(\*this);  
    //std::cout << "Calculation " << C.S << " & " << B.S << std::endl;  
    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);  
    //std::cout << "; Received " << S << "(" << n << ") = ["  
    // << \*A << "] = " << C.S << "|" << B.S << std::endl;  
    return \*this;  
   }  
   Set Set::operator | (const Set& B) const  
   {  
    Set C(\*this);  
    //std::cout << "Calculation " << C.S << " | " << B.S << std::endl;  
    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;  
    }  
    // std::cout << "; Received " << C.S << "(" << C.n << ") = [" << \*C.A  
    //<< "] = ~" << S << std::endl;  
    return C;  
   }  
     
   Set& Set::operator = (const Set & B)  
   {  
    if (this != &B)  
    { //std::cout << "\nDeleted " << S << "(" << n << ") = [" << \*A << "]";  
    //delete A;  
    //A = nullptr;  
    n = 0;  
    for(El \* p = B.A; p; p = p->next)  
    A = new El(p->e, A), ++n;  
    // S = 'A'+num++;  
    }  
    //std::cout << "; Created " << S << "(" << n << ") = [" << \*A << "] from "  
    // << B.S << std::endl;  
    return \*this;  
   }  
     
   Set& Set::operator = (Set && B)  
   { //std::cout << "\nDeleted " << S << "(" << n << ") = [" << \*A << "]";  
    swap(B);  
    //delete B.A; B.A = nullptr;  
    //S = 'A'+num++;  
    //std::cout << "; Received " << S << "(" << n << ") = ["  
    // << \*A << "] from " << B.S << std::endl;  
    return \*this;  
   }  
     
     
   void Set::Show()  
   {  
    std::cout<<'\n'<< 'E' << "(" << ") = [ ";  
    for(El \* p = A; p; p = p->next) std:: cout << p->e << " ";  
    std::cout << "]";  
   }  
     
   const int Set :: N = 26;  
   int Set :: num = 0;  
     
   \*/  
   const long q0 = 100000;  
     
   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
     
   int main()  
   {  
    srand(time(nullptr));  
    cout << "\n======== Multitudes ========\n";  
    Set A('A'), B('B'), C('C'), D('D'), E;  
    clock\_t begin = clock();  
    E=((A | (B & C)) & ~D);  
    for (int i = 0;i<q0;i++) {  
    ((A | (B & C)) & ~D);  
    }  
    clock\_t end = clock();  
    cout << "\nResult:\n";  
    E.Show();  
    cout << '\n'<<"Time=" << int(end - begin) <<"/"<<q0<<'\n'<<'\n';  
    return 0;  
   }