**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчЁт

**по лабораторной работе № 4**

**по дисциплине «Программирование»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент гр. 4354 |  |  | И.В. Чучалин |
| Преподаватель |  |  | В.А. Калмычков |

# Оглавление

[Оглавление 2](#_Toc197897692)

[Задание 2](#_Toc197897693)

[Контрольные примеры 2](#_Toc197897694)

[Организация интерфейса пользователя 2](#_Toc197897695)

[Организация входного файла 2](#_Toc197897696)

[Организация выходного файла 3](#_Toc197897697)

[Организация консоли 3](#_Toc197897698)

[Реализация ввода/вывода 3](#_Toc197897699)

[Модульное представление 3](#_Toc197897700)

[Внутреннее представление данных 3](#_Toc197897701)

[Описание внутренних функций 4](#_Toc197897702)

[Определение функций 4](#_Toc197897703)

[Назначение функций 5](#_Toc197897704)

[Описание алгоритма 5](#_Toc197897705)

[Общее описание алгоритма программы 5](#_Toc197897706)

[Принцип работы иерархического списка 6](#_Toc197897707)

[Блок-схема 7](#_Toc197897708)

[Текст программы 7](#_Toc197897709)

[Результат работы программы 7](#_Toc197897710)

[Вывод 7](#_Toc197897711)

# Задание

Задан текст, представленный иерархическим списком. Необходимо проверить истинность утверждения: S3 с (S1 п S2).

# Контрольные примеры

1. Утверждение истинно.

Входные данные: Полученный результат:

okaywow S1 п S2: okaywow

dont read Утверждение S3 с (S1 п S2) истинно.

this text

okaywow

you did it

okay wow

okaywow

2. Утверждение ложно.

Входные данные: Полученный результат:

okay S1 п S2: okay

dont read Утверждение S3 с (S1 п S2) ложно.

this text

okay

you did it

okaywow

ok

# Организация интерфейса пользователя

## Организация входного файла

О1. Текст, представленный в виде трех групп строк, разделенных пустыми строками.

## Организация выходного файла

1. Вывод считанных списков.
2. Вывод пересечения.
3. Вывод результата.

## Организация консоли

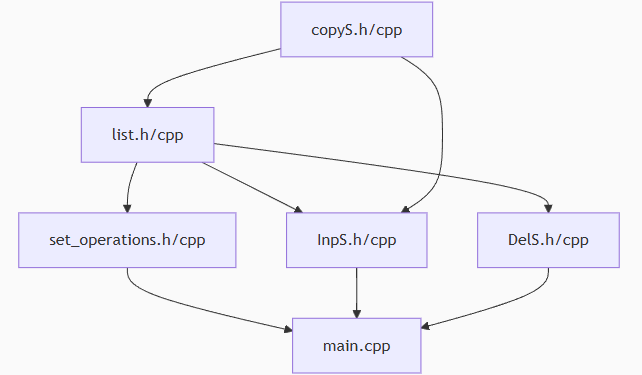
О1. Вывод ошибок.

# Реализация ввода/вывода

Использованы потоки ввода (cin>>) и вывода (cout<<) библиотеки «iostream», а также функции объекта cin: getline, clear, eof.

Использованы потоки чтения из файла (>>) и записи в файл (<<) библиотеки «fstream», а также функции объекта типа fstream: open, is\_open, fail, seekg, eof, close. Для настройки ввода были использованы функции setiosflags, resetiosflags библиотеки «iomanip».

# Модульное представление



# Внутреннее представление данных

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Переменная | Назначение |
| EL\_Stroka | struct | char A[N] | Хранит строку данных |
| EL\_Stroka\* next | Связывает элементы в односвязный список |
| Form\_Stroka | EL\_Stroka\* h, \* l | Указывает на первый и последний элемент списка строк |
| unsigned L | Счётчик элементов в списке |
| EL\_V | Form\_Stroka F | Хранит список строк (горизонтальный уровень) |
| EL\_V\* next | Связывает элементы в вертикальный список (иерархия) |
| Form\_V | EL\_V\* h, \* cur, \* last | Указывает на первый, текущий и последний уровень списка. |

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип | Назначение |
| N | const unsigned | Максимальная длина строки |
| fin | ifstream | Поток для чтения из файла |
| sets | |  |  | | --- | --- | | Form\_V\*[3] |  | | Массив указателей на множества |
| currentSet | int | Индекс текущего множества |
| temp | char[N] | Буфер для временного хранения строки из файла. |
| curV | EL\_V\* | Текущий уровень списка. |
| pos | int | Позиция в буфере |
| ch | char | Очередной символ из файла |
| level1, level2 | EL\_V\* | Указатели на текущие уровни списков |
| elem1, elem2 | EL\_Stroka\* | Указатели на текущие строки в уровнях |
| current1, current2 | const char\* | Текущие строки для сравнения. |
| resLevel | EL\_V\* | Указатель на текущий уровень результирующего списка |
| newLevel | EL\_V\* | Временный указатель для создания новых уровней |
| cmp | int | Результат сравнения строк |
| subLevel, mainLevel | EL\_V\* | Указатели на уровни списков |
| subElem, mainElem | |  |  | | --- | --- | | EL\_Stroka\* |  | | Указатели на строки в уровнях. |
| currentSub, currentMain | const char\* | Текущие строки для сравнения |
| curV | EL\_V\* | Текущий уровень списка |
| curS | EL\_Stroka\* | Текущая строка в уровне |
| tempV, tempS | EL\_V\*, EL\_Stroka\* | Временные указатели для освобождения памяти |
| S1, S2, S3 | Form\_V | Множества для хранения данных из файла |
| intersection | Form\_V | Результат пересечения |
| fout | ofstream | Поток для записи результатов в файл |
| isSubset | bool | Флаг результата проверки |

# Описание внутренних функций

## Определение функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя функции | Аргументы | | | | Возвращаемое значение |
| Входные | Выходные | Модифи-цируемые | Транзитные |
| CopyString | const char\* src, unsigned max | char\* dest | dest |  | void |
| CompareStrings | const char\* s1, const char\* s2 |  |  |  | int |
| InitELStroka | const char\* value | EL\_Stroka\*& el | el |  | void |
| InitFormStroka |  | Form\_Stroka& fs | fs.h, fs.l, fs.L |  |
| InitELV |  | EL\_V\*& el | el |  |
| InitFormV |  | Form\_V& fv | fv.h, fv.cur, fv.last |  |
| InsertString | const char\* value | Form\_Stroka& fs | fs.h, fs.l, fs.L |  |
| PrintHierarchy | |  |  | | --- | --- | | const Form\_V& list, const char\* name |  | | std::ostream& out |  |  |
| InpThreeSets | const char\* filename | Form\_V& S1, Form\_V& S2, Form\_V& S3 | S1, S2, S3 | std::ifstream fin, char temp[N] | void |
| |  |  | | --- | --- | | Intersection |  | | const Form\_V& s1, const Form\_V& s2 | Form\_V& result | result | EL\_V\* level1, EL\_V\* level2, EL\_Stroka\* elem1, EL\_Stroka\* elem2 | void |
| IsSubset | const Form\_V& subset, const Form\_V& mainSet |  |  | EL\_V\* subLevel, EL\_V\* mainLevel, EL\_Stroka\* subElem, EL\_Stroka\* mainElem | bool |
| Delete | Form\_V& fv |  | fv.h, fv.cur, fv.last | EL\_V\* curV, EL\_Stroka\* curS, EL\_V\* tempV, EL\_Stroka\* tempS | void |
| main |  |  | S1, S2, S3, intersection | std::ofstream fout | int |

## Назначение функций

* CopyString: Копирует строку src в dest .
* CompareStrings: Сравнивает строки лексикографически.
* InsertString: Добавляет строку value в конец списка fs.
* PrintHierarchy: Рекурсивно выводит список list в поток out.
* Init-функции инициируют.
* InpThreeSets читает данные из файла filename и заполняет множества S1, S2, S3.
* Intersection: Записывает в result пересечение множеств s1 и s2.
* IsSubset: Возвращает true, если subset является подмножеством mainSet.
* Delete рекурсивно удаляет все элементы списка fv и освобождает память.

# Описание алгоритма

## Общее описание алгоритма программы

Программа предназначена для проверки утверждения, является ли множество S3 подмножеством пересечения множеств S1 и S2. Работа программы начинается с чтения трех множеств строк из входного файла, где каждое множество отделено пустой строкой. Данные хранятся в виде иерархического списка, состоящего из двух уровней: вертикального (EL\_V) для групп строк и горизонтального (EL\_Stroka) для самих строк. После загрузки данных программа вычисляет пересечение S1 и S2, последовательно сравнивая элементы обоих множеств в отсортированном порядке. Результат пересечения сохраняется в отдельном списке. Затем осуществляется проверка, все ли элементы S3 присутствуют в полученном пересечении. Если это условие выполняется, программа возвращает истину, в противном случае — ложь. Весь процесс сопровождается выводом промежуточных результатов (исходных множеств, пересечения и итога проверки) в выходной файл. Для работы с памятью используются динамические структуры данных, которые создаются и удаляются вручную, что исключает утечки ресурсов.

## Принцип работы иерархического списка

Иерархический список организован как двухуровневая структура. На верхнем уровне находятся элементы типа EL\_V, которые образуют вертикальный список и содержат ссылки на нижний уровень — односвязные списки строк (Form\_Stroka). Каждый элемент Form\_Stroka (EL\_Stroka) хранит строку фиксированной длины (определенной константой N) и указатель на следующий элемент. Вертикальные уровни (EL\_V) связаны между собой через поле next, что позволяет создавать сложные иерархии данных. Для управления списками используются служебные структуры Form\_Stroka и Form\_V, которые содержат указатели на начало (h), конец (l или last) и текущий элемент (cur), а также счетчик элементов (L). Такая организация позволяет эффективно добавлять новые элементы в конец списка и обходить структуры как по горизонтали (перебор строк в уровне), так и по вертикали (переход между уровнями). Все операции вставки, удаления и сравнения оптимизированы для работы с предварительно отсортированными данными, что обеспечивает высокую производительность.

## Блок-схема

# Текст программы

# Результат работы программы

# Вывод

В ходе работы реализованы иерархические списки для хранения множеств строк. Разработаны алгоритмы поиска пересечения и проверки подмножества с линейной сложностью Освоены динамические структуры данных.