МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК ИМЕНИ
ПРОФЕССОРА Н.И.ЧЕРВЯКОВА

ЛАБОРАТОРНАЯ РАБОТА №18

Алгоритмизация и программирование

Вектор

Выполнил студент:
Сивко Иван Андреевич
студент 2 курса
группа ПМИ-б-о-23-2,
направление подготовки 01.03.02

Проверил:
Ассистент кафедры вычислительной
математики и кибернетики, к.ф.-м.н.,
Черкашина Анастасия Андреевна

2024 г.

Вариант 9

Цель:

- Совершенствование навыков разработки программ в среде программирования MS VStudio
- Совершенствование навыков в программировании с использованием векторов
- Исследование процесса формирования вектора
- Исследование операций с элементами векторов

Задание 1

1 Условие:

Используя полученную при выполнении лабораторной работы 10 в Задании II программу, реализовать возможность сохранения и обработки данных с использованием вектора.

Условие задания 2 лабораторной 10:

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

- *максимальный по модулю элемент массива;*
- *преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.*

2 Алгоритм / Мат. модель

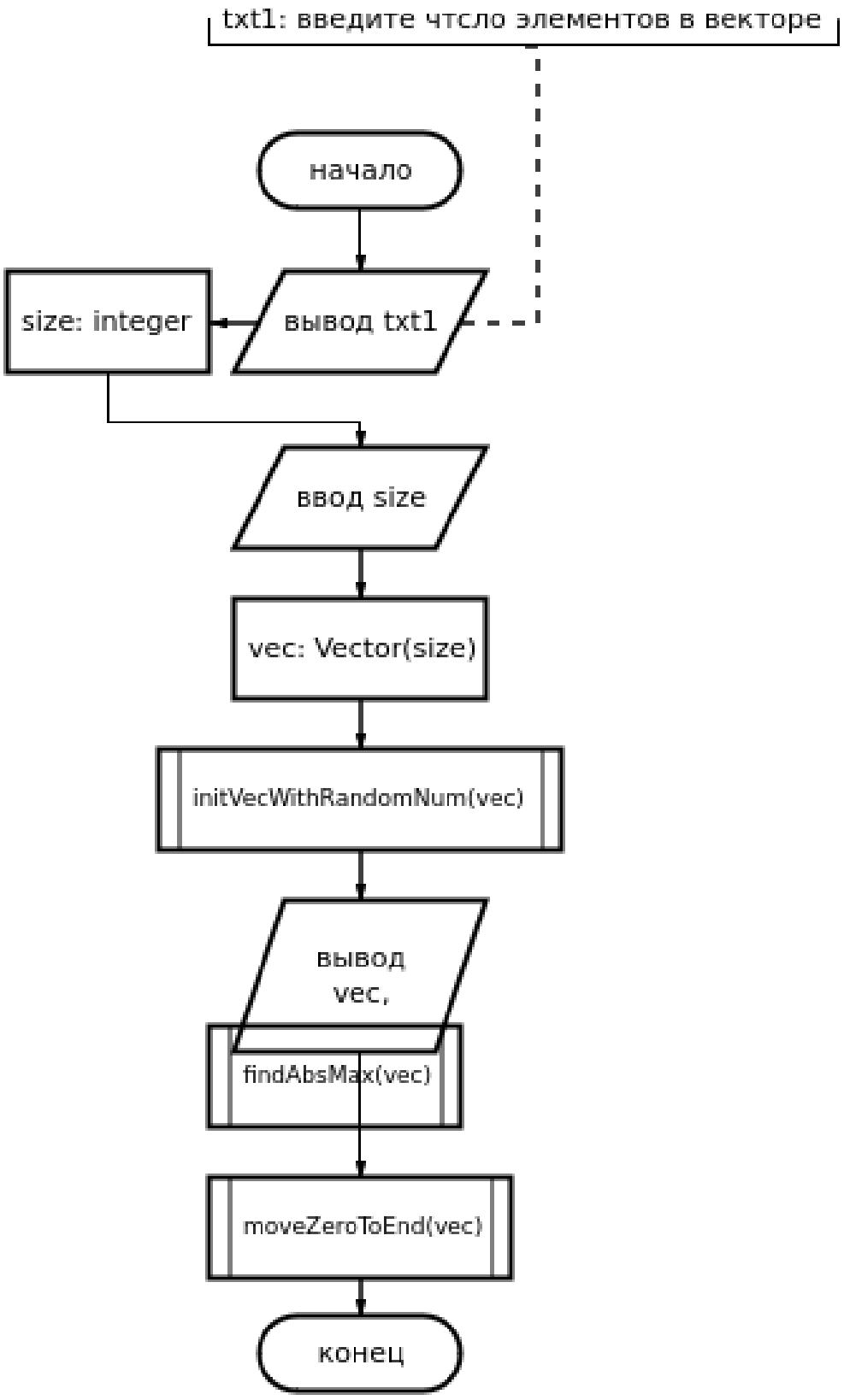
Программа инициализирует а затем заполняет вектор размера введенного пользователем числами от -100 до 100 затем выводит этот вектор, после выводит максимальный элемент массива и преобразует его таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

1. ввод размера для векторв `vec` (в переменную `size` типа *size_t (aka unsigned long)*)
2. инициализация вектора `vec` типа *double*
3. Заполнеине вектора лучайными значениями в пределах от -100 до 100
4. вывод вектора `vec` после заполнения и вывод максимального по абсалютной велечине значения вектора
5. перемещения всех 0 в `vec` в конец вектора
6. вывод вектора `vec` после перемещения всех 0 в конец вектора

Название	**Тип**	**Описание**
Классы и структуры		
Randgen	class	Генератор случайных чисел с различными распределениями.
Переменные-члены Randgen		
state	std::mt19937	Генератор случайных чисел Mersenne Twister, инициализированный функцией helpInitMt.
Функции-члены Randgen		
helpInitMt()	static std::mt19937	Инициализирует генератор случайных чисел с использованием текущего времени и случайных значений от std::random_device.
get<T>(const T&, const T&)	static T	Генерирует случайное число типа T в указанном диапазоне (целое или вещественное).
Другие функции		
abs(T n)	constexpr T	Возвращает модуль числа n.
swap(T& a, T& b)	void	Обменивает значения переменных a и b.
operator«	std::ostream&	Перегрузка оператора « для вывода элементов вектора std::vector<T> в поток.
initVecWithRandomNum	void	Инициализирует вектор случайными числами (используя функцию генерации случайных чисел, по умолчанию Randgen::get).
findAbsMax	const T&	Находит и возвращает элемент вектора с наибольшим абсолютным значением.
moveZerosToTheEnd	void	Перемещает все нулевые элементы вектора в конец, сохраняя порядок остальных элементов.
Переменные main		
size	size_t	Количество элементов вектора, вводимое пользователем.
vec	std::vector<double>	Вектор вещественных чисел, инициализируемый случайными значениями.

Таблица 1: Переменные функции и классы используемы при решении задачи

3 Диаграмма:


txt1: введите чтсло элементов в векторе
начало
size: integer
вывод txt1
ввод size
vec: Vector(size)
initVecWithRandomNum(vec)
вывод
vec,
findAbsMax(vec)
moveZeroToEnd(vec)
конец

4 Код:

```
#include <cmath>
#include <chrono>
#include <random>
#include <type_traits>
#include <vector>
#include <iostream>

class Randgen {
    static std::mt19937 helpInitMt() {
        std::random_device rd{};
        std::seed_seq seedSeq {
            static_cast<std::seed_seq::result_type>(
                std::chrono::steady_clock::now().time_since_epoch().count(),
                rd(), rd(), rd(), rd(), rd(), rd(), rd()) };
        return std::mt19937{ seedSeq };
    }
    inline static std::mt19937 state{ helpInitMt() };
public:
    template <typename T>
    using RandomFunc = T(*)(const T&, const T&);

    template <typename T>
    static T get(const T& min, const T& max) {
        if constexpr (std::is_integral<T>::value) {
            return std::uniform_int_distribution<T>(min, max)(state);
        } else if constexpr (std::is_floating_point<T>::value) {
            return std::uniform_real_distribution<T>(min, max)(state);
        } else {
            static_assert(std::is_arithmetic<T>::value, "Unsupported type for Randgen::get
        }
    }
};

template <typename T>
constexpr T abs(T n) {
    return n<0 ? -n : n;
}

template <typename T>
void swap(T& a, T& b) {
    T temp{a};
    a = b;
    b = temp;
}

template <typename T>
std::ostream& operator<<(std::ostream& out, const std::vector<T>& vec) {
    bool isFirst{true};
    for (double el : vec) {
        out << (isFirst?"":", ") /*<< std::setprecision(3)*/ << el;
```

```
        isFirst = false;
    }
    return out;
}

template <typename T>
void initVecWithRandomNum(std::vector<T>& vec, Randgen::RandomFunc<T> func=Randgen::get<T>
    for (double& el : vec) {
        el = trunc(
            func(0., 2.)>1 ? func(-100., 100) : 0
        );
    }
}

template <typename T>
const T& findAbsMax(const std::vector<T>& vec) {
    const T *absMax{&vec[0]};
    for (size_t i{1}; i<vec.size(); ++i)
        if (abs(vec[i]) > abs(*absMax))
            absMax = &vec[i];
    return *absMax;
}

template <typename T>
void moveZerosToTheEnd(std::vector<T>& vec) {
    size_t lnz{ vec.size() - 1};
    for (size_t i = 0; i < lnz; ++i) {
        if (vec[i] == 0) {
            while (lnz > i && vec[lnz] == 0)
                --lnz;
            std::swap(vec[i], vec[lnz]);
        }
    }
}

int main(int32_t, const char**) {
    std::cout << "Enter number of elements in arr: ";
    size_t size;
    std::cin >> size;
    std::vector<double> vec(size);
    initVecWithRandomNum(vec);
    std::cout << vec << '\n' << findAbsMax(vec) << '\n';
    moveZerosToTheEnd(vec);
    std::cout << vec << '\n';
    return EXIT_SUCCESS;
}
```

source code

5 Результат работы программы:

```
[john@arch data]$ ./a.out
Enter number of elements in arr: 8
65, 7, 0, 0, 33, 0, 96, 0
96
65, 7, 96, 33, 0, 0, 0, 0
[john@arch data]$ ./a.out
Enter number of elements in arr: 8
-29, 0, 0, 0, 24, 0, 0, -51
-51
-29, -51, 24, 0, 0, 0, 0, 0
[john@arch data]$ ./a.out
Enter number of elements in arr: 10
0, 0, 0, -40, 0, 0, 0, 0, 0, -46
-46
-46, -40, 0, 0, 0, 0, 0, 0, 0, 0
```

Задание 2

1 Условие:

1. Разработать соответствующую варианту программу с использованием имеющегося описания, отладить ее и перерешать с использованием векторов.
2. Подготовить набор тестов, подтверждающих правильность работы программы.
3. Оформить отчет, включив в него постановку задачи коды программ и результаты их работы.

   Задано описание:

   typedef float Vector[100];*
   Vector x;

   Считая, что все элементы вектора x отличны от *NULL*, описать

   - процедуру del(x), которая в векторе x все ссылки указывающие на повторяющиеся элементы заменяет значением NULL;
   - процедуру Inp(x) – формирования вектора x;
   - процедуру Out(x) – вывода чисел, на которые ссылаются элементы вектора x отличные от NULL.

2 Алгоритм / Мат. модель

Программа использует динамическую структуру данных для работы с векторами указателей. Она инициализирует вектор случайными числами, удаляет повторяющиеся элементы, заменяя их на `nullptr`, и выводит оставшиеся элементы.

1. **Инициализация вектора:** Создаётся вектор `vec1` типа *float*[100]* (размером 100), элементы которого являются указателями на тип *float*.

2. **Формирование вектора:** Вектор `vec1` заполняется случайными числами от -100 до 100 или последовательностью чисел от 0 до 9 с помощью функции `fillOneToNine` или `fillRandom`.

3. **Вывод начального состояния:** Выводится содержимое вектора `vec1` с использованием перегруженного operator«, который выводит только значения, на которые указывают ненулевые указатели.

4. **Удаление повторяющихся элементов:** Функция `del(vec)` заменяет все ссылки на повторяющиеся элементы вектора `vec1` на `nullptr`.

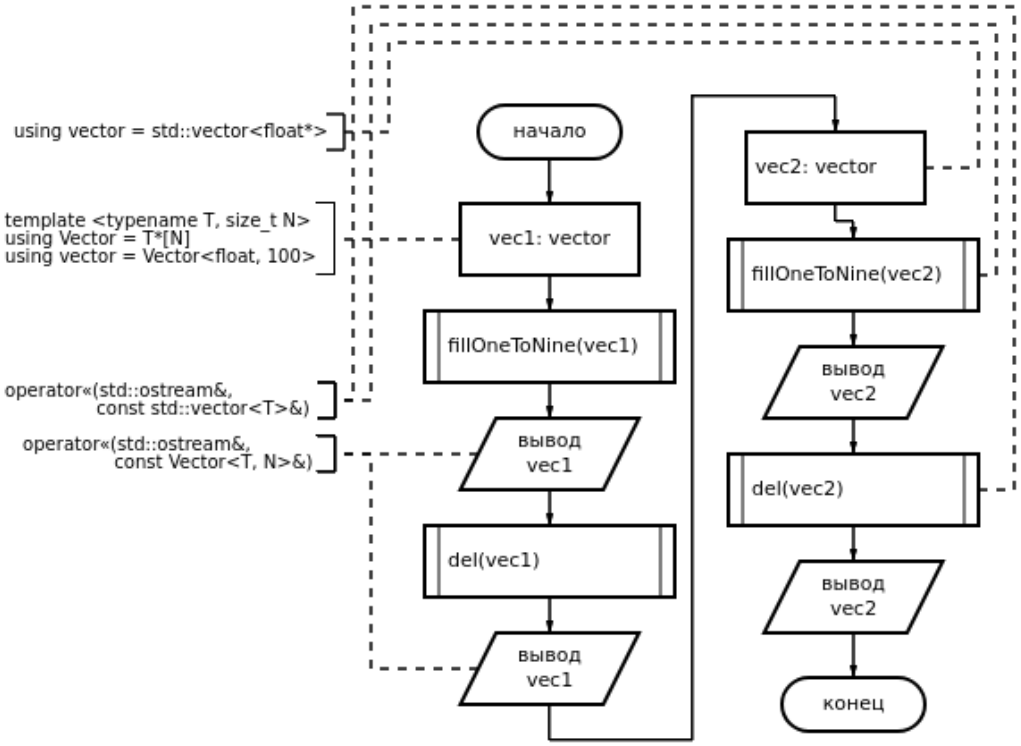
5. **Вывод после обработки:** Выводится содержимое вектора после удаления повторяющихся элементов.

6. **Реализация с помощью std::vector:** Аналогичные действия выполняются с использованием контейнера *std::vector<float*>*

Название	**Тип**	**Описание**
Классы и структуры		
Randgen	Randgen	Генератор случайных чисел с различными распределениями.
Vector	template T*[N]	Шаблон массива указателей фиксированного размера.
Vec100PtrtoFloat	Vector<float, 100>	Шаблон массива указателей на float размера 100.
Переменные-члены Randgen		
state	static std::mt19937	Генератор случайных чисел Mersenne Twister, инициализированный функцией helpInitMt.
Функции-члены Randgen		
helpInitMt()	static std::mt19937	Инициализирует генератор случайных чисел с использованием текущего времени и случайных значений от std::random_device.
get<T>(const T&, const T&)	static T	Генерирует случайное число типа T в указанном диапазоне (целое или вещественное).
Шаблонные функции		
del(Vector<T, N>&)	void	Удаляет дубликаты указателей в массиве фиксированного размера.
del(std::vector<T>&)	void	Удаляет дубликаты указателей в векторе std::vector.
operator«(std::ostream&, const Vector<T, N>&)	std::ostream&	Перегрузка оператора « для вывода массива указателей фиксированного размера.
operator«(std::ostream&, const std::vector<T>&)	std::ostream&	Перегрузка оператора « для вывода std::vector указателей.
operator»(std::istream&, Vector<T, N>&)	std::istream&	Перегрузка оператора » для ввода массива указателей фиксированного размера.
operator»(std::istream&, std::vector<T>&)	std::istream&	Перегрузка оператора » для ввода std::vector указателей.
fillRandom(Vector<T, N>&, Randgen::RandomFunc<T>)	void	Заполняет массив указателей фиксированного размера случайными числами.
fillRandom(std::vector<T>&, Randgen::RandomFunc<T>)	void	Заполняет std::vector указателей случайными числами.
fillOneToNine(Vector<T, N>&)	void	Заполняет массив указателей фиксированного размера числами от 0 до 9 циклично.
fillOneToNine(std::vector<T>&)	void	Заполняет std::vector указателей числами от 0 до 9 циклично.
Переменные main		
vec1	Vec100PtrtoFloat	Массив указателей на float фиксированного размера 100, заполняемый числами от 0 до 9.
vec2	std::vector<float*>	Вектор указателей на float, заполняемый числами от 0 до 9.

Таблица 2: Переменные, функции и классы, используемые в программе

3 Диаграмма:



4 Код:

```
#include <cstdlib>
#include <chrono>
#include <random>
#include <vector>
#include <iostream>

class Randgen {
    static std::mt19937 helpInitMt() {
        std::random_device rd{};
        std::seed_seq seedSeq {
            static_cast<std::seed_seq::result_type>(
                std::chrono::steady_clock::now().time_since_epoch().count(),
                rd(), rd(), rd(), rd(), rd(), rd(), rd()) };
        return std::mt19937{ seedSeq };
    }
    inline static std::mt19937 state{ helpInitMt() };
public:
    template <typename T>
    using RandomFunc = T(*)(const T&, const T&);

    template <typename T>
    static T get(const T& min, const T& max) {
        if constexpr (std::is_integral<T>::value) {
            return std::uniform_int_distribution<T>(min, max)(state);
        } else if constexpr (std::is_floating_point<T>::value) {
            return std::uniform_real_distribution<T>(min, max)(state);
```

```
        } else {
            static_assert(std::is_arithmetic<T>::value, "Unsupported type for Randgen::get
        }
    }
};

template <typename T, size_t N>
using Vector = T*[N];

using Vec100PtrtoFloat = Vector<float, 100>;

// for T*[N]
template <typename T, size_t N>
void del(Vector<T, N>& vec) {
    for (size_t i{0}; i<N; ++i)
        if (vec[i])
            for (size_t j{i+1}; j<N; ++j)
                if (vec[j] && *vec[i] == *vec[j])
                    vec[j] = nullptr;
}

template <typename T, size_t N>
std::ostream& operator<<(std::ostream& out, const Vector<T, N>& vec) {
    for (size_t i{0}; i<N; ++i)
        if (vec[i])
        out << (i>0 ? ", ": "") << *vec[i];
    return out;
}

template <typename T, size_t N>
std::istream& operator>>(std::istream& in, Vector<T, N>& vec) {
    for (size_t i{0}; i<N; ++i) {
        T val;
        in >> val;
        vec[i] = new T{val};
    }
    return in;
}

constexpr auto min{-100};
constexpr auto max{ 100};

template <typename T, size_t N>
void fillRandom(Vector<T, N>& vec, Randgen::RandomFunc<T> func=Randgen::get<T>) {
    for (size_t i{0}; i<N; ++i)
        vec[i] = new T{func(min, max)};
}

template <typename T, size_t N>
void fillOneToNine(Vector<T, N>& vec) {
    for (size_t i{0}; i<100; ++i)
        vec[i] = new float{(float)(i % 10)};
```

```
}

// std::vector
template <typename T>
void del(std::vector<T>& vec) {
    size_t N = vec.size();
    for (size_t i{0}; i<N; ++i)
        if (vec[i])
            for (size_t j{i+1}; j<N; ++j)
                if (vec[j] && *vec[i] == *vec[j])
                    vec[j] = nullptr;
}

template <typename T>
std::ostream& operator<<(std::ostream& out, const std::vector<T>& vec) {
    size_t N = vec.size();
    for (size_t i{0}; i<N; ++i)
        if (vec[i])
        out << (i>0 ? ", ": "") << *vec[i];
    return out;
}

template <typename T>
std::istream& operator>>(std::istream& in, std::vector<T>& vec) {
    size_t N = vec.size();
    for (size_t i{0}; i<N; ++i) {
        T val;
        in >> val;
        vec[i] = new T{val};
    }
    return in;
}

template <typename T>
void fillRandom(std::vector<T>& vec, Randgen::RandomFunc<T> func=Randgen::get<T>) {
    size_t N = vec.size();
    for (size_t i{0}; i<N; ++i)
        vec[i] = new T{func(min, max)};
}

template <typename T>
void fillOneToNine(std::vector<T>& vec) {
    size_t N = vec.size();
    for (size_t i{0}; i<100; ++i)
        vec[i] = new float{(float)(i % 10)};
}


int32_t main(int32_t, const char**) {
    {
        using vector = Vec100PtrtoFloat;
        vector vec1{};
```

```
        fillOneToNine(vec1);
        std::cout << "float*[100] version:\n"
            "before deleting a copy:\n" << vec1 << '\n';
        del(vec1);
        std::cout << "after deleting a copy:\n" << vec1 << '\n';
    }
    {
        using vector = std::vector<float*>;
        vector vec2(100);
        fillOneToNine(vec2);
        std::cout << "std::vector<float*>(100):\n"
            "before deleting a copy:\n" << vec2 << '\n';
        del(vec2);
        std::cout << "after deleting a copy:\n" << vec2 << '\n';
    }
    return EXIT_SUCCESS;
}
```

source code

5 Результат работы программы:

```
[john@arch data]$ ./a.out
float*[100] version:
before deleting a copy:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
after deleting a copy:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
std::vector<float*>(100):
before deleting a copy:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
after deleting a copy:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

Задание 3

В соответствии с вариантом написать и отладить программу, используя структуру данных: a) массив ссылок на строки; b) вектор строк;

1 Условие:

Используя данное представление текста, описать

- функцию find(T, c), определяющую, сколько раз литера c входит в текст T;
- функцию void Inp(T), считывающую из входного файла последовательность литер до первой * (- символ окончания текста) и формирующую из них текст T (последнюю строку, если надо, дополнить пробелами);
- функцию void out(T), печатающую построчно текст T.

a)

2 Алгоритм / Мат. модель

Программа использует динамическую структуру данных для работы с текстом, хранящим строки, и анализирует их на наличие символов. Она читает строки из файла, делит их на части заданной длины, подсчитывает количество вхождений заданного символа и выводит результат. Также освобождает динамическую память, используемую для хранения строк.

1. **Инициализация структуры данных:** Создаётся структур `TextLines`, которая содержит:

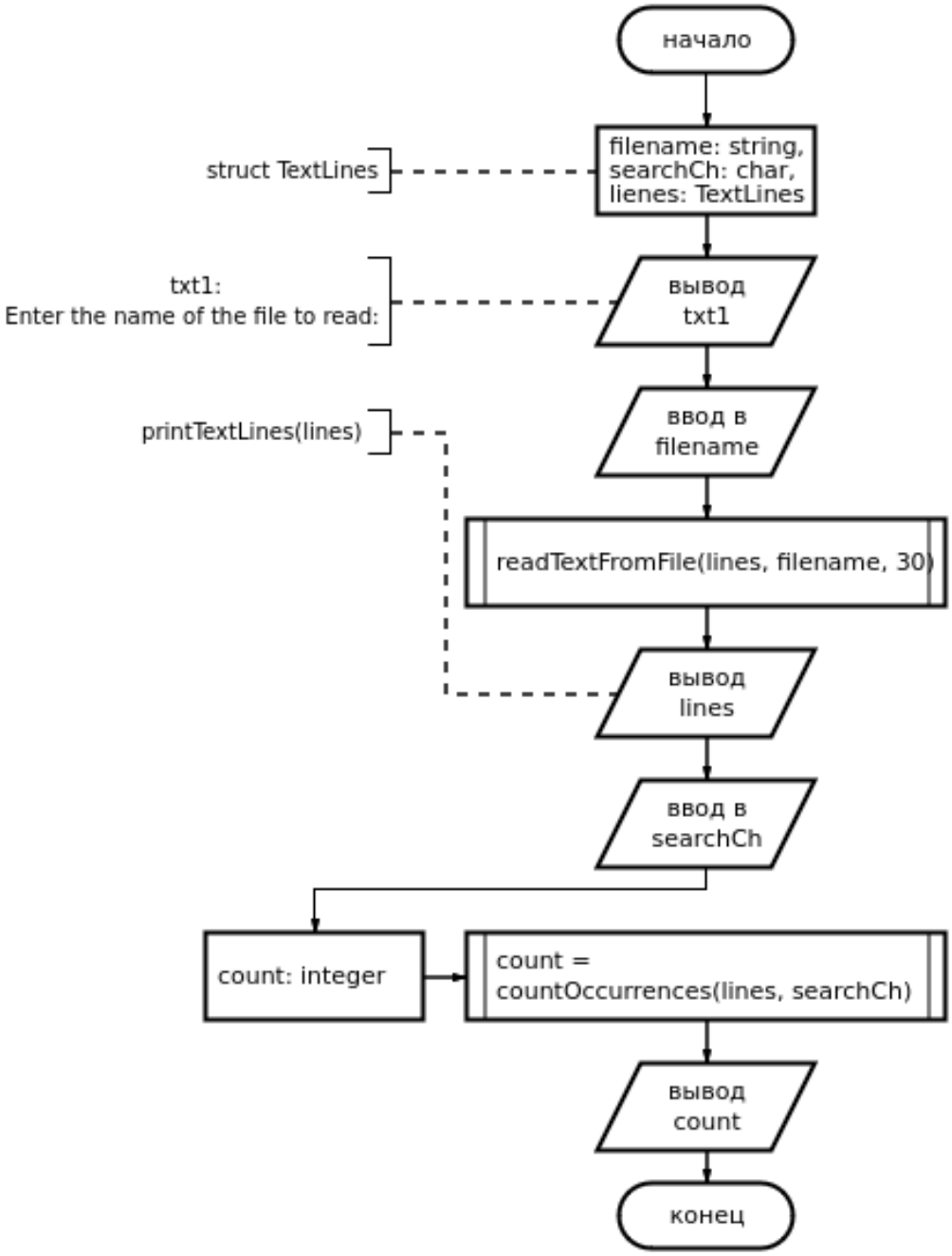
- `lines` — динамический массив указателей на строки типа *char**,
- `numLines` — количество строк в массиве,
- `maxLines` — максимальный размер массива строк.

2. **Чтение текста из файла:** Функция `readTextFromFile` читает содержимое файла и разбивает строки на части длиной не более `MAX_LINE_LENGTH` символов с учетом пробелов. Для каждой части выделяется память, и она добавляется в массив строк.
3. **Перераспределение памяти:** Если количество строк превышает текущий лимит `maxLines`, то выполняется увеличение размера массива в два раза.
4. **Подсчёт вхождений символа:** Функция `countOccurrences` подсчитывает количество вхождений заданного символа `searchChar` в строках, используя два вложенных цикла — по строкам и по символам внутри каждой строки.
5. **Вывод строк:** Функция `printTextLines` выводит каждую строку с её индексом.
6. **Освобождение памяти:** Функция `freeTextLines` освобождает всю выделенную память, связанную с текстовыми строками, и обнуляет указатели на массив строк.
7. **Реализация через командную строку:** Программа получает имя файла для чтения и, если указано, символ для подсчёта его вхождений. Результат выводится в консоль.

Название	**Тип**	**Описание**
Структуры		
TextLines	struct	Структура для хранения текста, содержащая динамический массив строк `char*`, количество строк `numLines` и максимальное количество строк `maxLines`.
Функции		
readTextFromFile()	bool	Читает текст из файла, разбивает его на строки длиной не более `maxLineLength`, и сохраняет их в структуре `TextLines`.
countOccurrences()	size_t	Подсчитывает количество вхождений символа `searchChar` во всех строках, сохранённых в структуре `TextLines`.
freeTextLines()	void	Освобождает память, выделенную для строк в структуре `TextLines`, и обнуляет указатели.
printTextLines()	void	Выводит все строки из структуры `TextLines` на экран с их индексами.
Переменные main		
filename	char[256]	Строка для хранения имени файла, которое будет считано с консоли или из командной строки.
searchChar	char	Символ, количество вхождений которого подсчитывается в тексте.
file	FILE*	Указатель на файл, который открывается для чтения.
lines	TextLines	Структура, в которой сохраняются строки, прочитанные из файла.

Таблица 3: Переменные, функции и структуры, используемые в программе

3 Диаграмма:



4 Код:

```
#include <stddef.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define BUFFER_SIZE 1024
#define MAX_LINE_LENGTH 30
```

```
typedef struct {
    char **lines; // dynamic array of 'char*' - lines
    size_t numLines;
    size_t maxLines;
} TextLines;

void freeTextLines(TextLines *lines);

bool readTextFromFile(TextLines *lines, FILE *file, size_t maxLineLength) {
    if (!lines->lines) {
        lines->numLines = 0;
        lines->maxLines = 10;
        lines->lines = malloc(lines->maxLines * sizeof(char *));
    }
    if (!lines->lines)
        return false;

    char buffer[BUFFER_SIZE];
    while (fgets(buffer, sizeof(buffer), file)) {
        size_t len = strlen(buffer);
        buffer[strcspn(buffer, "\n")] = '\0';

        size_t start = 0;
        while (start < len) {
            size_t end = start + maxLineLength;
            if (end >= len) {
                end = len;
            } else {
                size_t spacePos = end;
                while (spacePos > start && buffer[spacePos] != ' ') {
                    --spacePos;
                }
                if (spacePos > start) {
                    end = spacePos;
                }
            }

            size_t chunkLength = end - start;
            char *lineChunk = malloc(chunkLength + 1);
            if (!lineChunk) {
                freeTextLines(lines);
                return false;
            }
            strncpy(lineChunk, buffer + start, chunkLength);
            lineChunk[chunkLength] = '\0';

            if (lines->numLines >= lines->maxLines) {
                size_t newMaxLines = lines->maxLines * 2;
                char **newLines = realloc(lines->lines, newMaxLines * sizeof(char *));
                if (!newLines) {
                    freeTextLines(lines);
                    return false;
```

```
                }
                lines->lines = newLines;
                lines->maxLines = newMaxLines;
            }

            lines->lines[lines->numLines] = lineChunk;
            ++lines->numLines;

            start = end;
            while (start < len && buffer[start] == ' ') {
                ++start;
            }
        }
    }
    return true;
}

size_t countOccurrences(const TextLines *lines, char searchChar) {
    size_t count = 0;
    for (size_t i = 0; i < lines->numLines; ++i)
        for (size_t j = 0; j < strlen(lines->lines[i]); ++j)
            if (lines->lines[i][j] == searchChar)
                ++count;
    return count;
}

void freeTextLines(TextLines *lines) {
    for (size_t i = 0; i < lines->numLines; ++i)
        free(lines->lines[i]);
    free(lines->lines);
    lines->lines = NULL;
    lines->numLines = lines->maxLines = 0;
}

void printTextLines(const TextLines *lines) {
    for (size_t i = 0; i < lines->numLines; ++i)
        printf("%zu: %s\n", i, lines->lines[i]);
}

int32_t main(int32_t argc, const char **argv) {
    char filename[256];
    char searchChar;
    if (argc >= 2) {
        strcpy(filename, argv[1]);
    } else {
        printf("Enter the name of the file to read: ");
        if (!fgets(filename, sizeof(filename), stdin)) {
            fprintf(stderr, "Error reading file name\n");
            return EXIT_FAILURE;
        }
        filename[strcspn(filename, "\n")] = '\0';
    }
```

```
    FILE *file = fopen(filename, "r");
    if (!file) {
        fprintf(stderr, "Could not open file %s\n", filename);
        return EXIT_FAILURE;
    }
    TextLines lines = { NULL };
    if (readTextFromFile(&lines, file, MAX_LINE_LENGTH)) {
        printTextLines(&lines);
        if (argc == 3) {
            searchChar = argv[2][0];
        } else {
            printf("Enter a character to count occurrences of: ");
            if (scanf(" %c", &searchChar) != 1) {
                fprintf(stderr, "Error reading character\n");
                return EXIT_FAILURE;
            }
        }
        printf("the character '%c' occurs %zu times\n", searchChar,
               countOccurrences(&lines, searchChar));
    } else {
        fprintf(stderr, "Error reading from file\n");
    }
    fclose(file);
    freeTextLines(&lines);
    return EXIT_SUCCESS;
}
```

source code

5 Результат работы программы:

```
[john@arch data]$ touch input.txt
[john@arch data]$ cat > input.txt
Hello
world
[john@arch data]$ ./a.out input.txt l
0: Hello
1: world
the character 'l' occurs 3 times
```

b) вариант с векторами

1 Алгоритм / Мат. модель

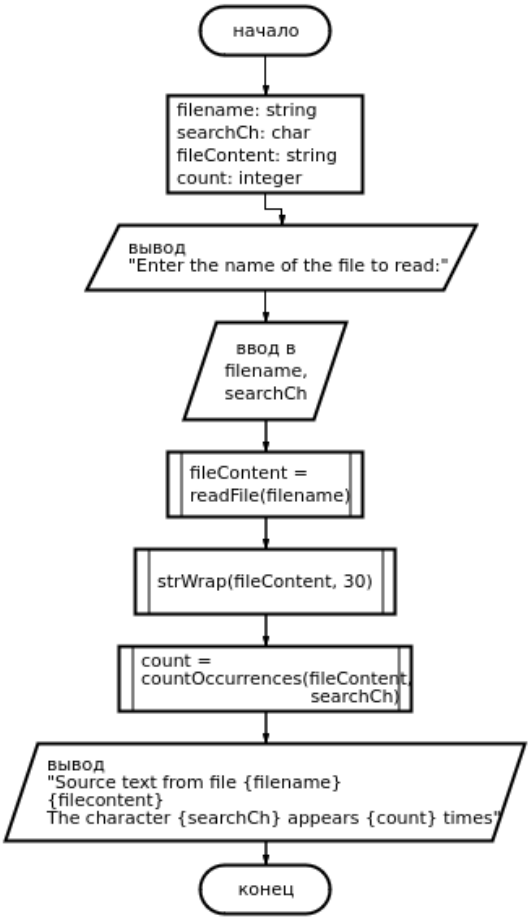
Программа читает содержимое текстового файла, форматирует его строки, заменяя пробелы на новые строки, и подсчитывает количество вхождений определённого символа в текст.

1. **Инициализация переменных:**
   Программа инициализирует несколько переменных
   - `filename` - строка для хранения имени файла, которое вводит пользователь.
   - `searchSymbol` - символ, количество вхождений которого нужно подсчитать, вводится пользователем.
2. **Чтение файла:**
   - Функция `readFile` открывает файл с именем, переданным пользователем в качестве параметра.
   - Если файл не удаётся открыть, выбрасывается исключение с сообщением об ошибке.
   - Если файл открыт успешно, его содержимое считывается и возвращается как строка.
3. **Форматирование текста::**
   - Функция `strWrap` форматирует текст, заменяя пробелы на символы новой строки, если длина строки превышает максимальное значение (`maxStrLen`).
   - Программа перезаписывает текст с разбивкой по строкам, где максимально возможная длина строки — 30 символов.
   - Это достигается путём нахождения последнего пробела в строке, который будет заменён на новую строку, если длина текущей строки превышает заданный лимит.
4. **Подсчёт вхождений символа:**
   - Функция `countOccurrences` подсчитывает количество вхождений символа `searchSymbol` в строке, переданной ей как параметр.
   - Для каждого символа строки, если он совпадает с `searchSymbol`, счётчик увеличивается.
5. **Вывод результатов:**
   - Программа выводит отформатированный текст из файла.
   - Затем выводится количество вхождений символа, введённого пользователем, в этот текст.

Название	Тип	Описание
Типы данных		
exitState	enum	Перечисление, определяющее возможные состояния завершения программы: FAILURE (false) и SUCCESS (true).
Функции		
readFile()	std::string	Читает содержимое файла и возвращает его в виде строки. Выбрасывает исключение в случае ошибки.
strWrap()	void	Форматирует строку, разбивая её на подстроки длиной не более maxStrLen, заменяя пробелы на символы новой строки.
countOccurrences()	size_t	Подсчитывает количество вхождений символа search в строке str.
Переменные main		
filename	std::string	Строка для хранения имени файла, которое вводит пользователь.
searchSymbol	char	Символ для поиска, который вводит пользователь.
fileContent	std::string	Строка для хранения содержимого файла, считанного функцией readFile().
file	std::ifstream	Поток для чтения файла, используемый в функции readFile().

Таблица 4: Переменные, функции и константы, используемые в программе

2 Диаграмма:



3 Код:

```
#include <cassert>
#include <fstream>
#include <sstream>
#include <string>
#include <stdexcept>
#include <iostream>

constexpr bool IS_RELEASE_VERSION{ true };

enum exitState : bool {
    FAILURE = false,
    SUCCESS = true
};

std::string readFile(const char *filename) {
    std::ifstream file(filename);
    if (!file) {
        if constexpr (IS_RELEASE_VERSION)
            assert("Could not open file");
        throw std::runtime_error("Could not open file");
    }
```

```
    std::ostringstream oss;
    oss << file.rdbuf();
    return oss.str();
}

void strWrap(std::string& str, size_t maxStrLen) {
    size_t strLen{ 1 };
    size_t lastWhitespaceIdx{ 0 };
    for (size_t i{ 0 }; i < str.length(); ++i, ++strLen) {
        if (str[i] == ' ')
            lastWhitespaceIdx = i;
        if (strLen > maxStrLen && lastWhitespaceIdx) {
            str[lastWhitespaceIdx] = '\n';
            strLen = i - lastWhitespaceIdx;
        } else if (str[i] == '\n')
            str[i] = ' ';
    }
}

size_t countOccurrences(const std::string& str, char search) {
    size_t count{ 0 };
    for (char ch : str)
        if (ch == search)
            ++count;
    return count;
}

int main() {
    std::string filename;
    std::cout << "Enter the name of the file to read: ";
    std::getline(std::cin, filename);

    char searchSymbol;
    std::cout << "Enter the character to search for: ";
    std::cin >> searchSymbol;

    try {
        std::string fileContent{ readFile(filename.c_str()) };

        strWrap(fileContent, 30);

        std::cout << "Source text from file " << filename << ":\n"
                  << fileContent << '\n'
                  << "The character '" << searchSymbol << "' appears "
                  << countOccurrences(fileContent, searchSymbol) << " times.\n";
    } catch (const std::exception& e) {
        std::cerr << "Error: " << e.what() << std::endl;
        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

source code

4 Результат работы программы:

```
Enter the name of the file to read: data/input.txt
Enter the character to search for: l
Source text from file data/input.txt:
Hello world
The character 'l' appears 3 times.
```