# Лабораторная работа №4

## *Содержание отчета*

Текст набирается 12 кеглем, интервал в тексте 1. Отступ 1,25 см.

Разделы нумеруются по порядку, выделяются из текста жирным шрифтом.

Таблицы и рисунки должны содержать номер и название. Комментарии к рисункам обязательны. Если таблица не содержит комментариев и примечаний, то комментарий к таблице обязателен.

После выполнения лабораторной работы студент должен защитить ее, пояснив процесс обработки данных, схемы алгоритмов и тексты программы, а также ответив на ряд контрольных вопросов.

- Титульный лист с указанием темы четвертой лабораторной работы.

Актуальное оформление титульного листа располагается по адресу: http://guap.ru/guap/standart/titl_main.shtml

- Постановка задачи

Необходимо написать оба полных задания, которые требуется выполнить в четвертой лабораторной работе.

- Формализация задачи, представленная двумя разделами

Каждый раздел посвящен отдельному заданию.

Формализация (от лат. forma — вид, образ) — отображение результатов мышления в точных понятиях и утверждениях. При формализации изучаемым объектам, их свойствам и отношениям ставятся в соответствие некоторые устойчивые, хорошо обозримые и отождествимые конструкции, дающие возможность выявить и зафиксировать существенные стороны объектов.

- Исходный код, представленный двумя разделами

Привести полный исходный код, который полностью выполняет требования и оба задания четвертой лабораторной работы.

- Результаты работы программы, представленные двумя разделами

Привести примеры работы программы для каждого задания с описанием действий пользователя и описанием того, что программа выводит на экран.

- Выводы

Сделать выводы о проделанной работе и изученном материале.

## *Требования к защите*

При оценке преподавателем ЛР будут учитываться следующие показатели:

1. Листинг программы и удобство использования программы. То есть будет оценен стиль программирования, выделение и очищение памяти, обоснованность типов возвращаемых данных. Будет оцениваться меню пользователя - насколько реализован интерфейс для пользователя и позволяет ли он полноценно взаимодействовать с программой.
2. Ответы по листингу программы. Обоснованность написания алгоритма и исходного кода при решении задачи.
3. Ответ на теоретические вопросы.

Для защиты лабораторной работы необходимо продемонстрировать работу программы, ответить на вопросы и предоставить корректно оформленный отчет. Лабораторная работа считается успешно сданной при подписанном отчете преподавателем и студентом, и выложенном отчете в личном кабинете студента на сайте ГУАП и подтвержденным преподавателем.

Максимальное число баллов за четвертую лабораторную работу - 15. Но студент может получить меньшее число баллов, в зависимости от его ответов и представленной программы.

## *Лабораторная работа 4. Шаблоны*

Важно учитывать при реализации:

- Необходимо выполнить разделение на h и cpp файлы для каждого класса. h файлы содержат определение, cpp файлы содержат реализацию. функция main обязана располагаться в отдельном cpp файле.
- Реализовать динамическое выделение памяти и очищение.
- Данные класса обязаны находиться в области доступа private. Должны быть созданы функции для получения значения данных и установки значения. Прямого доступа к данным быть не должно.
- Если требуется реализовать стек/очередь/дек/список, то элемент такой дисциплины обязан быть выполнен в виде класса. В классе элемента данные и ссылка (-и) обязаны располагаться в области private. Обязательно создание функций для извлечения данных и изменения этих данных и ссылок (-и)..
- Обязательно реализовать работу с исключительными ситуациями - генерация и обработку. Если в ЛР студенту не очевидно какую исключительную ситуацию следует обрабатывать, следует обратиться к преподавателю. Обработка исключений должна производиться в обоих заданиях.

- Реализовать пользовательское меню согласно заданию. Обязательно реализовать возможность выбора типа данных, с которыми возможно взаимодействие в каждом задании: int, char, float, double, char*. Не должно быть в программе параметров, которые задаются в main, все, что может задать пользователь должно задаваться с клавиатуры, если в задании не указано иначе (имеется в виду заполнение случайными данными).
- Класс должен содержать конструктор со списком инициализации, конструктор с параметром, деструктор.
- По списку студент определяет свой порядковый номер. Необходимо выполнить оба задания.

## Вариант 1

Задание 1

Написать функцию-шаблон последовательного поиска в массиве по ключу. Функция возвращает индексы всех элементов, найденных в массиве, равных ключу. Размер массива и данные задаются пользователем.

Задание 2

Создать параметризованный класс «множество» и перегрузить операторы != проверка на неравенство множеств, == проверка на равенство множеств, [] для доступа по индексу.

## Вариант 2

Задание 1

Написать функцию-шаблон, вычисляющую среднее арифметическое по значениям в массиве. Размер массива задается пользователем, значения массива генерируются случайным образом (от 0 до 50).

Задание 2

Создать параметризованный стек с перегруженными операторами потокового ввода/вывода, оператор присваивания =, оператор += для добавления в стек, оператор – для извлечения из стека.

## Вариант 3

Задание 1

Написать функцию-шаблон, переставляющую элементы в массиве с шагом, заданным пользователем. Размер и значения массива генерируются случайным образом (до 20).

Задание 2

Создать параметризованный массив с перегруженными операторами [] для доступа по индексу, = для присваивания массивов друг другу, вывода и ввода в поток, == для сравнения массивов.

## Вариант 4

Задание 1

Написать функцию-шаблон двоичного поиска. Двоичный поиск можно использовать только в том случае, если есть массив, все элементы которого упорядочены (отсортированы).

Шаги работы алгоритма:

Шаг 1. Зона поиска (на первом шаге ей является весь массив) делиться на две равные части, путем определения ее среднего элемента;

Шаг 2. Средний элемент сравнивается с искомым (key), результатом этого сравнения будет один из трех случаев:

- key<mid. Крайней правой границей области поиска становится элемент, стоящий перед средним (right ← mid-1).
- key>mid. Крайней левой границей области поиска становится следующий за средним элемент (left ← mid+1).
- key=mid. Значения среднего и искомого элементов совпадают, следовательно элемент найден, работа алгоритма завершается.

Шаг 3. Если для проверки не осталось ни одного элемента, то алгоритм завершается, иначе выполняется переход к пункту 1.

Приведем пример работы алгоритма. Предположим у нас есть отсортированный массив: 1 2 3 4 5 6 7 8 9 0. Пользователь указывает число, которое ему следует найти: 7. Сначала размер массива делится на 2: 10/2 = 5. Проверяем элемент массива, который расположен на этом индексе: [5] = 6. 6!=7. Искомый элемент больше элемента, расположенного посередине, поэтому поиск не стоит проводить в левой половине массива, так как там расположены числа меньше 6. Поиск сместится в правую часть исходного массива, выделяется подмассив:6 7 8 9 0. Размер делится на 2: 5/2=2. [7]=8. 8!=7. Поиск смешается в левую часть подмассива: 6 7 8. Размер делится на 2: 3/2 = 1. [6] = 7. 7==7. Таким образом, найден исходный элемент, его индекс [6].

Задание 2

Создать параметризованную очередь с перегруженным оператором потокового ввода/вывода и перегруженным оператором + для сложения двух очередей, ! для проверки на пустоту и – для вычитания двух очередей.

## Вариант 5

Задание 1

Написать функцию-шаблон для инверсии массива. Размер массива и значения задаются случайным образом (до 30).

Задание 2

Создать параметризованный стек. Начальный размер стека вводится с клавиатуры, заполняется случайными значениями. Перегрузить оператор = для присваивания двух стеков друг другу, + для сложения двух стеков, == для сравнения двух стеков, - для вычитания стеков.

## Вариант 6

Задание 1

Написать функцию-шаблон, вычисляющую среднее значение в массиве. Размер массива и значения задаются случайным образом (до 100).

Задание 2

Создать параметризованный дэк. Начальный размер дэка генерируется случайным образом (до 30), значения также генерируются случайным образом (до 10). Перегрузить оператор =, потокового ввода/вывода, - для извлечения последнего элемента, + для сложения двух дэков, < для сравнения двух дэков.

## Вариант 7

Задание 1

Написать функцию-шаблон, которая ищет разницу между максимальным и минимальным элементом в неупорядоченном массиве. Размер массива и значения задаются случайным образом (от 50 до 100).

Задание 2

Создать параметризированный класс «множество», перегрузить оператор * для пересечения множеств, + для объединения множеств, < для сравнения множеств.

## Вариант 8

Задание 1

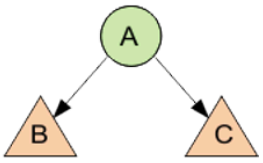
Написать функцию-шаблон, которая выполняет линейный поиск заданного пользователем значения, выводит на экран его индекс. Массив значений не

упорядочен, размер задается случайным образом (от 30 до 100). Значения массива также задаются случайным образом (до 100).

Задание 2

Создать параметризованный класс бинарного дерева. С методами - добавить элемент в дерево, прохождение по дереву в нисходящем и в восходящем порядке. Осуществить поиск по дереву.

Дерево состоит из узлов, которые хранят: данные, указатель на правую ветку, указатель на левую ветку. Основное правило формирования бинарного дерева в С++: если значение узла больше добавляемого – добавляется ветка справа, иначе создается ветка слева. Дерево строится относительно «корня» - пользовательского значения.




Обход дерева в нисходящем порядке: A, B, C. То сначала отображается корень дерева, затем рекурсивный «спуск» по левому поддереву, потом по правому.

Обход дерева в восходящем порядке: B, C, A. То есть сначала рекурсивный «подъем» по левому поддереву, потом по правому и отображение корня дерева.

## Вариант 9

Задание 1

Опишите параметризованную функцию для возведения в квадрат всех данных в массиве. Размер массива изначально неизвестен, пользователь вводит значения массива с клавиатуры.

Задание 2

Создайте шаблон класса массива, в котором есть методы для вычисления суммы и среднего значения хранимых в массиве чисел. Перегрузите потоковый ввод/вывод, оператор + для сложения двух массивов, - для вычитания массивов, != для сравнения массивов, < для сравнения массивов.

## Вариант 10

Задание 1

Написать функцию-шаблон, которая выполняет сортировку выбором как для сортировки по возрастанию, так и по убыванию.

Алгоритм сортировки выбором находит в исходном массиве максимальный или минимальный элементы, в зависимости от того как необходимо сортировать массив,

по возрастанию или по убыванию. Если массив должен быть отсортирован по возрастанию, то из исходного массива необходимо выбирать минимальные элементы. Если же массив необходимо отсортировать по убыванию, то выбирать следует максимальные элементы.

Приведем пример сортировки массива по возрастанию. В исходном массиве находим минимальный элемент, меняем его местами с первым элементом массива. В оставшейся части массива опять ищем минимальный элемент. Найденный минимальный элемент меняем местами со вторым элементом массива и т. д. Таким образом, суть алгоритма сортировки выбором сводится к многократному поиску минимального (максимального) элементов в неотсортированной части массива.

Пример, дан массив 5 2 2 7 8 4 5 0. Находим минимальный элемент в массиве, это 0. Меняем местами минимальный и первый элемент. Результат: 0 2 2 7 8 4 5 5. Находим минимальный элемент в неотсортированной части массива: 2. Менять местами не надо. Следующий элемент снова 2. Затем – 4, меняем местами с 7; результирующий массив: 0 2 2 4 8 7 5 5. И т.д. в итоге получаем отсортированный массив: 0 2 2 4 5 5 7 8.

Задание 2

Создать параметризованный список, перегрузить оператор потокового ввода/вывода, [] для доступа к элементу заданной позиции, + для объединения двух списков.

## Вариант 11

Задание 1

Написать функцию-шаблон, реализующую сортировку слиянием.

Список разделяется на равные или практически равные части, каждая из которых сортируется отдельно. После чего уже упорядоченные части сливаются воедино.

Массив рекурсивно разбивается пополам, и каждая из половин делиться до тех пор, пока размер очередного подмассива не станет равным единице. Далее выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива. После чего из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив; В конце операции слияния, элементы перезаписываются из результирующего массива в исходный.

Приведем пример для сортировки массива чисел: 3 9 6 1 3 6 8 0.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 9 | 6 | 1 | 3 | 6 | 8 | 0 |
| ↓ | | ↓ | | ↓ | | ↓ | |
| 3 | 9 | 1 | 6 | 3 | 6 | 0 | 8 |
| ↓ | | | | ↓ | | | |
| 1 | 3 | 6 | 9 | 0 | 3 | 6 | 8 |
| ↓ | | | | | | | |
| 0 | 1 | 3 | 3 | 6 | 6 | 8 | 9 |

Задание 2

Создать параметризированный класс циклической очереди. Размер очереди вводится с клавиатуры, очередь заполняется случайными числами (от 0 до 50). Перегрузить оператор + для добавления элемента в очередь, ! для проверки очереди на пустоту, - для извлечения элемента очереди.

## Вариант 12

Задание 1

Написать функцию-шаблон, которая выполняет сортировку вставками. Размер массива задается случайным образом (от 10 до 20 элементов). Значения заполняются случайным образом.

В начале сортировки первый элемент массива считается отсортированным, все остальные — не отсортированные. Начиная со второго элемента массива и заканчивая последним, алгоритм вставляет неотсортированный элемент массива в нужную позицию в отсортированной части массива. Таким образом, за один шаг сортировки отсортированная часть массива увеличивается на один элемент, а неотсортированная часть массива уменьшается на один элемент. На каждом шаге сортировки сравнивается текущий элемент со всеми элементами в отсортированной части.

Приведем пример для массива 4 1 4 5 9 0.

| шаг | Отсортированная часть массива | Текущий элемент |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 1 4 | 4 |
| 3 | 1 4 4 | 5 |
| 4 | 1 4 4 5 | 9 |
| 5 | 1 4 4 5 9 | 0 |
| 6 | 0 1 4 4 5 9 | |

Задание 2.

Создать параметризованный массив с перегруженными операторами ввода/вывода в поток, оператором индексирования [] для доступа по индексу, операторами поэлементного сравнения < и >.

## Вариант 13

Задание 1

Написать параметризованную функцию - сортировка методом Шелла. Метод построен на основе метода вставки с минимизацией промежуточных шагов. Общая схема метода состоит в следующем.

Шаг 1. Происходит упорядочивание элементов n/2 пар ($x_i$,$x_{n/2+i}$) для $1 < i < n/2$.

Шаг 2. Упорядочиваются элементы в n/4 группах из четырех элементов ($x_i$,$x_{n/4+i}$,$x_{n/2+i}$,$x_{3n/4+i}$) для $1 < i < n/4$.

Шаг 3. Упорядочиваются элементы уже в n/4 группах из восьми элементов и т.д.

На последнем шаге упорядочиваются элементы сразу во всем массиве x1,x2,...,xn. На каждом шаге для упорядочивания элементов в группах используется метод сортировки вставками.

Пример сортировки Шелла. Исходный массив чисел:

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 8 | 0 | 7 | 4 | 9 | 1 | 6 | 2 |

Шаг 1. 10/2 = 5. Числа расположены на расстоянии 5 друг от друга. Список пар следующий: (5,4), (3,9), (8,1), (0,6),(7,2). Отсортируем внутри пары по возрастанию и расставим в исходном массиве.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 3 | 1 | 0 | 2 | 5 | 9 | 8 | 6 | 7 |

Шаг 2. 5/2=2. Числа расположены на расстоянии 2 друг от друга. Отсортируем внутри пары по возрастанию и расставим в исходном массиве.

Выполняем сортировку последовательно. Пара (4,1):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 4 | 0 | 2 | 5 | 9 | 8 | 6 | 7 |

Пара (3,0):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 4 | 3 | 2 | 5 | 9 | 8 | 6 | 7 |

Пара (4,2):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 9 | 8 | 6 | 7 |

Пара (3,5):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 9 | 8 | 6 | 7 |

Пара (4,9):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 9 | 8 | 6 | 7 |

Пара (5,8):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 9 | 8 | 6 | 7 |

Пара (9,6):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 6 | 8 | 9 | 7 |

Пара (8,7):

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |

Шаг 3. 2/2 = 1. Числа расположены на расстоянии 2 друг от друга. Отсортируем внутри пары по возрастанию и расставим в исходном массиве.

Выполняем сортировку последовательно как на предыдущем шаге. Результат сортировки приведен ниже.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Задание 2

Написать параметризованный класс двусвязный список элементов. Определить операции сравнения != и ==, + для добавления элемента в конец списка.

## Вариант 14

Задание 1

Написать функцию-шаблон бинарного поиска. Если данные, по которым требуется провести поиск, отсортированы, то можно использовать бинарный поиск. Поэтому,

первоначально надо написать функцию сортировки (или использовать ту функцию, которую написали в задании №5.1 по практическим занятиям). При использовании бинарного метода на первом шаге проверяется срединный элемент. Если он больше ключа поиска, то проверяется срединный элемент второй половины массива. Эта процедура повторяется до тех пор, пока не будет найдено совпадение, или до тех пор, пока больше не останется элементов, которые можно было бы проверять.

Задание 2

Создать параметризованный класс односвязного списка. Перегрузить операторы – для извлечения первого элемента из списка, + для добавления элемента в начало списка, < и > для сравнения списков.