ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра информационных систем и технологий

Назаренко П.А.

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Методические указания

по выполнению лабораторных работ

Часть I. Базовые структуры данных и алгоритмы

Самара 2016

УДК 004.65+ 004.43
ББК 32.973
Н 19

Рекомендовано к изданию методическим советом ПГУТИ,
протокол № 27, от 11.05.2016 г.

Назаренко, П. А.
Н **Алгоритмы и структуры данных:** методические указания по выполнению лабораторных работ. Часть I. Базовые структуры данных и алгоритмы / П. А. Назаренко – Самара : ПГУТИ, 2016. – 32 с.

Методические указания по выполнению лабораторных работ для дисциплины «Алгоритмы и структуры данных» содержат семь лабораторных работ, ориентированных на использование языков Си/Си++.

1. Массивы автоматические, статические и динамические.
2. Структурированные типы данных и массивы.
3. Файлы.
4. Связные списки.
5. Сортировка одномерных массивов.
6. Поиск данных в массивах и связных списках.
7. Двоичные деревья поиска.

Методические указания разработаны в соответствии с Федеральным государственным образовательным стандартом высшего профессионального образования по направлению подготовки 02.03.03 – «Математическое обеспечение и администрирование информационных систем» и предназначено для студентов третьего курса факультета информационных систем и технологий, а также для студентов других специальностей, изучающих и использующих структуры данных и алгоритмы их обработки.

Содержание

Введение

Лабораторные работы по курсу «Алгоритмы и структуры данных» предназначены для практического закрепления знаний, а также для получения студентами навыков практической работы с изучаемыми структурами данных, применения алгоритмов обработки этих структур, самостоятельной разработки необходимых алгоритмов. Также при выполнении лабораторных работ решаются задачи оценки эффективности алгоритмов и оптимальности применения конкретных структур данных в каждом отдельном случае.

При выполнении лабораторных работ студенты на практике осваивают критерии оценки эффективности и сложности алгоритмов, учатся выбирать структуры данных и алгоритмы их обработки для решения поставленных задач, оценивать эффективность и сложность алгоритмов, разрабатывать программное обеспечение на основе выбранных алгоритмов и структур данных, овладевают навыками применения структур данных и алгоритмов в разрабатываемом или используемом программном обеспечении.

Лабораторный практикум по дисциплине «Алгоритмы и структуры данных» разделён на две части. В первую часть вошли лабораторные работы по основным структурам данных и базовым алгоритмам, изучаемым в лекционном курсе. Тематика работ практически соответствует основным темам лекционного курса. Во вторую части практикума предполагается включить как лабораторные работы по вспомогательным разделам дисциплины, так и работы исследовательского характера, предназначенные в том числе для магистрантов.

Для успешного выполнения лабораторных работ студенты должны изучить лекционный материал, рекомендованную и дополнительную литературу по темам лабораторных занятий, выбрать вариант задания, уяснить пункты задания, выполнить подготовку к работе по этим пунктам, разработать для каждой работы программу (одну или несколько) в соответствии с заданием к работе, выполнить отладку программы, получить с помощью программы результаты, подтверждающие (или не подтверждающие) положения теории и сделать выводы по отдельным пунктам задания и по всей работе.

В исходном виде лабораторный практикум ориентирован на применение языков программирования Си или Си++ (предпочтительным является Си++), но в равной степени может использоваться и язык Паскаль, тем более, что в учебном пособии [1] приведены примеры процедур на этом языке. По согласованию с ведущими лабораторные занятия преподавателями могут использоваться и другие языки программирования, например, Java, C#, Python, при условии, что студенты владеют этими языками в достаточной степени, а также, что семантика языка позволяет применять программы на нём для проверки теоретических положений дисциплины. Например, в языках Java, C# и Python отсутствуют явные указатели, но тем не менее, существует возможность создания динамических связных структур данных. В языке Python существует отдельный тип данных «список», которые не соответствует изучаемому в курсе понятию связного списка. Подобные обстоятельства должны учитываться студентами и преподавателями при выборе языка. **Не следует** использовать библиотечные функции, например, функцию быстрой сортировки, а также готовые шаблонные структуры данных в объектно-ориентированных языках программирования.

Контрольные вопросы к каждой работе ориентированы на материал, изложенный в учебном пособии **[**1**]**, и при подготовке к защите лабораторных работ в абсолютном большинстве случае нет необходимости прибегать к дополнительным источникам информации (хотя это и не возбраняется).

Защита лабораторных работ предполагается в виде предъявления отчёта по работе с её результатами и сделанными выводами. Студентам задаются контрольные вопросы, от двух до пяти, в зависимости от продемонстрированного уровня знаний, а также могут быть заданы дополнительные вопросы по результатам работы или её программному обеспечению.

1. Массивы автоматические, статические и динамические

1.1. Цель работы

Цель работы заключается в изучении:

– свойств одномерных и многомерных массивов как линейных и прямоугольных структур данных на примере одного из основных языков программирования;

– свойств автоматических, статических и динамических массивов;

– принципов использования массивов, в том числе динамических;

– принципов использования указателей, в том числе для работы с массивами.

1.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить темы «Типы данных», «Массивы», «Указатели», «Операции с динамической памятью».

Разработать программу в соответствии с заданием к лабораторной работе.

1.3. Варианты заданий

Вариант задания выбирается по номеру студента в списке подгруппы. Варианты заданий приведены в таблице 1.

Таблица 1

№	Переменные	№	Переменные
1	1. Одномерный символьный массив; 2. Указатель на тип **char**; 3. Статический одномерный массив целых чисел; 4. Указатель на массив	9	1. Одномерный массив плавающих чисел; 2. Указатель на тип **float**; 3. Статический одномерный массив беззнаковых целых чисел; 4. Указатель на массив

	целых чисел; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.		**unsigned int**; 5. Трехмерный массив символов; 6. Указатель на двумерный массив символов.
2	1. Одномерный массив целых чисел; 2. Указатель на тип **int**; 3. Статический одномерный символьный массив; 4. Указатель на массив символов; 5. Трехмерный массив плавающих чисел; 6. Указатель на двумерный массив плавающих чисел.	10	1. Одномерный символьный массив; 2. Указатель на тип **char**; 3. Статический одномерный массив типа **double**; 4. Указатель на массив типа **double**; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.
3	1. Одномерный символьный массив; 2. Указатель на тип **char**; 3. Статический одномерный массив длинных чисел; 4. Указатель на массив длинных целых чисел; 5. Трехмерный массив плавающих чисел; 6. Указатель на двумерный массив плавающих чисел.	11	1. Одномерный массив целых чисел; 2. Указатель на тип **int**; 3. Статический одномерный массив плавающих чисел; 4. Указатель на массив плавающих чисел; 5. Трехмерный массив символов; 6. Указатель на двумерный массив символов.
4	1. Одномерный массив длинных целых чисел; 2. Указатель на тип **long int**; 3. Статический одномер-	12	1. Одномерный массив типа **double**; 2. Указатель на тип **double**; 3. Статический одномерный массив беззнаковых це-

	ный массив символов; 4. Указатель на массив символов; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.		лых чисел; 4. Указатель на массив **unsigned int**; 5. Трехмерный массив символов; 6. Указатель на двумерный массив символов.
5	1. Одномерный массив типа **float**; 2. Указатель на тип **float**; 3. Статический одномерный массив символов; 4. Указатель на массив символов; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.	13	1. Одномерный массив беззнаковых целых чисел; 2. Указатель на тип **unsigned int**; 3. Статический одномерный массив символов; 4. Указатель на массив символов; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.
6	1. Одномерный символьный массив; 2. Указатель на тип **char**; 3. Статический одномерный массив коротких целых чисел; 4. Указатель на массив коротких целых чисел; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.	14	1. Одномерный массив типа **float**; 2. Указатель на тип **float**; 3. Статический одномерный массив символов; 4. Указатель на массив символов; 5. Трехмерный массив **unsigned int**; 6. Указатель на двумерный массив беззнаковых целых чисел.
7	1. Одномерный массив коротких целых чисел; 2. Указатель на тип **short int**;	15	1. Одномерный символьный массив; 2. Указатель на тип **char**; 3. Статический одномерный

	3. Статический одномерный массив символов; 4. Указатель на массив символов; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.		массив плавающих чисел; 4. Указатель на массив плавающих чисел; 5. Трехмерный массив целых чисел; 6. Указатель на двумерный массив целых чисел.
8	1. Одномерный массив типа **double**; 2. Указатель на тип **double**; 3. Статический одномерный массив целых чисел; 4. Указатель на массив целых чисел; 5. Трехмерный массив символов; 6. Указатель на двумерный массив символов.	16	1. Одномерный массив целых чисел; 2. Указатель на тип **int**; 3. Статический одномерный массив типа **double**; 4. Указатель на массив типа **double**; 5. Трехмерный массив символов; 6. Указатель на двумерный массив символов.

1.4. Порядок выполнения работы

1. Определить в функции main() переменные и массивы по таблице 1 в соответствии с вариантом (см. таблицу 1).

2. В функции main() выполнить следующие действия:

   2.1. Проверить содержимое массива №1 (с помощью цикла for и операции вывода cout<<).

   2.2. Ввести данные в массив №1 (с помощью цикла for и операции ввода cin>>).

   2.3. Еще раз проверить содержимое этого массива, сделать выводы.

   2.4. Присвоить указателю №2 адрес массива №1, вывести на экран адреса массива и указателя и содержимое указателя. Сделать выводы.

2.5. Повторить пункт 3 для указателя, содержащего адрес массива. Сделать выводы.
2.6. Повторить пункты 1 – 3 для статического массива №3. Сделать выводы.
2.7. Используя имеющийся указатель №2, создать динамический массив и повторить для него пункты 1 – 3. Сделать выводы.
2.8. Удалить динамический массив. Используя указатель №4, создать двумерный динамический массив и повторить для него пункты 2, 3. Сделать выводы. Удалить двумерный динамический массив.
2.9. Вывести на экран любой из элементов трехмерного массива №5, используя операцию индексации.
2.10. Повторить пункт 9, используя имя массива как указатель и операцию доступа по указателю.
2.11. Присвоить указателю №6 на двумерный массив адрес трехмерного массива №5. Повторить для этого указателя пункт 10. Сделать выводы.

Текст программы следует дополнить сообщениями о вводе и выводе данных. Такие сообщения облегчат контроль выполнения программы. По мере выполнения работы отлаженные фрагменты текста можно закомментировать, предварительно занеся в отчет полученные результаты.

Сохранить файл с текстом программы для следующей работы.

1.5. Содержание отчёта

Отчёт по лабораторной работе №1 и последующим работам должен однозначно отражать **личное участие** студента в получении результатов работы. **Допускается** как **рукописное**, так и машинное исполнение отчёта.

В отчёте должны быть указаны фамилия и инициалы студента, номер группы, номер и название работы. Должен быть приведён текст заранее составленной программы с необходимыми изменениями (если они были сделаны; **допускаются** рукописные исправления).

Отчёт должен содержать результаты работы составленной и отлаженной программы и сделанные студентом выводы. **Отсутствие выводов может послужить причиной недопуска работы к защите.**

1.6. Контрольные вопросы

1. По каким признакам классифицируются структуры данных?
2. К какой группе структур данных относятся автоматические массивы?
3. Что означает понятие «тип данных»?
4. Какую информацию можно извлечь из типа данных?
5. К какой группе структур данных относятся статические массивы?
6. К какой группе структур данных относятся динамические массивы?
7. Что такое указатели?
8. Какие операции можно выполнять над указателями?
9. В чем заключается связь между указателями и массивами?
10. Какие операции обязательны при работе с динамическими массивами?
11. Перечислите свойства динамических массивов.
12. В чем заключается отличие между автоматическими и статическими массивами?
13. Можно ли изменить размер динамического массива при исполнении программы? Если да, то как это сделать?
14. Какое требование нужно соблюдать при присваивании адреса массива указателю?
15. Какие ограничения накладываются на определение многомерных динамических массивов?
16. В чем заключается отличие между именем массива и указателем?

2. Структурированные типы данных и массивы

2.1. Цель работы

Получить навыки работы со структурированными типами, их объектами и компонентами, а также массивами объектов, в том числе динамическими. Научиться использовать объекты структурных типов в функциях, изучить способы передачи объектов в функции. Закрепить знания о динамических массивах.

2.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить темы «Массивы», «Указатели», «Структуры», «Операции с динамической памятью».

Разработать программу в соответствии с заданием к лабораторной работе.

2.3. Варианты заданий

Вариант задания выбирается по номеру студента в списке подгруппы. Варианты заданий приведены в таблице 2.

Таблица 2

№	Переменные	№	Переменные
1	1. Переменная длинного целого типа; 2. Указатель на целый тип; 3. Переменная целого типа.	9	1. Переменная плавающего типа; 2. Указатель на тип **float**; 3. Переменная беззнакового целого типа.
2	1. Переменная целого типа; 2. Указатель на тип **int**; 3. Переменная плавающего типа.	10	1. Переменная типа **double**; 2. Указатель на тип **double**; 3. Переменная целого типа.
3	1. Переменная беззнако-	11	1. Переменная целого типа;

	вого длинного целого типа; 2. Указатель на длинный целый тип; 3. Переменная плаваю-щего типа.		2. Переменная плавающего типа; 3. Указатель на плавающий тип.
4	1. Переменная длинного целого типа; 2. Указатель на тип **long int**; 3. Переменная целого типа.	12	1. Переменная беззнакового целого типа; 2. Указатель на тип **unsigned int**; 3. Переменная символьного типа.
5	1. Переменная типа **float**; 2. Указатель на тип **float**; 3. Переменная символь-ного типа.	13	1. Переменная беззнакового целого типа; 2. Указатель на тип **unsigned int**; 3. Переменная целого типа.
6	1. Переменная символь-ного типа; 2. Указатель на короткий целый тип; 3. Переменная короткого целого типа.	14	1. Переменная типа **float**; 2. Указатель на тип **float**; 3. Переменная типа **unsigned int**.
7	1. Переменная короткого целого типа; 2. Указатель на тип **short int**; 3. Переменная символь-ного типа.	15	1. Переменная плавающего типа; 2. Указатель на плавающий тип; 3. Переменная целого типа.
8	1. Переменная типа **double**; 2. Указатель на тип **double**; 3. Переменная символь-ного типа.	16	1. Указатель на тип **int**; 2. Переменная типа **double**; 3. Переменная символьного типа.

2.4. Порядок выполнения работы

1. Определить структурный тип, содержащий следующие поля:

– символьный массив, используемый для хранения строки, например, с именем студента,

– указатель на тип **char** – для организации динамического массива, хранящего строку, например, с фамилией студента.

Остальные поля выбрать по вариантам, приведенным в таблице 2.

Использовать одну из переменных для хранения некоторого идентификатора (номера); указатель на не символьный тип – для организации динамического массива целых или плавающих чисел; другую переменную – для хранения размера этого массива.

Дополнить структурный тип любыми полями по своему выбору.

2. Определить функции:

– инициализации структуры;

– заполнения массива чисел;

– вывода на экран массива чисел;

– ввода информации в строки имени и фамилии и другие поля;

– вывода на экран всех полей структуры, кроме массива чисел;

– функцию освобождения динамической памяти.

У половины функций, по выбору студента, одним из аргументов должен быть указатель на структурный тип, у другой половины – ссылка на структурный тип.

3. Определить функцию main(), в которой создать:

– объект ранее определенного структурного типа

– указатель на этот структурный тип.

С помощью указателя создать динамический массив объектов структурного типа из 3-х – 4-х элементов.

Для объекта последовательно вызывать функции инициализации, заполнения массива чисел, ввода данных в остальные поля, показа массива, показа полей.

Для каждого элемента массива структур выполнить в цикле (for) функции инициализации, заполнения массива и ввода данных.

Вывести на экран содержимое полей каждого элемента массива структур в цикле (for) с помощью соответствующих функций.

В конце функции main() вызвать функцию освобождения памяти для объекта структурного типа и в цикле для каждого элемента массива объектов.

Удалить динамический массив.

Ход выполнения программы контролировать по выводимым на экран сообщениям.

Сохранить файл с текстом программы для следующей работы.

2.5. Дополнительное задание

Создать в функции main() блок, в котором определить локальный объект структурного типа. Ввести данные в поля локального объекта.

Попытаться вывести на экран содержимое полей локального объекта за пределами блока. Сделать выводы.

2.6. Контрольные вопросы

1. Что представляет собой структурный тип данных?
2. Данные каких типов могут входить в состав структур?
3. Данные каких типов не могут входить в состав структур?
4. По каким признакам классифицируются структуры данных?
5. К какой группе структур данных относятся автоматические массивы?
6. Что означает понятие «тип данных»?
7. Какую информацию можно извлечь из типа данных?
8. К какой группе структур данных относятся статические массивы?
9. К какой группе структур данных относятся динамические массивы?

10. Что такое указатели?

11. Какие операции можно выполнять над указателями?

12. В чем заключается связь между указателями и массивами?

13. Какие операции обязательны при работе с динамическими массивами?

14. Каковы свойства динамических массивов?

15. Как обеспечить связь между массивами или структурами и функциями?

16. Можно ли использовать одно и то же имя для глобальной и локальной переменных?

17. Можно ли использовать для разных функций аргумент с одним и тем же именем?

3. Файлы

3.1 Цель работы

Получить навыки работы с файлами как структурами данных. Научиться работать с текстовыми и двоичными файлами. Научиться обрабатывать потоки данных с блоками переменного размера.

3.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить темы «Массивы», «Структуры», «Файлы».

Разработать программу в соответствии с заданием к лабораторной работе.

3.3. Порядок выполнения работы

1. Определить функции записи и считывания структуры для двоичного и текстового файлов. У всех функций одним из аргументов должна быть ссылка на структурный тип, у функций работы с двоичными файлами еще одним аргументом должен быть указатель на тип FILE.

Использовать структурный тип и функции ввода и вывода на экран данных, определенные в программе к лабораторной работе №2.

2. Определить глобальные указатели на тип FILE для функций работы с двоичными файлами.

3. Определить функцию main(), в которой создать:

– указатели на тип FILE для функций работы с текстовыми файлами;

– указатель на структурный тип для организации динамического массива структур;

– первый динамический массив структур небольшого размера (3 – 4 элемента) – для ввода данных и их сохранения в файлах;

– второй динамический массив структур такого же размера – для считывания данных из текстового файла;

– третий динамический массив структур – для считывания данных из двоичного файла.

4. Ввести данные в элементы первого динамического массива с помощью функций, разработанных в лабораторной работе №2.

5. Сохранить содержимое элементов массива структур в текстовом и двоичном файлах.

6. Считать содержимое текстового файла в элементы второго массива и вывести на экран. Сравнить содержимое первого и второго массивов.

7. Считать содержимое двоичного файла в элементы третьего массива и вывести на экран. Сравнить содержимое первого и третьего массивов.

8. Просмотреть содержимое двоичного и текстового файлов с помощью любого тестового редактора или программы-просмотрщика (для просмотра тестового файла можно использовать «Блокнот» Windows, для просмотра двоичного файла рекомендуется Total Commander или аналогичная программа). Установить соответствие между элементами данных в первом массиве и текстовом файле. Попытаться сделать то же самое для двоичного файла.

Сохранить файл с тестом программы для последующих работ.

3.4. Контрольные вопросы

1. Что такое файл?
2. К какой группе структур данных относятся файлы?
3. Какие действия необходимо выполнить для работы с файлом?
4. Различаются ли файлы по типам?
5. Как в программах устанавливается связь с файлами?
6. Какие способы организации связи с файлами вам известны?
7. Какие операции можно выполнять над файлами?
8. Как открыть файл для записи?
9. Как открыть файл для считывания?
10. Какая функция позволяет узнать длину файла?

11. Как проверить, можно ли произвести запись в выбранный файл?

12. Можно ли считать данные из произвольного места в файле? Если да, то как это сделать?

13. Можно ли перемещаться по файлу? Если да, то с помощью какой функции?

14. Чем отличается запись действительных чисел в текстовый и двоичный файлы?

15. Как обеспечить связь между файлами и функциями?

4. Связные списки

4.1. Цель работы

Научиться работать со связными списками, линейными и кольцевыми, односвязными и двусвязными. Изучить операции добавления и удаления звеньев, просмотра списка.

4.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить тему «Связные списки».

Разработать программу в соответствии с заданием к лабораторной работе.

4.3. Порядок выполнения работы

1. Для организации односвязного списка определить структурный тип, содержащий указатель на свой тип и поля, использованные в работе №2.

2. Определить функции вставки нового звена в односвязный линейный список, удаления звена из списка, просмотра содержимого списка.

3. В функции main() создать заглавное звено списка и проверить работу функций вставки, удаления и просмотра. В функции просмотра предусмотреть вывод адресов каждого звена списка. Сделать выводы.

4. Преобразовать односвязный список в двусвязный. Внести соответствующие изменения в функции работы со списком. Проверить работу функций.

5. Преобразовать линейный список в кольцевой. Внести необходимые изменения в функции работы со списком. Проверить работу функций. Сделать выводы.

Сохранить файл с тестом программы для последующих работ.

4.4. Контрольные вопросы

1. Что представляют собой связные списки?

2. К каким классификационным группам структур данных относятся связные списки?

3. Какие существуют разновидности списков?

4. В чем состоит отличие несвязного списка от массива?

5. В чем состоит отличие связного списка от массива?

6. В чем состоит отличие линейного списка от кольцевого?

7. В чем заключаются недостатки односвязного списка?

8. В чем состоит отличие односвязного списка от двусвязного?

9. Какие операции применяются для связных списков?

10. В чем отличие считывания информации из списка от считывания из очереди или стека?

11. Каковы особенности операций вставки и удаления для связных списков?

12. В чем отличие операции вставки в двусвязный список от вставки в односвязный список?

13. В чем отличие операции удаления из двусвязного списка от удаления из односвязного списка?

14. В чем заключаются особенности работы с кольцевыми списками?

15. Что означает понятие «динамическая структура данных»?

16. Какой тип должно иметь звено связного списка? Почему?

17. Что обязательно должно содержать звено связного списка?

18. В чем состоит отличие звена двусвязного списка от звена односвязного списка?

19. В чем состоит отличие простого связного списка от стека, организованного в виде связного списка?

5. Сортировка одномерных массивов

5.1. Цель работы

Изучить основные алгоритмы сортировки массивов и освоить их на практике. Проверить работу алгоритмов на различных наборах данных.

5.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить тему «Сортировка».

Разработать программу в соответствии с заданием к лабораторной работе.

5.3. Варианты заданий

Вариант задания выбирается по номеру студента в списке подгруппы. Варианты заданий приведены в таблице 3.

Таблица 3

№	Алгоритмы сортировки	№	Алгоритмы сортировки
1	1. Пузырьковая. 2. Быстрая.	9	1. Быстрая. 2. Отбором.
2	1. Отбором. 2. Шелла.	10	1. Шелла. 2. Вставками.
3	1. Вставками. 2. Быстрая.	11	1. Пузырьковая. 2. Отбором.
4	1. Пузырьковая. 2. Шелла.	12	1. Быстрая. 2. Пузырьковая.
5	1. Отбором. 2. Быстрая.	13	1. Шелла. 2. Отбором.
6	1. Вставками. 2. Шелла.	14	1. Вставками. 2. Пузырьковая.
7	1. Отбором. 2. Пузырьковая.	15	1. Быстрая. 2. Вставками.
8	1. Пузырьковая. 2. Вставками.	16	1. Шелла; 2. Пузырьковая.

5.4. Порядок выполнения работы

1. Определить массив, элементы которого будут упорядочиваться. Тип массива выбрать по таблице №1 – массив №1.

2. Разработать функцию сортировки массива методами, выбранными по таблице 3 в соответствии с вариантом.

3. Любым способом заполнить элементы массива значениями.

4. Выполнить сортировку массива первым алгоритмом и проконтролировать ее результат. Проверить все варианты исходного заполнения массива: случайным образом, отсортированного в обратном порядке, отсортированного в требуемом порядке. Убедиться в правильности сортировки во всех случаях. Сделать выводы.

5. Повторить пункты 3 и 4 для второго алгоритма сортировки.

Сохранить файл с тестом программы для последующих работ.

5.5. Контрольные вопросы

1. Что представляет собой операция сортировки?
2. Сколько существует групп алгоритмов сортировки?
3. Сколько существует алгоритмов сортировки?
4. По каким признакам характеризуются алгоритмы сортировки?
5. Что нужно учитывать при выборе алгоритма сортировки?
6. Какой алгоритм сортировки считается самым простым?
7. Какой алгоритм сортировки считается самым эффективным?
8. Что означает понятие «скорость сортировки»?
9. В чем заключается метод пузырьковой сортировки?
10. В чем заключается метод сортировки отбором?
11. В чем заключается метод сортировки вставками?
12. В чем заключается метод сортировки разделением?
13. В чем заключается метод быстрой сортировки?
14. В чем заключается метод сортировки Шелла?
15. В чем заключается метод сортировки Бэтчера?

16. Как зависит скорость сортировки от размера структуры данных для разных алгоритмов?

17. Почему метод Бэтчера называется параллельным?

18. Какой из алгоритмов сортировки лучше всех остальных подходит для связных списков?

19. Можно ли применить метод Шелла для сортировки связного списка?

20. Можно ли применить быструю сортировку для упорядочения связного списка?

21. Оправданно ли с точки зрения эффективности применение сортировки Шелла для связного списка?

22. Оправданно ли с точки зрения эффективности применение быстрой сортировки для связного списка?

6. Поиск данных в массивах и связных списках

6.1. Цель работы

Изучить основные алгоритмы поиска данных в массивах и связных списках и освоить их на практике.

6.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить темы «Поиск», «Массивы», «Списки».

Разработать программу в соответствии с заданием к лабораторной работе.

6.3. Порядок выполнения работы

1. Определить массив и список, в которых будет выполняться поиск. Использовать массив, созданный в работе №5, и любой из списков, созданных в работе №4.

2. Разработать функции линейного поиска в массиве, двоичного поиска в массиве и поиска в списке. Для обеспечения двоичного поиска в массиве использовать одну из функций сортировки массива, созданных в работе №5.

3. Ввести информацию в массив и список.

4. Выполнить линейный поиск в массиве.

5. Упорядочить элементы массива функцией сортировки и выполнить двоичный поиск.

6. Выполнить поиск в связном списке.

Для всех случаев проверить варианты успешного и неуспешного поиска. Сделать выводы.

Сохранить файл с тестом программы для последующих работ.

6.4. Контрольные вопросы

1. Что представляет собой операция поиска?
2. Что называется ключом поиска?
3. Какие известны методы поиска?
4. Какой алгоритм поиска является наиболее эффективным?
5. Какое требование предъявляется к структуре данных, в которой выполняется двоичный поиск?
6. Чем отличается поиск в массиве от поиска в списке?
7. Чем отличаются процедуры поиска в односвязном и двусвязном списках?
8. В чем заключается метод линейного поиска?
9. В чем заключается метод двоичного поиска?
10. В чем заключается поиск в списке?
11. Почему двоичный поиск получил такое название?
12. Какие известны варианты двоичного поиска?
13. Пригоден ли двоичный метод для поиска данных в неупорядоченной структуре?
14. Какой из методов поиска данных в массиве является более универсальным?
15. Существуют ли какие-нибудь недостатки у линейного поиска? Если да, то какие?

7. Двоичные деревья поиска

7.1. Цель работы

Изучить древовидные структуры данных, в том числе двоичные деревья поиска. Получить практические навыки работы с деревьями. Научится использовать операции с деревьями.

7.2. Подготовка к работе

По рекомендованной литературе и конспекту лекций повторить темы «Древовидные структуры данных», «Двоичные деревья поиска», «Операции с деревьями».

Разработать программу в соответствии с заданием к лабораторной работе.

7.3. Порядок выполнения работы

1. Для организации двоичного поиска дерева определить соответствующий структурный тип, содержащий поля, использованные в работе №2. Для хранения ключа использовать любое не символьное поле.

2. Разработать функции:

– вставки узла в дерево,

– удаления узла из дерева,

– прохождения (просмотра) дерева в восходящем, нисходящем, последовательном порядке,

– создания идеально сбалансированного дерева с заданным числом узлов,

– поиска в дереве.

3. Создать двоичное дерево поиска с помощью функции вставки узла.

4. Выполнить просмотр дерева в восходящем, нисходящем и последовательном порядке с помощью соответствующих функций.

5. Удалить несколько узлов из дерева, проверяя состояние дерева после каждого удаления с помощью просмотра в нисходящем порядке.

6. Создать вырожденное дерево путём ввода в него упорядоченных данных. Выполнить просмотр этого дерева в восходящем, нисходящем и последовательном порядке. Зафиксировать результаты.

7. Создать идеально сбалансированное дерево и выполнить его просмотр в нисходящем и последовательном порядках. Сравнить результаты просмотров с результатами пункта 6. Сделать выводы.

8. Дополнить идеально сбалансированное дерево несколькими узлами и проверить его состояние, просматривая дерево в нисходящем порядке.

9. Выполнить поиск информации в дереве. Проверить варианты успешного и неуспешного поиска.

7.4. Контрольные вопросы

1. Что представляют собой древовидные структуры данных?
2. Какие существуют виды деревьев?
3. Что представляет собой двоичное дерево?
4. Что представляет собой двоичное дерево поиска?
5. Чем отличается двоичное дерево от двусвязного списка?
6. Что означает термин «вырожденное дерево»?
7. В каком порядке должны вводиться данные, чтобы получилось вырожденное дерево?
8. Чем вырожденное дерево отличается от односвязного списка?
9. Что означает термин «идеально сбалансированное дерево»?
10. В каком порядке должны вводиться данные, чтобы получилось сбалансированное дерево?
11. В чем заключается особенность дерева как структуры данных?
12. Каковы области применения древовидных структур данных?
13. Процедуры какого характера наиболее эффективны при работе с деревьями?
14. В чем заключается вставка узла в дерево?
15. В чем заключается удаление узла из дерева?

16. Каковы особенности удаления элемента из древовидной структуры данных?

17. Какого рода процедуры часто оказываются эффективными для деревьев?

18. В чем заключается поиск в дереве?

19. Что такое «прохождение дерева»? Какие возможны варианты прохождения дерева?

20. Что такое высота дерева?

21. Как сохранить сбалансированность дерева при вставке и удалении узлов?

22. На каких структурах данных могут строится деревья?

Рекомендуемая литература

1. Назаренко, П. А. Алгоритмы и структуры данных [Текст] / П. А. Назаренко – Самара : Издательство учебной и научной литературы ПГУТИ, 2015. – 196 с.

2. Алексеев, В. Е. Графы и алгоритмы. Структуры данных. Модели вычислений [Текст] : учебник для вузов / В. Е. Алексеев, В. А. Таланов. – М. : Интернет Ун-т Информ. Технологий : БИНОМ, 2006. – 320 с.

3. Вирт, Н. Алгоритмы и структуры данных с примерами на Паскале [Текст] / Н. Вирт. – 2-е изд. – СПб. : Невский диалект, 2005. – 352 с.

4. Кнут, Д. Э. Искусство программирования для ЭВМ. Т. 3 : Сортировка и поиск [Текст] : пер. с англ. / Д. Э. Кнут. – М. : Мир, 1978. – 846 с.

5. Кубенский, А. А. Структуры и алгоритмы обработки данных: объектно-ориентированный подход и реализация на C++ [Текст] : учеб. пособие / А. А. Кубенский. – СПб. : БХВ-Петербург, 2004. – 464 с.

6. Седжвик, Р. Фундаментальные алгоритмы на C++. Ч. 1 – 4: Анализ. Структуры данных. Сортировка. Поиск [Текст] : пер. с англ. / Р. Седжвик. – Киев : ДиаСофт, 2002. – 688 с.

7. Топп, У. Структуры данных в C++ [Текст] : пер. с англ. / У. Топп, У. Форд. – М. : Бином, 2000. – 815 с.

8. Алгоритмы: построение и анализ [Текст] : пер. с англ. / Т. Х. Кормен [и др.]. – 2-е изд. – М. : Издательский дом «Вильямс», 2005. – 1296 с.

9. Шилдт, Д. Теория и практика C++ [Текст] / Г. Шилдт. – СПб. : BHV – Санкт-Петербург, 1996. – 416 с.

10. Алгоритмы, методы, исходники [Электронный ресурс] / Илья Кантор, 2015. – Режим доступа: http://algolist.manual.ru/, свободный. – Загл. с экрана.

11. Язык программирования C++. Динамические структуры данных [Электронный ресурс] / Курган. гос. ун-т. Каф. информ. технологий. – Режим доступа: http://it.kgsu.ru/C_DIN/, свободный. – Загл. с экрана.