МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Государственное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Учебно – методическое пособие по курсу «Алгоритмы и структуры данных» по выполнению практических работ и самостоятельной работы для студентов ВУЗа

Томск
2018

Пособие составлено в соответствии с тематикой практических работ и самостоятельной работы по дисциплине «Алгоритмы и структуры данных». Пособие содержит темы и содержание практических работ, методические указания к их проведению.
Для преподавателей, аспирантов, студентов и магистрантов.

СОСТАВИТЕЛЬ: Е.А. Шельмина

СОДЕРЖАНИЕ

Раздел 1. Практические работы

Практическая работа №1
Алгоритмизация

Цель: изучить основные понятия алгоритмизации.

Теоретические сведения

Алгоритм — заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.

Алгоритмизация – процесс разработки алгоритма (плана действий) для решения задачи.

Исполнитель алгоритма — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

- среда;
- элементарные действия;
- система команд;
- отказы.

Среда (или обстановка) — это "место обитания" исполнителя.

Система команд. Каждый исполнитель может выполнять команды только из некоторого строго заданного списка — системы команд исполнителя. Для каждой команды должны быть заданы условия применимости (в каких состояниях среды может быть выполнена команда) и описаны результаты выполнения команды.

После вызова команды исполнитель совершает соответствующее элементарное действие.

Отказы исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды.

Основные свойства алгоритмов

Понятность для исполнителя — т.е. исполнитель алгоритма должен знать, как его выполнять.

Дискретность (прерывность, раздельность) — т.е. алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

Определенность — т.е. каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

Результативность (или конечность). Это свойство состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов.

Массовость. Это означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными.

Формы записи алгоритмов

На практике наиболее распространены следующие формы представления алгоритмов:

- словесная (записи на естественном языке);
- графическая (изображения из графических символов);
- псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);

- программная (тексты на языках программирования).

Базовые алгоритмические структуры

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных базовых (т.е. основных) элементов. Естественно, что при таком подходе к алгоритмам изучение основных принципов их конструирования должно начинаться с изучения этих базовых элементов.

Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: *следование, ветвление, цикл.*

Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.

Линейная алгоритмическая структура - описание последовательности действий, которые выполняются однократно и в заданном порядке. Это такие алгоритмы, в которых все действия совершаются одно за другим, независимо ни от чего. Такие алгоритмы называются линейными, характерная для них форма организации действий – последовательное выполнение.

Ветвление (развилка) - такая форма организации действий, при которой в зависимости от выполнения или невыполнения конкретного условия, совершается либо одна, либо другая последовательность действий.

Циклический алгоритм. При построении алгоритмов решения задач очень часто возникает ситуация, когда требуется многократно повторить ту или иную часть алгоритма. Многократно повторяемые при некоторых условиях части алгоритма называются циклами, а вычислительные процессы содержащие циклы называют циклическими.

Циклы бывают двух основных видов:

циклы по количеству повторений (к моменту выполнения цикла заранее известно сколько раз его нужно выполнить);

циклы по условию (когда циклические действия продолжаются до тех пор, пока не будет выполнено какое-либо условие).

Задания

Задание 1. Найти наибольший общий делитель чисел 551 и 116 методом Евклида.

Задание 2. Разработать алгоритм перевода десятичного числа 123 в двоичную, пятеричную и шестнадцатеричную системы счисления и осуществить проверку.

Практическая работа №2
Структуры данных

Цель: получить навыки работы со структурами данных в языке Си.

Теоретические сведения

Данные – это представление фактов, понятий, инструкций, идей или какой-либо другой информации в формализованном виде, приемлемом для обработки, интерпретации, общения или передачи как человеком, так и техническими средствами, при помощи некоторых процессов или алгоритмов.

Данные – непременный атрибут любой программы. Ими могут быть отдельные биты, последовательности независимых битов, числа в разных формах представления (с фиксированной или плавающей точкой, обычной или удвоенной точностью и т.д.), байты и группы независимых байтов, представляющие символы в различных системах кодирования, массивы чисел, информация хранимая в памяти вычислительной машины в форме связанных списков, а также информация на устройствах внешней памяти, организованная в виде отдельных файлов и систем взаимосвязанных файлов.

Структуру данных можно определить двумя способами. Во-первых, структура данных – это логическая или математическая модель организации данных. Фактически, структура данных может рассматриваться как представление именно этих данных в памяти ЭВМ

(физическая структура), и является общим свойством любого информационного объекта, с которым имеет дело какая-либо программа.

Во-вторых, структура данных – это реализация логического понятия данных, объект (больший или меньший) в программе и в памяти ЭВМ. Это может быть отдельная переменная, массив или более сложный программный объект, например, список, дерево и т.п.

Структуры данных можно классифицировать по нескольким различным признакам. По уровню сложности структуры данных разделяются на: простые структуры – обычные переменные или константы стандартных для языков программирования типов, а также динамические переменные этих же типов; наборы однотипных данных – массивы одномерные (или векторы), двумерные (матрицы) и многомерные; составные структуры, отличные от массивов – записи и объекты классов и им подобные структуры; динамические структуры с внутренними связями – связные списки, деревья, графы.

С точки зрения архитектуры можно выделить: линейные структуры – одномерные массивы (или векторы), линейные списки, линейные очереди, стеки; прямоугольные структуры – двумерные (матрицы) и многомерные массивы; кольцевые структуры – кольцевые списки, кольцевые очереди, некоторые реализации графов; ветвящиеся структуры – деревья различных видов, некоторые реализации графов; сетевые структуры – графы.

По способу создания структуры данных можно разделить на обычные – переменные стандартных типов, обычные (т.е. не динамические) массивы, записи и т.п.; динамические (создаваемые и разрушаемые с помощью специальных операций или процедур динамического выделения и освобождения памяти) – динамические массивы, динамические переменные, связные списки, деревья.

Вычислительный процесс на ЭВМ реализуется с помощью программы и данных, которые программа использует или формирует. Но и сама программа представляет собой совокупность данных, и поэтому можно считать, что данные описывают любую информацию, с которой может работать ЭВМ. Все данные в общем случае характеризуются рядом признаков, или атрибутов, в том числе своим значением, смысл которого различен для разных типов данных.

Тип некоторых данных определяется множеством значений этих данных и набором операций, которые можно выполнять над этими значениями в соответствии с их известными свойствами.

К простейшим стандартным типам относятся целые числа, логические значения, символы, вещественные числа.

Задание. Разработать алгоритм и написать программу на языке Си для вычисления корней уравнения $ax^2 + bx + c = 0$ при заданных коэффициентах a,b,c.

Практическая работа №3
Линейные статические структуры данных

Цель: научиться работать с массивами и строками.

Для выполнения лабораторной работы необходимо ознакомиться со следующими разделами дисциплины:

Массивы;
Указатели;
Строки;
Записи;
Перечисления;
Объединения.

Задание 1. Разработать алгоритм на языке Си для вычисления определителя матрицы A

$$A=\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

Задание 2. Дано предложение, в котором имеются буквы с и т. Определить, какая из них встречается позже (при просмотре слова слева направо). Если таких букв несколько, то должны учитываться последние из них.

Практическая работа №4
Алгоритмы обработки данных. Сортировка

Цель: изучить алгоритмы сортировки.

Теоретические сведения

Под сортировкой обычно понимают процесс перестановки объектов заданного множества в определенном порядке.

Цель сортировки - облегчение последующего поиска элементов в отсортированном множестве. В этом смысле элементы сортировки присутствуют почти во всех задачах.

Из всех задач программирования сортировка, возможно, имеет самый богатый выбор алгоритмов решения – несколько десятков видов. Рассмотрим некоторые из них.

Сортировка простой выборкой. Порядок алгоритма простой выборки - $O(N^2)$. Количество пересылок - N.

Алгоритм сортировки простой выборки на языке Си можно записать следующим образом:

```
void SortSelect(int[] a, int[] b)
{ int i, j, m;
  int N = a.Length;
  bool[] c = new bool[N];  //состояние эл-тов вход. множества
  for (i = 0; i < N; i++)
  c[i] = true;              // сброс отметок
  for (i = 0; i < N; i++)
  {        //поиск 1-го невыбранного элемента во входном множестве
    j = 0;
    while (!c[j])
      j++;
    m = j;                  // поиск минимального элемент а
    for (j = 1; j < N; j++)
      if (c[j] && (a[j] < a[m]))
        m = j;
    b[i] = a[m];                  // запись в выходное множество
    c[m] = false;                 // во входное множество - "пусто"
  }
}
```

Обменная сортировка простой выборкой. Алгоритм сортировки простой выборкой, однако, редко применяется в том варианте, в каком он описан выше. Гораздо чаще применяется его, так называемый, обменный вариант. При обменной сортировке выборкой входное и выходное множество располагаются в одной и той же области памяти; выходное - в начале области, входное - в оставшейся ее части. В исходном состоянии входное множество занимает всю область, а выходное множество - пустое. По мере выполнения сортировки входное множество сужается, а выходное - расширяется.

Алгоритм обменной сортировки простой выборкой на языке СИ:

```
void SortSelect2(int[] a)
{ int x, i, j, m;
  int N = a.Length;
  for (i = 0; i < N - 1; i++)
  { // перебор элементов выходного множества}
    // входное множество - [i:N]; выходное - [1:i-1]
    m = i;
    for (j = i + 1; j < N; j++)
      // поиск минимума во входном множестве
      if (a[j] < a[m])
        m = j;
    // обмен 1-го элемента вх. множества с минимальным
    if (i != m)
    {
      x = a[i];
      a[i] = a[m];
      a[m] = x;
    }
  }
}
```

Пузырьковая сортировка. Входное множество просматривается, при этом попарно сравниваются соседние элементы множества. Если порядок их следования не соответствует заданному критерию упорядоченности, то элементы меняются местами. В результате одного такого просмотра при сортировке по возрастанию элемент с самым большим значением ключа переместится («всплывет») на последнее место в множестве. При следующем проходе на свое место «всплывет» второй по величине ключа элемент и т.д. Для постановки на свои места N элементов следует сделать N-1 проходов. Выходное множество, таким образом, формируется в конце сортируемой последовательности, при каждом следующем проходе его объем увеличивается на 1, а объем входного множества уменьшается на 1.

Порядок пузырьковой сортировки - $O(N^2)$. Среднее число сравнений - $N*(N-1)/2$ и таково же среднее число перестановок, что значительно хуже, чем для обменной сортировки простым выбором. Однако, то обстоятельство, что здесь всегда сравниваются и перемещаются только соседние элементы, делает пузырьковую сортировку удобной для обработки связных списков. Перестановка в связных списках также получается более экономной.

Шейкер–сортировка. Одна из модификаций пузырьковой сортировки носит название шейкер-сортировки. Суть ее состоит в том, что направления просмотров чередуются: за просмотром от начала к концу следует просмотр от конца к началу входного множества. При просмотре в прямом направлении запись с самым большим ключом ставится на свое место в последовательности, при просмотре в обратном направлении - запись с самым маленьким. Этот алгоритм весьма эффективен для задач восстановления упорядоченности, когда исходная последовательность уже была упорядочена, но подверглась не очень значительным изменениям. Упорядоченность в последовательности с одиночным изменением будет гарантированно восстановлена всего за два прохода.

Пример шейкер-сортировки на языке Си:

```
void SortSheiker(int[] mas)
{ int beg;  //индекс начала сортируемой части
  int end;  //индекс конца сортируемой части
  int j, x;
  beg = 1;
```

```
  end = mas.Length;
  do {   //прямой проход
    x = beg; // признак перестановок (позиция для верхней границы end
    for (j = beg; j < end; j++) //перебор несортированной части
      if (mas[j - 1] > mas[j]) // сравнение соседних элементов
      { // перестановка
        x = mas[j - 1];
        mas[j - 1] = mas[j];
        mas[j] = x;
        x = j; // запоминание позиции
      }
    end = x ; //корректировка верхней границы
    //обратный проход
    x = end;// признак перестановок (позиция для нижней границы beg)
    for (j = end-1; j >= beg; j--) //перебор несортиров-й части
      if (mas[j - 1] > mas[j]) // сравнение соседних элементов
      { // перестановка
        x = mas[j - 1];
        mas[j - 1] = mas[j];
        mas[j] = x;
        x = j; // запоминание позиции
      }
    beg = j + 1; //корректировка нижней границы
  } while (beg < end); // пока несортированная часть не пустая
}
```

Сортировка Шелла. Это еще одна модификация пузырьковой сортировки. Суть ее состоит в том, что здесь выполняется сравнение ключей, отстоящих один от другого на некотором расстоянии *d*. Исходный размер *d* обычно выбирается соизмеримым с половиной общего размера сортируемой последовательности. Выполняется пузырьковая сортировка с интервалом сравнения *d*. Затем величина *d* уменьшается вдвое и вновь выполняется пузырьковая сортировка, далее *d* уменьшается еще вдвое и т.д. Последняя пузырьковая сортировка выполняется при $d = 1$. Качественный порядок сортировки Шелла остается $O(N^2)$, среднее же число сравнений, определенное эмпирическим путем - $\log_2(N)^2 * N$. Ускорение достигается за счет того, что выявленные «не на месте» элементы при $d > 1$, быстрее «всплывают» на свои места.

Задания

Задание 1. В массиве A(n) отсортировать элементы массива, стоящие на нечетных местах, в порядке возрастания тремя различными алгоритмами сортировки.

Задание 2. Оценить порядок алгоритмов сортировки из задания 1.

Практическая работа №5
Алгоритмы обработки данных. Поиск

Цель: изучить алгоритмы поиска и получить навыки их реализации.

Для выполнения практической работы необходимо ознакомиться со следующими алгоритмами поиска:

последовательный поиск

двоичный поиск

поиск с использованием чисел Фибоначчи

Задание. Найти в матрице A(nxn) заданный элемент тремя различными алгоритмами.

Практическая работа №6
Файлы

Цель: получить навыки работы с файлами.

Для выполнения практической работы необходимо ознакомиться со следующими разделами дисциплины:

Файлы;

Операции с данными на внешних носителях: внешний поиск, внешняя сортировка;

Сортировка прямым слиянием;

Сортировка естественным слиянием;

Сбалансированное многопутевое слияние.

Задание. Дан файл f, элементы которого являются действительными числами. Найти сумму наибольшего и наименьшего из значений элементов. Результат вывести в файл g.

Практическая работа №7
Основные алгоритмы обработки данных

Цель: изучить получисленные, комбинаторные и рекурсивные алгоритмы.

Для выполнения практической работы необходимо ознакомиться со следующими разделами дисциплины:

Получисленные алгоритмы;

Комбинаторные алгоритмы;

Рекурсивные алгоритмы.

Задание. Реализовать на языке Си получисленный, комбинаторный и рекурсивный алгоритм.

Раздел 2. Самостоятельная работа

2.1. Проработка лекционного материала.

2.2. Подготовка к практическим занятиям согласно разделу 1.

Список литературы

1. Пермякова, Н. В. Информатика и программирование: Учебное пособие [Электронный ресурс] / Н. В. Пермякова — Томск: ТУСУР, 2016. — 188 с. — Режим доступа: https://edu.tusur.ru/publications/7678 дата обращения: 16.06.2018

2. Перемитина, Т. О. Математическая логика и теория алгоритмов: Учебное пособие [Электронный ресурс] / Т. О. Перемитина. — Томск: ТУСУР, 2016. — 132 с. — Режим доступа: https://edu.tusur.ru/publications/5949 , дата обращения: 16.06.2018