МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Статическое кодирование и декодирование текстового файла методами Хаффмана и Фано-Шеннона — текущий контроль.

Студент гр. 9304 _____ Краев Д.В.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Краев Д.В.

Группа 9304

Тема работы : Статическое кодирование и декодирование текстового файла методами Хаффмана и Фано-Шеннона — текущий контроль.

Содержание пояснительной записки:
«Содержание», «Постановка задачи», «Описание алгоритмов», «Выполнение работы», «Примеры работы программы», «Введение», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 22 страниц.

Дата выдачи задания: 23.11.2020

Дата сдачи реферата: 24.12.2020

Дата защиты реферата: 24.12.2020

Студент _____ Краев Д.В.

Преподаватель _____ Филатов А.Ю.

АННОТАЦИЯ

В данной работе разработана программа, в которой реализованы статические методы кодирования Хаффмана и Шеннона-Фано. Программа генерирует задания на кодирование строки с помощью методов Хаффмана или Шеннона-Фано, далее создает 2 файла: файл, в который записывается текст заданий и файл, в который записываются ответы на эти задания. Для указания файлов для заданий и ответов, а также для установления количества заданий был реализован Command line interface.

СОДЕРЖАНИЕ

	Введение	5
1.	Постановка задчи	6
2.	Описание алгоритмов	7
2.1.	Алгоритм статического кодирования Хаффмана	7
2.2	Алгоритм статического кодирования Шеннона-Фано	7
3.	Выполнение работы	9
3.1	Структуры и классы	9
3.2	Функции	9
3.3	Command line interface	10
4.	Примеры работы программы	11
	Заключение	16
	Список использованных источников	17
	Приложение А. Исходный код	18

ВВЕДЕНИЕ

Цель работы: изучить кодирование статическими методами Хаффмана и Шеннона-Фано. Создать программу для генерации заданий с ответами к ним для проведения текущего контроля среди студентов.

1. ПОСТАНОВКА ЗАДАЧИ

Вариант 2

Статическое кодирование и декодирование текстового файла методами Хаффмана и Фано-Шеннона — **текущий контроль**

«Текущий контроль» - создание программы для генерации заданий с ответами к ним для проведения текущего контроля среди студентов. Задания и ответы должны выводиться к ним для проведения текущего контроля среди студентов. Задания и ответы должны выводиться в файл в удобной форме: тексты заданий должны быть готовы для передачи студентам, проходящим ТК; все задания должны касаться конкретных экземпляров структуры данных(т.е. не должны быть вопросами по теории); ответы должны позволять удобную проверку правильности выполнения заданий.

2. ОПИСАНИЕ АЛГОРИТМОВ

2.1 Алгоритм статического кодирования Хаффмана

Алгоритм статического кодирования Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (Н-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.
2. Выбираются два свободных узла дерева с наименьшими весами.
3. Создается их родитель с весом, равным их суммарному весу.
4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.
5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.
6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

2.2 Алгоритм статического кодирования Шеннона-Фано

Символы первичного алфавита выписывают по убыванию вероятностей.

1. Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.
2. В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
3. Полученные части рекурсивно делятся, и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Когда размер подалфавита становится равен нулю или единице, то дальнейшего удлинения префиксного кода для соответствующих ему символов первичного

алфавита не происходит, таким образом, алгоритм присваивает различным символам префиксные коды разной длины. На шаге деления алфавита существует неоднозначность, так как разность суммарных вероятностей может быть одинакова для двух вариантов разделения (учитывая, что все символы первичного алфавита имеют вероятность больше нуля).

3. ВЫПОЛНЕНИЕ РАБОТЫ

3.1. Структуры и классы

Code

Структура содержит 2 поля типа std::string: первое поле хранит сам название символа, второе поле хранит его код.

Node

Структура является узлом бинарного дерева, поэтому содержит указатели 2 поля, которые указывает на левого и правого потомка. Структура так же содержит название символа и его количество повторений в заданном тексте.

Args

Структура была создана для реализации CLI. Поэтому содержит информацию об аргументах, подаваемых программе. Структура содержит 3 поля: поле tasks содержит название файла с заданиями, поле answers — название файла с ответами, поле number — количество заданий. По умолчанию answers = «answers.txt», tasks = «tasks.txt», number = 5.

3.2 Функции

Для реализации алгоритма цикличной сортировки были написаны несколько функций.

void comp(const std::shared_ptr<Node> &a, const std::shared_ptr<Node> &b)

Данная функция является компоратором для сортировки функцией std::sort. Функция принимает две ссылки на умные указатели на Node и возвращает 1, если a больше b, 0 в ином случае.

count0letters(const std::string text)

Функция принимает на вход текст и считает количество каждого символа в тексте. Возвращает вектор, состоящий из структур Code, содержащую имя символа и его количество в тексте.

createTree1(std::vector<std::shared_ptr<Node>> &elems)

Функция создает бинарное дерево из символов текста по методу Хаффмана. Возвращает умный указатель на корень дерева.

createTree2(std::vector<std::shared_ptr<Node>> &elems)

Функция создает бинарное дерево из символов текста по методу Шеннона-Фано. Возвращает умный указатель на корень дерева.

setCodes(const std::shared_ptr<Node> &nde, std::vector<Code>& codes, std::string code = "")

Функция принимает умный указатель на корень бинарного дерева, полученного предыдущей функцией и ссылку на вектор, в который будут записаны символы и их коды. Функция рекурсивно высчитывает коды символов .

writeCodes(std::vector<Code> codes, int num)

Функция принимает на вход вектор кодов символов и номер задания, создает строку кодов символов и возвращает ее.

genWord(int size)

Функция принимает на вход размер строки, генерирует строку, состоящую из случайных символов заданного размера, и возвращает ее.

main

Сначала в функции main происходит работа CLI, задаются файлы для заданий и ответом, а также задается число заданий. Далее в цикле происходит генерация строки, символы которой нужно закодировать, подсчет количества символов, случайный выбор метода(Хаффман или Шеннон-Фано), кодирование символов заданным методом и вывод заданий и ответов в файлы.

3.3 Command line interface

Для реализации программы был создан CLI. С помощью него можно совершить 4 команды:

- t «название файла с заданиями» — установить название файла с заданиями.
- a «название файла с ответами» — установить название файла с ответами.
- n «число» — установить количество заданий.
- h — вызвать справку.

4. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Программу можно запустить с помощью файла genTasks, его можно получить, запустив команду make, находясь в папке с файлом Makefile.
Если не задать аргументы программе, они будут выбраны по умолчанию.

4.1 Пример 1

Команда: ./genTasks

Файл tasks.txt:

1. Закодируйте символы заданной строки по статическому методу Хаффмана.
Слово: cozzequmzennrwltokqx

2. Закодируйте символы заданной строки по статическому методу Шеннона-Фано.
Слово: vfdzurjdyahobjvtdhsi

3. Закодируйте символы заданной строки по статическому методу Хаффмана.
Слово: hzstuidkfdilgjhzsmau

4. Закодируйте символы заданной строки по статическому методу Шеннона-Фано.
Слово: oxejqisksmsthnopsavv

5. Закодируйте символы заданной строки по статическому методу Хаффмана.
Слово: idtpdldfhytgffwnxjhm

Файл answers.txt:

1.
x - 0000
w - 0001
q - 001
r - 0100
m - 0101

u - 0110

t - 0111

e - 100

c - 1010

l - 10110

k - 10111

z - 110

o - 1110

n - 1111

2.

d - 000

h - 001

v - 010

j - 0110

a - 0111

z - 1000

y - 1001

s - 1010

u - 10110

t - 10111

b - 1100

f - 1101

r - 1110

i - 11110

o - 11111

3.

h - 000

d - 001

s - 010

i - 011

f - 1000

a - 1001

j - 1010

g - 1011

z - 1100

u - 1101

l - 11100

k - 11101

t - 11110

m - 11111

4.

s - 00

v - 010

o - 0110

e - 01110

a - 01111

x - 1000

t - 1001

n - 1010

q - 10110

p - 10111

h - 1100

i - 1101

m - 1110

j - 11110

k - 11111

5.

n - 0000

m - 0001

w - 0010

p - 0011

i - 0100

g - 0101

l - 0110

j - 0111

h - 100

f - 101

d - 110

y - 11100

x - 11101

t — 1111

2.Пример 2

Команда: ./getTasks -n 1

Файл tasks.txt:

1. Закодируйте символы заданной строки по статическому методу Хаффмана.

Слово: nxivmgiemaqumadnqpwh

Файл answers.txt:

1.

x - 0000

w - 0001

q - 001

p - 0100

h - 0101

v - 0110

u - 0111

a - 100

d - 1010

g - 10110

e - 10111

m - 110

n - 1110

i - 1111

3.Пример 3

Команда:./genTasks -h

Вывод:

a - set a file for asnwers

t - set a file for tasks

h - display usage

n - set a number of tasks

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены алгоритмы статического кодирования Хаффмана и Шеннона-Фано. Написана программа на языке C++ с Command Line Interface, позволяющая генерировать задания для проведения текущего контроля.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм Шеннона-Фано

https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD%D0%B0_%E2%80%94_%D0%A4%D0%B0%D0%BD%D0%BE

2. Код Хаффмана

https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Файл main.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <iterator>
#include <algorithm>
#include <memory>
#include <fstream>
#include <getopt.h>
#include <ctime>
#include <cstdlib>

struct Node{
	std::string str;
	int count;
	std::shared_ptr<Node> left = nullptr;
	std::shared_ptr<Node> right = nullptr;
};

struct Code{
	std::string c;
	std::string code;
};

bool comp(const std::shared_ptr<Node> &a, const std::shared_ptr<Node> &b){
	return a->count > b->count;
}

std::vector<std::shared_ptr<Node>> count0letters(const std::string text){
	std::vector<std::shared_ptr<Node>> elems1(256);

	for(int i=0;i<elems1.size();i++){
		elems1[i] = std::make_shared<Node>();
		elems1[i]->str = (char)i;
	}
	for(std::string::const_iterator it = text.begin();it != text.end();it++){
			elems1[*it]->count++;
	}

	std::vector<std::shared_ptr<Node>> elems2;
	for(int i=0;i<elems1.size();i++){
```

```
            if(elems1[i]->count)
                elems2.push_back(elems1[i]);
        }
        std::sort(elems2.begin(),elems2.end(),comp);
        return elems2;
}


std::shared_ptr<Node> createTree1(std::vector<std::shared_ptr<Node>> elems){
        std::shared_ptr<Node> nde;

        while(elems.size() != 1){
            nde = std::make_shared<Node>();
            nde->count = elems[elems.size()-1]->count + elems[elems.size()-2]-
>count;
            nde->str = elems[elems.size()-1]->str + elems[elems.size()-2]->str;
            nde->left = elems[elems.size()-1];
            nde->right = elems[elems.size()-2];
            elems.pop_back();
            elems.pop_back();
            elems.push_back(nde);
            std::sort(elems.begin(),elems.end(),comp);
        }
        return nde;
}

std::shared_ptr<Node> createTree2(std::vector<std::shared_ptr<Node>> elems){
        int sum1=0, sum2=0;
        std::vector<std::shared_ptr<Node>> vec1,vec2;
        std::shared_ptr<Node> nde = std::make_shared<Node>();
        std::sort(elems.begin(),elems.end(),comp);
        for(int i=0;i<elems.size();i++){
            nde->str += elems[i]->str;
            nde->count += elems[i]->count;
        }
        if(elems.size() != 1){
            while(elems.size() != 0){
                if(sum1>=sum2){
                    sum2 += elems[elems.size()-1]->count;
                    vec2.push_back(elems[elems.size()-1]);
                    elems.pop_back();
                }else{
                    sum1 += elems[0]->count;
                    vec1.push_back(elems[0]);
```

```
                elems.erase(elems.begin());
            }
        }
        nde->left = createTree2(vec1);
        nde->right = createTree2(vec2);
    }
    return nde;
}

void setCodes(const std::shared_ptr<Node> &nde, std::vector<Code>& codes,
std::string code = "" ){
    Code cd;
    if(nde->right == nullptr &&  nde->left == nullptr){
        codes.push_back({nde->str, code});
    }
    if(nde->left != nullptr)
        setCodes(nde->left,codes, code+'0');
    if(nde->right != nullptr)
        setCodes(nde->right,codes, code+'1');

}


std::string genWord(int size){
    std::string str;
    for(int i=0;i<size;i++){
        str.push_back((char)(97 + rand() % 26));
    }
    return str;
}

std::string writeCodes(std::vector<Code> codes, int num){
    std::string str;
    str.push_back(num+48);
    str += ".\n";
    for(int i=0;i<codes.size();i++)
        str +=  codes[i].c + " - " + codes[i].code + "\n";
    str += "\n\n";
    return str;
}

struct Args{
    std::string tasks = "tasks.txt";
    std::string answers = "answers.txt";
```

```
        int number = 5;
};

int main(int argc, char** argv){

        Args args;
        char* optString = "a:t:hn:?";
        bool a=true;
        int opt = getopt( argc, argv, optString);
   while( opt != -1 ) {
      switch( opt ) {
         case 'a':
               args.answers = optarg;
                        break;
         case 't':
            args.tasks = optarg;
                        break;
         case 'n':
                        args.number = atoi(optarg);
                        break;
         case 'h':
         case '?':
                        std::cout << "a - set a file for asnwers\nt - set a file for tasks\
nh - display usage\nn - set a number of tasks\n";
            a = false;
                        break;

         default:
            break;
      }
      opt = getopt( argc, argv, optString);
        }

        std::string str;
        std::vector<std::shared_ptr<Node>> vec;
        std::vector<Code> codes;
        std::shared_ptr<Node> head;
        if(a){
        srand(time(0));
        std::ofstream file1, file2;
        file1.open(args.tasks);
        file2.open(args.answers);
        for(int i=0;i<args.number;i++){
                str = genWord(20);
```

```
            vec = count0letters(str);
            if(rand()%2){
                head = createTree1(vec);
                file1 << i+1 << ". Закодируйте символы заданной строки по статическому методу Хаффмана.\nСлово: " << str << "\n\n";
            }else{
                head = createTree2(vec);
                file1 << i+1 << ". Закодируйте символы заданной строки по статическому методу Шеннона-Фано.\nСлово: " << str << "\n\n";
            }
            setCodes(head, codes);
            file2 << writeCodes(codes, i+1);
            codes.clear();
        }
        file1.close();
        file2.close();
        }
        return 0;
}
```