МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Систем и Технологий

Кафедра «Информационные системы»

Дисциплина «Конструирование программного обеспечения»

**КУРСОВАЯ РАБОТА(ПРОЕКТ)**

Тема

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Соловьев.А.В. /

подпись инициалы, фамилия

Курс первый Группа ПИбд-14

Направление 09.03.04 «Программная инженерия»

Руководитель к.т.н., доцент Гуськов Г.Ю.

должность, ученая степень, ученое звание, фамилия, имя, отчество

Дата сдачи:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Дата защиты:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Оценка:

Ульяновск

2021 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных систем и технологий

Кафедра «Информационные системы»

Дисциплина «Конструирование программного обеспечения»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ(ПРОЕКТ)**

Студенту ПИбд-14 Соловьев А. В.

группа фамилия, инициалы

Тема проекта (работы)\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок сдачи законченного проекта «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Исходные данные к проекту УлГТУ, курс лекций по дисциплине «Конструирование программного обеспечения», интернет-источники \_

Содержание пояснительной записки (перечень подлежащих разработке вопросов)

Введение.

Планирование разработки проекта. Техническое задание. Планирование и формирование списка задач.

Описание проектирования. Техническое проектирование. Представление диаграмм с их описанием.

Описание процедур процесса конструирования.

Алгоритмическое обеспечение, описание алгоритма. Описание созданных проектов, классов, интерфейсов и т.п.

Описание технологий, использованных при разработке.

Пример работы приложения.

Качество конструирования. План и итоги тестирования.

Заключение

Перечень графического материала (с точным указанием обязательных чертежей)

Диаграммы UML: диаграмма вариантов использования (use-case), диаграмма последовательности (sequence), диаграмма проектирования классов приложения. Скриншоты разработанного программного продукта. Примеры работы пирложения.

Руководитель к.т.н, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ / Гуськов Г.Ю. /

должность подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **/** **\_\_\_\_\_\_\_\_\_/**

подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовую работу (проект)**

студента \_\_\_\_\_\_\_\_\_\_\_Соловьева Антона Владимировича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя и отчество

Факультет информационных систем и технологий группа ПИбд-14 курс первый

Дисциплина «Конструирование программного обеспечения»

Тема проекта Визуализация алгоритма Хаффмана «Жадный Алгоритм»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель к.т.н. доцент кафедры «Информационные системы» /Гуськов Г. Ю. /

должность, учёная степень, ученое звание подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

***Оглавление***

**Введение**….

**Планирование разработки проекта**…

Техническое задание…

**Описание работы Алгоритма**…

**Алгоритмическое обеспечение**…

Описание алгоритмической реализации…

**Описание технологий, использованных при разработке**…

**Примеры работы приложения**….

**Заключение**….

**Список Литературы**….

Исходный код программы……

**Введение**

Темой данной курсового проекта является визуализация алгоритма Хаффмана

Суть проекта показать и объяснить как работает Алгоритм Хаффмана

Сжатие информации - проблема, имеющая достаточно давнюю историю, гораздо более давнюю, нежели история развития вычислительной техники, которая (история) обычно шла параллельно с историей развития проблемы кодирования и шифровки информации.

Все алгоритмы сжатия оперируют входным потоком информации, минимальной единицей которой является бит, а максимальной - несколько бит, байт или несколько байт.

Кодирование Хаффмана является простым алгоритмом для построения кодов переменной длины, имеющих минимальную среднюю длину. Этот весьма популярный алгоритм служит основой многих компьютерных программ сжатия текстовой и графической информации. Некоторые из них используют непосредственно алгоритм Хаффмана, а другие берут его в качестве одной из ступеней многоуровневого процесса сжатия. Метод Хаффмана производит идеальное сжатие (то есть, сжимает данные до их энтропии), если вероятности символов точно равны отрицательным степеням числа 2. Алгоритм начинает строить кодовое дерево снизу вверх, затем скользит вниз по дереву, чтобы построить каждый индивидуальный код справа налево (от самого младшего бита к самому старшему). Начиная с работ Д.Хаффмана 1952 года, этот алгоритм являлся предметом многих исследований.

**Планирование разработки проекта**

Техническое задание

* Визуализировать Алгоритм Хаффмана
* Рассказать об Алгоритме ,как он работает
* Какие инструменты и языки программирования использовались в проекте
* Показать примеры работы программы

**Описание работы Алгоритма**

Метод сжатия информации на основе двоичных кодирующих деревьев был предложен Д. А. Хаффманом в 1952 году задолго до появления современного цифрового компьютера. Обладая высокой эффективностью, он и его многочисленные адаптивные версии лежат в основе многих методов, используемых в современных алгоритмах кодирования. Код Хаффмана редко используется отдельно, чаще работая в связке с другими алгоритмами кодирования. Метод Хаффмана является примером построения кодов переменной длины, имеющих минимальную среднюю длину. Этот метод производит идеальное сжатие, то есть сжимает данные до их энтропии, если вероятности символов точно равны отрицательным степеням числа 2.

Этот метод кодирования состоит из двух основных этапов:

1. Построение оптимального кодового дерева.
2. Построение отображения код-символ на основе построенного дерева.

Алгоритм основан на том, что некоторые символы из стандартного 256-символьного набора в произвольном тексте могут встречаться чаще среднего периода повтора, а другие - реже. Следовательно, если для записи распространенных символов использовать короткие последовательности бит, длиной меньше 8, а для записи редких символов - длинные, то суммарный объем файла уменьшится. В результате получается систематизация данных в виде дерева («двоичное дерево»).

* Бинарным деревом называется ориентированное дерево, полустепень исхода любой из вершин которого не превышает двух.
* Вершина бинарного дерева, полустепень захода которой равна нулю, называется корнем. Для остальных вершин дерева полустепень захода равна единице.

Пусть Т- бинарное дерево, А=(0,1)- двоичный алфавит и каждому ребру Т-дерева приписана одна из букв алфавита таким образом, что все ребра, исходящие из одной вершины, помечены различными буквами. Тогда любому листу Т-дерева можно приписать уникальное кодовое слово, образованное из букв, которыми помечены ребра, встречающиеся при движении от корня к соответствующему листу. Особенность описанного способа кодирования в том, что полученные коды являются префиксными.

Очевидно, что стоимость хранения информации, закодированной при помощи Т-дерева, равна сумме длин путей из корня к каждому листу дерева, взвешенных частотой соответствующего кодового слова или длиной взвешенных путей.

Классический алгоритм 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;

Создается их родитель с весом, равным их суммарному весу;

Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка;

Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой - бит 0;

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.

Допустим, у нас есть следующая таблица частот:

А – 15

Б – 7

В – 6

Г – 6

Д – 5

На первом шаге из листьев дерева выбираются два с наименьшими весами - Г и Д. Они присоединяются к новому узлу- родителю, вес которого устанавливается 5+6=11. Затем узлы Г и Д удаляются из списка свободных. Узел Г соответствует ветви 0 родителя, узел Д - ветви 1.

На следующем шаге то же происходит с узлами Б и В, так как теперь эта пара имеет самый меньший вес в дереве. Создается новый узел с весом 13, а узлы Б и В удаляются из списка свободных.

На следующем шаге «наилегчайшей» парой оказываются узлы Б/В и Г/Д.

Для них еще раз создается родитель, теперь уже с весом 24. Узел Б/В соответствует ветви 0 родителя, Г/Д - ветви 1.

На последнем шаге в списке свободных осталось только 2 узла - это узел А и узел Б (Б/В)/(Г/Д). В очередной раз создается родитель с весом 39, и бывшие свободные узлы присоединяются к разным его ветвям. Поскольку свободным остался только один узел, то алгоритм построения дерева кодирования Хаффмана завершается. Каждый символ, входящий в сообщение, определяется как конкатенация нулей и единиц, сопоставленных ребрам дерева Хаффмана, на пути от корня к соответствующему листу. Для данной таблицы символов коды Хаффмана будут выглядеть так:

А 01 Б 100 В 101 Г 110 Д 111

Наиболее частый символ сообщения А закодирован наименьшим количеством бит, а наиболее редкий символ Д - наибольшим. Стоимость хранения кодированного потока, определенная как сумма длин взвешенных путей, определится выражением 15\*1+7\*3+6\*3+6\*3+5\*3=87, что существенно меньше стоимости хранения входного потока (312). Поскольку ни один из полученных кодов не является префиксом другого, они могут быть однозначно декодированы при чтении их из потока. Алгоритм декодирования предполагает просмотр потоков битов и синхронное перемещение от корня вниз по дереву Хаффмана в соответствии со считанным значением до тех пор, пока не будет достигнут лист, то есть декодировано очередное кодовое слово, после чего распознавание следующего слова вновь начинается с вершины дерева. Классический алгоритм Хаффмана имеет один существенный недостаток. Для восстановления содержимого сжатого сообщения декодер должен знать таблицу частот, которой пользовался кодер. Следовательно, длина сжатого сообщения увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию сообщения. Кроме того, необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по сообщению: одного для построения модели сообщения (таблицы частот и дерева Хаффмана), другого собственно для кодирования.

**Алгоритмическое обеспечение**

Описание алгоритмической реализации

Инструменты:

Язык программирования – С++

*Принцип Работы Алгоритма*

1. На вход программы поступает текст - вводится с клавиатуры.
2. Создается список узлов дерева, где каждый узел содержит символ и количество его повторений в тексте.
3. Создается дерево Хаффмана, объединяя узлы с наименьшим количеством повторений.
4. Программа кодирует исходный текст и таблицу с символами на экран.
5. Потом программа декодирует закодированный текст с помощью обратной таблицы.
6. Декодированный текст выходит на экран.

**Описание технологий, использованных при разработке**

Я решил взять именно С++ как язык программирования в своем проекте, потому что он имеет много достоинств по сравнению с остальными языками программирования

Этот язык программирования подходит для создания игр, операционных систем и другого.

И язык С++ очень гибкий и подходит под очень многие задачи.

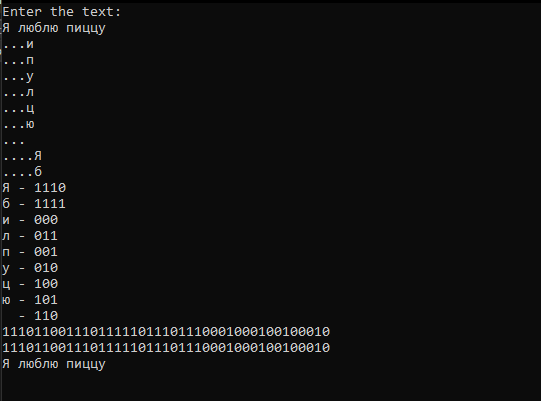
Преимущества C++:

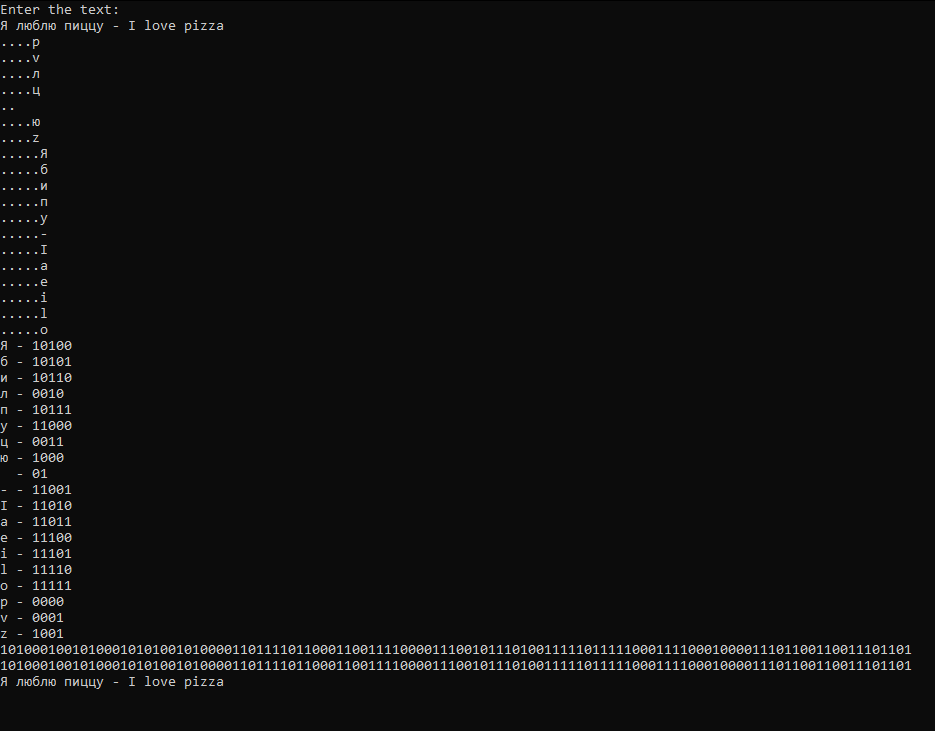
1. Возможность поручать большую часть работы компилятору .
2. C++ — строгий язык, в нём уже на этапе компиляции ясно, что будет делать программа. А значит, на этапе компиляции можно сделать ряд вычислений и оптимизаций, чтобы программа работала быстрее.
3. С++ позволяет писать гибкие и быстро работающие программы.

Недостатки C++:

1. Сложный язык для изучения.
2. Легко можно написать неправильную программу, даже этого не заметив.
3. В С++ сложно работать с зависимостями, в ручную приходится скачивать исходники нужной библиотеки со всеми зависимостями.

***Примеры работы программы***





**Заключение**

Проведена демонстрация проекта по визуализации алгоритма Хаффмана

В процессе написания курсового проекта были:

* Улучшено знание языка С++
* Получены новые знания о языке С++
* Повторен материал связанный с Алгоритмом Хаффмана

Программа хорошо работает и проблем в её работе во время тестов не было, программа хорошо работает с русским и английским алфавитом.

**Список литературы**

Исходный код

