



Федеральное агентство по рыболовству
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Астраханский государственный технический университет»
Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована DQS
по международному стандарту ISO 9001:2015

Институт информационных технологий и коммуникаций
Направление подготовки 09.03.04 Программная инженерия
Профиль «Разработка программно-информационных систем»
Кафедра «Автоматизированные системы обработки информации и управления»

КУРСОВОЙ ПРОЕКТ

Учебно-демонстрационная программа

«Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»

по дисциплине «Алгоритмы и структуры данных»

Допущен к защите
«__» _____ 20__ г.
Руководитель

Оценка, полученная на защите
«_____»

Проект выполнен
обучающимся группы ДИПР6-21
Катуниным С.С.

Руководитель
доцент Лаптев В.В.

Члены комиссии:

_____ Лаптев В.В.

ФЕДЕРАЛЬНОЕ АГЕНТСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ ПО РЫБОЛОВСТВУ
АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ

Заведующий кафедрой

к.т.н., доцент

С.В. Белов _____

« ____ » _____ 201 ____ г.

Кафедра

«Автоматизированные системы

обработки информации и управления»

ЗАДАНИЕ

на выполнение курсового проекта

Обучающийся *Катунин Сергей Сергеевич*

Группа *ДИПР6-21*

Дисциплина *Алгоритмы и структуры данных*

Тема *Учебно-демонстрационная программа «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»*

Дата получения задания « ____ » _____ 201 ____ г.

Срок представления обучающимся КП на кафедру « ____ » _____ 201 ____ г.

Руководитель *доцент* _____ *Лантев В.В.* « ____ » _____ 201 ____ г.

должность, степень, звание подпись ФИО

Обучающийся _____ *Катунин С.С.* « ____ » _____ 201 ____ г.

подпись ФИО

Задачи

Разработка программного продукта, который

- предоставляет пользователю теоретический материал по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»;
- визуализирует работу алгоритма поиска подстроки в строке Кнута – Морриса – Пратта;
- предоставляет пользователю возможность тестирования по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»;

Список рекомендуемой литературы

1. Вирт. Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD / Пер. с англ. Ткачев Ф. В. - М.: ДМК Пресс, 2010. – 272 с.: ил.
2. Кормен, Томас Х. и др. Алгоритмы: построение и анализ, 3-е изд. : Пер. с англ. – М.: ООО «И. Д. Вильямс», 2013. – 1328 с. : ил. – Парал. тит. англ.

УТВЕРЖДАЮ

Заведующий кафедрой

к.т.н., доцент

С.В. Белов _____

« ____ » _____ 20 ____ г.

К заданию на курсовой проект
по дисциплине
«Алгоритмы и структуры данных»

КАЛЕНДАРНЫЙ ГРАФИК
курсового проектирования

№ п/п	Разделы, темы и их содержание, графический материал	Дата сдачи	Объем, %
1	Выбор темы	15.09.2019	1
2	Техническое задание	30.09.2019	3
3	Разработка модели, проектирование системы <ul style="list-style-type: none">▪ введение,▪ технический проект,▪ программа и методика испытаний,▪ литература	31.10.2019	25
4	Программная реализация системы <ul style="list-style-type: none">▪ работающая программа,▪ рабочий проект▪ скорректированное техническое задание (при необходимости)	30.11.2019	40
5	Тестирование и отладка системы, эксперименты <ul style="list-style-type: none">▪ работающая программа с внесёнными изменениями,▪ окончательные тексты всех разделов	15.12.2019	50
6	Компоновка текста Подготовка презентации и доклада <ul style="list-style-type: none">▪ пояснительная записка▪ презентация▪ электронный носитель с текстом пояснительной записки, исходным кодом проекта, презентацией и готовым программным продуктом	20.12.2019	59
7	Защита курсового проекта	30.12.2019	60-100

С графиком ознакомлен « ____ » _____ 20 ____ г.

Катунин С.С. _____, обучающийся группы ДИПР6-21
(фамилия, инициалы, подпись)

График курсового проектирования выполнен
без отклонений / с незначительными отклонениями / со значительными отклонениями

нужное подчеркнуть

Руководитель курсового проекта _____ доцент Лаптев В. В.
подпись, ученая степень, звание, фамилия, инициалы

СОДЕРЖАНИЕ

Введение.....	6
1 Технический проект.....	7
1.1 Анализ предметной области.....	7
1.1.1 Алгоритм поиска подстроки в строке.....	7
1.1.2 Алгоритм Кнута – Морриса – Пратта.....	7
1.1.3 Проверка знаний.....	13
1.2 Технология обработки информации.....	13
1.2.1 Диаграмма классов.....	14
1.2.2 Форматы данных для подсистемы теории.....	18
1.2.3 Форматы данных для демонстрации первого этапа алгоритма.....	18
1.2.4 Форматы данных для демонстрации второго этапа алгоритма.....	18
1.2.5 Форматы данных для подсистемы тестирования.....	18
1.2.6 Алгоритм формирования массива префикс-функции.....	19
1.2.7 Алгоритм Кнута – Морриса – Пратта.....	20
1.2.8 Алгоритм визуализации первого этапа алгоритма КМП.....	21
1.2.9 Алгоритм визуализации результата первого этапа алгоритма КМП.....	21
1.2.10 Алгоритм визуализации второго этапа алгоритма КМП.....	22
1.2.11 Алгоритм вывода теоретического материала.....	22
1.2.12 Алгоритм чтения базы вопросов из файла.....	22
1.2.13 Алгоритм тестирования.....	23
1.2.14 Основной алгоритм.....	24
1.3 Входные и выходные данные.....	24
1.4 Системные требования.....	25
2 Рабочий проект.....	26
2.1 Общие сведения о работе системы.....	26
2.2 Функциональное назначение программного продукта.....	26
2.3 Установка и выполнение программного продукта.....	26
2.4 Описание программы.....	27
2.5 Разработанные меню и интерфейсы.....	34
2.6 Сообщения системы.....	43
3 Программа и методика испытаний.....	45
3.1 Проверка работоспособности режима чтения теории.....	45
3.2 Проверка работоспособности режима демонстрации.....	45
3.3 Проверка работоспособности режима тестирования.....	46

Заключение.....	47
Список использованных источников.....	48
Приложение 1 Техническое задание.....	49
Приложение 2 Диаграмма классов.....	53
Приложение 3 База текстовых вопросов.....	55

ВВЕДЕНИЕ

Информационный поиск – очень распространённая задача, которая представляет собой процесс выявления в некотором множестве документов (текстов) той информации, которая удовлетворяет заранее определенному условию поиска или содержит необходимые (соответствующие информационной потребности) факты, сведения, данные. Мы каждый день сталкиваемся с процессом поиска подстроки в строке, когда пользуемся браузером и формируем поисковые запросы, а также в текстовых редакторах, в офисных редакторах, и так далее.

Существует большое количество различных видов алгоритмов поиска подстроки в строке, выполняющих одну и ту же задачу, многие из которых являются оптимальными, а большинство имеет какие-либо преимущества по сравнению с остальными. Поэтому иногда от выбора оптимального алгоритма может зависеть эффективность программы. Чаще всего у студентов возникают некоторые сложности в понимании принципа работы этих алгоритмов, а также в процессе применения их при решении различных задач. Студент, не уделивший должного внимания этой теме, не может считаться хорошим программистом, так как не может реализовать эффективный поиск данных для необходимой задачи. Таким образом, разработка учебно-демонстрационной программы, предлагающей теоретический материал по теме «Алгоритм Кнута – Морриса – Пратта», а также тестирование с вопросами по этой теме, не только позволит начинающему программисту, впервые столкнувшемуся с таким методом поиска, понять принцип его работы, но и поможет быстро научиться применять приобретенные знания на практике студенту, у которого возникают трудности с использованием поиска подстроки в строке.

Целью создания учебно-демонстрационной программы «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта» является автоматизация обучения и контроля знаний по теме «алгоритм Кнута – Морриса – Пратта».

Назначение программы – повышение качества знаний студентов, снижение нагрузки на преподавателя.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

1.1.1 Алгоритм поиска подстроки в строке

Алгоритм поиска подстроки в строке – это специфический поиск, так называемый поиском строки (поиском в строке). Пусть есть некоторая строка (или текст) T и подстрока (или образ) W . Необходимо найти первое вхождение этой подстроки в указанной строке. Это действие типично для любых систем обработки текстов. (Элементы массивов T и W – символы некоторого конечного алфавита – например, $\{0, 1\}$, или $\{a, \dots, z\}$, или $\{a, \dots, я\}$.)

Поиск подстроки формально определяется следующим образом. Пусть задан массив T из N элементов и массив W из M элементов, причем $0 < M \leq N$. Поиск подстроки обнаруживает первое вхождение W в T , результатом будем считать индекс i , указывающий на первое с начала строки (с начала массива T) совпадение с подстрокой (образом).

Пример. Требуется найти вхождение подстроки $W = abaa$ в строке $T = abcabaabscabca$. На рисунке 1.1 визуализирован результат поиска.

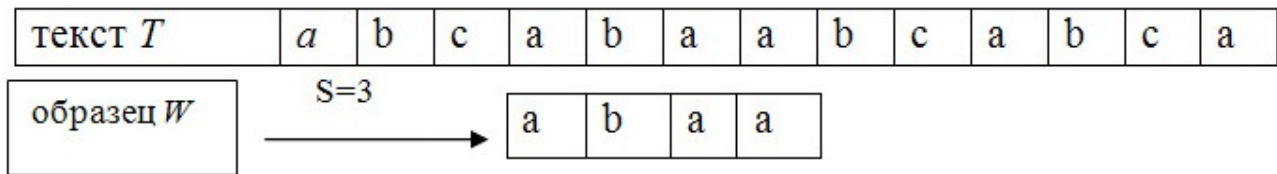


Рисунок 1.1 — Алгоритм поиска в строке

1.1.2 Алгоритм Кнута – Морриса – Пратта

Алгоритм Кнута – Морриса – Пратта (далее КМП-алгоритм) – эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных. У КМП-алгоритма число сравнений литер имеет порядок $M+N$ (где M – длина подстроки W , а N – длина всей строки T), что гораздо лучше, чем $M*N$ в простом алгоритме поиска. Указатель на символ во время поиска всегда движется по тексту только вперед, тогда как в простом поиске просмотр текста всегда начинается с первой литеры образца после обнаружения неравенства литер, и поэтому уже просмотренные литеры могут просматриваться снова. Таким образом, типичная и максимальная сложность КМП-алгоритма не превосходит $O(2N)$, следовательно, этот алгоритм страдает невысокой скоростью на «хороших» данных, что однако компенсируется отсутствием регрессии на «плохих».

Алгоритм основывается на том соображении, что после частичного совпадения начальной части подстроки W с соответствующими символами строки T фактически известна

пройденная часть строки и можно «вычислить» некоторые сведения (на основе самой подстроки W), с помощью которых затем можно быстро продвинуться по строке T .

Основным отличием КМП-алгоритма от алгоритма прямого поиска является выполнение сдвига искомой подстроки W не на один символ на каждом шаге алгоритма, а на некоторое переменное количество символов. Перед тем как осуществлять очередной сдвиг, необходимо найти величину сдвига, которая определяется префикс-функцией. Для повышения эффективности алгоритма необходимо, чтобы сдвиг на каждом шаге был как можно большим.

Следующий пример, в котором ищется слово Hooligan, иллюстрирует идею алгоритма. Подчеркнуты литеры, которые уже сравнивались. Каждый раз, когда сравниваемые литеры оказываются не равны, подстрока W сдвигается на весь уже пройденный путь, так как полное совпадение со строкой T заведомо невозможно для слова Hooligan при меньшем сдвиге. На рисунке 1.2 представлен данный пример.

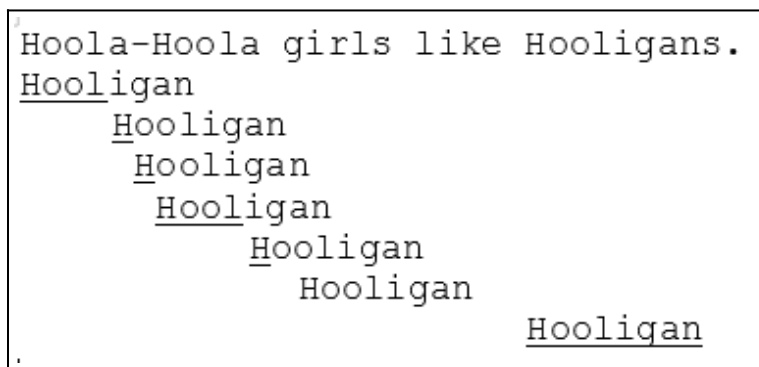


Рисунок 1.2 — Алгоритм Кнута – Морриса – Пратта

Алгоритм КМП можно условно поделить на два этапа:

1. Формирование массива P (префикс-функция), содержащего величины сдвигов.
2. Поиск подстроки W в строке T .

Первый этап: Для создания массива P необходима префикс-функция, которая для каждого i -го символа образа возвращает некоторое значение. Эти значения хранятся в $P[i]$.

Чтобы понять как работает префикс-функция, нужно определить такие понятия как префикс и суффикс. Префикс — это символы, стоящие в начале подстроки W . Суффикс — символы образа стоящие в конце W . У суффикса и префикса есть длина (количество символов). В данном алгоритме не рассматриваются суффиксы и префиксы с длиной равной длине всего образа. Префикс-функция для i -го символа подстроки W возвращает значение, равное максимальной длине совпадающих префикса и суффикса подстроки, которая заканчивается i -м символом в подстроке W (подстрока в подстроке).

Пример. Сформируем массив P для следующей подстроки W : **abcabca**. Текущий префикс подчеркнут, а суффикс выделен серым цветом. Префикс-функция ищет наибольшую длину совпадающих суффикса и префикса, поэтому для каждого i -го символа проверяются все возможные длины суффикса и префикса начиная с длины 1. В таблице 1.1 визуализирован процесс формирования массива с помощью префикс-функции.

Таблица 1.1 — Формирование массива P с помощью префикс-функции.

i	Образ	$\pi[i]$
0	<u>a</u> b c a b c a	<u>0</u>
1	a <u>b</u> c a b c a	<u>0</u>
2	a b <u>c</u> a b c a	0
2	a <u>b</u> c a b c a	<u>0</u>
3	a b c <u>a</u> b c a	1
3	a <u>b</u> c a b c a	1
3	a <u>b</u> c a b c a	<u>1</u>
4	a b c a <u>b</u> c a	0
4	a <u>b</u> c a b c a	2
4	a b <u>c</u> a b c a	2
4	a <u>b</u> c a b c a	<u>2</u>
5	a b c a b <u>c</u> a	0
5	a <u>b</u> c a b c a	0
5	a b c <u>a</u> b c a	3
5	a <u>b</u> c a b c a	3
5	a <u>b</u> c a b c a	<u>3</u>
6	a b c a b c <u>a</u>	1
6	a <u>b</u> c a b c a	1
6	a b c a <u>b</u> c a	1
6	a b c <u>a</u> b c a	4
6	a <u>b</u> c a b c a	4
6	a <u>b</u> c a b c a	<u>4</u>

Таким образом имеем массив $P[i]$ со значениями: 0 для $i = 0$, так как $P[0]$, согласно алгоритму, всегда равен нулю. Значение 0 для подстроки **ab**, 0 для подстроки **abc**, 1 для **abca**, так как первый и последний символ: **a** — совпадают; 2 для **abcab**, так как это длина

совпадающих суффикса и префикса: **ab**; 3 для подстроки **abcabc** и 4 для самого образца **W**, на котором выполнялся алгоритм формирования массива значений префикс-функции.

Алгоритм формирования массива префикс-функции приведен ниже в виде псевдокода:

Дано: **W** – образец (подстрока), **M** – длина образца, **P[N]** – массив префикс-функции, **i, j** — индексы алгоритма.

```

P[0] = 0
если M > 1 то
|   i = 1
|   j = 0
|   пока (i < M)
| |   если W[j] == W[i] то
| | |   P[i] = j+1
| | |   ++i
| | |   ++j
| |   иначе
| | |   если j == 0 то
| | | |   P[i] = 0
| | | |   ++i
| | |   иначе
| | | |   j = P[i-j-1]
| | |   конец ветвления
| |   конец ветвления
|   конец цикла
конец ветвления

```

Второй этап: на втором этапе происходит непосредственный поиск подстроки в строке.

У нас есть подстрока (образ) **W** и строка **T**, в которой мы находим вхождение образа. Для поиска используется два индекса: **k** для строки и **l** для подстроки, которые указывают на соответствующий символ. Когда символы строки и подстроки совпадают с заданными **k** и **l**, мы смещаем индексы вправо, в том случае, когда происходит несовпадение, мы используем заранее подготовленный массив **P** (префикс-функция). Мы обращаем внимание на символ в образе, который предшествует несовпадшему, и нам нужно соответствующее значение из массива **P**, которое является новым значением индекса **l** образа **W**. Таким образом алгоритм продолжает свою работу, пока **l** не станет равен длине образа **W**, что означает, что образ был найден в строке **T**, либо пока **k** не станет равен длине строки **T**, что соответственно означает,

что искомой подстроки W в строке T нет. В таблице 1.2 визуализирован процесс поиска подстроки с помощью алгоритма КМП.

Таблица 1.2 — Поиск подстроки в строке с помощью алгоритма КМП

Шаг 1 – символы совпадают															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 2															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 3															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 4															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 5															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 6 – несовпадение символов, обращаем внимание на $\pi[5-1]$ (для символа b)															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока	a	b	c	a	b	d									
Шаг 7 – индекс l для подстроки меняется на $\pi[4]=2$, снова несовпадение символов															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока				a	b	c	a	b	d						
Шаг 8 – индекс l для подстроки меняется на $\pi[2-1]=0$															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			
Шаг 9 – символы совпадают															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			
Шаг 10															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			
Шаг 11															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			

Продолжение таблицы 1.2

Шаг 12															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			
Шаг 13 – несовпадение символов, обращаем внимание на $\pi[5-1]$ (для символа b)															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока							a	b	c	a	b	d			
Шаг 14 – индекс l для подстроки меняется на $\pi[4]=2$, символы совпадают															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока										a	b	c	a	b	d
Шаг 15 – символы совпадают															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока										a	b	c	a	b	d
Шаг 16															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока										a	b	c	a	b	d
Шаг 17 – так как $l = m$ (индекс равен длине образа), значит образ был найден															
Строка	a	b	c	a	b	e	a	b	c	a	b	c	a	b	d
Подстрока										a	b	c	a	b	d

Алгоритм второго этапа поиска подстроки Кнута — Морриса — Пратта представлен ниже в виде псевдокода:

Дано: T – строка, W – образец (подстрока), N – длина строки, M – длина подстроки, $P[M]$ – массив префикс-функции, k – индекс строки, l – индексы подстроки.

```

k = 0
l = 0
пока (k < N)
|   если  $T[k] == W[l]$  то
|   |   ++i
|   |   ++j
|   |   если (l == M) то вернуть истина
|   |   иначе
|   |   |   если j == 0 то
|   |   |   |   ++k
|   |   |   |   если (k == N) то вернуть ложь
|   |   |   иначе

```

```

|   |   |    $l = P[l-1]$ 
|   |   |   конец ветвления
|   |   |   конец ветвления
конец цикла
вернуть ложь

```

1.1.3 Проверка знаний

После того, как студент ознакомился с теоретическим материалом по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта», он может пройти тестирование, для того чтобы выяснить, насколько хорошо был усвоен полученный теоретический материал.

В случае если вопросов будет слишком много, студент может утомиться в процессе тестирования, однако, если вопросов будет слишком мало, то по результату тестирования нельзя будет утверждать, что тестируемый усвоил материал. Следует предоставить студенту вопросы закрытого типа с выбором одного правильного ответа из четырех предложенных, а также открытые вопросы с кратким ответом, и подсчитать процент верных ответов (традиционно считается, что уровень знаний тестируемого достаточен, если количество правильных ответов составляет не менее 60%, на хорошем уровне при 75% правильных ответов, и на отличном уровне при 85%).

В базе будет содержаться не менее десяти вопросов, количество вопросов устанавливается перед тестированием, выбор количества варьируется от 1 до 10, по умолчанию количество вопросов равно пяти. Соответствующие вопросы будут выбраны случайным образом из базы вопросов и добавлены в тест, порядок вариантов ответов каждого вопроса при повторном прохождении теста будет меняться.

По завершении тестирования студент сможет ознакомиться со своим результатом. Каждый правильный ответ на вопрос будет стоить по одному баллу. Процент правильных ответов высчитывается по формуле:

$$P = R / N * 100 \%,$$

где P – процент правильных ответов,

R – количество баллов,

N – общее количество вопросов.

1.2 Технология обработки информации

Анализ предметной области показал, что программа рассчитана на одного пользователя, который может ознакомиться с теоретическим материалом, перейти в режим демонстрации работы алгоритма и пройти тестирование.

В ходе анализа предметной области была построена диаграмма вариантов использования (рис. 1.3).

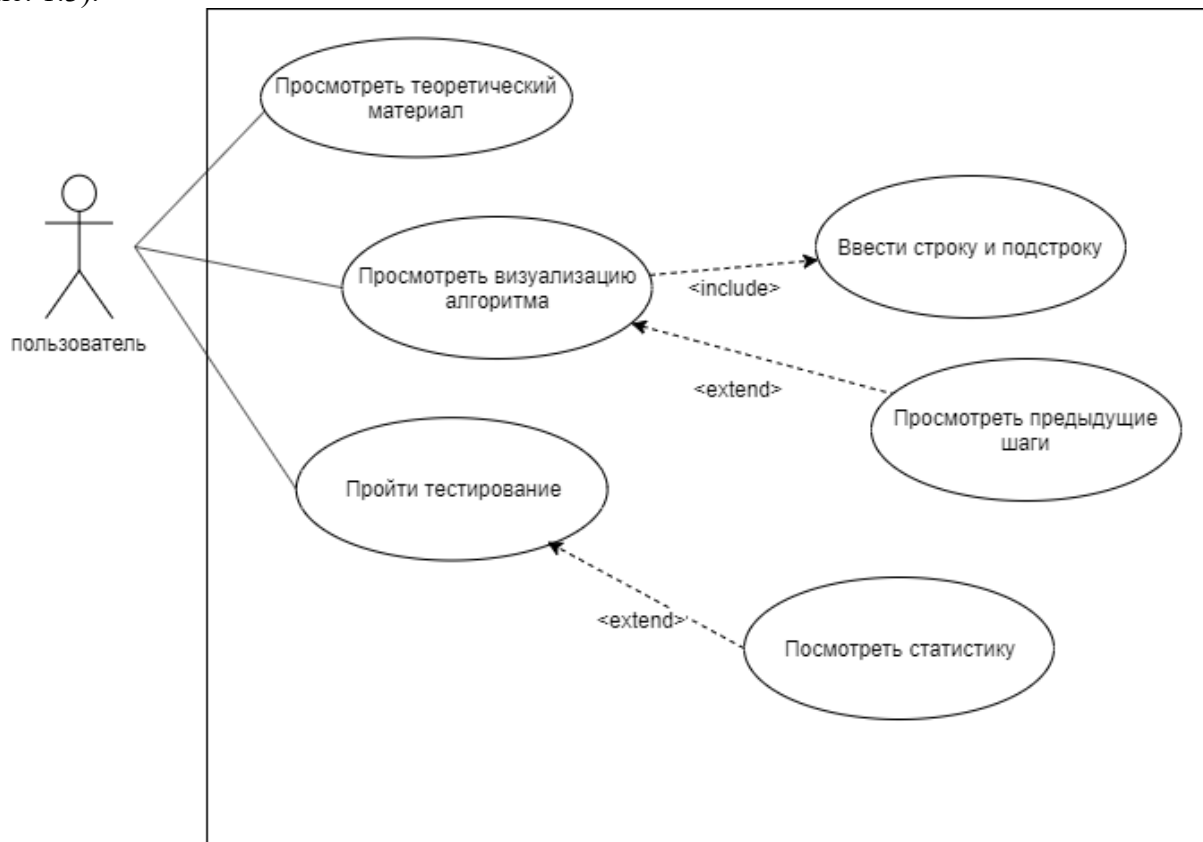


Рисунок 1.3 – Диаграмма вариантов использования

Вариант использования «Просмотреть визуализацию алгоритма» расширен функцией «Просмотреть предыдущие шаги», а также включает обязательную функцию «Ввести строку и подстроку».

Вариант использования «Пройти тестирование» расширен функцией «Посмотреть статистику».

1.2.1 Диаграмма классов

В ходе анализа предметной области была разработана диаграмма классов. Она представлена в *Приложении 2* в нотации UML с некоторыми нестандартными Qt обозначениями, такими как: < – для обозначения сигнала Qt (функция, которая ничего не делает и вызывается когда происходит определенное событие) и \$ – для обозначения слота Qt (метод-функция, которая вызывается в ответ на определенный сигнал).

Главным классом данной программы является класс *MainWindow* (главное окно – контроллер), являющегося наследником класса Qt – *Qmainwindow*. *MainWindow* включает в себя:

- указатель на интерфейс Qt;
- указатель на объект класса *Teoria* (теоретическая подсистема программы);
- указатель на объект класса *Demo* (демонстрационная подсистема программы);
- указатель на объект класса *Test* (подсистема тестирования).

Методы:

- функция – слот обработки нажатия на кнопку перехода к теории;
- функция – слот обработки нажатия на кнопку перехода к демонстрации;
- функция – слот обработки нажатия на кнопку перехода к тестированию;
- функция, которая делает главное меню – активным окном приложения.

Класс *Teoria* (подсистема теории) является наследником класса Qt – *QWidget* и включает в себя указатель на интерфейс Qt.

Методы:

- функция – сигнал для возврата в главное меню;
- функция – слот обработки нажатия на кнопку перехода в главное меню;
- функция, которая переопределяет поведение программы при нажатии на ссылку HTML текста.

Класс *Demo* (подсистема демонстрации) является наследником класса Qt – *QmainWindow* и включает в себя:

- массив, содержащий строки псевдокода алгоритма, выводимого на экран во время демонстрации;
- указатель на объект класса *PictureBox* (реализует графическую демонстрацию алгоритма);
- указатель на объект класса *kmp* (функциональная часть демонстрационной подсистемы – непосредственно сам КМП-алгоритм).

Методы:

- функция для задержки вывода анимации на экран;
- функция – сигнал для возврата в главное меню;
- функция – слот обработки нажатия на кнопку запуска демонстрации первого этапа КМП-алгоритма;
- функция – слот обработки нажатия на кнопку перехода к демонстрации результатов первого этапа КМП-алгоритма;
- функция – слот обработки нажатия на кнопку запуска демонстрации второго этапа;
- функция – слот обработки нажатия на кнопку перехода в главное меню;
- функция – слот обработки ввода пользователем начальной строки *T* и искомой подстроки *W*;

- функция, обрабатывающая нажатия на клавиши стрелок влево и вправо клавиатуры (переход к следующему и предыдущему шагу алгоритма).

Класс **kmp** (функциональная часть подсистемы демонстрации) включает в себя:

- строка **T**;
- искомая подстрока **W**;
- матрица, содержащая дополнительную информацию о шаге алгоритма префикс-функции;
- матрица, содержащая дополнительную информацию о шаге второго этапа КМП-алгоритма.

Методы:

- функция, вычисляющая массив значений префикс-функции;
- функция, реализующая КМП-алгоритм (второй этап алгоритма – непосредственный поиск подстроки);
- функции получения матриц с дополнительной информации для обоих алгоритмов.

Класс **PictureBox** (графическая часть подсистемы демонстрации) включает в себя:

- объект класса **QPixmap** (графическое полотно, на котором визуализируется работа КМП-алгоритма);
- объект класса **drawSettings** (настройки вывода), содержащего объекты класса **QFont** – шрифт для букв и шрифт для чисел, целое число – размер ячейки, в которой записана буква и объект класса **QPoint** – координата для рисования.;
- строка **T**;
- подстрока **W**;
- номер текущего шага демонстрации;
- количество шагов демонстрации;
- массив объектов **pref_func_step_info** (матрицы с дополнительной информацией о шаге алгоритма префикс-функции);
- массив объектов **kmp_step_info** (матрицы с дополнительной информацией о шаге КМП-алгоритма);
- массив значений префикс-функции;
- номер текущего демонстрируемого этапа (1-й этап, результаты 1-ого этапа, 2-й этап).

Методы:

- функция, рисующая ячейки, текст, числа;
- функция, подготавливающая настройки для рисования;
- функция – демонстрация первого этапа алгоритма;
- функция – демонстрация результатов первого этапа алгоритма;

- функция – демонстрация второго этапа алгоритма;
- функция, отображающая данный шаг алгоритма;
- функция, реализующая анимацию при демонстрации 2-ого этапа алгоритма;

Класс *Test* (подсистема тестирования) является наследником класса Qt – *Qwidget* и включает в себя:

- строка, содержащая весь текст из файла с базой вопросов;
- массив объектов класса *Question*, состоящего из строки с текстом вопроса, целого числа – типа вопроса, и массива из четырех объектов класса *Answer*, который в свою очередь состоит из строки – текста ответа и булевой переменной – правильный ли ответ.
- объект класса *user_answers*, состоящего из массива целых чисел – ответов пользователя на вопросы закрытого типа и массива строк – ответов пользователя на вопросы открытого типа с кратким ответом;
- количество вопросов N;
- индекс текущего вопроса *i_step*;
- указатель на интерфейс Qt.

Методы:

- функция – сигнал для возврата в главное меню;
- функции обработки нажатий на кнопки перехода к следующему и предыдущему вопросам;
- функции – слоты, обрабатывающие выбор пользователем флаговых кнопок (чекбоксы);
- функция – слот обработки нажатия на кнопку завершения тестирования;
- функция – слот обработки ввода пользователем ответа на вопрос открытого типа;
- функция – слот обработки нажатия на кнопку начала тестирования;
- функция – слот обработки нажатия на кнопку перехода в главное меню;
- функция – слот обработки нажатия на кнопку просмотра статистики тестирования;
- функция, которая начинает процесс тестирования;
- функция формирования вопросов;
- функция вывода текущего вопроса (как закрытого, так и открытого типа) на экран;
- функция, проверяющая ответы пользователя;
- функция, которая выводит результаты тестирования;
- функция, которая сохраняет результаты в файл со статистикой;
- функция, которая читает и выводит статистику на экран.

1.2.2 Форматы данных для подсистемы теории

Теоретический материал, словарь для теоретического материала, база вопросов и статистика результатов тестирования хранятся в текстовых файлах с названиями «text.txt», «dictionary.txt», «quest.txt» и «stats.txt» соответственно.

Теоретический материал представляет собой текстовый файл, содержащий гипертекст (HTML). Текст в файле должен быть отформатирован вручную с использованием стандартных HTML-тегов.

1.2.3 Форматы данных для демонстрации первого этапа алгоритма

Для визуализации первого этапа алгоритма поиска подстроки КМП – «формирование массива значений P (префикс-функция)» используется подстрока W и матрица, содержащая дополнительную информацию о шаге: в которой хранятся индексы i и j , указывающие на символы подстроки, значение массива P на текущем шаге (значение префикс-функции для i -го элемента), а так же индекс выделяемой строки пояснения к алгоритму. Пояснение к алгоритму представляет собой текст – псевдокод выполняемого алгоритма, фон текущей «выполняемой» строки (индекс которой находится в матрице) алгоритма выделяется синим цветом, фон остальных строк является белым. Матрица, содержащая дополнительную информацию о шаге состоит из N строк, где N – количество шагов демонстрации первого этапа алгоритма, и соответственно 4-х столбцов.

1.2.4 Форматы данных для демонстрации второго этапа алгоритма

Для визуализации второго этапа алгоритма поиска подстроки КМП – «непосредственный поиск подстроки» используется строка T , подстрока W , массив значений P (префикс-функция) и матрица, содержащая дополнительную информацию о шаге: в которой хранятся индексы k и l , где k – индекс, указывающий на символ строки, а l – индекс, указывающий на символ подстроки; а так же индекс выделяемой строки пояснения к алгоритму. Формат представления пояснения к алгоритму для второго этапа алгоритма аналогичен формату пояснения к алгоритму первого этапа, и отличается только непосредственно текстом (псевдокодом). Матрица, содержащая дополнительную информацию о шаге второго этапа состоит из N строк, где N – количество шагов демонстрации второго этапа алгоритма, и соответственно 3-х столбцов.

1.2.5 Форматы данных для подсистемы тестирования

База вопросов представляет собой текстовый файл, который имеет следующую структуру: каждый вопрос начинается с символа «?», после которого указывается число — тип вопроса. Если указана цифра «1», значит описывается вопрос закрытого типа, в случае, если

«2», то вопрос – открытый с кратким ответом. Вопрос с закрытым типом занимает пять строк – на первой строке (после символов «?1») находится текст вопроса, на оставшихся четырех — дистракторы, причем первый вариант ответа является правильным. Вопрос с открытым типом занимает две строки – на первой строке (после символов «?2») находится текст вопроса, на второй строке – текст правильного ответа. В программе вопрос будет представлен в виде структуры, в которой хранятся строка с вопросом, тип вопроса и четыре дистрактора. В случае вопроса открытого типа, используется только первый дистрактор из четырех в качестве правильного краткого ответа. Дистрактор, в свою очередь, тоже является структурой и состоит из строки (варианта ответа), и переменной логического типа, которая определяет правильность данного ответа.

1.2.6 Алгоритм формирования массива префикс-функции

Дано: W – образец (подстрока), M – длина образца, $P[N]$ – массив префикс-функции, v_pref_info - матрица с дополнительной информацией ($P[i], i, j, i_code$), где $P[i]$ – значение P для i -ого элемента, i, j — индексы алгоритма, i_code — соответствующий индекс подсвечиваемой строки пояснения к алгоритму (-1, если ничего не подсвечивается и 0, 1, 2 подсвечивать 1,2,3 строку соответственно).

$P[0] = 0$

если $M > 1$ то

| $i = 1$

| $j = 0$

| поместить в v_pref_info следующие значения: $\{P[0], i, j, -1\}$

| пока ($i < M$)

| | поместить в v_pref_info следующие значения: $\{-1, i, j, -1\}$

| | если $W[j] == W[i]$ то

| | | $P[i] = j + 1$

| | | поместить в v_pref_info следующие значения: $\{P[i], i, j, 2\}$

| | | $++i$

| | | $++j$

| | иначе

| | | если $j == 0$ то

| | | | $P[i] = 0$

| | | | поместить в v_pref_info следующие значения: $\{P[i], i, j, 0\}$

| | | | $++i$

| | | иначе

```

|   |   |   |    $j = P[j-1]$ 
|   |   |   |   поместить в  $v\_pref\_info$  следующие значения:  $\{-1, i, j, 1\}$ 
|   |   |   |   конец ветвления
|   |   |   |   конец ветвления
|   |   |   |   конец цикла
|   |   |   |   конец ветвления

```

1.2.7 Алгоритм Кнута – Морриса – Пратта

Дано: T – строка, W – образец (подстрока), N – длина строки, M – длина подстроки, $P[M]$ – массив Пи, v_kmp_info - матрица с дополнительной информацией (k, l, i_code), где k – индекс строки, l — индексы подстроки, (-1, если ничего не подсвечивается и 0, 1, 2 подсвечивать 1,2,3 строку соответственно).

$k = 0$

$l = 0$

поместить в v_kmp_info следующие значения: $\{k, l, -1\}$

пока ($k < N$)

```

|   |   |   |   если  $T[k] == W[l]$  то
|   |   |   |   ++ $i$ 
|   |   |   |   ++ $j$ 
|   |   |   |   если  $(l == M)$  то вернуть истина
|   |   |   |   иначе поместить в  $v\_kmp\_info$  следующие значения:  $\{k, l, 2\}$ 
|   |   |   |   конец ветвления
|   |   |   |   иначе
|   |   |   |   если  $j == 0$  то
|   |   |   |   ++ $k$ 
|   |   |   |   если  $(k == N)$  то вернуть ложь
|   |   |   |   иначе поместить в  $v\_kmp\_info$  следующие значения:  $\{k, l, 0\}$ 
|   |   |   |   конец ветвления
|   |   |   |   иначе
|   |   |   |    $l = P[l-1]$ 
|   |   |   |   поместить в  $v\_kmp\_info$  следующие значения:  $\{k, l, 1\}$ 
|   |   |   |   конец ветвления
|   |   |   |   конец ветвления
|   |   |   |   конец цикла
|   |   |   |   вернуть ложь

```

1.2.8 Алгоритм визуализации первого этапа алгоритма КМП

Дано: W – образец (подстрока), M – длина образца, v_pref_info – матрица с дополнительной информацией ($P[i]$, I , j , i_code), где $P[i]$ – значение P для i -ого элемента, i , j – индексы алгоритма, i_code – соответствующий индекс подсвечиваемой строки пояснения к алгоритму (-1, если ничего не подсвечивается и 0, 1, 2 подсвечивать 1,2,3 строку соответственно), i_step – номер текущего шага, N_step – количество шагов первого этапа алгоритма

$i_step = 0$

пока пользователь не дал команду завершить процесс демонстрации

| если пользователь запросил следующий шаг и $i_step < N_step - 1$ то увеличить i_step на 1

| если пользователь запросил предыдущий шаг и $i_step > 0$ то уменьшить i_step на 1

| получить информацию из v_pref_info для текущего шага

| очистить окно для рисования и нарисовать подстроку W , индексы i, j , значения P для текущего шага

| вывести текст подсветить соответствующую строку в части окна с пояснением к алгоритму

конец цикла

предложить пользователю перейти к демонстрации результата первого этапа (алгоритм 1.2.5)

1.2.9 Алгоритм визуализации результата первого этапа алгоритма КМП

Дано: W – образец (подстрока), M – длина образца, массив $P[N]$ – массив Π , i_step – номер текущего шага, N_step – количество шагов демонстрации результата первого этапа алгоритма

$i_step = 0$

пока пользователь не дал команду завершить процесс демонстрации

| если пользователь запросил следующий шаг и $i_step < N_step - 1$ то увеличить i_step на 1

| если пользователь запросил предыдущий шаг и $i_step > 0$ то уменьшить i_step на 1

| очистить окно для рисования и нарисовать подстроку W , выделить символы в зависимости от значения P для текущего шага

конец цикла

предложить пользователю перейти к демонстрации второго этапа (алгоритм 1.2.6)

1.2.10 Алгоритм визуализации второго этапа алгоритма КМП

Дано: T – строка, W – образец (подстрока), N – длина строки, M – длина подстроки, $P[M]$ – массив P_i , v_kmp_info – матрица с дополнительной информацией (k, l, i_code), где k – индекс строки, l – индексы подстроки, (-1, если ничего не подсвечивается и 0, 1, 2 подсвечивать 1,2,3 строку соответственно), i_step – номер текущего шага, N_step – количество шагов второго этапа алгоритма

$i_step = 0$

пока пользователь не дал команду завершить процесс демонстрации

- | если пользователь запросил следующий шаг и $i_step < N_step - 1$ то увеличить i_step на 1
- | если пользователь запросил предыдущий шаг и $i_step > 0$ то уменьшить i_step на 1
- | получить информацию из v_kmp_info для текущего шага
- | очистить окно для рисования и нарисовать строку T , подстроку W , индексы k, l и массив P (нарисовать подстроку a , и под каждым символом подстроки нарисовать соответствующие значения массива P_i)
- | вывести текст подсветить соответствующую строку в части окна с пояснением к алгоритму

конец цикла

1.2.11 Алгоритм вывода теоретического материала

Дано: $text.txt$ – текстовый файл с теорией, $dictionary.txt$ – текстовый файл со словарем, $window$ – окно, в которое выводится текст теории, $dictionary$ – вспомогательное окно, в которое выводится текст из словаря

пока пользователь не дал команду выйти в главное меню

- | вывести содержимое файла $text.txt$ в окно $window$
- | вывести содержимое файла $dictionary.txt$ в окно $dictionary$

конец цикла

1.2.12 Алгоритм чтения базы вопросов из файла

Дано: $Answer$ – структура с полями $text$ (строковая константа, текст варианта ответа) и переменной логического типа $right$ (определяет правильность ответа), $Question$ – структура с полями $text$ (строковая константа, текст вопроса), $mode$ – тип вопроса (1 – закрытый или 2 – открытый) и $arr[4]$ – массив с элементами типа $Answer$. $Questions[N]$ – массив с элементами типа $Question$, N – количество вопросов в базе вопросов, $quest.txt$ – текстовый файл с базой вопросов.

пока (в файле есть строки с текстом)

```
|   прочитать строку temp_string
|   объявить переменную temp_question типа Question
|   прочитать тип вопроса mode для temp_question из строки temp_string
|   прочитать текст вопроса text для temp_question из строки temp_string
|   если mode = 1 (закрытый тип вопроса) то
|   |   отметить первый вариант ответа temp_question.arr[0] правильным
|   |   пока i < 4
|   |   |   прочитать следующую строку из файла в temp_string
|   |   |   прочитать текст ответа temp_question.arr[i].text из temp_string
|   |   конец цикла
|   а если mode = 2 (открытый тип вопроса) то
|   |   прочитать следующую строку из файла в temp_string
|   |   прочитать текст ответа temp_question.arr[0].text из temp_string
|   конец ветвления
|   поместить temp_question в массив Questions[N]
конец цикла
```

1.2.13 Алгоритм тестирования

Дано: *Answer* — структура с полями *text* (строковая константа, текст варианта ответа) и переменной логического типа *right* (определяет правильность ответа), *Question* — структура с полями *text* (строковая константа, текст вопроса), *mode* - тип вопроса (1 - закрытый или 2 - открытый) и *arr[4]* – массив с элементами типа *Answer*. *Questions[N]* – массив с элементами типа *Question*, *N* – количество вопросов в базе вопросов, *size* – количество вопросов в тесте.

перемешать вопросы в массиве вопросов

перемешать варианты ответов в вопросах закрытого типа

цел *i* = 0

ch[size] – массив целых чисел, заполненный значением -1 (ответы пользователя на вопросы закрытого типа)

ch_str[size] – массив пустых строк (ответы пользователя на вопросы открытого типа)

пока пользователь не дал команду завершить тестирование

```
|   вывести на экран текст вопроса
|   если Questions[i].mode = 1 (вопрос закрытого типа) то
|   |   вывести на экран четыре варианта ответа
```

```

|   |   если ch[i] = -1 то записать ответ пользователя в ch[i]
|   |   а если Questions[i].mode = 2 (вопрос открытого типа)
|   |       то предоставить пользователю поле для ввода краткого ответа
|   |   если ch_str[i] — пуста строка то записать краткий ответ пользователя в ch_str[i]
|   |   конец ветвления
|   |   если пользователь запросил следующий вопрос и i < size -1 то увеличить i на 1
|   |   если пользователь запросил предыдущий вопрос и i > 0 то уменьшить i на 1
|   |   конец цикла
|   |   подсчитать количество правильных ответов и процент правильных ответов
|   |   вывести результаты тестирования

```

1.2.14 Основной алгоритм

```

вывести условия меню
пока пользователь не дал команду завершить программу
|   |   если пользователь дал команду «перейти в режим теории» то вызвать алгоритм
|   |   вывода теоретического материала 1.2.7
|   |   если пользователь дал команду «перейти в режим демонстрации» то вызвать
|   |   алгоритм визуализации первого этапа алгоритма КП 1.2.4
|   |   если пользователь дал команду «перейти в режим тестирования» то вызвать
|   |   алгоритм тестирования 1.2.9
|   |   конец цикла

```

1.3 Входные и выходные данные

Входные данные:

- Выбор пунктов меню.
- Строка и подстрока, для демонстрации алгоритма;
- Ответ пользователя на вопрос тестирования;

Выходные данные:

- Теоретический материал, предоставляемый пользователю для изучения.
- Демонстрация первого этапа алгоритма КП;
- Демонстрация результата первого этапа алгоритма КП (массив Р);
- Демонстрация второго этапа алгоритма КП;
- Тестовые вопросы;
- Оценка, полученная в результате тестирования.

1.4 Системные требования

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 512 МБ ОЗУ;
- не менее 50 МБ свободного места на диске;
- дисковод CD-ROM/DVD-ROM

Операционная система: Windows XP (x86) с пакетом обновления 3 (SP3) или более поздние. Среда разработки – интегрированная среда Qt Creator (версия Qt 5.4), язык C++ (стандарт C++ 11 и более поздние).

2 РАБОЧИЙ ПРОЕКТ

2.1 Общие сведения о работе системы

Программный продукт разработан в интегрированной среде Qt Creator (версия Qt 5.4) на языке C++ (стандарт C++ 11 и более поздние), с использованием компилятора MinGW GCC (версия 4.9.2). Программа работает под управлением операционной системы Windows XP (x86) Professional (SP3) и более поздними.

2.2 Функциональное назначение программного продукта

Разработанный программный продукт предназначен для повышения качества знаний пользователей по теме «Поиск подстроки в строке: Алгоритм Кнута – Морриса – Пратта», а также для снижения нагрузки на преподавателя. Программа имеет следующие функциональные возможности:

- предоставление пользователю теоретического материала по теме «Поиск подстроки в строке: Алгоритм Кнута – Морриса – Пратта»;
- визуализация работы алгоритма Кнута – Морриса – Пратта;
- тестирование пользователя по теме «Поиск подстроки в строке: Алгоритм Кнута – Морриса – Пратта»;

Программа имеет следующие функциональные ограничения:

- количество символов в строке/подстроке не может быть более 15;
- программа не предоставляет возможность шифровать / расшифровывать текстовые файлы с базой вопросов и с теорией;
- программа не должна обеспечивать редактирование статистики результатов тестирования.

2.3 Установка и выполнение программного продукта

Программу можно запустить с помощью проводника операционной системы, запустив скомпилированный исполняемый .exe файл.

Для выполнения программы необходимо:

1. Скопировать на жесткий диск компьютера папку KMP_coursework, содержащую исполняемый файл KMP_coursework.exe, необходимые файлы для функционирования программы (quest.txt, stats.txt, text.txt, dictionary.txt, img1.jpg, img2.jpg), библиотеки .dll для функционирования интерфейсной части и папку source с исходными текстами программы.
2. Запустить исполняемый файл KMP_coursework.exe.

2.4 Описание программы

Программа состоит из 7 классов – 3 класса для модуля демонстрации, 2 класса – для модуля тестирования, 1 класс – для модуля теории, 1 класс – класс «интерфейс-контроллер».

В таблице 2.1 приведено описание вложенной структуры (в классе kmp), содержащей дополнительную информацию о шаге первого этапа алгоритма КМП (заголовочный файл kmp.h).

Таблица 2.1 – Описание структуры pref_func_step_info

Поле	Тип	Назначение
Pi	int	Значение префикс-функции для i-ого элемента
i	int	Индекс i для символа
j	int	Индекс j для символа
code_i	int	Индекс подсвечиваемой строки в пояснении к алгоритму

В таблице 2.2 приведено описание вложенной структуры (в классе kmp), содержащей дополнительную информацию о шаге второго этапа алгоритма КМП (заголовочный файл kmp.h).

Таблица 2.2 – Описание структуры kmp_step_info

Поле	Тип	Назначение
k	int	Индекс k для символа строки
l	int	Индекс l для символа подстроки
code_i	int	Индекс подсвечиваемой строки в пояснении к алгоритму

В таблице 2.3 приведено описание класса kmp, реализующего функциональную часть модуля демонстрации и используемое в классе Demo (композиция) (заголовочный файл kmp.h). Для реализации строк был выбран QString вместо std::string для лучшей совместимости с юникодом и интерфейсной частью программы.

Таблица 2.3 – Описание класса kmp

Поле	Тип	Назначение
str	QString	Строка, в которой ищется подстрока
substr	QString	Подстрока, образец
v_pref_info	std::vector<pref_func_step_info>	Матрица, содержащая дополнительную информацию о шаге первого этапа алгоритма
v_kmp_info	std::vector<kmp_step_info>	Матрица, содержащая дополнительную информацию о шаге второго этапа алгоритма

Продолжение таблицы 2.3

Метод	Описание
kmp(QString _str = "", QString _substr = "")	Конструктор для класса, с инициализацией строк
std::vector<int> prefix_function();	Префикс-функция
bool kmp_algorithm(const std::vector<int> &Pi)	Поиск подстроки с помощью алгоритма КМП
const std::vector<pref_func_step_info>& get_pref_step_info() const	Получить матрицу с информацией для первого этапа алгоритма
const std::vector<kmp_step_info>& get_kmp_step_info() const;	Получить матрицу с информацией для второго этапа алгоритма

В таблице 2.4 приведено описание вложенного перечисления DemoMode (в классе PictureBox), в котором перечислены все режимы отрисовки — этапы демонстрации алгоритма (заголовочный файл picturebox.h).

Таблица 2.4 – Описание перечисления DemoMode

Название	Номер	Назначение
drawPrefFunc	0	Режим отрисовки первого этапа алгоритма
drawPrefFunc_result	1	Режим отрисовки результата первого этапа алгоритма
drawKMP	2	Режим отрисовки второго этапа алгоритма

В таблице 2.5 приведено описание вложенной структуры drawSettings (в классе PictureBox), содержащей необходимые настройки для отрисовки этапов алгоритма (заголовочный файл picturebox.h).

Таблица 2.5 – Описание структуры drawSettings

Поле	Тип	Назначение
font	QFont	Шрифт (цвет, размер) для символов строки/подстроки
font_for_num	Qfont	Шрифт (цвет, размер) для чисел и цифр (нумерация символов)
cell_size	int	Размер ячейки, в которой находится символ строки/подстроки
p	QPoint	Координаты для отрисовки ячеек с символами

В таблице 2.6 приведено описание класса PictureBox, реализующего визуализацию (отрисовку) для модуля демонстрации и используемое в классе Demo (композиция) (заголовочный файл picturebox.h).

Таблица 2.6 – Описание класса PictureBox

Поле	Тип	Назначение
m_Pixmap	QPixmap	Буфер для отрисовки
settings	drawSettings	Настройки отрисовки
str	QString	Отрисовываемая строка
substr	QString	Отрисовываемая подстрока
i_step	size_t	Номер текущего шага
N_step	size_t	Количество шагов демонстрации алгоритма
pref_steps	std::vector<pref_func_step_info>	Матрица с информацией для первого этапа алгоритма (шаги первого этапа алгоритма)
kmp_steps	std::vector<kmp_step_info>	Матрица с информацией для второго этапа алгоритма (шаги второго этапа алгоритма)
Pi	std::vector<int>	Массив со значениями префиксной функции
current_mode	DemoMode	Текущий режим демонстрации
Метод		Описание
PictureBox(QWidget *parent = nullptr)		Конструктор для класса
void paintEvent(QPaintEvent*)		Метод, отрисовывающий буфер m_Pixmap
drawSettings prepare_drawSettings(int length)		Подготовка настроек (length – длина строки, используется для корректного выравнивания отрисовываемой строки по центру экрана)
void drawPrefFunc(QPainter &painter)		Нарисовать первый этап алгоритма (ячейки с символами, индексы i и j, числа, значения префиксной-функции под ячейками с символами)
void drawPrefFunc_result(QPainter &painter)		Нарисовать результат первого этапа алгоритма (ячейки с символами, значения префиксной-функции и выделить совпадающие суффиксы и префиксы для i-ого символа)
void drawKMP(QPainter &painter, int anim_x)		Нарисовать второй этап алгоритма (ячейки с символами для строки и подстроки, массив со значениями префиксной функции, индексы k и l)
void drawPiArray(QPainter &painter)		Нарисовать массив со значениями префиксной функции (используется в методе drawKMP)

Продолжение таблицы 2.6

Метод	Описание
void animation_podstroka(QPainter &painter, int x, int y)	Используется для реализации анимации двигающейся подстроки при демонстрации второго этапа алгоритма (используется в методе drawKMP)
void do_step(bool next)	Нарисовать текущий шаг: вызывает методы drawPrefFunc, drawPrefFunc_result, do_next_step_kmp и do_prev_step_kmp в зависимости от текущего режима демонстрации
void do_next_step_kmp(QPainter &painter)	Нарисовать следующий шаг с анимацией двигающейся вперед подстроки второго этапа демонстрации (вызывает метод drawKMP)
void do_prev_step_kmp(QPainter &painter)	Нарисовать предыдущий шаг с анимацией двигающейся назад подстроки второго этапа демонстрации (вызывает метод drawKMP)

В таблице 2.7 приведено описание класса Demo, реализующего интерфейсную часть модуля демонстрации и используемое в классе-контроллере MainWindow (композиция) (заголовочный файл demo.h).

Таблица 2.7 – Описание класса Demo

Поле	Тип	Назначение
code	std::vector<QString>	Коды, отображаемые в пояснении к алгоритму
pb_frame	PictureBox*	Указатель на виджет, в котором происходит отрисовка демонстрации алгоритма
ui	Ui::Demo*	Поле, необходимое для работы ui форм в Qt
Метод	Описание	
Demo(QWidget *parent = nullptr);	Конструктор класса	
~Demo()	Деструктор класса	
inline static void wait(int ms)	Метод, реализующий задержку при отрисовке (необходимо для анимаций во время демонстрации)	
void return_to_menu()	Сигнал о возврате из модуля демонстрации в главное меню	
void on_pushButton_start_clicked()	Нажатие на кнопку «Начать» (начинает инициализацию pb_frame и запускает демонстрацию первого этапа)	
void on_pushButton_result_clicked()	Нажатие на кнопку «Результат 1-го этапа» (запускает демонстрацию результата первого этапа)	

Продолжение таблицы 2.7

Метод	Описание
void on_pushButton_step2_clicked()	Нажатие на кнопку «Этап» (запускает демонстрацию второго этапа)
void on_pushButton_finish_clicked()	Нажатие на кнопку «Завершить демонстрацию» (прекращает демонстрацию и вызывает сигнал return to menu)
void on_lineEdit_str_textChanged(const QString &arg1)	Метод, который проверяет, чтобы длина подстроки не превышала длину строки
void keyPressEvent(QKeyEvent *event)	Метод, обрабатывающий нажатия на кнопки клавиатуры
void update_code()	Метод, обновляющий текст (и выделяющий «выполняемую» подстроку) в части окна с пояснением к алгоритму

В таблице 2.8 приведено описание класса Teoria, реализующего модуль теории и используемое в классе-контроллере MainWindow (заголовочный файл teoria.h).

Таблица 2.8 – Описание класса Teoria

Поле	Тип	Назначение
ui	Ui::Teoria*	Поле, необходимое для работы ui форм в Qt
Метод	Описание	
Teoria(QWidget *parent = nullptr)	Конструктор класса	
~Teoria()	Деструктор класса	
void return_to_menu()	Сигнал о возврате из модуля теории в главное меню	
void on_textBrowser_anchorClicked(const QUrl &arg1)	Метод, обрабатывающий нажатия на ссылки в основном тексте, для перемещения текущего отображаемого раздела в части окна со словарём	
void on_pushButton_clicked()	Нажатие на кнопку «Вернуться в главное меню» (вызывает сигнал return to menu)	

В таблице 2.9 приведено описание класса loader, реализующего загрузку текста из файла. Статический метод данного класса используется в классе Test.

Таблица 2.9 – Описание класса loader

Поле	Тип	Назначение
-	-	-
Метод	Описание	
loader()	Конструктор класса	
static std::string getText_from_file(const std::string &filename)	Получить текст в виде строки из файла с названием filename, если файл не существует, то вернуть пустую строку	

В таблице 2.10 приведено описание вложенного перечисления `quest_mode` (в классе `Test`), в котором перечислены все типы вопросов (заголовочный файл `test.h`).

Таблица 2.10 – Описание перечисления `quest_mode`

Название	Номер	Назначение
<code>closed_answer</code>	0	Закрытый тип вопроса
<code>open_answer</code>	1	Открытый тип вопроса с кратким ответом

В таблице 2.11 приведено описание вложенной структуры `user_answers` (в классе `Test`), содержащей ответы пользователя (заголовочный файл `test.h`).

Таблица 2.11 – Описание структуры `user_answers`

Поле	Тип	Назначение
<code>num</code>	<code>std::vector<int></code>	Ответы пользователя на вопросы закрытого типа в виде чисел
<code>str</code>	<code>std::vector<QString></code>	Ответы пользователя на вопросы открытого типа с кратким ответом в виде строк

В таблице 2.12 приведено описание вложенной структуры `Answer` (в классе `Test`), являющейся вариантом ответа на вопрос (заголовочный файл `test.h`).

Таблица 2.12 – Описание структуры `Answer`

Поле	Тип	Назначение
<code>text</code>	<code>QString</code>	Текст дистрактора
<code>right</code>	<code>bool</code>	Правильный ли ответ

В таблице 2.13 приведено описание вложенной структуры `Answer` (в классе `Test`), являющейся дистрактором вопроса (заголовочный файл `test.h`).

Таблица 2.13 – Описание структуры `Question`

Поле	Тип	Назначение
<code>text</code>	<code>QString</code>	Текст дистрактора
<code>mode</code>	<code>int</code>	Тип вопроса
<code>arr</code>	<code>std::array<Answer, 4></code>	4 варианта ответа

В таблице 2.14 приведено описание класса `Test`, реализующего модуль тестирования и используемое в классе-контроллере `MainWindow` (композиция) (заголовочный файл `test.h`).

Таблица 2.14 – Описание класса `Test`

Поле	Тип	Назначение
<code>all_text</code>	<code>std::string</code>	Содержимое текстового файла с базой вопросов, полученное с помощью статического метода класса <code>loader</code>
<code>Questions</code>	<code>std::vector<Question></code>	Сформированные вопросы
<code>user_choices</code>	<code>user_answers</code>	Ответы пользователя
<code>N</code>	<code>size_t</code>	Количество вопросов в тесте
<code>i_step</code>	<code>size_t</code>	Номер текущего отображаемого вопроса

Продолжение таблицы 2.14

Метод	Описание
Test(QWidget *parent = nullptr)	Конструктор класса
~Test()	Деструктор класса
void return_to_menu()	Сигнал о возврате из модуля тестирования в главное меню
void on_next_clicked()	Нажатие на кнопку «>» (переход к следующему вопросу)
void on_prev_clicked()	Нажатие на кнопку «<» (переход к предыдущему вопросу)
void on_Answer1_checkBox_clicked()	Выбор 1 варианта ответа
void on_Answer2_checkBox_clicked()	Выбор 2 варианта ответа
void on_Answer3_checkBox_clicked()	Выбор 3 варианта ответа
void on_Answer4_checkBox_clicked()	Выбор 4 варианта ответа
void on_finish_clicked();	Нажатие на кнопку «Завершить»
void on_answer_textEdited(const QString &arg1)	Обработка ввода краткого ответа
void on_spinBox_valueChanged(int arg1)	Ввод количества вопросов в тесте
void on_pushButton_start_clicked()	Нажатие на кнопку «Начать тестирования»
void on_return_to_menu_button_clicked()	Нажатие на кнопку «Вернуться в главное меню» (вызывает сигнал return_to_menu)
void on_pushButton_results_clicked()	Нажатие на кнопку «Посмотреть статистику»

В таблице 2.15 приведено описание класса MainWindow, главного интерфейсного класса-контроллера (заголовочный файл mainwindow.h). Классы Demo, Test и Teoria являются частью MainWindow (композиция), что не отображено в таблице 2.15, поскольку они были привязаны к классу MainWindow в дизайнера форм Qt, который автоматически генерирует файл ui_mainwindow.h, в котором уже и содержатся указатели на классы Demo, Test и Teoria.

Таблица 2.15 – Описание класса MainWindow

Поле	Тип	Назначение
ui	Ui::MainWindow*	Поле, необходимое для работы ui форм в Qt
Метод	Описание	
MainWindow(QWidget *parent = nullptr)	Конструктор класса	
~MainWindow()	Деструктор класса	
void on_pushButton_teorica_clicked()	Скрывает все виджеты и делает текущим отображаемым виджетом — виджет с теорией	
void on_pushButton_test_clicked()	Скрывает все виджеты и делает текущим отображаемым виджетом — виджет с тестированием	

Продолжение таблицы 2.15

Метод	Описание
<code>void on_pushButton_demo_clicked()</code>	Скрывает все виджеты и делает текущим отображаемым виджетом — виджет с демонстрацией
<code>void return_to_menu()</code>	Скрывает все виджеты и делает текущим отображаемым виджетом — виджет с главным меню

2.5 Разработанные меню и интерфейсы

После запуска программы на выполнение появится окно, содержащее главное меню (рис. 2.1), который помимо самого меню содержит информацию о теме курсового проекта и об авторе программы.

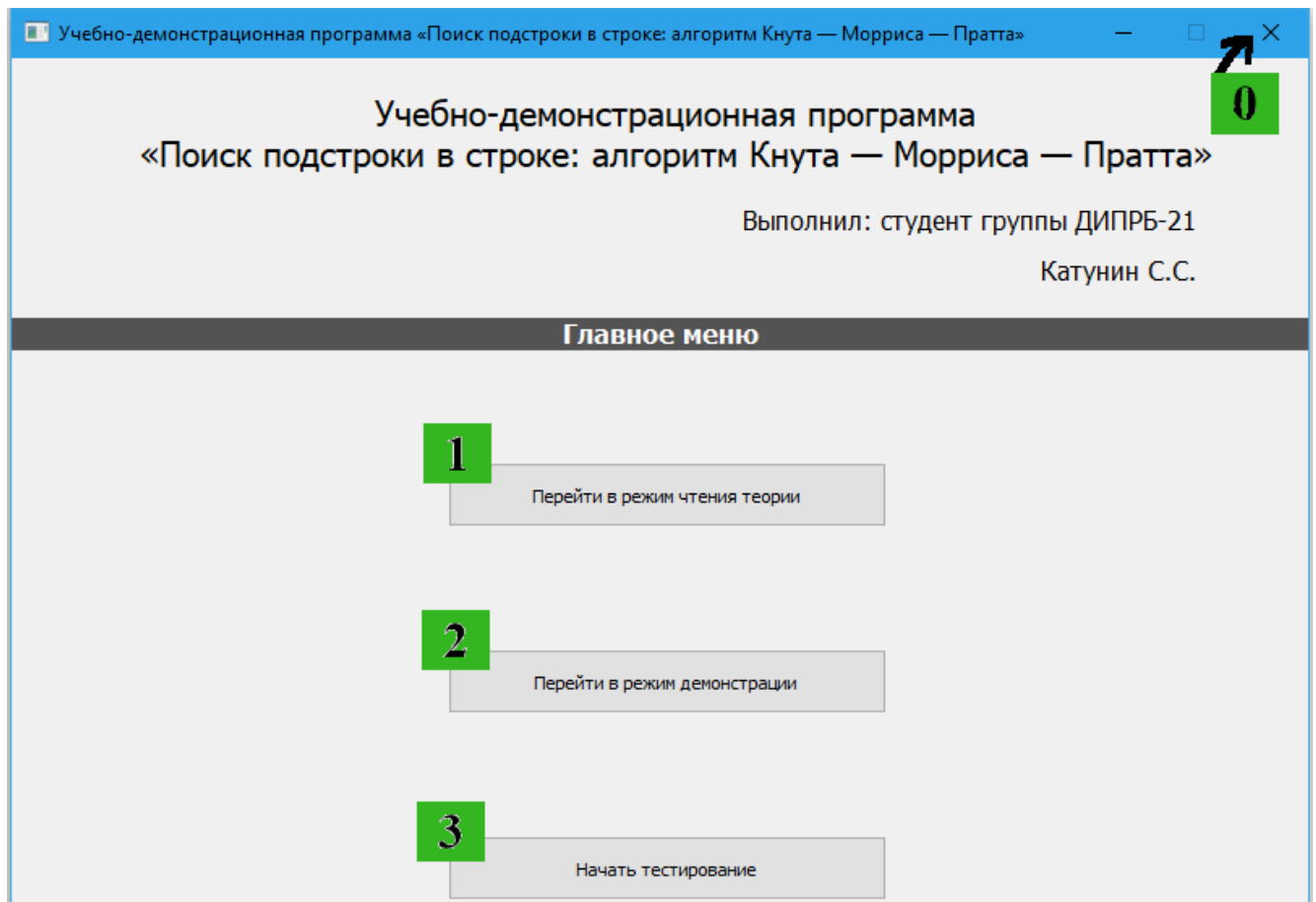


Рисунок 2.1 – Окно программы с главным меню

Кнопка «1» позволяет перейти к режиму чтения теории, кнопка «2» – к режиму демонстрации, кнопка «3» – к режиму тестирования, а кнопка «0» - завершает работу программы.

При нажатии на кнопку «1» главное меню сменится на режим чтения теории (рис. 2.2).

В правой части окна можно наблюдать теоретический материал, который может включать рисунки, таблицы, различные стили для текста и гиперссылки. В левой части окна представлен словарь, в котором вынесены некоторые определения, на которые указывают гиперссылки из теории. При нажатии на гиперссылку словарь автоматически прокрутится до места начала определения, чтобы пользователь мог ознакомиться с ним подробнее.

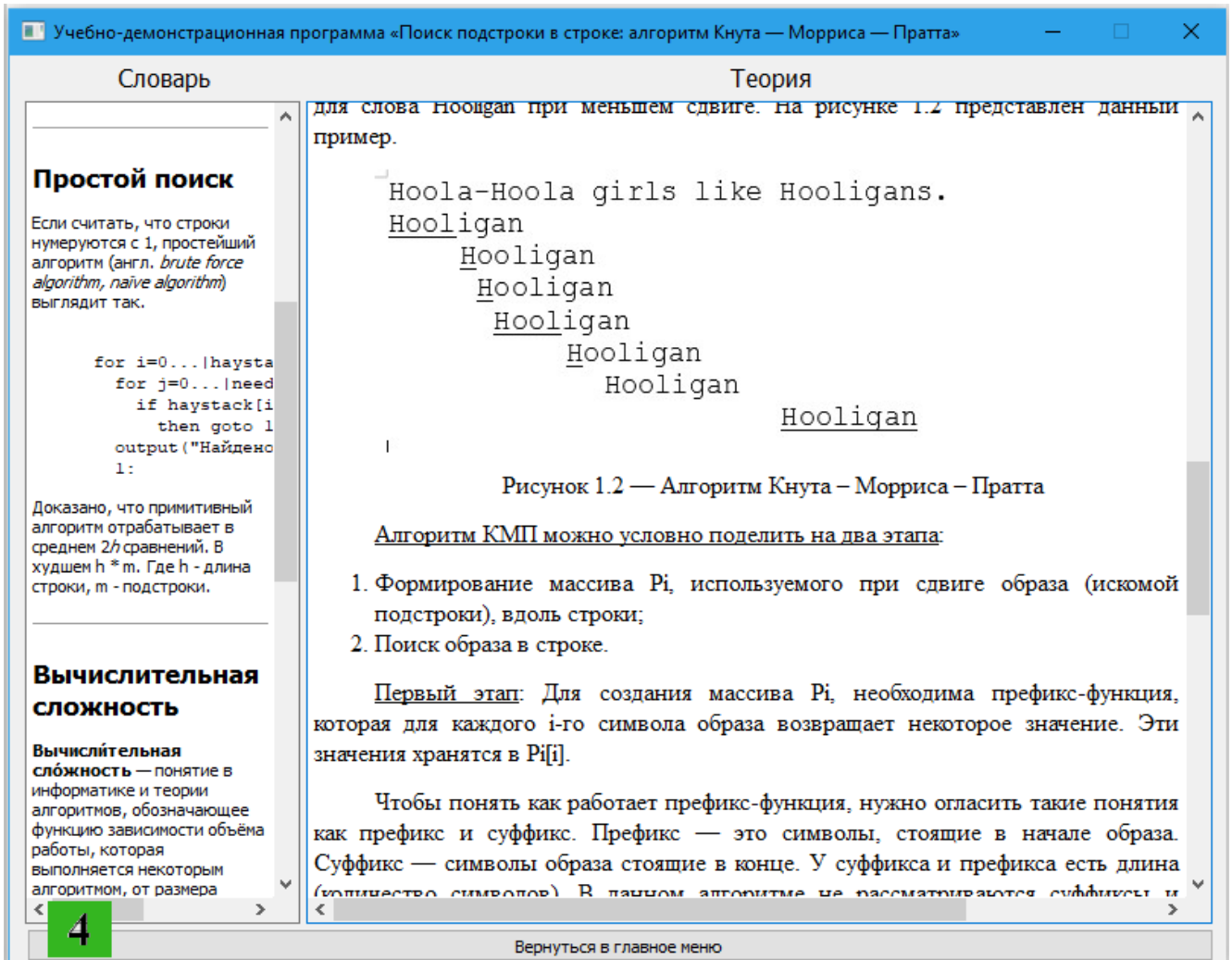


Рисунок 2.2 – Окно с выбранным режимом чтения теории

После нажатия кнопки «4» происходит возврат в главное меню (см. рис. 2.1).

После нажатия кнопки «2» в главном меню осуществляется переход в меню демонстрации (рис. 2.3).

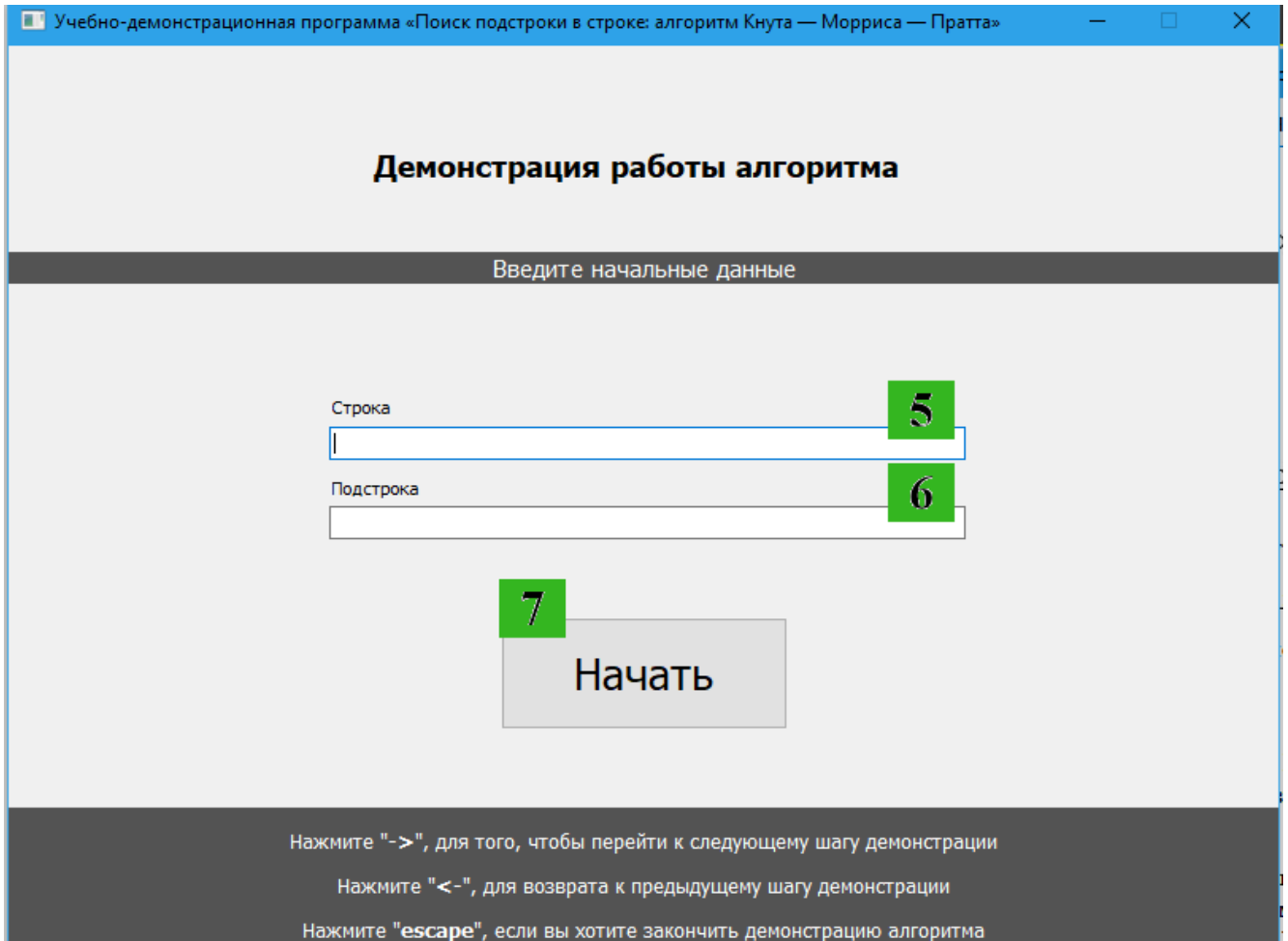


Рисунок 2.3 – Меню демонстрации

Программа приглашает пользователя ввести начальные данные для работы алгоритма. Текстовые поля «5» и «6» используются для ввода строки T и подстроки W соответственно. Если пользователь оставит данные поля пустыми, то алгоритм демонстрации начнет свою работу со строками «abcabeabscababd» для строки T и «abcabd» для подстроки W . В нижней части окна представлена инструкция по навигации между шагами демонстрации КМП-алгоритма.

При нажатии на кнопку «7» начинается процесс демонстрации первого этапа алгоритма Кнута – Морриса – Пратта (рис. 2.4).

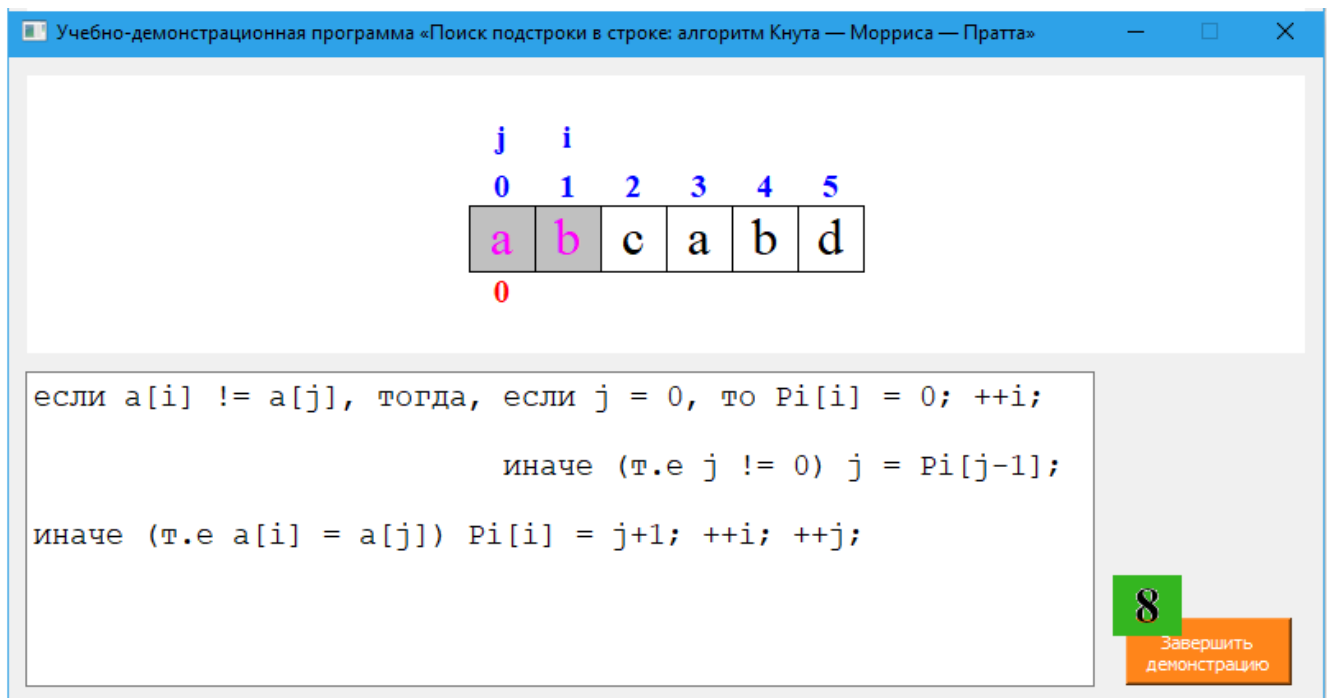


Рисунок 2.4 – Начало демонстрации первого этапа алгоритма КМП

При помощи нажатия на клавишу клавиатуры «стрелка вправо» осуществляется переход к следующему шагу первого этапа алгоритма (рис. 2.5), а при помощи нажатия клавиши «стрелка влево» осуществляется возврат к предыдущему шагу.

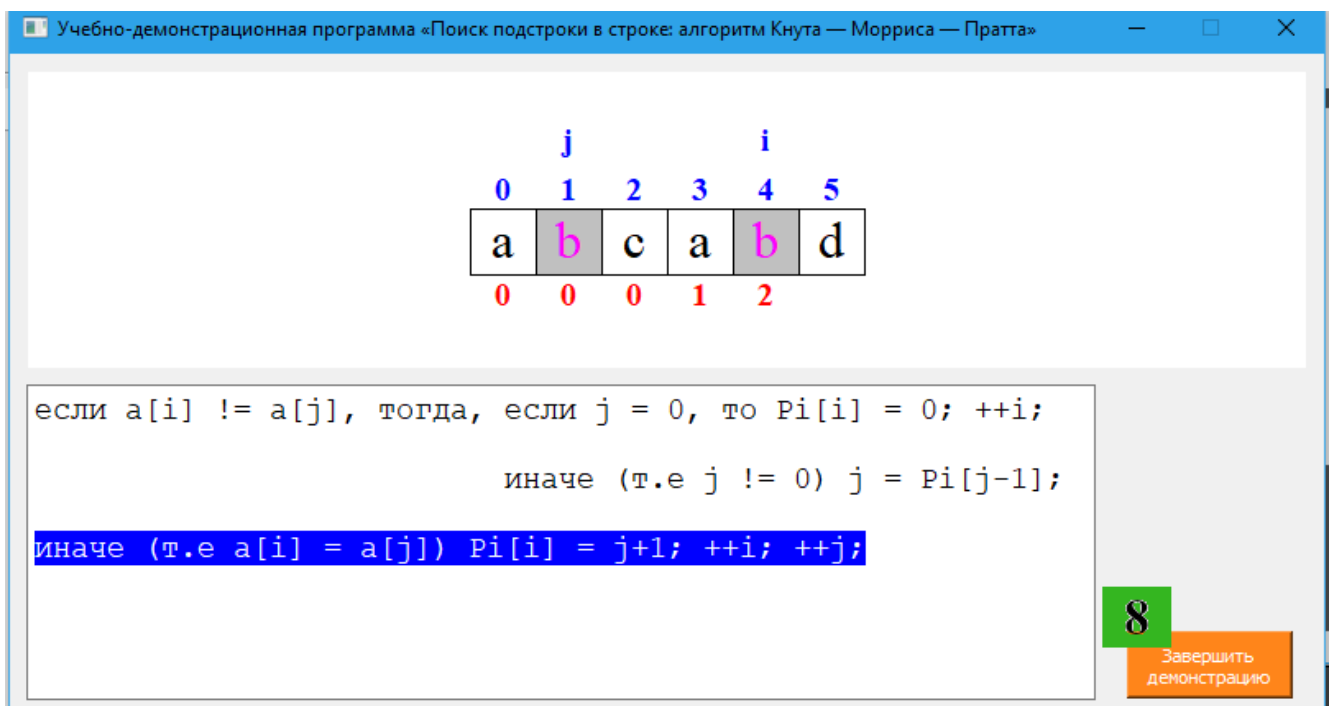


Рисунок 2.5 – Демонстрация первого этапа алгоритма КМП

Нажав кнопку «8» можно досрочно завершить демонстрацию и вернуться в главное меню программы (см. рис. 2.1).

На рисунке 2.6 представлен конец первого этапа демонстрации работы алгоритма.

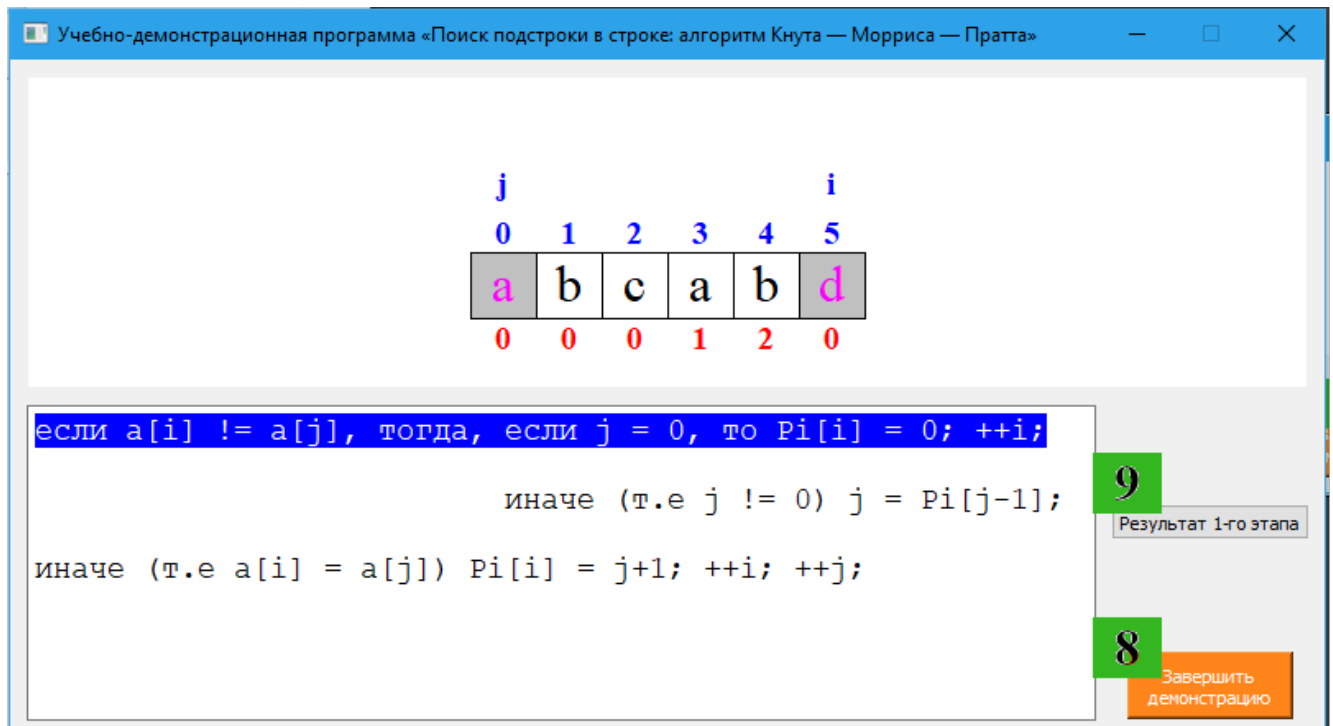


Рисунок 2.6 – Конец демонстрации первого этапа алгоритма КМП

После нажатия на кнопку «9» происходит демонстрация результата первого этапа. Навигация по шагам происходит аналогично демонстрации первого этапа (рис. 2.7).



Рисунок 2.7 – Демонстрация результата первого этапа алгоритма КМП

По завершении демонстрации результата первого этапа становится доступна кнопка «10» для перехода к демонстрации второго этапа алгоритма КМП (рис. 2.8).

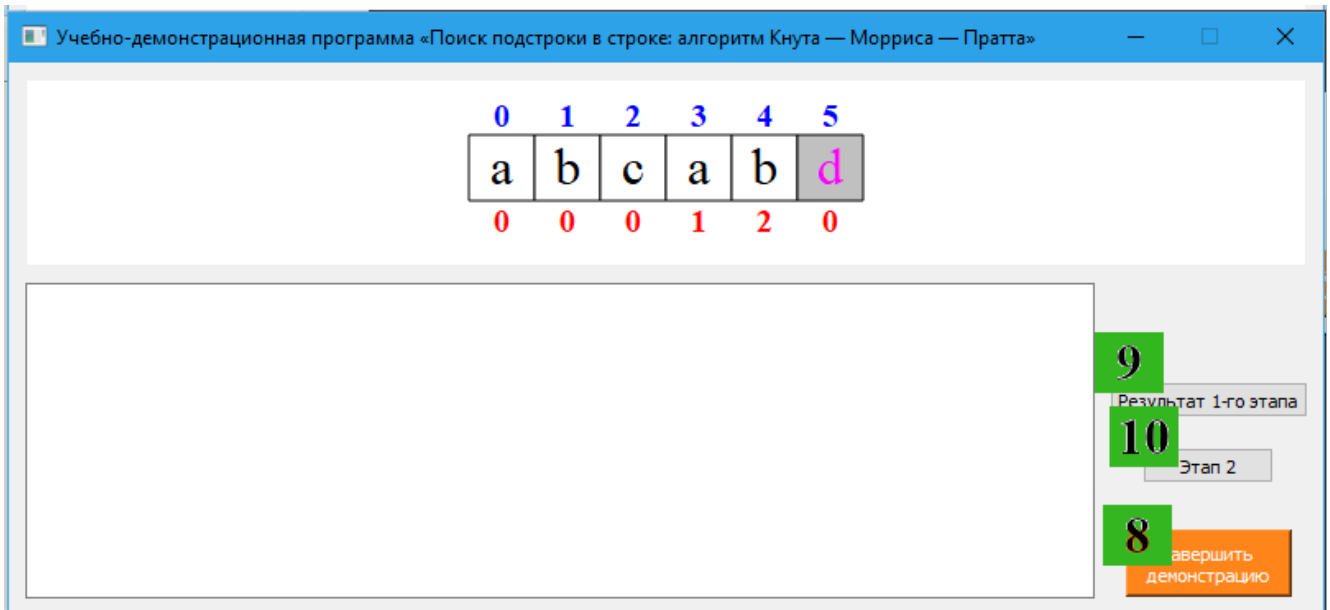


Рисунок 2.8 – Конец демонстрации результата первого этапа алгоритма КМП

После нажатия кнопки «10» начинается процесс демонстрации второго этапа алгоритма КМП. Один из шагов демонстрации проиллюстрирован на рисунке 2.9.

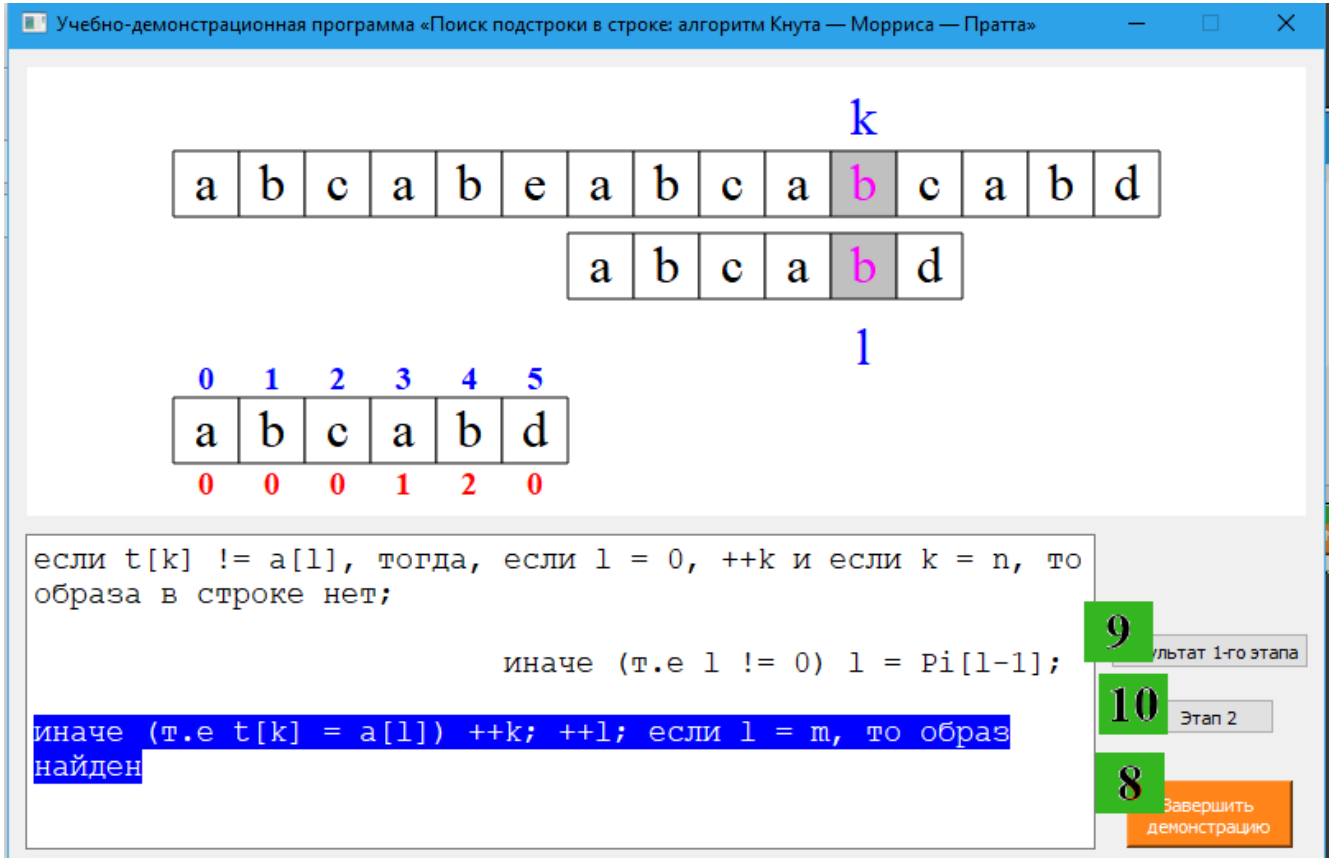


Рисунок 2.9 – Демонстрация второго этапа алгоритма КМП

При нажатии на кнопку «3» в главном меню перед пользователем предстает меню тестирования (рис. 2.10). Числовое поле «11» используется для задания количества вопросов в тесте, текстовое поле «12» – для задания имени тестируемого. Кнопка «13» начинает процесс тестирования, нажав на кнопку «14» можно ознакомиться со статистикой, и с помощью кнопки «15» можно вернуться в главное меню.

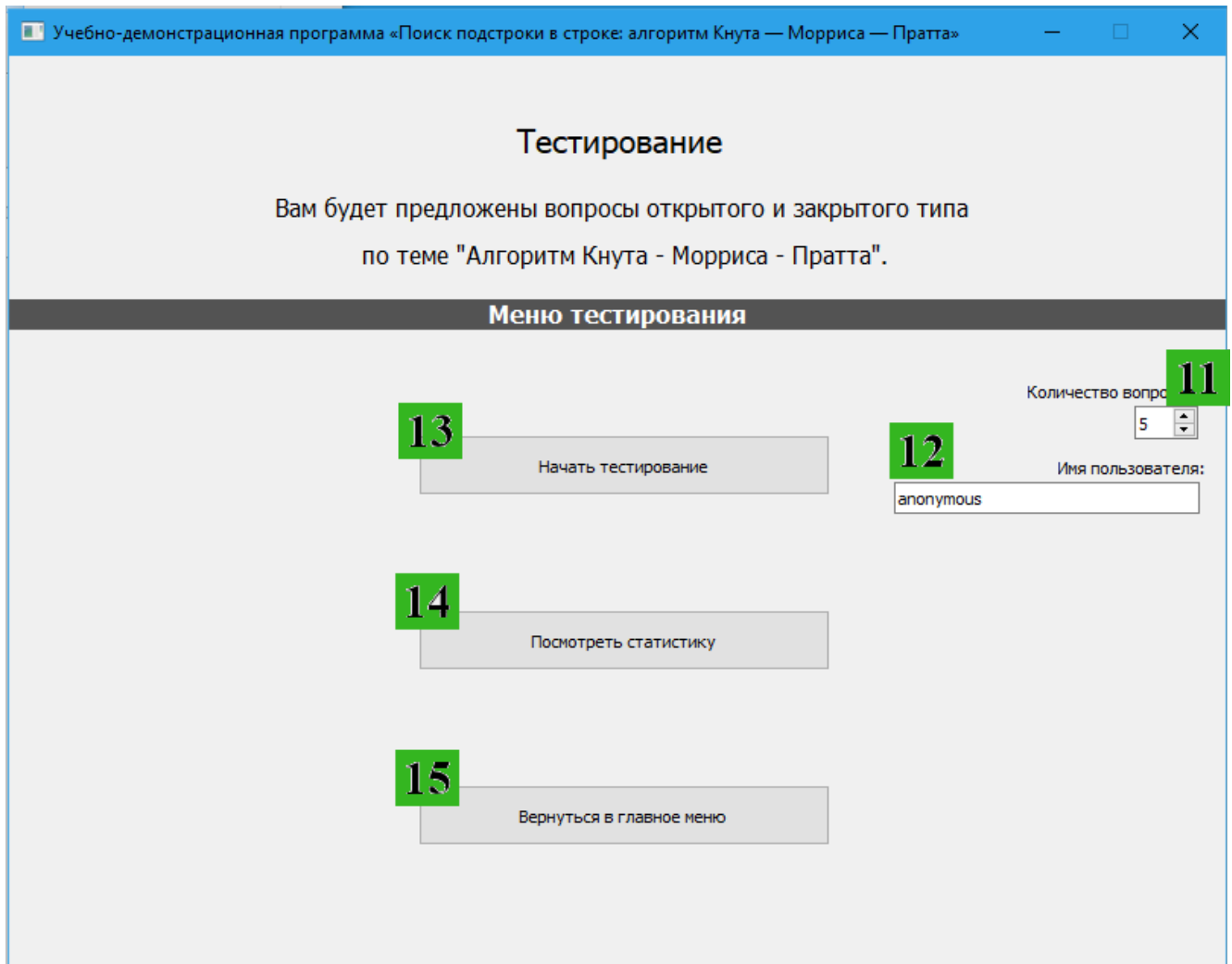


Рисунок 2.10 – Меню тестирования

После нажатия кнопки «13» начинается процесс тестирования, который может включать вопросы закрытого типа с 4-мя вариантами ответа и открытого типа с кратким ответом, который пользователь вводит с клавиатуры.

На рисунке 2.11 отображен вопрос закрытого типа.

Рисунок 2.11 – Вопрос закрытого типа

С помощью кнопки «16» можно завершить тестирование. Кнопки «17» и «18» используются для навигации, а чекбоксы «19» используются для того, чтобы пользователь мог дать ответ на вопрос. На рисунке 2.12 представлен вопрос открытого типа.

Рисунок 2.12 – Вопрос открытого типа

Текстовое поле «20» используется для того, чтобы пользователь мог дать краткий ответ на вопрос. При нажатии на кнопку «16» пользователю становятся доступны результаты тестирования (рис. 2.13).

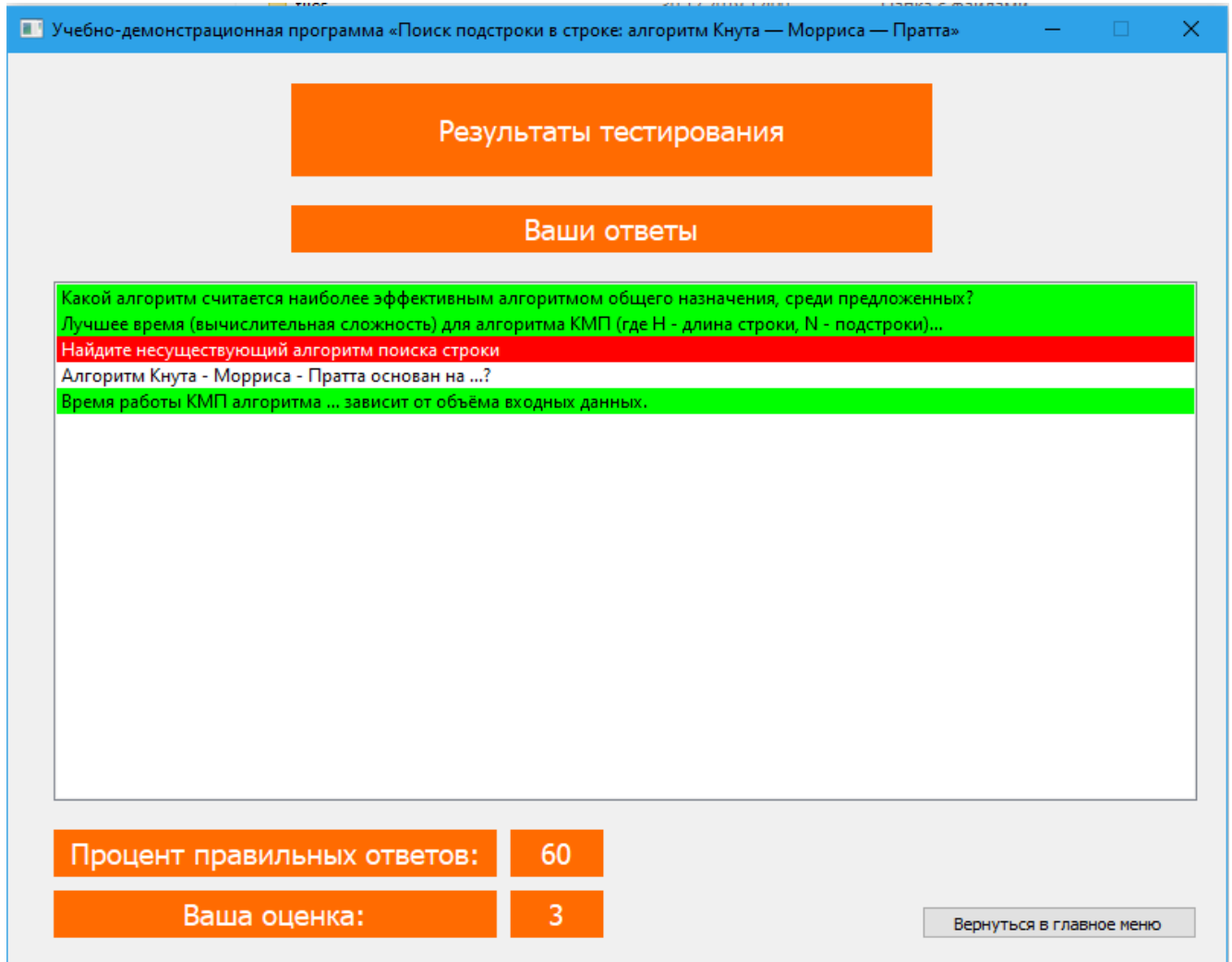


Рисунок 2.13 – Результаты тестирования

Зеленым цветом выделяются те вопросы, на которые был дан правильный ответ, красным выделяются те, на которые был дан неправильный ответ, а в случае, если вопрос был пропущен, то он не выделяется. В данном окне пользователь может ознакомиться с процентом правильных ответов и с оценкой. После нажатия на кнопку «21» происходит возврат в главное меню программы (см. рис. 2.1).

Если пользователь нажмет на кнопку «14» в меню тестирования, ему станет доступен просмотр статистики (рис. 2.14).

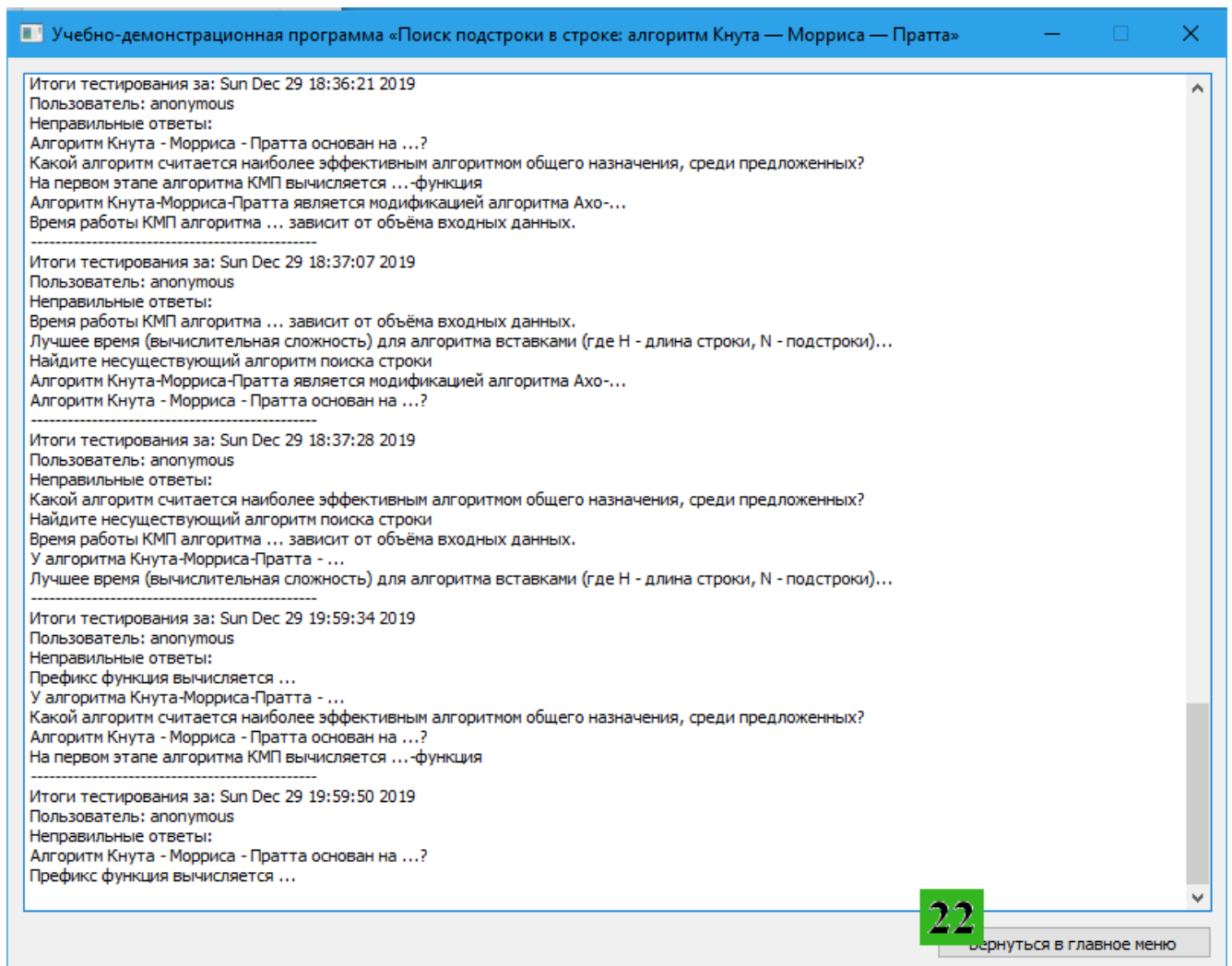


Рисунок 2.14 – Просмотр статистики тестирования

2.6 Сообщения системы

Программа сообщает об ошибках с помощью месседжбоксов. На рисунке 2.15 представлено сообщение об ошибке, возникающее в случае, если программа не может открыть файл с теорией.

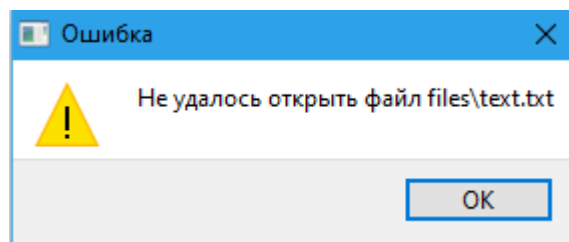


Рисунок 2.15 – Ошибка: не удалось открыть файл с теорией

На рисунке 2.16 представлено сообщение об ошибке, которое возникает, если программа не может открыть файл со словарем.

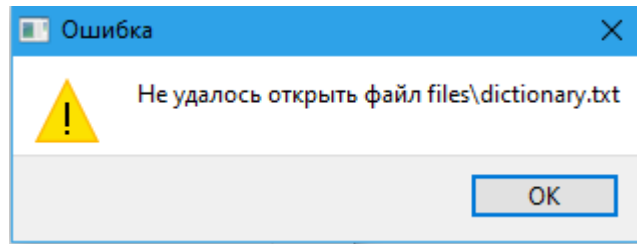


Рисунок 2.16 – Ошибка: не удалось открыть файл со словарем

На рисунке 2.17 представлена сообщение об ошибке, которое возникает, если программа не может открыть файл с базой вопросов.

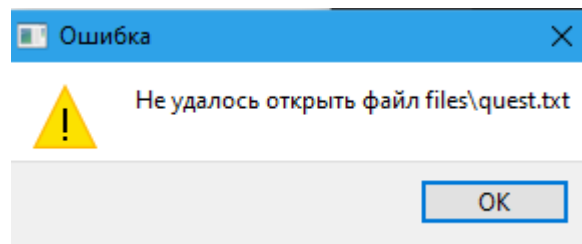


Рисунок 2.17 – Ошибка: не удалось открыть файл с базой вопросов

В случае появления других сообщений следует обратиться к разработчику.

3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

3.1 Проверка работоспособности режима чтения теории

1. Запустить программу на выполнение. Появится главное меню. (см. рис. 2.1).
2. Нажать кнопку «Перейти в режим чтения теории», убедиться, что программа перешла в режим чтения теории (см. рис.2.2).
3. Нажать кнопку «Вернуться в главное меню», убедиться, что произошел возврат в главное меню (см. рис. 2.1).
4. Нажать кнопку «X (закрыть окно)» в правом верхнем углу окна, убедиться, что программа завершила свою работу.

3.2 Проверка работоспособности режима демонстрации

1. Запустить программу на выполнение. Появится главное меню. (см. рис. 2.1).
2. Нажать кнопку «Перейти в режим демонстрации», убедиться, что на экране появилось меню демонстрации (см. рис. 2.3).
3. Попробовать ввести подстроку длиннее строки в поля «5» и «6» и убедиться, что программа не дает этого сделать.
4. Ввести строку и подстроку в поля «5» и «6», нажать кнопку «7», убедиться, что программа начала демонстрацию первого этапа алгоритма (см. рис. 2.4).
5. Нажать клавишу «стрелка вправо», убедиться, что программа демонстрирует следующий шаг (см. рис. 2.5).
6. Нажать клавишу «стрелка влево», убедиться, что программа демонстрирует предыдущий шаг (см. рис. 2.5).
7. Повторять пункт 5 пока демонстрация первого этапа алгоритма не будет закончена и не станет доступна для нажатия кнопка «Результат 1-го этапа» (см. рис. 2.6).
8. Нажать на кнопку «Результат 1-го этапа», убедиться, что программа приступила к демонстрации результата первого этапа алгоритма. Повторить пункты 5 и 6 (см. рис. 2.7).
9. Повторять пункт 5 пока демонстрация результата первого этапа не будет закончена и не станет доступна для нажатия кнопка «Этап 2» (см. рис. 2.8).
10. Нажать на кнопку «Этап 2», убедиться, что программа приступила к демонстрации второго этапа алгоритма. Повторить пункты 5 и 6 (см. рис. 2.9).
11. Повторять пункт 5 пока демонстрация результата второго этапа не будет закончена.

12. Убедиться, что алгоритм завершил свою работу и результат верный. Нажать кнопку «Завершить демонстрацию», убедиться, что произошел возврат в главное меню (см. рис. 2.1).
13. Нажать кнопку «X (закрыть окно)», убедиться, что программа завершила свою работу.

3.3 Проверка работоспособности режима тестирования

1. Запустить программу на выполнение. Появится главное меню. (см. рис. 2.1).
2. Нажать кнопку «Начать тестирование», убедиться, что на экране появилось меню тестирования (см. рис. 2.10).
3. Нажать кнопку «Начать тестирование», убедиться, что началось тестирование (см. рис. 2.11 или рис. 2.12)
4. Нажать кнопку «>», убедиться, что был совершен переход к следующему вопросу (см. рис. 2.11 или рис. 2.12)
5. Нажать кнопку «<», убедиться, что был совершен переход к предыдущему вопросу (см. рис. 2.11 или рис. 2.12)
6. Отметить чекбокс «19» в случае вопроса закрытого типа или ввести ответ в поле «20» (см. рис. 2.11 или рис. 2.12).
7. Нажать кнопку «Завершить», убедиться, что тестирование было завершено и был совершен переход к результатам тестирования, которые соответствуют выбранным ответам на вопросы (см. рис. 2.13).
8. Нажать кнопку «Вернуться в главное меню», убедиться, что был совершен возврат в главное меню программы (см. рис. 2.1).
9. Повторять пункты 2, 3, 4, 5, 6, 7 несколько раз, таким образом, чтобы получить оценки 2, 3, 4 и 5 в результатах тестирования (см. рис. 2.13).
10. Повторить пункт 2, после чего нажать кнопку «Посмотреть статистику», убедиться, что была открыта статистика тестирования.
11. Нажать кнопку «Вернуться в главное меню», убедиться, что был совершен возврат в главное меню программы (см. рис. 2.1).
12. Нажать кнопку «X (закрыть окно)», убедиться, что программа завершила свою работу.

ЗАКЛЮЧЕНИЕ

В результате курсового проектирования разработана учебно-демонстрационная программа «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта». Программа предоставляет теоретический материал по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта», визуализирует работу алгоритма, позволяет пройти тестирование, по окончании которого выводятся результаты тестирования.

Программа отвечает поставленным требованиям и может быть использована для обучения студентов, обучающихся программированию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вирт. Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD / Пер. с англ. Ткачев Ф. В. - М.: ДМК Пресс, 2010. – 272 с.: ил.
2. Кормен, Томас Х. и др. Алгоритмы: построение и анализ, 3-е изд. : Пер. с англ. – М.: ООО „И. Д. Вильямс“, 2013. – 1328 с. : ил. – Парал. тит. Англ.
3. Шлее М. Qt 5.10. Профессиональное программирование на C++. – СПб.: БХВ-Петербург, 2018. – 1072 с.: ил. – (В подлиннике)
4. Саттер Г., Александреску А. Стандарты программирования на C++.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005.— 224 с. :ил.— Парал. тит. англ.
5. Лаптев В. В. C++. Объектно-ориентированное программирование: Учебное пособие. – СПб.: Питер, 2008. – 464 с.: ил. – (Серия «Учебное пособие»).
6. Qt Documentation – [Электронный ресурс] режим доступа: <https://doc.qt.io> (27.10.2019)

ПРИЛОЖЕНИЕ 1**ТЕХНИЧЕСКОЕ ЗАДАНИЕ****на выполнение курсового проекта**

по дисциплине «Алгоритмы и структуры данных»

Направление 090304 – Программная инженерия

Исполнитель: обучающийся гр. ДИПР621 **Катунин С.С.**

**Тема: Учебно-демонстрационная программа «Поиск подстроки в строке:
алгоритм Кнута – Морриса – Пратта»**

1 Назначение, цели и задачи разработки

Цель разработки – автоматизация обучения и контроля знаний по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта».

Назначение разработки:

- повышение качества знаний пользователей;
- снижение нагрузки на преподавателя.

Основные задачи, решаемые разработчиком в процессе выполнения курсового проекта:

- анализ предметной области;
- разработка программного продукта в соответствии с требованиями;
- документирование проекта в соответствии с установленными требованиями.

2 Характер разработки: прикладная квалификационная работа.**3 Основания для разработки**

- Учебный план направления 09.03.04 «Программная инженерия» 2018 года набора.
- Рабочая программа дисциплины Программирование и информатика».
- Распоряжение по кафедре АСОИУ №____ от «__»_____ 201__г.

4 Плановые сроки выполнения – осенний семестр 2019/20 учебного года:

Начало « 15 » сентября 2019 г.

Окончание « 30 » декабря 2019 г.

5 Требования к проектируемой системе**5.1 Требования к функциональным характеристикам**

Проектируемая система представляет собой графическое приложение и должна обеспечивать выполнение следующих основных функций:

- Предоставление пользователю теоретического материала по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»
 - ✓ теоретический материал организован в виде гипертекста (HTML);
 - ✓ теоретический материал может включать: текст, таблицы, рисунки;

- ✓ определения, упомянутые, но не прописанные в теоретическом материале собраны в отдельный словарь и в тексте теоретического материала снабжены гиперссылками
- Визуализация работы алгоритма Кнута – Морриса – Пратта:
 - ✓ пошаговая демонстрация первого этапа алгоритма: на каждом шаге происходит отрисовка подстроки, индексов i и j , участвующих в работе алгоритме и значения массива π (префикс-функция) на экран; выделяются символы, соответствующие индексам i и j на данном шаге; после завершения демонстрации первого этапа будет предложено перейти к демонстрации результата первого этапа;
 - ✓ пошаговая демонстрация результата первого этапа алгоритма: на каждом шаге происходит отрисовка подстроки и значения массива π (префикс-функция) на экран; выделяются совпадающие суффиксы и префиксы, соответствующие данному шагу; после завершения демонстрации результатов первого этапа будет предложено перейти к демонстрации второго этапа;
 - ✓ пошаговая демонстрация результата второго этапа алгоритма: на каждом шаге происходит отрисовка строки, подстроки, индексов k и l , участвующих в работе алгоритма, значения массива π (префикс-функция) на экран; выделяются символ строки, соответствующий индексу k , и символ подстроки, соответствующий индексу l на данном шаге;
 - ✓ должна быть предусмотрена возможность просмотра предыдущих шагов;
 - ✓ пользователь задает строку и подстроку с клавиатуры.
- Тестирование пользователя по теме «Поиск подстроки в строке: алгоритм Кнута – Морриса – Пратта»:
 - ✓ вопросы закрытого типа с выбором одного правильного ответа, количество предлагаемых вариантов ответа – 4, при показе пользователю вопроса варианты ответа должны быть перемешаны;
 - ✓ вопросы открытого типа с кратким ответом;
 - ✓ при тестировании пользователю предлагается то количество вопросов, которое было указано перед тестированием, общее количество вопросов в базе не менее 10, база представляет собой текстовый файл;
 - ✓ формат файла и вопросы разрабатываются исполнителем самостоятельно;
 - ✓ перед тестированием пользователю должна быть предоставлена инструкция о проведении теста;
 - ✓ программа должна подсчитывать процент правильных ответов;
 - ✓ если ответ правильный, то к количеству правильных ответов добавляется 1 балл;
 - ✓ пользователь может пропустить вопрос с возможностью вернуться к нему;

- ✓ по завершении тестирования общий итог выводится как на экран, так и в текстовый файл со статистикой (с указанием данных о тестируемом, даты и времени прохождения теста, текста вопросов, в которых была допущена ошибка);
- ✓ тестирование считается успешным, если процент правильных ответов не менее 60%.
- **Интерфейс программы:** текст русский, шрифты кириллический и латинский, заголовки, термины и другая важная информация выделены цветом.

Система имеет функциональные ограничения:

- количество символов в строке/подстроке не может быть более 15;
- программа не предоставляет возможность шифровать / расшифровывать текстовые файлы с базой вопросов и с теорией;
- программа не должна обеспечивать редактирование статистики результатов тестирования.

5.2 Требования к эксплуатационным характеристикам

Программа не должна аварийно завершаться при любых действиях пользователя

Время реакции программы на действия пользователя не должно превышать 10 секунд.

5.3 Требования к программному обеспечению:

Средства разработки: интегрированная среда Qt Creator (версия Qt 5.4), язык C++ (стандарт C++ 11 и более поздние).

Операционная система: Windows XP (x86) с пакетом обновления 3 (SP3) или более поздние.

5.4 Требования к аппаратному обеспечению:

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 512 МБ ОЗУ;
- не менее 50 МБ свободного места на диске;
- Дисковод CD-ROM/DVD-ROM.

6 Стадии и этапы разработки

6.1 Эскизный проект (ЭП)

- Анализ предметной области.
- Подготовка проектной документации.

6.2 Технический проект (ТП)

- Разработка структур и форм представления данных.
- Разработка структуры программного комплекса.
- Подготовка пояснительной записки.

6.3 Рабочий проект (РП)

- Программная реализация.
- Тестирование и отладка программы.
- Подготовка программной и эксплуатационной документации.

6.4 Эксплуатация (Э)

Описание и анализ результатов проведенного исследования.

7 Требования к документированию проекта

К защите курсового проекта должны быть представлены следующие документы:

- Пояснительная записка к курсовому проекту:
- Презентация доклада.
- Программа, презентация и пояснительная записка к курсовому проекту на оптическом носителе.

Требования к структуре документов определены соответствующими стандартами ЕСПД.

Требования к оформлению определены соответствующими методическими указаниями.

8 Порядок контроля и приемки

Контроль выполнения курсового проекта проводится руководителем поэтапно в соответствии с утвержденным графиком выполнения проекта.

На завершающем этапе руководитель осуществляет нормоконтроль представленной исполнителем документации и принимает решение о допуске (недопуске) проекта к защите.

Защита курсового проекта проводится комиссией в составе не менее двух человек, включая руководителя проекта.

В процессе защиты проекта исполнитель представляет документацию, делает краткое сообщение по теме разработки и демонстрирует ее программную реализацию.

При выставлении оценки учитывается:

- степень соответствия представленной разработки требованиям технического задания;
- качество программной реализации, документации и доклада по теме проекта;
- соблюдение исполнителем графика выполнения курсового проекта.

9 Литература

1. Вирт. Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD / Пер. с англ. Ткачев Ф. В. - М.: ДМК Пресс, 2010. – 272 с.: ил.
2. Кормен, Томас Х. и др. Алгоритмы: построение и анализ, 3-е изд. : Пер. с англ. – М.: ООО „И. Д. Вильямс“, 2013. – 1328 с. : ил. – Парал. тит. Англ.
3. Шлее М. Qt 5.10. Профессиональное программирование на C++. – СПб.: БХВ-Петербург, 2018. – 1072 с.: ил. – (В подлиннике)

ПРИЛОЖЕНИЕ 2

Диаграмма классов

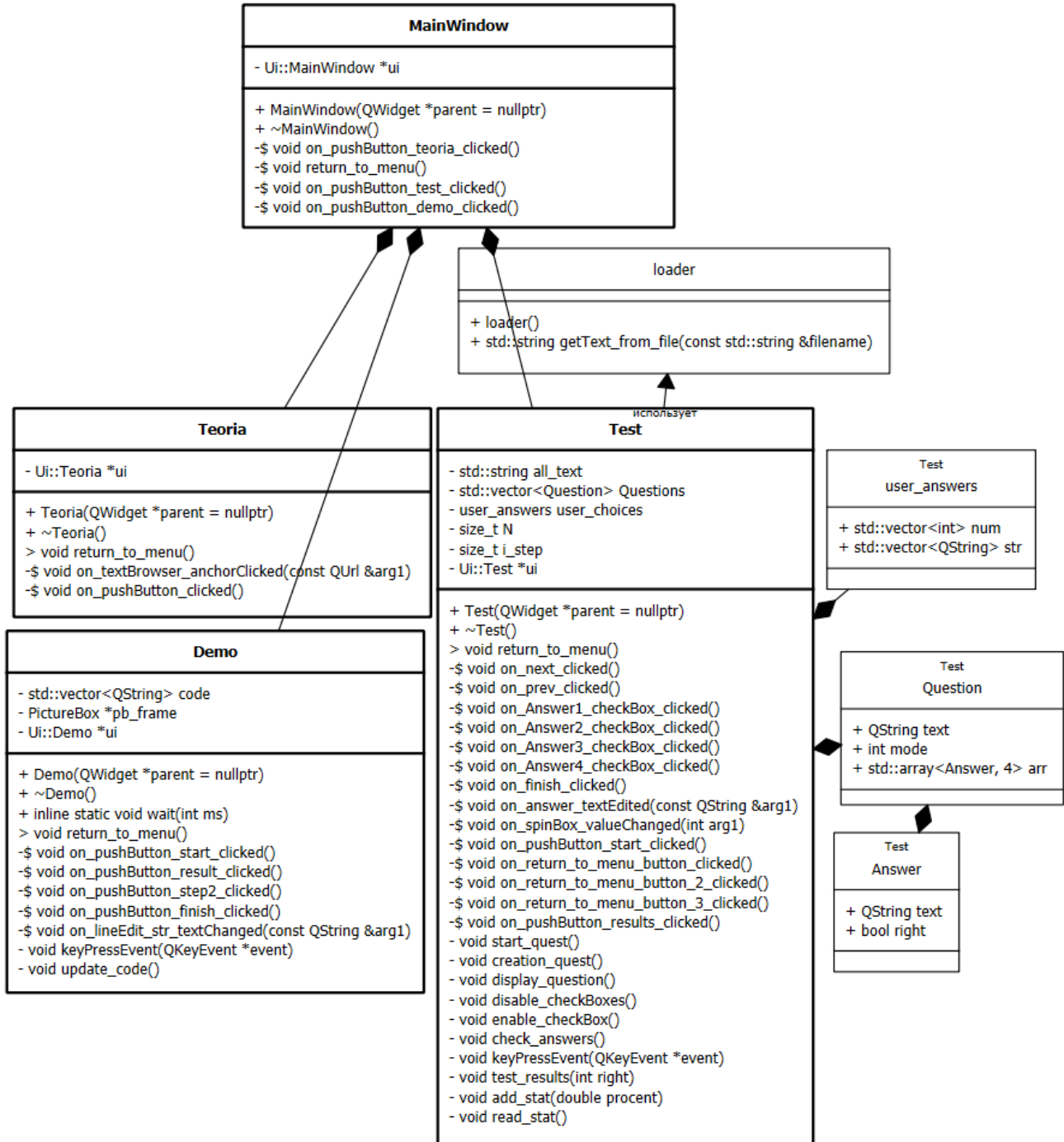


Рисунок П.2.1 — Диаграмма классов (1 часть)

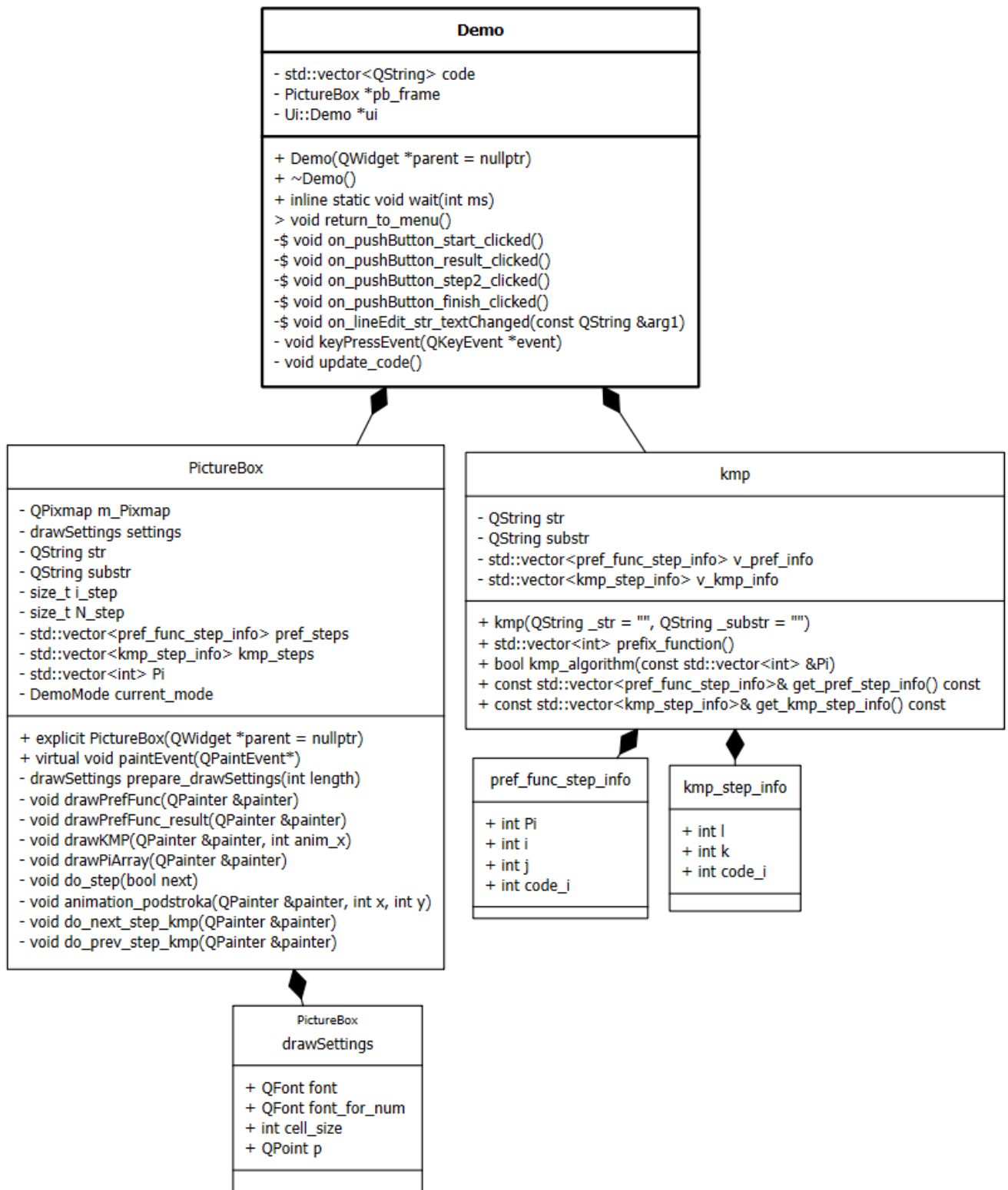


Рисунок П.2.2 — Диаграмма классов (2 часть)

ПРИЛОЖЕНИЕ 3**База текстовых вопросов
(содержание файла quest.txt,
первый ответ является правильным)**

?1Алгоритм Кнута - Морриса - Пратта основан на ...?

сравнении с начала

сравнении с конца

сравнении как "чёрном ящике"

сравнении в необычном порядке

?1Какой алгоритм считается наиболее эффективным алгоритмом общего назначения, среди предложенных?

Алгоритм Бойера-Мура

Алгоритм Кнута-Морриса-Пратта

Алгоритм Рабина-Карпа

Алгоритм Апостолико-Джанкарло

?2На первом этапе алгоритма КМП вычисляется ...-функция

префикс

?1У алгоритма Кнута-Морриса-Пратта - ...

отсутствует регрессия на "плохих" данных

высокая скорость на "хороших" данных

отсутствует предварительная обработка подстроки

используется таблица стоп-символов

?1Найдите несуществующий алгоритм поиска строки

алгоритм Кнута-Бойера-Мура

алгоритм Кнута-Морриса-Пратта

алгоритм Бойера-Мура-Хорспула

алгоритм Ахо-Корасик

?1Префикс функция вычисляется ...

для подстроки

для строки

с использованием таблицы стоп-символов

с использованием хеширования

?1На каком из предложенных языков программирования невозможно реализовать алгоритм Кнута-Морриса-Пратта?

возможно на всех предложенных

C++

Python

Java

?1 Лучшее время (вычислительная сложность) для алгоритма КМП (где H - длина строки, N - подстроки)...

$\leq O(2 * H)$

$O(H)$

$\leq O(1.5 * H)$

$O(H * N)$

?2 Алгоритм Кнута-Морриса-Пратта является модификацией алгоритма Ахо-...

Корасик

?2 Время работы КМП алгоритма ... зависит от объёма входных данных.

линейно