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 «Разрботка программно-информационных систем»
Кафедра «Автоматизированные системы обработки информации и управления»

КУРСОВОЙ ПРОЕКТ

Учебно-демонстрационная программа

«Двусвязный список»

по дисциплине «Программирование и информатика»

Допущен к защите
«__» _____20__г.
Руководитель
_____

Проект выполнен
обучающимся группы ДИПРб-11
Тагировым Р.Р.
_____

Оценка, полученная на защите
«_____»

Руководитель
ст. преп. Толасова В.В.

Астрахань – 2020

ФЕДЕРАЛЬНОЕ АГЕНТСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ ПО РЫБОЛОВСТВУ

АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Заведующий кафедрой
к.т.н., доцент
Т. В. Хоменко _____
«____»_____202 __ г.

Кафедра
«Автоматизированные системы
обработки информации и управления»

ЗАДАНИЕ

на выполнение курсового проекта

Обучающийся ***Тагиров Руслан Расимович***

Группа ***ДИПРб-11***

Дисциплина ***Программирование и информатика***

Тема ***Учебно-демонстрационная программа «Двусвязный список»***

Дата получения задания «_____»_____202___г.

Срок представления обучающимся КП на кафедру «_____»_____202___г.

Руководитель ***ст. преподаватель***_____ ***Толасова В.В.*** «_____»_____202___г.

должность, степень, звание подпись ФИО

Обучающийся _____ ***Тагиров Р.Р.*** «_____»_____202___г.

подпись ФИО

Задачи

Разработка программного продукта, который

- Позволяет просматривать теорию по теме «Двусвязный список»
- Показывает пользователю визуализацию списка
- Предоставляет пользователю тест по теме «Двусвязный список»

Рекомендуемая литература

1. Прата С. Язык программирования C++. Лекции и упражнения – М.: ООО «И.Д. Вильямс» 2016 – 1248 стр.
2. Васильев А. Программирование на C++ в примерах и задачах – М.: Эксмо, 2018 – 368с.
3. Белов С.В., Лаптев В.В., Морозов А.В., Толасова В.В., Мамлеева А.Р. Требования к оформлению студенческих работ. - Астрахань, АГТУ, 2017. 104 с.

УТВЕРЖДАЮ
Заведующий кафедрой

к.т.н., доцент

Т.В. Хоменко_____

«____»_____20 ___г.

К заданию на курсовой проект

по дисциплине

«Программирование и информатика»

КАЛЕНДАРНЫЙ ГРАФИК
курсового проектирования

№ п/п	Разделы, темы и их содержание, графический материал	Дата сдачи	Объем, %
1	Выбор темы	25.02.2020	1
2	Техническое задание	11.03.2020	3
3	Разработка модели, проектирование системы • *введение,* • *технический проект,* • *программа и методика испытаний,* • *литература*	15.04.2020	25
4	Программная реализация системы • *работающая программа,* • *рабочий проект* • *скорректированное техническое задание (при необходимости)*	13.05.2020	40
5	Тестирование и отладка системы, эксперименты *работающая программа с внесёнными изменениями, окончательные тексты всех разделов*	20.05.2020	50
6	Компоновка текста Подготовка презентации и доклада *пояснительная записка* *презентация* *электронный носитель с текстом пояснительной записки, исходным кодом проекта, презентацией и готовым программным продуктом*	27.05.2020	59
7	Защита курсового проекта	03.06.2020-07.06.2020	60-100

С графиком ознакомлен «____» _____ 20 ___г.
Тагиров Р.Р._____, обучающийся группы ДИПРб-11
(фамилия, инициалы, подпись)

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

нужное подчеркнуть

Руководитель курсового проекта_____ ст. преподаватель Толасова В.В.
подпись, ученая степень, звание, фамилия, инициалы

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Структуры данных — неотъемлемая часть любой сложной программы. Они способны увеличить эффективность при работе с данными, а также облегчить работу с ними. Правильно подобранные структуры данных — первый шаг к созданию стабильного программного продукта. Неудивительно, что знание внутреннего устройства, а также умение работать со структурами данных — один из критериев, по которым можно отличить опытного программиста. Даже при приеме на работу программиста обязательно попросят реализовать какую-нибудь структуру данных.

Одной из важнейших структур данных является двусвязный список. Его преимущество в том, что он позволяет за константное время добавлять и удалять данные, а также то, что в памяти данные не обязательно должны располагаться последовательно, за счет чего достигается максимальная гибкость при работе с ним.

Однако существует много людей, которые готовы использовать его даже не зная, как он устроен. Каждая структура данных имеет свои преимущества и недостатки, и такое незнание зачастую приводит к созданию крайне неэффективного кода.

Причем было бы гораздо удобнее изучать эту тему, если бы вся необходимая информация вкупе с системой тестов и визуализацией находились в одном месте, тогда бы пользователю не пришлось отвлекаться ни на что Проходя тест, пользователь мог бы проверить, насколько хорошо он знает двусвязный список, а, глядя на визуализацию, он сможет наглядно проследить изменения, происходящие со списком во время выполнения различных операций.

Целью создания учебно-демонстрационной программы «Двусвязный список» является автоматизация процесса обучения использованию двусвязного списка.

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

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

1.1.1 Двусвязный список

Базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и два указателя («связки»), содержащие адреса следующего и предыдущего узла списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

На рисунке 1.1 изображено строение двусвязного списка

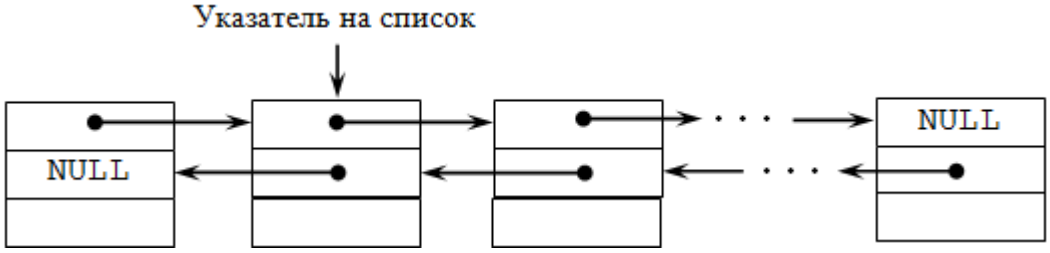


Рисунок 1.1 – Двусвязный список

Головой списка называется указатель на какой-то из узлов списка (чаще всего первый), также у списка может быть взят указатель на хвост — последний узел списка.

Над списком возможны операции: добавления в начало, добавления в середину, удаления из начала и удаления из середины, а также, если используется реализация с указателем на конец списка, то доступны операции добавления после последнего и удаления последнего элемента списка.

На рисунке 1.2 изображена схема добавления в начало списка

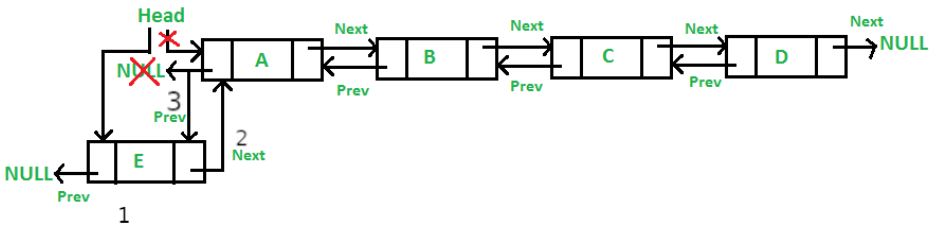


Рисунок 1.2 – Добавление в начало списка

На рисунке 1.3 изображена схема добавления в середину списка

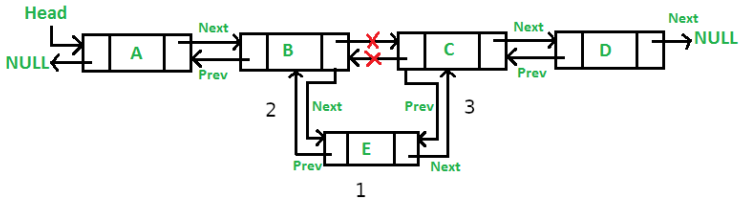


Рисунок 1.3 – Добавление в середину списка

На рисунке 1.4 изображена схема добавления в конец списка

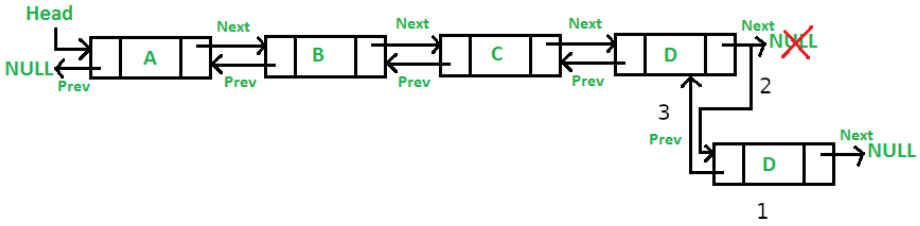


Рисунок 1.4 – Добавление в конец списка

На рисунке 1.5 изображена схема удаления

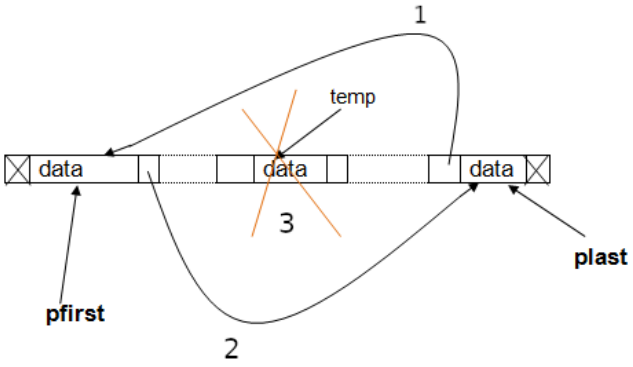


Рисунок 1.5 – Удаление элемента из списка

1.1.2 Визуализация

Для большей наглядности в программе будет предусмотрена визуализация работы двусвязного списка. Пользователю будут предложены на выбор 7 операций: добавить в начало, удалить из начала, добавить после, удалить после, удалить из конца, добавить в конец и найти элемент двусвязного списка с данным содержимым.

На рисунке 1.6 изображено как будет выглядеть визуализация

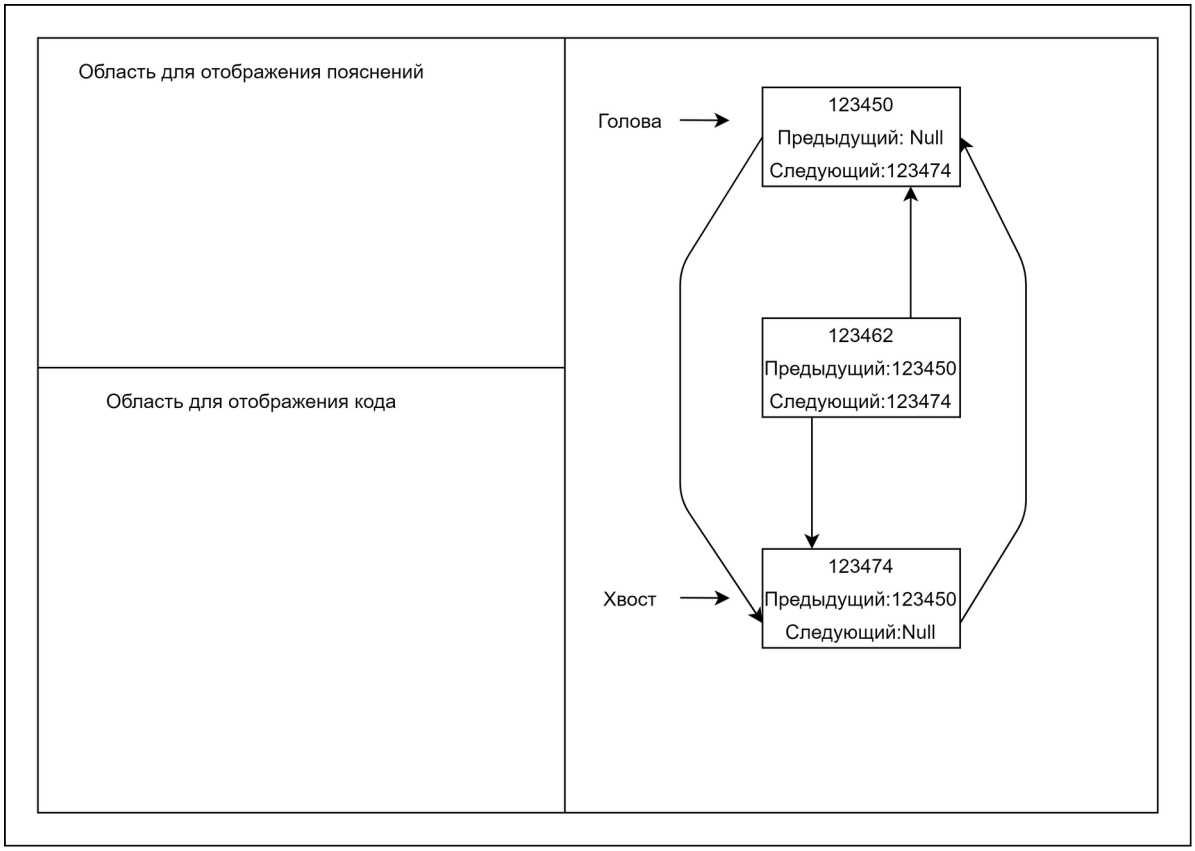


Рисунок 1.6 – Эскиз визуализации

Все действия, производимые со списком, будут отображаться на экране в трех различных областях окна. В перовой области будут визуально изображены узлы и связи между ними (адресное поле , указатель на следующий и предыдущий и поле данных). Во втором будет показан код, реализующий данную операцию и будет выделена строчка кода, которая сейчас выполняется. А в третьем разделе будет описываться что выполняется со списком в данный момент из зачем это происходит.

Список выводится в виде прямоугольников, изображающих узлы списка и линий, изображающих связи между узлами. Между узлами будет 2 связи, означающие указатель на следующий и на предыдущий узлы соответственно. Эти связи будут отражать содержимое соответствующих полей узла.

На каждом шаге у пользователя будет запрашиваться ввод. Пользователь может продвинуть визуализацию на шаг вперед или посмотреть предыдущие шаги. Пользователь может откатываться на произвольное число шагов назад и вперед вплоть до последнего и первого шагов.

1.1.3 Тестирование

В программе будет предусмотрена система тестов по теме «Двусвязный список». Сначала пользователю будет предложено прочитать теорию в виде текста расположенного на 4 страницах, по которым он сможет свободно перемещаться. Так же ему будет предложено пройти небольшой тест по данной теме состоящий из 5 вопросов, в каждом из которых предусмотрено больше двух вариантов ответа, один или несколько из которых будут верными. Вопросы будут браться из базы с вопросами, в которой будет находится 10 вопросов, вопросы будут браться в случайном порядке. Перед прохождением теста, пользователю нужно ввести свое имя. Во время теста пользователь будет с начала помечать ответы, которые он считает верными, а после отправлять их на проверку. Ответ пользователя считается неверным, если хотя бы один из выбранных им вариантов оказался неверным, за ответ с ошибками баллы не начисляются. Пройдя тест он увидит свои результаты (бинарную оценку — прошел/не прошел тест). Также если на какой-то вопрос он ответит неверно, он увидит сообщение об этом. После прохождения теста его результаты (дата и время проведения имя пользователя и результаты теста) будут записаны в специальный файл. Тест длится 30 минут, если пользователь не укладывается в это время — тест считается не пройденным.

1.1.4 Шифрование

Программе нужно хранить базу с вопросами для теста, а так же теорию к ним в файлах, но продвинутый пользователь компьютера может открыть их и посмотреть ответы. Это недопустимо. Поэтому перед использованием базу вопросов нужно сделать не читаемой.

1.2 Технология обработки информации

Анализ предметной области показал, что программа рассчитана на одного пользователя. На рис. 1.7 показана диаграмма вариантов использования.

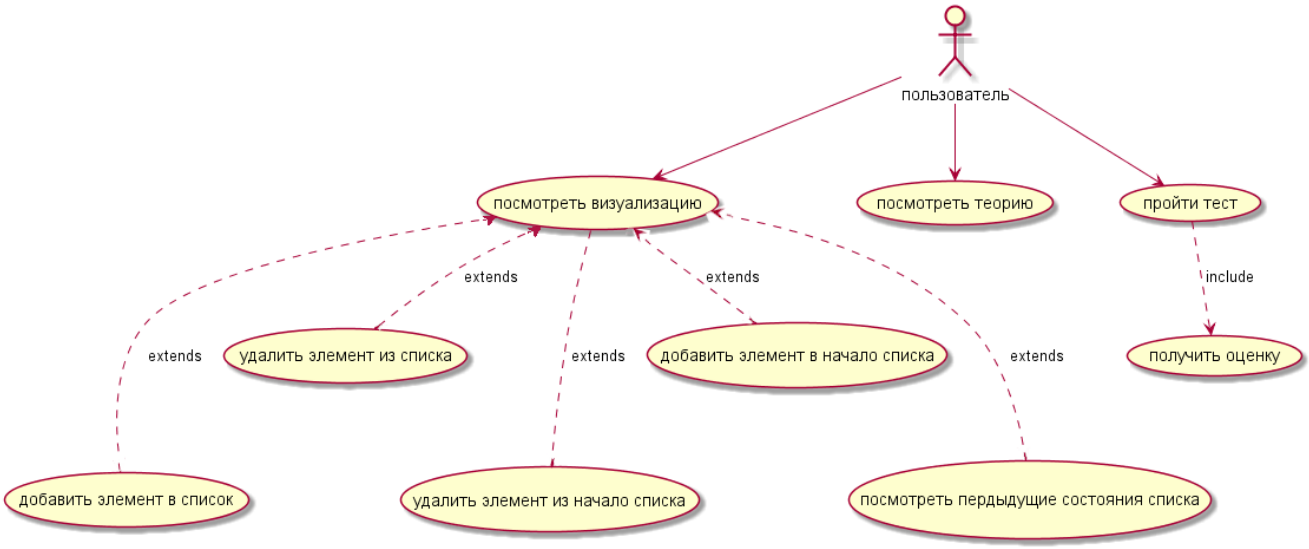


Рисунок 1.7 – Диаграмма вариантов использования

1.2.1 Форматы файлов с вопросами и теорией

Для хранения данных теории и базы с вопросами в программе будут предусмотрены специальные текстовые файлы.

Файлы с теорией будут иметь следующий формат разметки: каждая страница в них будет отделяться новой строкой с тремя символами «+».

Структура файлов с вопросами будет следующей: на строчке N будет находиться текст в на следующей строчке будет находиться его стоимость (сколько баллов за него присуждается) на последующих строчках будут варианты ответа, правильные варианты нужно выделить знаком «+» перед вариантом, а отделяются вопросы стремя знаками «#»

На рисунке 1.8 изображена структура файла с вопросами.

```
За какое время происходит добавление элемента в двусвязный список?
1
+за O(n)
за O(1)
за O(n^2)
###
За какое время происходит удаление элемента из двусвязного списка?
1
за O(1)
+за O(n)
за O(n^2)
###
За какое время происходит добавление/удаление в начале списка?
```

Рисунок 1.8 Формат файла с вопросами

1.2.2 Шифрование

Чтобы сделать базу с вопросами нечитаемой обычными средствами, нужно в ней каждый байт изменить операцией XOR с определенным ключом, в программе в качестве ключа будет число 42. В программе будет предусмотрена специальная постоянная-ключ, отвечающая за дешифровку, а файлы будут шифроваться с использованием специальной программы или любого другого XOR-шифрователя. Даже зная ключ, пользователь не сможет узнать ответы на вопросы.

1.2.3 Визуализация

Для корректного отображения визуализации, в программе будут следующие константы:

VSIZE = (вертикальная длина окна консоли в символах)
HSIZE (горизонтальная длина окна консоли в символах)
NODE_WIDTH 30
NODE_HEIGHT 6
DISTANCE_BETWEEN_NODES 4
NODE_SUMMARY_SIZE (DISTANCE_BETWEEN_NODES + NODE_HEIGHT)
NODE_ORIGIN (HSIZE / 4)
Они отвечают за то, как будут выглядеть визуальные представления узлов.

Для реализации алгоритмов необходима возможность перемещения указателя консоли на нужные координаты. Началом координат является верхний левый угол консоли. Ось ординат направлена вниз. В дальнейшем будет так же использоваться вывод относительно каких-либо координат, что значит, что результирующие координаты будут получаться путем сложения

данных координат и координат смещения. К примеру: «Вывести символ «A» на позиции (2, 1) относительно w», где w = (3, 5) , означает , что символ A должен быть выведен на позиции (2+3, 1+5). Если смещение не указано, то подразумевается, что вывод происходит относительно начала координат.

1.2.4 Алгоритмы

1.2.2.1 Алгоритм расшифровки файлов

Дано: имя файла

Вернуть расшифрованный файл в виде строки

1. Открыть файл как бинарный
2. Объявить строку res
3. Если это конец файла то перейти к шагу 8
4. Прочитать символ из файла
5. Произвести с ним операцию побитового ИЛИ и ключом 42
6. Прибавить символ к строке res
7. Перейти к шагу 3
8. Вернуть res

1.2.4.3 Алгоритм чтения файла с теорией

Вернуть: pages – массив строк-страниц теории, page_count – количество страниц

1. Расшифровать файл с теорией в строковый поток theoryfile (см. 1.2.2.0)
2. page_count = 0
3. Если это конец theoryfile, перейти к шагу 8
4. Объявить строку current_page
5. Прочитать строку из theoryfile
6. Если строка не равна «+++», то прибавить ее к current_page и перейти к шагу 4
7. Присвоить current_page pages[page_count] и увеличить page_count на 1
8. Перейти к шагу 2
9. Конец

1.2.4.4 Алгоритм чтения файла с вопросами

Вернуть: question_count – количество вопросов, questions – массив строк-вопросов, options – массив вариантов ответа, answers – массив ответов, costs – массив цен вопросов

1. question_count = 0
2. Расшифровать файл с вопросами в строку questionfile (см. 1.2.2.0)
3. Если это конец файла, то перейти к шагу 16
4. Объявить строчку question
5. Прочитать в нее строку из questionfile
6. questions[question_count] = question
7. Считать число в cost[question_count]
8. Прочитать строку, и если эта строка не равна «###» то сохранить ее в options[question_count][options_count[question_count]] и увеличить options_count[question_count] на 1
9. answer = 0
10. mul = 1
11. Для каждого I пробегающего от 0 до options_count[question_count]:
    Если options[question_count][i][0] равен «+» то прибавить к answer mul и убрать первый символ из options[question_count][i][0]. Умножить mul на 2
12. answers[question_count] = answer
13. Увеличить question_count на 1

1.2.4.5 Алгоритм вывода вопроса

Дано: question – номер вопроса, choose – выбранный вариант ответа, answer - отмеченные варианты ответа, options_count – массив количеств вариантов ответа, questions – массив вопросов

1. Вывести строку с форматированием "Вопрос: %s" questions[question] на позиции (0, 5)
2. Получить текущее положение координаты у указателя консоли и присвоить его переменной offset
3. Вывести строку "Варианты ответа:" на позиции (0,offset + 1)
4. Вывести симовол '>' на позиции (0, offset + 2 + choose)
5. Для каждого целого I пробегающего от 0 до options_count[question] выполнить:
    1. Если I равно choose вывести «[выбрано]» используя на позиции (1,offset + 2 + i)
    2. Вывести номер варианта ответа равный I + 1 используя на позиции

(1,offset + 2 + i)

3. Вывести вариант ответа из options[question][i] используя на позиции (1 ,offset + 2 + i)

1.2.4.6 Алгоритм показа теории

Дано: pages – массив страниц теории page_count – количество страниц

Перед использованием функции должны быть вызваны функция чтения файла теории (см. 1.2.2.1)

1. current_page = 0
2. Вывести строку pages[current_page % page_count] на позиции (0,0)
3. Вывести строку "Нажмите стрелку влево, чтобы листать влево и стрелку вправо, чтобы листать вправо или нажмите q, чтобы выйти" на позиции (0,VSIZE - 2)
4. Вывести строку с форматированием "Вы на странице: %i", current_page % page_count на позиции (VSIZE - 1,0)
5. choice = 0
6. Если choice равен q, то очистить экран консоли
7. Если choice равен KEY_LEFT, то current_page = (current_page – 1)
8. Если choice равен KEY_RIGHT, то current_page = (current_page – 1)
9. Очистить консоль
10. Вывести строку pages[current_page % page_count] на позиции (0, 0)
11. Вывести строку "Нажмите стрелку влево, чтобы листать влево и стрелку вправо, чтобы листать вправо или нажмите q, чтобы выйти" на позиции (0,VSIZE- 2)
12. Вывести строку с форматированием "Вы на странице: %i", current_page % page_count на позиции (0, VSIZE - 1)
13. Ожидать ввода клавиши, и сохранить ее в choice
14. Перейти к шагу 10

1.2.4.7 Алгоритм тестирования

Перед использованием функции должна быть вызвана функция чтения базы с вопросами (см. 1.2.4.3)

Дано: question – номер вопроса, choose – выбранный вариант ответа, answer - отмеченные варианты ответа, options_count – массив количеств вариантов ответа, questions – массив вопросов, costs – массив стоимостей вопросов

1. choose = 0

2. question = 0
3. solved_count = 0
4. total_passed = 0
5. user_answer = 0
6. Вывести строку «Введите имя пользователя» на позиции (0, 0)
7. Запросить у пользователя ввод строки и сохранить ее в переменную username
8. Сохранить текущее время и дату в переменную begin
9. Выбрать случайный question меньший question_count
10. Вывести строку "Нажимайте стрелку вниз или стрелку вверх, чтобы листать варианты ответа." на позиции (0, 0)
11. Вывести строку "Нажимите стрелку вправо, чтобы выбрать варианты ответа" на позиции (0, 1)
12. Вывести строку "Нажмите enter, чтобы отправить ответ или стрелку влево, чтобы пропустить вопрос и вернуться к нему позже." на позиции (0, 2)
13. Вывести строку "Или нажмите q чтобы досрочно выйти из теста." на позиции (0, 3)
14. choice = 0
15. Если total_passed больше или равен 5 , то перейти к шагу 27
16. Очистить окно консоли
17. Вывести строку "Нажимайте стрелку вниз или стрелку вверх, чтобы листать варианты ответа." на позиции (0, 0)
18. Вывести строку "Нажимите стрелку вправо, чтобы выбрать вариант ответа" на позиции (0, 1)
19. Вывести строку "Нажмите enter, чтобы отправить ответ или стрелку влево, чтобы пропустить вопрос и вернуться к нему позже." на позиции (0, 2)
20. Вывести строку "Или нажмите q чтобы досрочно выйти из теста." на позиции (0, 3)
21. Если choice равен KEY_LEFT, то выбрать такой случайный question меньший question_count, что не будет выполняться solved[question], и обнулить choose и user_answer
22. Если choice равен KEY_RIGHT, то инвертировать бит choose числа user_answer
23. Если choice равен KEY_UP, то choose = (choose - 1) % options_count[question]
24. Если choice равен KEY_DOWN, то choose = (choose + 1) % options_count[question]
25. Если choice равен «\n» то:
    25.1 Если user_answer равен answers[question], то увеличить solved_count на 1 иначе вывести на позиции (0,10) строку «Ответ неверный»

25.2 solved[question] = true;

25.3 Увеличить total_passed на 1

25.4 Выбрать такой случайный question меньший question_count, что не будет выполняться solved[question], и обнулить choose и user_answer

26. Если choice равен «q» то: увеличить total_passed до ста
27. Вывести вопрос (см. 1.2.4.4)
28. Ожидать ввода клавиши, и сохранить ее в choice
29. Перейти к шагу 15
30. Очистить экран консоли
31. Сохранить текущее время и дату в переменную end
32. Открыть файл result.dat как текстовый на дозапись и записать в него содержимое переменных begin и username через пробел
33. Если solved_count меньше 5 или время длительности теста больше пулу часа (отнять от end begin) вывести "Вы не прошли тест" в консоль и в файл иначе вывести "Вы прошли тест" в консоль и в файл
34. Вывести в файл знак перевода строки
35. Закрыть файл
36. Очистить экран консоли

1.2.4.8 Алгоритм сохранения содержимого прямоугольной области в строку

Дано: width, height — высота и ширина области, xoffset yoffset — координаты смещения

Вернуть: строку save – содержимое заданной прямоугольной области

1. Пустая строка save
2. Для j , пробегающего от 0 до height выполнить:
   2.1 Для i пробегающего от 0 до width прочитать символ на позиции с экрана xoffset + i yoffset + j и прибавить его к строке save
   2.2 Добавить к строке символ „\n“
3. Вернуть save

1.2.4.9 Алгоритм рисования вертикальной линии относительно данных координат

Дано: offset — координаты смещения (если не указаны, значит относительно нынешних координат указателя), x y – начальные координаты линии. L – длина линии, C — символ.

Выполнить:

Для i пробегающего от 0 до L вывести символ C на позиции (x, y + I) относительно offset

1.2.4.10 Алгоритм рисования горизонтальной линии относительно данных координат

Дано: offset — координаты смещения(если не указаны, значит относительно нынешних координат указателя), x y – начальные координаты линии. L – длина линии,

C — символ.

Выполнить:

Для I пробегающего от 0 до L вывести символ C на позиции (x+ i, y) относительно offset

1.2.4.11 Алгоритм рисования прямоугольника относительно данных координат

Дано: w — координаты смещения, rheight – высота прямоугольника, rwidth – ширина прямоугольника, x y – начальные координаты прямоугольника

1. Нарисовать горизонтальную линию (см. 1.2.4.9) относительно w на позиции (x,y) длиной width
2. Нарисовать горизонтальную линию относительно w на позиции (x,y + height) длиной width
3. Нарисовать вертикальную (см. 1.2.4.8) линию относительно w на позиции (x + width,y) длиной height
4. Нарисовать вертикальную линию относительно w на позиции (x,y) длиной height
5. Вывести симовол '+' относительно w на позиции (x,y)
6. Вывести симовол '+' относительно w на позиции (x,y+height)
7. Вывести симовол '+' относительно w на позиции (x+width,y)
8. Вывести симовол '+' относительно w на позиции (x+width,y+height)

1.2.4.12 Алгоритм очистки прямоугольной области

Дано: w — координаты смещения, width height – ширина и высота области, x y – начальные координаты области.

Выполнить:

Для i пробегающего от 0 до width и j , пробегающего от 0 до height выполнить

Вывести символ ' ' на позиции (x + i, y + j) относительно w

1.2.4.13 Алгоритм перемещения указателя в начало стрелки, указывающей на следующий узел узла с данным номером

Дано: w — координаты смещения, порядковый номер узла в списке - node_id

Выполнить:

Переместить указатель относительно w на координаты

(NODE_ORIGIN + 10,NODE_SUMMARY_SIZE * node_id - DISTANCE_BETWEEN_NODES)

1.2.4.14 Алгоритм перемещения указателя в начало стрелки, указывающей на предыдущий узел узла с данным номером

Дано: w — координаты смещения, порядковый номер узла в списке - node_id

Выполнить:

Переместить указатель относительно w на координаты

(NODE_ORIGIN — 10,NODE_SUMMARY_SIZE * node_id + NODE_HEIGHT)

1.2.4.15 Алгоритм сохранения отката

Дано: loopbacks – массив откатов desc_window- смещение области с описанием deb_window — смещение области просмотра кода vis_window — смещение области с визуализацией.

1. Массив из 3 строк windows;
2. Сохранить содержимое прямоугольной области (см. 1.2.4.7) относительно desc_window размером HSIZE*3/8 на VSIZE/2 в windows[0]
3. Сохранить содержимое прямоугольной области относительно deb_window размером HSIZE*3/8 на VSIZE/2 в windows[1]
4. Сохранить содержимое прямоугольной области относительно vis_window размером HSIZE*5/8 на VSIZE в windows[2]
5. Добавить window в loopbacks

1.2.4.16 Алгоритм вывода отката

Дано: loopback_id - номер отката loopbacks – массив откатов desc_window — смещение области с описанием deb_window — смещение области просмотра кода vis_window — смещение области с визуализацией.

1. Вывести loopback[loopback_id][0] на позиции 0,0 относительно desc_window
2. Вывести loopback[loopback_id][1] на позиции 0,0 относительно deb_window
3. Вывести loopback[loopback_id][2] на позиции 0,0 относительно vis_window

1.2.4.17 Алгоритм вывода узла

Дано: node *n – указатель на узел, position — номер узла в списке, offset — смещение относительно центра окна (по умолчанию 0) loopbacks – массив откатов vis_window — смещение области с визуализацией.

1. node_y = position * NODE_SUMMARY_SIZE
2. node_x = NODE_ORIGIN - NODE_WIDTH / 2 + offset
3. Нарисовать прямоугольник (см. 1.2.4.10) относительно vis_window на позиции

(node_x,node_y) и с шириной и высотой (NODE_WIDTH, NODE_HEIGHT)

4. строка s_address = перевести n в строку как адрес
5. Вывести строку s_address относительно vis_window на позиции node_y + 1, NODE_ORIGIN + offset
6. строка s_prev = «Предыдущий = » + (преобразовать в строку n→previous)
7. Вывести строку s_prev относительно vis_window на позиции node_x + 1,node_y + 2
8. строка s_next = "следующий = " + (преобразовать в строку n→next)
9. Вывести строку s_next относительно vis_window на позиции node_x + 1,node_y+3
10. строка s_data = "данные = " + (преобразовать в строку n→data)
11. Вывести строку s_data относительно vis_window на позиции node_x + 1,node_y+4

1.2.4.18 Алгоритм вывода списка

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. node *n = list
2. i = 0
3. Вывести строку "Голова списка -------→" относительно vis_window на позиции (0, i * NODE_SUMMARY_SIZE + 2)
4. Если n равно нулю то перейти к шагу 24
5. Вывести узел n, под номером i (см. 1.2.4.16)
6. Переместить указатель в начало стрелки, указывающей на следующий за i узлом в окне vis_window
7. Вывести вертикальную линию (см. 1.2.4.8) из символов 'Y' относительно vis_window длиной DISTANCE_BETWEEN_NODES
8. Переместить указатель в начало стрелки, указывающей на предыдущий за i узлом в окне vis_window
9. Если n→previous не равно нулю выполнить
   Вывести вертикальную линию из символов '^' относительно vis_window длиной DISTANCE_BETWEEN_NODES
10. Увеличить i на 1
11. n = n→next
12. Перейти к шагу 16
13. Вывести строку "NULLPTR" относительно vis_window на позиции

(NODE_ORIGIN — 13,NODE_SUMMARY_SIZE * i)

14. Вывести строку "Хвост списка ----------→" относительно vis_window на позиции (0, (i - 1) * NODE_SUMMARY_SIZE + 1)

1.2.4.19 Алгоритм вывода кода

Дано: индекс в массиве кодов операций — op deb_window — смещение области просмотра кода list_size – размер списка

1. Очистить прямоугольную область относительно deb_window с шириной HSIZE * 3/ 8 и высотой VSIZE / 2
2. Вывести строку codes[operation] относительно deb_window на позиции 1,0

1.2.4.20 Алгоритм шага визуализации

Дано: description — объяснение, line — номер строки кода deb_window — смещение области просмотра кода loopback – динамический массив откатов list_size – размер списка

1. Очистить прямоугольную область (см. 1.2.4.11) относительно deb_window с шириной HSIZE * 3/ 8 и высотой VSIZE / 2
2. Вывести строку description относительно desc_window на позиции 0, 0
3. Для i пробегающего от 0 до HSIZE/2 выполнить:
   Вывести символ ' ' относительно deb_window на позиции 0,i
4. Вывести символ '>' относительно deb_window на позиции 0, line
5. Добавить откат (см. 1.2.4.14)
6. current_step = (количество элементов в массиве loopback) - 1
7. choice = 0
8. Если current_step равен (количество элементов в массиве loopback) выйти из функции
9. Ожидать ввода символа, считанный символ сохранить в choice
10. Если choice равен KEY_RIGHT то увеличить current_step на 1 и если после этого он не стал равен (количество элементов в массиве loopback), то вывести откат (см. 1.2.4.16) под номером current_step
11. Если choice равен KEY_LEFT то, если current_step не равен нулю уменьшить current_step на 1 и вывести откат под номером current_step
12. Перейти к шагу 9

1.2.4.21 Алгоритм визуализации прохождения по списку

Дано: where – номер искомого узла в списке, list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

Вернуть: указатель на найденный узел или 0, если индекс слишком большой

1. Сделать шаг визуализации (см. 1.2.4.19) с описанием "Указатель указывает на первый элемент." и указанием на 2 строчку кода.
2. node *n = list
3. i = 0
4. arrow_x = NODE_ORIGIN + 20
5. Если n равно нулю или адрес n равно where перейти к шагу 13
6. n = n->next;
7. Вывести строку "<--- ptr" относительно vis_window на позиции (arrow_x, i * NODE_SUMMARY_SIZE + 1)
8. Если i не равно нулю вывести строку " " относительно vis_window на позиции (arrow_x, (i - 1) * NODE_SUMMARY_SIZE + 1)
9. Сделать шаг визуализации (см. 1.2.4.19) с описанием «перейти к следующему узлу» и указанием на 5 строчку кода.
10. Увеличить i на 1
11. Перейти к шагу 5
12. Вывести строку "<--- ptr" относительно vis_window на позиции (arrow_x, i * NODE_SUMMARY_SIZE + 1)
13. Если i не равно нулю вывести строку " " относительно vis_window на позиции (arrow_x, (i - 1) * NODE_SUMMARY_SIZE + 1)
14. Вернуть n

1.2.4.22 Алгоритм визуализации вставки в начало списка

Дано: data – данные для вставки list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. Вывести список (см. 1.2.4.17)
2. Вывести код операции вставки в начало
3. node* begin = list
4. Если begin равен нулю то перейти к шагу 19
5. begin→previous = new node; begin→previous→data = data
6. Вывести узел begin→previous под номером 0 со смещением NODE_WIDTH / 2 + NODE_WIDTH (см. 1.2.4.16)
7. Вывести узел begin под номером 0 (см. 1.2.4.16)
8. Нарисовать горизонтальную линию (см. 1.2.4.9) из символов '>' относительно vis_window на позиции NODE_ORIGIN + NODE_WIDTH / 2 + 1, 2 длиной NODE_WIDTH / 2 - 1

9. Сделать шаг визуализации (см. 1.2.4.19) с описанием "Создать новый узел и указателю на предыдущий первого узла присвоить адрес созданного узла" и указанием на 2 строчку
10. begin->previous->next = begin
11. Вывести узел begin→previous под номером 0 со смещением NODE_WIDTH / 2 + NODE_WIDTH (см. 1.2.4.16)
12. Нарисовать горизонтальную линию (см. 1.2.4.9) из символов '<' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH / 2 + 1), 4 длиной NODE_WIDTH / 2 - 1
13. Сделать шаг визуализации (см. 1.2.4.19) с описанием "Указателю на следующий узел созданного узла присвоить адрес первого узла"и указанием на 3 строчку
14. list = begin->previous
15. Очистить прямоугольную область относительно vis_window с шириной и высотой: HSIZE / 2 и VSIZE (см. 1.2.4.11)
16. Вывести список (см. 1.2.4.17)
17. Сделать шаг визуализации с описанием "Указателю на начало списка присвоить адрес созданного узла" и указанием на 4 строчку (см. 1.2.4.19)
18. Перейти к шагу 23
19. list = new node
20. list.data = data
21. Вывести список
22. Сделать шаг визуализации с описанием «Создать новый узел и указателю на первый узел присвоить его» и указанием на 9 строчку
23. Увеличить list_size на 1

1.2.4.23 Алгоритм визуализации удаления из начала списка

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. Вывести список
2. Вывести код операции удаления из начала
3. node *begin = list
4. Сделать шаг визуализации с описанием "Сохранить данные первого узла" и указанием на 3 строчку (см. 1.2.4.19)
5. Если begin→next равно нулю перейти к шагу 12
6. begin->next->previous = nullptr;

7. Переместить указатель в начало стрелки, указывающей на предыдущий за 1 узлом в окне vis_window
8. Нарисовать вертикальную линию из символов ' ' относительно vis_window длиной DISTANCE_BETWEEN_NODES (см. 1.2.4.8)
9. Очистить прямоугольную область в окне vis_window с координатами (NODE_ORIGIN - NODE_WIDTH / 2 , NODE_SUMMARY_SIZE) и шириной и высотой (NODE_WIDTH, NODE_HEIGHT) (см. 1.2.4.11)
10. Вывести узел begin→next под номером 1
11. Сделать шаг визуализации с описанием "Если это не последний в списке узел, то указатель на предыдущий следующего за первым узла обнулить" и указанием на 4 строчку(см. 1.2.4.19)
12. node *next = begin->next;
13. Сделать шаг визуализации с описанием "Сохранить указатель на следующий узел первого узла" и указанием на 5 строчку (см. 1.2.4.19)
14. delete begin
15. Очистить прямоугольную область относительно vis_window с координатами (NODE_ORIGIN - NODE_WIDTH / 2,0) и шириной и высотой (NODE_WIDTH + 1, NODE_SUMMARY_SIZE) (см. 1.2.4.11)
16. Сделать шаг визуализации с описанием "Удалить первый узел" и указанием на 6 строчку (см. 1.2.4.19)
17. list = next;
18. Очистить прямоугольную область относительно vis_window с шириной HSIZE * 5 / 8 и высотой VSIZE (см. 1.2.4.11)
19. Вывести список
20. Сделать шаг визуализации с описанием "Присвоить указателю на начало списка адрес сохраненного узла" и указанием на 7 строчку (см. 1.2.4.19)
21. Сделать шаг визуализации с описанием "Вернуть сохраненные данные" и указанием на 8 строчку (см. 1.2.4.19)
22. Уменьшить list_size на 1

1.2.4.24 Алгоритм визуализации вставки в середину списка

Дано: where — номер узла, после которого нужно вставить, data — данные для вставки list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. Если where равно нулю, то визуализировать вставку в начало и выйти из функции (см. 1.2.4.21)
2. Вывести список (см. 1.2.4.17)
3. Вывести код операции вставки в середину (см. 1.2.4.18)
4. Визуализировать прохождение по списку до узла where, сохранить найденный узел в переменную n (см. 1.2.4.20)
5. Сделать шаг визуализации с описанием "Если узел не существует вернуть ложь." и указанием на 8 строчку (см. 1.2.4.19)
6. Если n равно нулю выйти из функции
7. node *temp = new node
8. temp→data = data
9. Очистить прямоугольную область относительно vis_window с шириной HSIZE * 5/ 8 и высотой VSIZE (см. 1.2.4.11)
10. Вывести список (см. 1.2.4.17)
11. Вывести узел temp под номером where со смещением NODE_WIDTH + NODE_WIDTH / 4 (см. 1.2.4.16)
12. Сделать шаг визуализации с описанием "Создать новый узел." и указанием на 9 строчку (см. 1.2.4.19)
13. Нарисовать вертикальную линию из символов 'Y' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH + NODE_WIDTH / 4 + 10, where * NODE_SUMMARY_SIZE + NODE_HEIGHT)
    длиной DISTANCE_BETWEEN_NODES + 4 (см. 1.2.4.8)
14. Вывести символ '+' относительно vis_window на позиции
    (NODE_ORIGIN + NODE_WIDTH + NODE_WIDTH / 4 + 10, where * NODE_SUMMARY_SIZE + NODE_HEIGHT)
15. Вывести горизонтальную линию из символов '<' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH / 2 + 1, where * NODE_SUMMARY_SIZE+NODE_HEIGHT + DISTANCE_BETWEEN_NODES + 4)
    длиной NODE_WIDTH + 1 (см. 1.2.4.9)
16. temp->next = n->next
17. Вывести узел temp под номером where со смещением NODE_WIDTH + NODE_WIDTH / 4 (см. 1.2.4.16)

18. Сделать шаг визуализации с описанием "Указателю на следующий узел созданного узла присвоить указатель на следующий узел узла с выбранным номером." и указанием на 10 строчку (см. 1.2.4.19)
19. Нарисовать горизонтальную линию из символов '<' относительно vis_window на позиции
    (NODE_ORIGIN + NODE_WIDTH / 2 + 1,where*NODE_SUMMARY_SIZE+ 1)
    длиной NODE_WIDTH / 4 — 1 (см. 1.2.4.9)
20. Вывести узел temp под номером where со смещением NODE_WIDTH + NODE_WIDTH / 4 (см. 1.2.4.16)
21. temp->previous = n
22. Сделать шаг визуализации с описанием "Указателю на предыдущий узел созданного узла присвоить адрес узла с выбранным номером." и указанием на 11 строчку (см. 1.2.4.19)
23. Если n→next равно нулю перейти к шагу 32
24. Нарисовать вертикальную линию из символов '^' относительно vis_window
    на позиции
    (NODE_ORIGIN + NODE_WIDTH + NODE_WIDTH / 4 + 7, where * NODE_SUMMARY_SIZE + NODE_HEIGHT)
    длиной DISTANCE_BETWEEN_NODES + 2 (см. 1.2.4.8)
25. Вывести симовол '+' относительно vis_window на позиции
    (NODE_ORIGIN+NODE_WIDTH+NODE_WIDTH/4+7, where*NODE_SUMMARY_SIZE+ NODE_HEIGHT+DISTANCE_BETWEEN_NODES+2)
26. Нарисовать горизонтальную линию из символов '>' относительно vis_window
    на позиции
    (NODE_ORIGIN + NODE_WIDTH / 2 + 1, where * NODE_SUMMARY_SIZE + NODE_HEIGHT + DISTANCE_BETWEEN_NODES + 2)
    длиной NODE_WIDTH — 2 (см. 1.2.4.9)
27. n->next->previous = temp
28. Вывести узел n→next под номером where + 1
29. Переместить указатель в начало стрелки, указывающей на предыдущий за where + 1 узлом в окне vis_window (см. 1.2.4.13)
30. Нарисовать вертикальную линию из символов ' ' относительно vis_window длиной DISTANCE_BETWEEN_NODES (см. 1.2.4.8)

31. Сделать шаг визуализации с описанием "Если узел не последний, то указателю на предыдущий узел следующего за выбранным узла присвоить адрес созданного узла." и указанием на 13 строчку (см. 1.2.4.19)
32. Нарисовать горизонтальную линию из символов '>' относительно vis_window на позиции (NODE_ORIGIN +NODE_WIDTH / 2 + 1, where*NODE_SUMMARY_SIZE+ 3) длиной NODE_WIDTH / 4 — 1 (см. 1.2.4.9)
33. Переместить указатель в начало стрелки, указывающей на следующий за where узлом в окне vis_window (см. 1.2.4.12)
34. Нарисовать вертикальную линию из символов ' ' относительно vis_window длиной DISTANCE_BETWEEN_NODES (см. 1.2.4.8)
35. Вывести узел n под номером where (см. 1.2.4.16)
36. n->next = temp
37. Сделать шаг визуализации с описанием "Указателю на следующий выбранного узла присвоить адрес созданного узла." и указанием на 14 строчку (см. 1.2.4.19)
38. Сделать шаг визуализации с описанием "Вернуть истину." и указанием на 15 строчку (см. 1.2.4.19)
39. Увеличить list_size на 1

1.2.4.25 Алгоритм визуализации удаления из середины списка

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. Если where равно 0 то визуализировать удаление из начал списка и выйти из функции (см. 1.2.4.22)
2. Вывести список (см. 1.2.4.17)
3. Вывести код операции удаления из середины (см. 1.2.4.18)
4. Визуализировать прохождение по списку до узла where, сохранить найденный узел в переменную n (см. 1.2.4.20)
5. Сделать шаг визуализации с описанием "Если узел не существует вернуть ложь." и указанием на 8 строчку (см. 1.2.4.19)
6. Если n равно нулю выйти из функции
7. n->previous->next = n->next
8. Нарисовать горизонтальную линию из символов '<' относительно vis_window на позиции (NODE_ORIGIN - NODE_WIDTH, (where - 1)*NODE_SUMMARY_SIZE+1) длиной NODE_WIDTH / 2 (см. 1.2.4.9)

9. Нарисовать вертикальную линию из символов 'Y' относительно vis_window на позиции (NODE_ORIGIN-NODE_WIDTH-1,(where - 1)*NODE_SUMMARY_SIZE+1) длиной NODE_SUMMARY_SIZE * 2 (см. 1.2.4.8)
10. Нарисовать горизонтальную линию из символов '>' относительно vis_window на позиции (NODE_ORIGIN - NODE_WIDTH, (where +1)*NODE_SUMMARY_SIZE + 1) длиной NODE_WIDTH / 2 (см. 1.2.4.9)
11. Вывести символ '+' относительно vis_window на позиции NODE_ORIGIN - NODE_WIDTH - 1, (where - 1) * NODE_SUMMARY_SIZE + 1
12. Вывести символ '+' относительно vis_window на позиции (NODE_ORIGIN - NODE_WIDTH - 1, (where + 1) * NODE_SUMMARY_SIZE + 1)
13. Переместить указатель в начало стрелки, указывающей на следующий за where - 1 узлом в окне vis_window (см. 1.2.4.12)
14. Нарисовать вертикальную линию из символов ' ' относительно vis_window длиной DISTANCE_BETWEEN_NODES (см. 1.2.4.8)
15. Вывести узел n→previous под номером where - 1
16. Сделать шаг визуализации с описанием "Указателю на следующий предыдущего за данным узла присвоить указатель на следующий данного узла." и указанием на 10 строчку (см. 1.2.4.19)
17. Если n→next равен нулю, то перейти к шагу 28
18. n->next->previous = n->previous;
19. Нарисовать горизонтальную линию из символов '<' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH, (where - 1)*NODE_SUMMARY_SIZE+1) длиной NODE_WIDTH / 2 (см. 1.2.4.9)
20. Нарисовать вертикальную линию из символов '^' относительно vis_window на позиции (NODE_ORIGIN+NODE_WIDTH+1,(where - 1)*NODE_SUMMARY_SIZE+1) длиной NODE_SUMMARY_SIZE * 2 (см. 1.2.4.8)
21. Нарисовать горизонтальную линию из символов '>' относительно vis_window на позиции (NODE_ORIGIN+NODE_WIDTH, (where + 1)*NODE_SUMMARY_SIZE +1) длиной NODE_WIDTH / 2 (см. 1.2.4.9)
22. Вывести символ '+' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH + 1, (where - 1) * NODE_SUMMARY_SIZE + 1)
23. Вывести символ '+' относительно vis_window на позиции (NODE_ORIGIN + NODE_WIDTH + 1, (where + 1) * NODE_SUMMARY_SIZE + 1)

24. Переместить указатель в начало стрелки, указывающей на предыдущий за where + 1 узлом в окне vis_window (см. 1.2.4.13)
25. Нарисовать вертикальную линию из символов ' ' относительно vis_window длиной DISTANCE_BETWEEN_NODES (см. 1.2.4.8)
26. Вывести узел n→next под номером where + 1
27. Сделать шаг визуализации с описанием "Если выбранный узел не последний то указателю на предыдущий следующего за данным узла присвоить указатель на предыдущий данного узла." и указанием на 12 строчку (см. 1.2.4.19)
28. Очистить прямоугольную область относительно vis_window на позиции:
    (NODE_ORIGIN- NODE_WIDTH/ 2, (where-1)*NODE_SUMMARY_SIZE+NODE_HEIGHT)
    с шириной и высотой:
    NODE_WIDTH + 1, NODE_SUMMARY_SIZE + DISTANCE_BETWEEN_NODES
    (см. 1.2.4.11)
29. Сделать шаг визуализации с описанием "Удалить данный узел." и указанием на 13 строчку (см. 1.2.4.19)
30. Очистить прямоугольную область относительно vis_window с шириной HSIZE * 5 / 8 и высотой VSIZE (см. 1.2.4.11)
31. Вывести список
32. Сделать шаг визуализации с описанием "Вернуть истину." и указанием на 14 строчку (см. 1.2.4.19)
33. Уменьшить list_size на 1

1.2.4.26 Алгоритм визуализации добавления в конец

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка data – вставляемые данные

1. Вывести код операции вставки в конец списка
2. node* n = list
3. Если n не равно нулю то n = n→next до тех пор, пока n→next не станет равен нулю
4. Вывести список
5. arrow_x = NODE_ORIGIN + 20
6. Вывести строку "<--- end" на позиции arrow_x, 1 если list_size не равен нулю или на позиции arrow_x, (list_size - 1) * NODE_SUMMARY_SIZE + 1 если равен
7. Сделать шаг визуализации с описанием "Указатель указывает на последний элемент" и указанием на 2 строку
8. Если list_size равен нулю то перейти к шагу

9. Создать узел, инициализировать его данными data и присвоить n→next
10. Вывести узел n→next на позиции list_size — 1 со смещением NODE_WIDTH / 2 + NODE_WIDTH
11. Вывести узел n на позиции list_size-1
12. Сделать шаг визуализации с описанием "Если узел не единственный, то создать новый узел и указателю на следующий узел последнего узла присвоить его." и указанием на 5 строку
13. n->next->previous = n
14. Вывести узел n→next на позиции list_size — 1 со смещением NODE_WIDTH / 2 + NODE_WIDTH
15. Вывести горизонтальную линию на позиции NODE_ORIGIN + NODE_WIDTH / 2 +1, 4 + NODE_SUMMARY_SIZE * (list_size — 1) из символов '<' NODE_WIDTH / 2 — 1
16. Сделать шаг визуализации с описанием "Указателю на предыдущий узел созданного узла присвоить адрес последнего узла" и указанием на 6 строку
17. Сделать шаг визуализации с описанием "Указателю на последний узел списка присвоить адрес созданного узла" и указанием на 7 строку
18. Перейти к шагу
19. Инициализировать узел данными data и присвоить его list
20. Вывести список
21. Сделать шаг визуализации с описанием "Создать новый узел и указателю на первый узел присвоить его" и указанием на 10 строку
22. Сделать шаг визуализации с описанием "Указателю на последний узел присвоить указатель на первый" и указанием на 11 строку
23. Увеличить list_size на 1

1.2.4.27 Алгоритм визуализации удаления из конца списка

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка

1. Вывести код операции вставки в конец списка
2. node* n = list
3. Если n не равно нулю то n = n→next до тех пор, пока n→next не станет равен нулю
4. Вывести список
5. arrow_x = NODE_ORIGIN + 20
6. Вывести строку "<--- end" на позиции arrow_x, (list_size — 1) NODE_SUMMARY_SIZE + 1

7. Сделать шаг визуализации с описанием "Указатель указывает на последний элемент" и указанием на 2 строку
8. Вывести строку " " на позиции arrow_x, (list_size — 1) NODE_SUMMARY_SIZE+1
9. Если list_size равен 1 то вывести строку "<--- end" на позиции arrow_x, (list_size — 2) NODE_SUMMARY_SIZE + 1 и присвоить n->previous->next = nullptr
10. delete n
11. Вывести список
12. Уменьшить list_size на 1

1.2.4.28 Алгоритм визуализации поиска в списке

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка data - искомое

1. Вывести код операции поиска в списке
2. n = list
3. Сделать шаг визуализации с описанием "Указатель указывает на первый элемент" и указанием на 2 строку
4. arrow_x = NODE_ORIGIN + 20
5. I = 0
6. n = n→next
7. Вывести строку "<--- ptr" на позиции arrow_x, i * NODE_SUMMARY_SIZE + 1 относительно vis_window
8. Если I не равен нулю то вывести строку " " на позиции arrow_x, (i - 1) * NODE_SUMMARY_SIZE + 1 относительно vis_window
9. Сделать шаг визуализации с описанием "Перейти к следующему узлу." и указанием на 4 строку
10. Увеличить I на 1
11. Если n не равен нулю и n->data не равна data то перейти к шагу 6
12. Вывести строку "<--- ptr" на позиции arrow_x, i * NODE_SUMMARY_SIZE + 1 относительно vis_window
13. Если I не равен нулю то вывести строку " " на позиции arrow_x, (i - 1) * NODE_SUMMARY_SIZE + 1 относительно vis_window
14. Сделать шаг визуализации с описанием "Вернуть найденный узел или указатель на ноль." и указанием на 5 строку

1.2.4.28 Алгоритм визуализации

Дано: list — указатель на голову списка vis_window — смещение области с визуализацией. list_size – размер списка desc_window - смещение области с описанием deb_window - смещение области с кодом

1. vis_window = (HSIZE * 3 / 8 + 1, 1)
2. desc_window = (1, 1)
3. deb_window = (1, VSIZE/2+1)
4. Нарисовать прямоугольник относительно vis_window на позиции (-1,-1) с шириной HSIZE * 5 / 8 и высотой VSIZE
5. Нарисовать прямоугольник относительно desc_window на позиции (-1,-1) с шириной HSIZE * 3 / 8 и высотой VSIZE/2
6. Нарисовать прямоугольник относительно deb_window на позиции (-1,-1) с шириной HSIZE * 3 / 8 и высотой VSIZE/2
7. Вывести строку "Во время визуализации нажмите стрелку вправо, чтобы продвинуться на шаг вперед или стрелку влево, чтобы посмотреть предыдущие шаги (только посмотреть)" относительно desc_window на позиции
   (0, VSIZE/2 - 2)
8. list = nullptr
9. list_size = 0
10. int choice
11. Вывести список (см. 1.2.4.17)
12. Вывести строку "Введите номер операции" относительно desc_window на позиции (0,0)
13. Если list_size не равен нулю то:
    - 13.1 Вывести строку "2.Удалить элемент из начала списка" относительно desc_window на позиции (0, 2)
    - 13.2 Вывести строку "4.Удалить элемент из середины списка" относительно desc_window на позиции (0, 4)
    - 13.3 Вывести строку "6.Удалить элемент из конца списка" относительно desc_window на позиции (0, 6)
    - 13.4 Вывести строку "7.Найти узел с указанным содержимым" относительно desc_window на позиции (0, 7)
14. Если list_size меньше 6 то:

14.1 Вывести строку "1.Вставить элемент в начало списка" относительно desc_window на позиции (0, 1)

14.2 Вывести строку "3.Вставить элемент в середине списка" относительно desc_window на позиции (0, 3)

14.3 Вывести строку "5.Вставить элемент в конец списка" относительно desc_window на позиции (0, 5)

15. Вывести строку "0.Выйти из визуализации" относительно desc_window на позиции (0, 8)
16. Если list_size равен нулю, то ввести choice такой, чтобы он был равен 0 1 3 или 5
17. Если list_size равен 6, то ввести choice такой, чтобы он был равен 0 2 4 6 или 7
18. Если оба условия выше не выполнились, то ввести такой choice, чтобы он был в пределах от 0 до 7
19. Если choice не равен 1 перейти к шагу 24
20. Вывести строку "Введите, что вставлять " относительно desc_window на позиции (0, 6)
21. Считать число с клавиатуры и сохранить его в переменную data
22. Очистить прямоугольную область относительно desc_window с шириной HSIZE * 3/8 и высотой VSIZE/2 (см. 1.2.4.11)
23. Визуализировать вставку data (см. 1.2.4.21)
24. Если choice не равен '2' перейти к шагу 26
25. Визуализировать удаление из начала списка
26. Если choice не равен '3' перейти к шагу 32
27. Вывести строку "Введите, что вставлять" относительно desc_window на позиции (0, 6)
28. Считать число с клавиатуры и сохранить его в переменную data
29. Вывести строку "Введите,где вставлять" относительно desc_window на позиции (0, 6)
30. Считать число с клавиатуры и сохранить его в переменную where
31. Визуализировать вставку data после узла с адресом where (см. 1.2.4.23)
32. Если choice не равен '4' перейти к шагу 37
33. Вывести строку "Введите, где удалять" относительно desc_window на позиции 0 6
34. Считать число с клавиатуры и сохранить его в переменную where
35. Визуализировать удаление узла с адресом where (см. 1.2.4.24)
36. Если choice не равен '5' перейти к шагу 40

37. Вывести строку "Введите, что вставлять " относительно desc_window на позиции (0, 6)
38. Считать число с клавиатуры и сохранить его в переменную data
39. Очистить прямоугольную область относительно desc_window с шириной HSIZE * 3/8 и высотой VSIZE/2 (см. 1.2.4.11)
40. Визуализировать вставку data в конец списка
41. Если choice не равен '6' перейти к шагу 43
42. Визуализировать удаление из конца списка
43. Если choice не равен '7' перейти к шагу 47
44. Вывести строку "Введите, что найти " относительно desc_window на позиции (0, 6)
45. Считать число с клавиатуры и сохранить его в переменную data
46. Визуализировать поиск data в списке
47. Если choice не равен 0, перейти к шагу 11

1.2.4.29 Основной алгоритм программы

1. Прочитать файлы с теорией и базу с вопросами (см. 1.2.2.1 и 1.2.4.3)
2. current_option = 0
3. Вывести "нажмите стрелку вверх или вниз, чтобы листать пункты, нажмите q чтобы выйти, или нажмите стрелку вправо чтобы выбрать данный пункт\n"
4. Вывести " Почитать теорию\n Пройти тест\n Посмотреть визуализацию\n"
5. Вывести символ '>' на позиции 0, current_option % 3 + 1
6. choice = 0
7. Если choice равен 'q' , то выйти из программы
8. Если choice равен KEY_UP то current_option = (current_option — 1) % 3
9. Если choice равен KEY_DOWN то current_option = (current_option + 1) % 3
10. Если choice равен KEY_DOWN то если current_option был равен 0 выполнить алгоритм показа теории (см. 1.2.4.4), или если current_option был равен 1 то выполнить алгоритм тестирования (см. 1.2.4.6), в противном случае выполнить алгоритм визуализации (см. 1.2.4.28)
11. Очистить консоль
12. Вывести на позиции 0,0 "нажмите стрелку ввверх или вниз, чтобы листать пункты, нажмите q чтобы выйти, или нажмите стрелку вправо чтобы выбрать данный пункт"
13. Вывести на позиции 0,1 " Почитать теорию\n Пройти тест\n Посмотреть визуализацию\n"
14. Вывести символ '>' на позиции 0, current_option % 3 + 1

15. Ожидать ввода символа и сохранить его в переменную choice

1.3 Входные и выходные данные

Входные данные:

- файл с теорией
- база с вопросами
- данные, вводимые пользователем во время визуализации
- выбор пунктов меню

Выходные данные:

- визуализация списка
- теория
- вопросы и варианты ответов
- результаты тестирования и оценка

1.4 Системные требования

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 2 ГБ ОЗУ;
- не менее 300КБ свободного места на диске.

Операционная система: Windows 7 и выше.

2 РАБОЧИЙ ПРОЕКТ

2.1 Общие сведения о работе системы

Программный продукт разработан в текстовом редакторе VS Code (версия 1.45) на языке C++11. Программа работает под управлением операционной системы Windows 7 и более поздними версиями. Стартовый файл — DLVis.exe

2.2 Функциональное назначение программного продукта

Разработанный программный продукт предназначен для получения и проверки знаний в использовании двусвязных списков.

Программа имеет следующие функциональные возможности:

- предоставление пользователю теоретический материал по данной теме
  - основные обозначения и определения
  - реализация алгоритмов для работы с двусвязным списком
  - полезные для ознакомления заметки
- предоставление в виде визуализации работы двусвязного списка
  - операция добавления в начало списка
  - операция добавления в середину списка
  - операция удаления из начала списка
  - операция удаления из середины списка
- предоставление возможности тестирования
  - проверка правильности ответа пользователя
  - сохранение пользовательских результатов тестирования
- прекращение тестирования:
  - по желанию пользователя
  - после ответа на 5 вопросов

Программа имеет следующие функциональные ограничения:

- Количество элементов списке во время визуализации не превышает 6
- Данные элементов визуализации должны быть меньше по модулю чем INT_MAX

2.3 Инсталляция и выполнение программного продукта

Для запуска пограммы достаточно скопировать файл DLVIS.exe на диск и запустить.

После первого запуска программа устанавливается в папку ProgrammFiles в папке DoublyLinkedListVisualizer. Появляются файлы DLL.exe libpdcurses.dll nircmd questions.bin start.bat theory.bin xor.py а так же ярлык на рабочем столе.

2.4 Описание программы

В таблице 2.1 приведен класс Theory , используемый в программе

Таблица 2.1 – Описание класса Theory

Поле	Тип	Назначение
pages	string[THEORY_PAGE_MAX_COUNT]	Массив страниц
questions	string[QUESTION_MAX_COUNT]	Массив вопросов
answers	int[QUESTION_MAX_COUNT]	Массив ответов
options	string[QUESTION_MAX_COUNT][8]	Массив массивов вариантов ответа
option_counts	int[QUESTION_MAX_COUNT]	Массив количеств вариантов ответа
cost	int[QUESTION_MAX_COUNT]	Массив стоимостей вопросов
question_count	int	Количество вопросов
pages_count	int	Количество страниц

Метод	Назначение
theory(string theorybase, string questionbase)	Конструктор класса
~Theory()	Деструктор класса
void print_page(uint8_t page)	Выводит страницу с данным номером
void print_question(int question,int choose, int answer)	Выводит вопрос с данным номером question и выводит указатель возле выбранного в данный момент варианта ответа choose, а так же выводит слово «выбрано» возле всех выбранных вариантов ответа answer
void start_read()	Начинает процесс демонстрации теории
void start_test()	Начинает процесс тестирования

В таблице 2.2 приведен класс visualization , используемый в программе

Таблица 2.2 – Описание класса visualization

Поле	Тип	Назначение
loopback	vector<string[3]>	Массив откатов
vis_window	WINDOW*	Дескриптор окна с визуализацией
desc_window	WINDOW*	Дескриптор окна с объяснениями и меню
deb_window	WINDOW*	Дескриптор окна с демонстрации кода
list	Node*	Указатель на голову списка
list_size	int	Размер списка
codes	String[7]	Массив строк, представляющих исходный код каждой операции над списком

Метод	Назначение
void print_node(node* n,int position,int offset)	Выводит данный узел на данной позиции в списке с данным смещением по X относительно середины окна vis_window
void print_list()	Выводит список
void restore_defaults()	Очищает все три окна и выводит список
node* v_search(int where)	Визуализирует прохождение по списку вплоть до узла с данным адресом
void v_push(int32_t data)	Визуализирует вставку в начало списка
void v_pop()	Визуализирует удаление из начала списка
void v_insert(int where,int32_t data)	Визуализирует вставку в середину списка
void v_remove(int where)	Визуализирует удаление из середины списка
void v_push_f(int32_t data)	Визуализирует вставку в конец списка
void v_pop_f()	Визуализирует удаление из конца списка
void v_search_data(int32_t data)	Визуализирует поиск данного числа в списке
void print_code(op operation)	Выводит исходный код операции с данным индексом в массиве codes
void step(string description,int line)	Останавливает визуализацию, давая пользователю возможность посмотреть предыдущие шаги, а так же обновляет описание и перемещает указатель в окне кода на данную строку.
void visualization_start()	Начинает визуализацию

В таблице 2.2 приведены функции, используемые в программе.

Таблица 2.2 – Функции программы

Прототип	Назначение
stringstream decryptFile(string filename, char key = 42)	Расшифровывает данный файл в строковый поток используя данный ключ
void read_theory_file(stringstream theory_file, string *pages, uint16_t &page_count)	Прочитывает данные из расшифрованного строкового потока как файл с теорией, заполняя pages и page_count
void read_question_file(stringstream question_file, string *questions, uint16_t &question_count,int*answers, uint16_t*options_count,string options[][8])	Прочитывает данные из расшифрованного строкового потока как файл с вопросами, заполняя questions, question_count answers options_count options

2.5 Разработанные меню и интерфейсы

После запуска программы на выполнение в консольном окне появится описание (рис. 2.1)

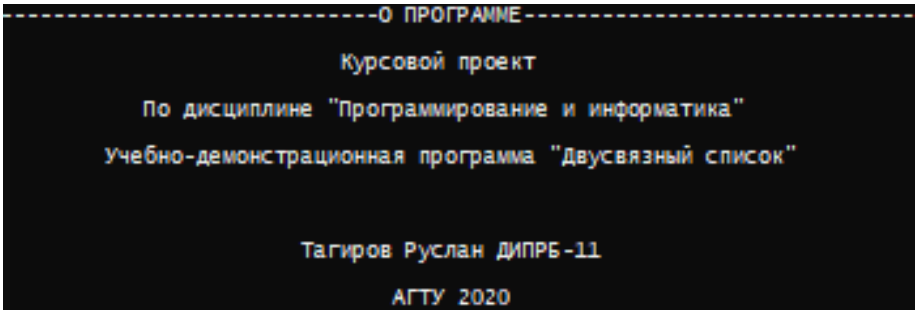


Рисунок 2.1 – Консоль программы с описанием

Под описанием расположено меню (рис. 2.2), которое позволяет выбрать один из трех блоков либо выйти из программы.

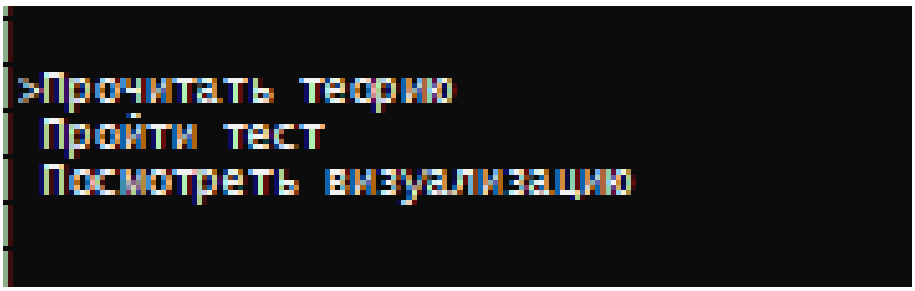


Рисунок 2.2 – Консоль программы с главным меню

Если пользователь желает ознакомиться с теоретическим материал, то ему следует выбрать первый пункт.

После выбора данного пункта, на экране пользователя будет показана (рис. 2.3) первая страница теории и пользователь сможет свободно перемещаться по страницам.

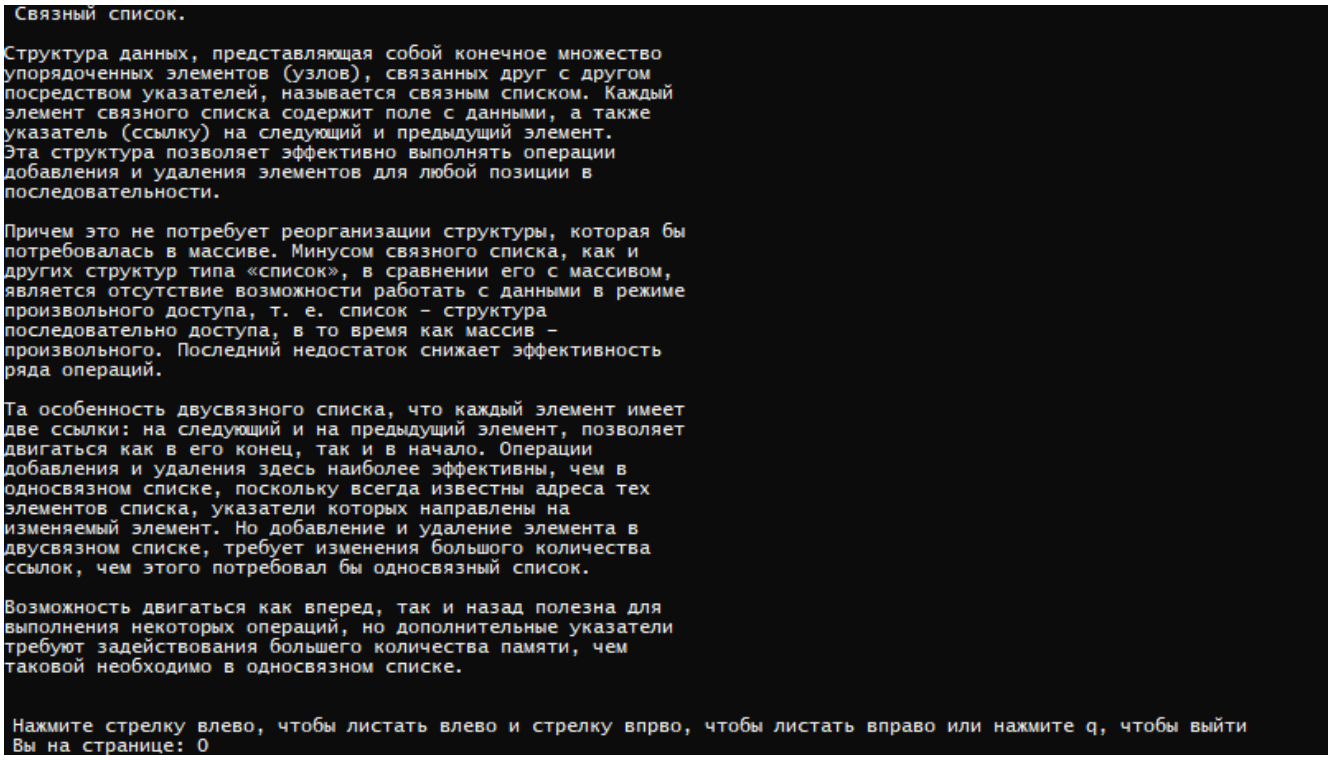


Рисунок 2.3 – Консоль программы со страницей теории

По страницам можно перемещаться, нажимая стрелки вправо и влево, а выйти можно нажав клавишу Q.

Если пользователь выбрал пункт «Пройти тест» то ему представляется возможность проверить свои знания. Сначала он должен будет ввести имя пользователя (рис. 2.4)

Рисунок 2.4 – Консоль программы с запросом имени пользователя

После ввода имени пользователя начинается собственно сам тест (рис. 2.5)

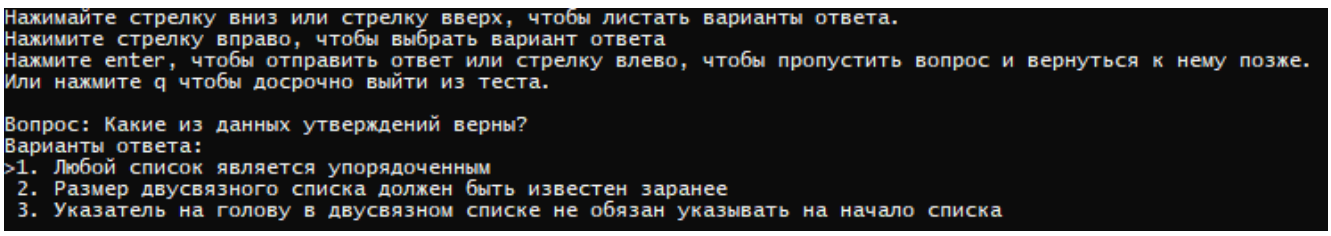


Рисунок 2.5 – Консоль программы с тестированием

Пользователь может переключаться между вариантами нажимая стрелки вверх и вниз, а также может пометить вариант как верный, нажав кнопку вправо (рис. 2.6). Пользователь может отправить свой ответ нажав кнопку ENTER или пропустить вопрос, нажав стрелку влево. Пропущенные вопросы не отнимают очков.

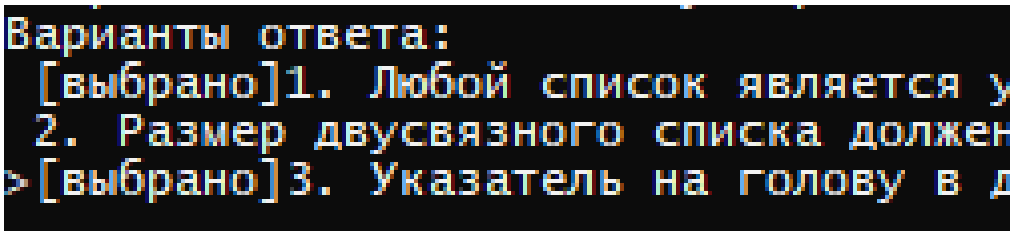
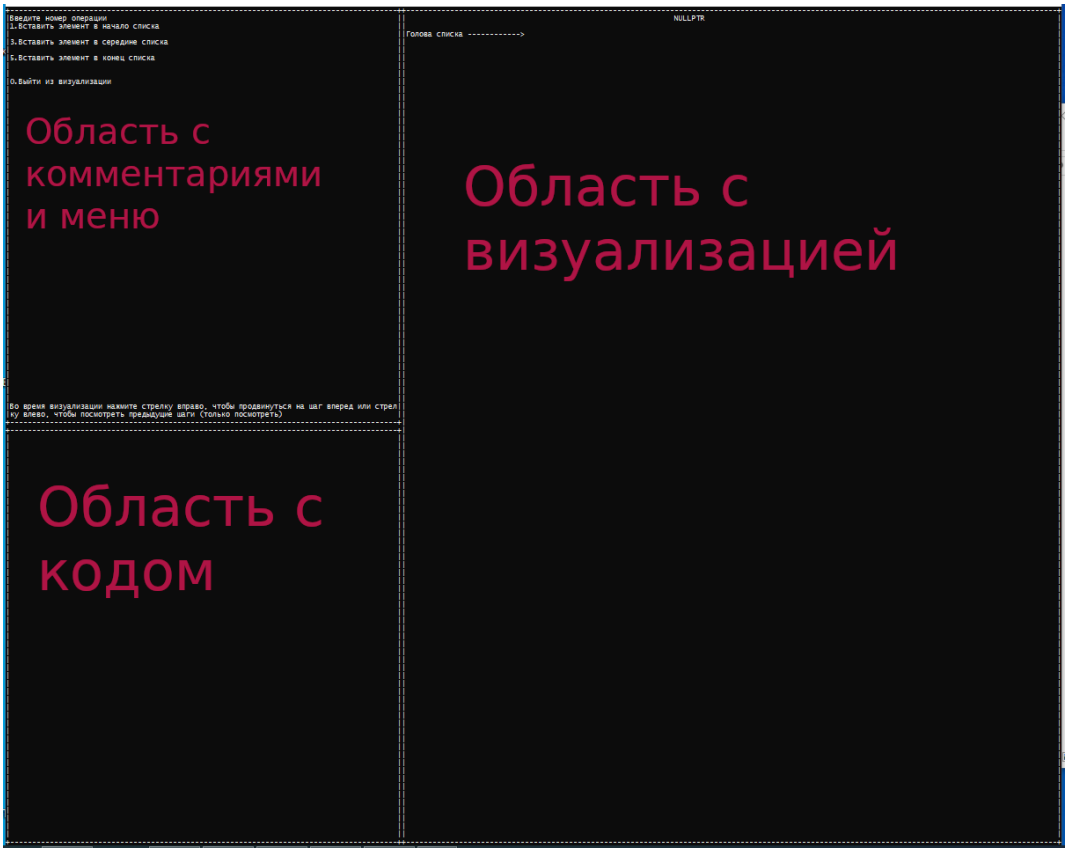


Рисунок 2.6 – Выбранные варианты ответа

Когда пользователь отвечает на 5 вопросов, ему сообщается прошел или не прошел он тест, а еще информация он нем и его результатах записывается в файл result.dat, из которого она может быть просмотрена администратором

Если пользователь проходит тест дольше 30 минут, тест не засчитывается. Пользователь также может завершить тестирование досрочно.

Если пользователь выберет в меню пункт «Посмотреть визуализацию» как понятно из названия он сможет посмотреть, как работает двусвязный список изнутри (рис. 2.7)



.Рисунок 2.7 – Главное меню визуализации

На выбор пользователю представляется 7 операций: вставка в начало, удаление из начала, вставка в середину, удаление из середины, вставка в конец, удаление из конца, поиск в списке (рис. 2.8).

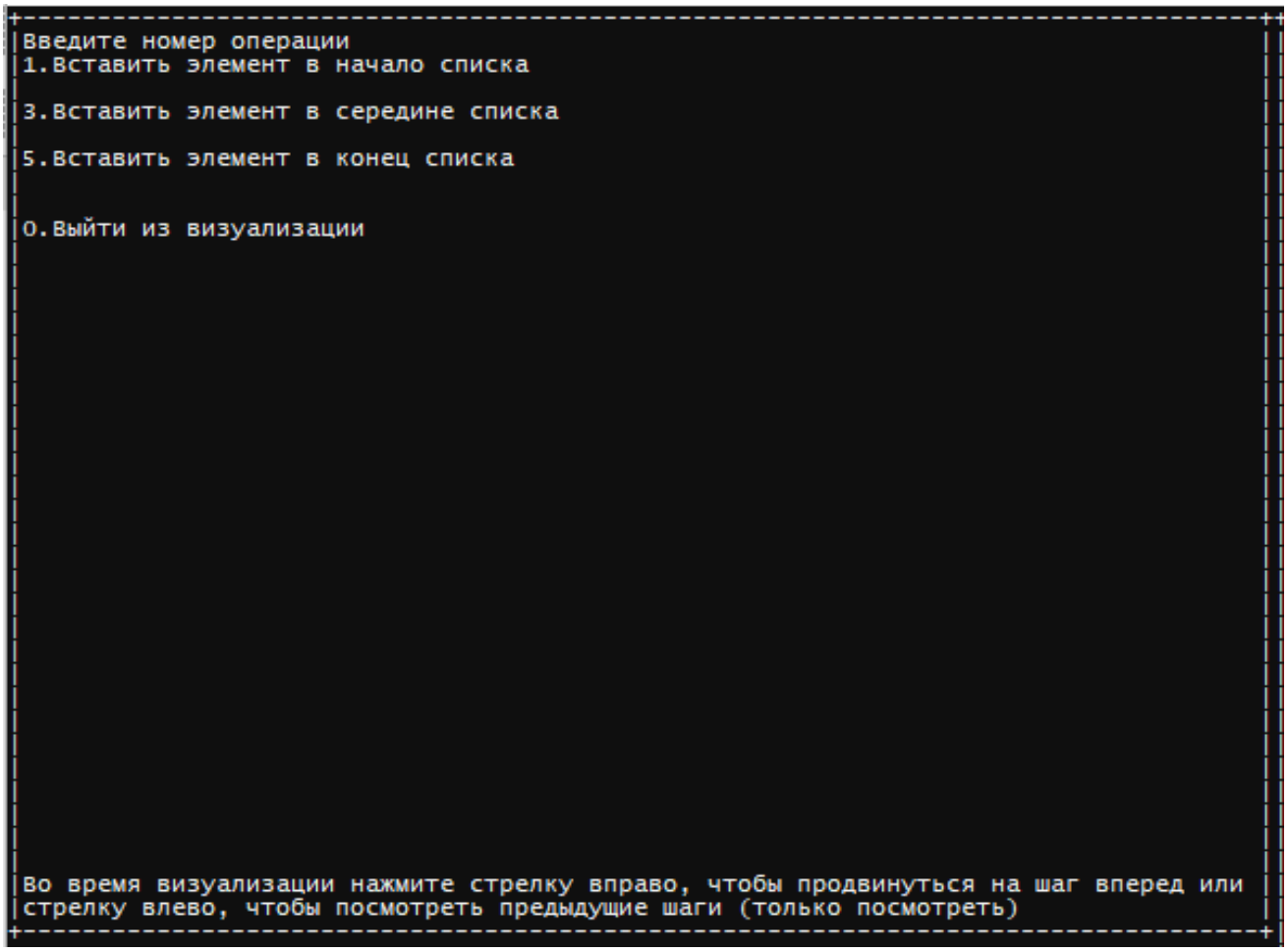


Рисунок 2.8 – Меню операций

В начале список пуст, и пользователь не может удалять ничего (потому что ничего и нет), но когда пользователь добавит хотя бы один элемент, он может использовать все операции. Если пользователь добавит 6 элементов, он не сможет больше ничего добавить, потому что на экране не помещается больше 6 элементов.

Когда пользователь просматривает визуализацию операции, он может видеть, какая строчка сейчас выполняется, для чего это нужно и как выглядит список в данный момент

На рисунке 2.9 показана область, в которой расположена визуализация кода операции

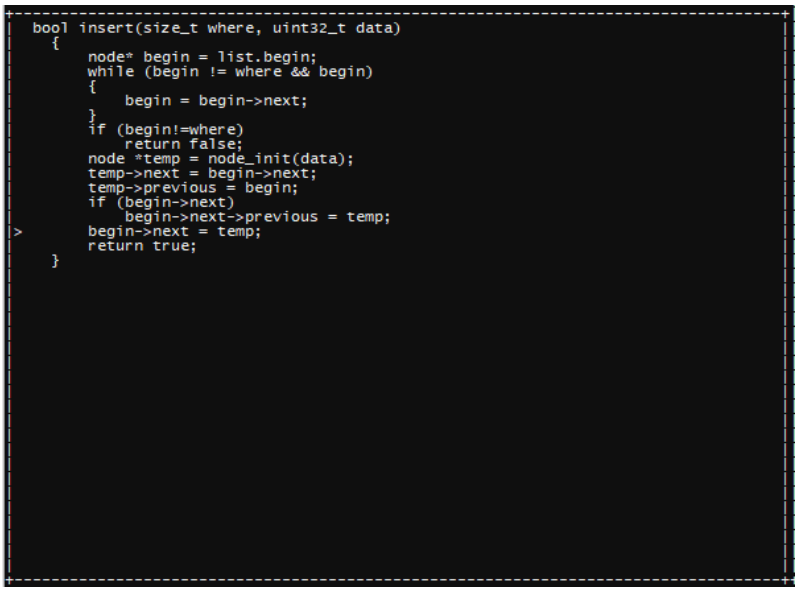


Рисунок 2.9 – Область визуализации с кодом

На рисунке 2.10 показана область, в которой расположены комментарии к выполняющейся в данный момент строчке кода

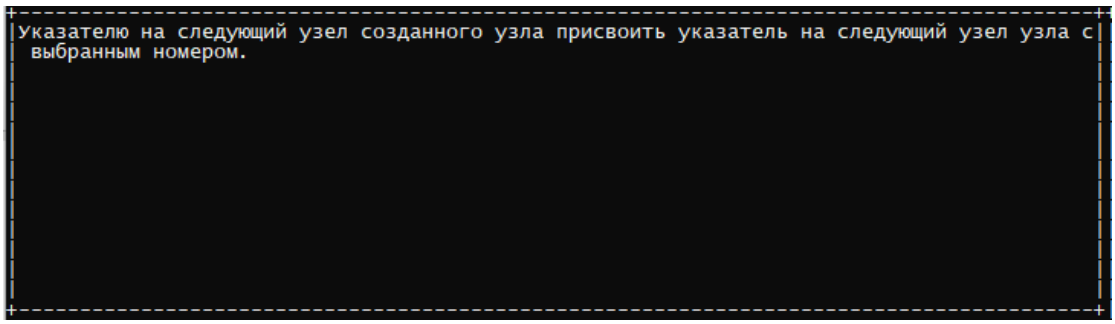


Рисунок 2.10 – Область визуализации с комментариями

На рисунке 2.11 показана область, в которой расположена сама визуализация списка

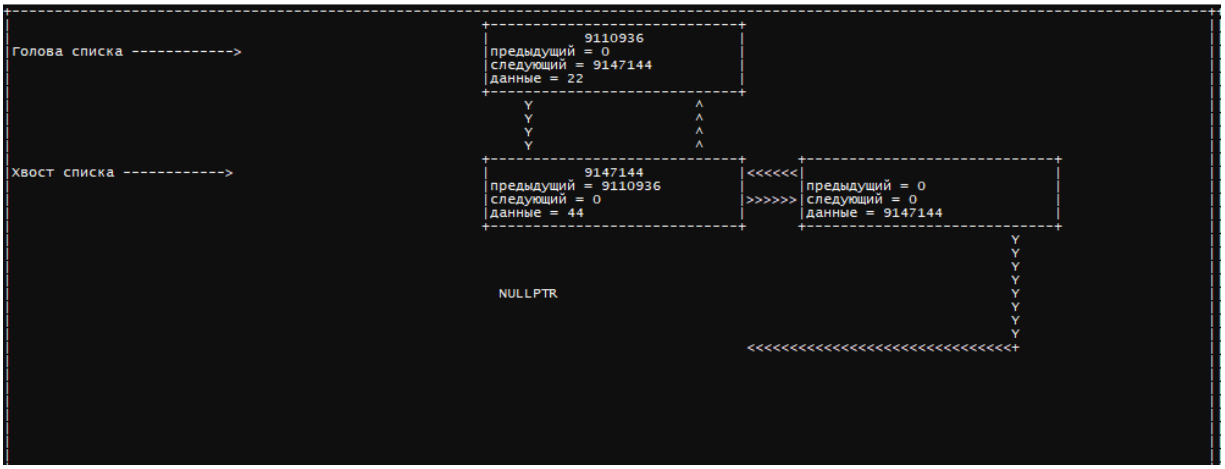


Рисунок 2.11 – Область визуализации с визуальным представлением списка

Когда пользователь начал визуализацию любой операции, он может на любом из шагов просмотреть предыдущие шаги, нажав стрелку влево.

Когда пользователь находится в меню, он может выйти из визуализации нажав кнопку 0

2.6 Сообщения системы

В таблице 2.4 приведены сообщения системы.

Таблица 2.4 – Сообщения системы

Сообщение	Причина возникновения
Ответ верный	Пользователь ответил на вопрос верно
Ответ неверный	Пользователь ответил на вопрос неверно
Вы прошли тест	Пользователь успешно прошел тест
Вы не прошли тест	Пользователь не прошел тест

В случае вывода программой других сообщений, не представленных в таблице, сообщите об этом разработчику программного обеспечения.

3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

3.1 Проверка работоспособности выдачи теоретического материала

1. Запустить программу на выполнение, в главном меню программы (см. рис. 2.2) выбрать пункт 1 и убедиться, что открыта первая страница теории (см. рис. 2.3).
2. Последовательно понажимав клавиши «стрелка влево», и «стрелка вправо», убедиться, что материал пролистывается назад и вперед.
3. Нажать клавишу Q и удостоверится, что появилось главное меню. (см. рис. 2.2)

3.2 Проверка работоспособности визуализации

1. Запустить программу на выполнение, в главном меню (см. рис. 2.2) программы выбрать пункт 3 и убедиться, что открыто меню с выбором операции (см. рис. 2.8).
2. Выбрать операцию 1 и ввести число 42 должна начаться визуализация операции вставки в начало списка
3. Нажать стрелку вправо 1 раз, должно снова появится меню (см. рис. 2.8).
4. Выбрать операцию 1 и ввести число 47 должна начаться визуализация операции вставки в начало списка
5. Нажать стрелку вправо 3 раза, должно снова появится меню (см. рис. 2.8).
6. Выбрать операцию 2 должна начаться визуализация операции удаления из начала списка
7. Выбрать операцию 1 и ввести число 47 должна начаться визуализация операции вставки в начало списка
8. Нажать стрелку вправо 3 раза, должно снова появится меню (см. рис. 2.8).
9. Выбрать операцию 3 ввести 33 и ввести адрес 2 элемента списка, должна начаться визуализация вставки в середину
10. Нажимать стрелку вправо до тех пор, пока не появится меню (см. рис. 2.8).
11. Выбрать операцию 4 и ввести адрес 2 элемента, должна начаться визуализация удаления второго элемента.
12. Нажимать стрелку вправо до тех пор, пока не появится меню (см. рис. 2.8).
13. Выбрать операцию 6 должна начаться визуализация удаления последнего элемента.
14. Нажимать стрелку вправо до тех пор, пока не появится меню (см. рис. 2.8).
15. Выбрать операцию 5 и ввести 1337 должна начаться визуализация добавления после последнего элемента.
16. Нажимать стрелку вправо до тех пор, пока не появится меню (см. рис. 2.8).

17. Выбрать операцию 7 и ввести 1337 должна начаться визуализация поиска
18. Нажать стрелку влево 10 раз, убедиться, что предыдущие шаги показываются
19. Нажимать стрелку вправо до тех пор, пока не появится меню, убедиться, что во время визуализации был найден последний элемент.
20. Нажать кнопку 0, убедиться, что вывелось главное меню. (см. рис. 2.2)

3.3 Проверка работоспособности тестирования

1. Запустить программу на выполнение, в главном меню программы выбрать пункт 2 и убедиться, что запрошено имя пользователя (см. рис. 2.4)
2. Ввести test
3. Удостовериться, что появился один из вопросов (см. рис. 2.5)
4. Нажать стрелку вправо, удостовериться, что перед первым вариантом появилось слово «[выбрано]» (см. рис. 2.6)
5. Нажать кнопку ENTER, удостовериться, что ответ был проверен и появилось сообщение «Ответ верный» или «Ответ неверный» и появился новый вопрос (см. табл. 2.4)
6. Нажать стрелку влево, удостовериться, что появился другой вопрос
7. Нажать на кнопку ENTER 4 раза, должно появиться сообщение «Вы не прошли тест» (см. табл. 2.4) и в папке DoublyLinkedListVisualizer должен появиться файл result.dat и если его открыть, то в нем будет «[<Дата и время проведения теста>] test не прошел тест»
8. Нажать кнопку ENTER, должно появиться главное меню (см. рис. 2.2)

ЗАКЛЮЧЕНИЕ

В результате курсового проектирования разработана учебно-демонстрационная программа «Двусвязный список». Программа предоставляет теоретический материал для ознакомления с внутренним устройством и реализацией двусвязного списка, а также тестирование, позволяющее проверить полученные знания.

Программа отвечает поставленным требованиям и может быть использована для обучения студентов высших учебных заведений.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Прата С. Язык программирования C++. Лекции и упражнения – М. ООО «И.Д. Вильямс» 2016 – 1248 стр.
2. Васильев А. Программирование на C++ в примерах и задачах – М: Эксмо, 2018 – 368с.
3. Белов С.В., Лаптев В.В., Морозов А.В., Толасова В.В., Мамлеева А.Р. Требования к оформлению студенческих работ. - Астрахань, АГТУ, 2017. 104 с.

ПРИЛОЖЕНИЕ 1

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на разработку учебно-демонстрационной программы

«Двусвязный список»

по дисциплине «Программирование и информатика»

Направление 09.03.04 Программная инжинерия

Исполнитель: обучающийся гр. ДИПРб-11 **Тагиров Р. Р.**

1. **Назначение, цели и задачи разработки**

Цель разработки – автоматизация обучения работе с двусвязным списком

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 инжинерия» 2019 года набора.
   - Рабочая программа дисциплины «Программирование и информатика».
   - Распоряжение по кафедре АСОИУ №____ от «___»_____ 2020 г.
4. **Плановые сроки выполнения –** весенний семестр 2019/20 учебного года:

Начало «25» февраля 2020 г.

Окончание «01» июня 2020 г.

5. **Требования к проектируемой системе**

5.1 Требования к функциональным характеристикам

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

- Предоставление пользователю теоретического материала по теме «Двусвязный список»
  - текст разбит на страницы не более 20 строк по 60 символов в строке
  - программа должна обеспечивать пролистывание как в прямом, так и в обратном направлении;
  - текст с теоретическим материалом разбит на абзацы и структурирован;

- текст хранится в файле, должен быть зашифрован (закодирован) таким образом, чтобы его нельзя было прочесть стандартными средствами операционной системы.

- Визуализация работы с двусвязным списком:
  - показано состояние списка с указанием адресов узлов и связей между ними
  - показан исходный код выполняющейся операции и указана строка
  - показаны комментарии к выполняющейся в данный момент строке
  - доступна операция добавления в начало
  - доступна операция добавления с после узла с данным адресом
  - доступна операция добавления в конец
  - доступна операция удаления из начала
  - доступна операция удаления из конца
  - доступна операция удаления узла с данным адресом
  - доступна операция поиска узла с данным содержимым
- Тестирование пользователя по теме «Двусвязный список»:
  - вопросы закрытого типа, количество предлагаемых вариантов ответа не менее 2 и не более 8, правильных ответов от 0 до 8
  - общее количество вопросов в базе не менее 10, база представляет собой файл, который должен быть зашифрован (закодирован) таким образом, чтобы его нельзя было прочесть стандартными средствами операционной системы;
  - формат файла и вопросы разрабатываются исполнителем самостоятельно;
  - пользователь выбирает ответ путем нажатия на кнопки «стрелка вверх» и «стрелка вниз» на клавиатуре; ответ считается верным, если выбраны все верные варианты ответа, за ответ присуждается количество баллов, написанное в безе с вопросами
  - пользователь может пропустить вопрос и вернуться к нему позже
  - после ввода ответа должно быть выведено сообщение о текущем результате, по завершении тестирования – общий итог;
  - тестирование считается успешным, если количество баллов, набранных пользователем превышает 5
- **Интерфейс программы:** текст русский, шрифт кириллический, заголовки, термины и другая важная информация выделены цветом.

Система имеет функциональные ограничения:

- количество элементов списка не превышает 6
- Данные в списке по модулю не превышают INT_MAX

5.2 Требования к эксплуатационным характеристикам

Программа не должна аварийно завершаться при любых действиях пользователя.

5.3 Требования к программному обеспечению:

Средства разработки: текстовый редактор VS Code (версия 1.45), компилятор gcc, язык C++ (стандарт C++ 11 и выше).

Операционная система: Windows 7 и выше.

5.4 Требования к аппаратному обеспечению:

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 2 ГБ ОЗУ;
- не менее 20 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. Прата С. Язык программирования C++. Лекции и упражнения – М.: ООО «И.Д. Вильямс» 2016 – 1248 стр.
2. Васильев А. Программирование на C++ в примерах и задачах – М.: Эксмо, 2018 – 368с.
3. Белов С.В., Лаптев В.В., Морозов А.В., Толасова В.В., Мамлеева А.Р. Требования к оформлению студенческих работ. - Астрахань, АГТУ, 2017. 104 с.

ПРИЛОЖЕНИЕ 2 База с вопросами и теорией

В таблице П2.1 представлено содержимое базы с вопросами

Таблица П2.1 — База с вопросами

Вопрос	Стоимость	Верный	Вариант
За какое время происходит добавление элемента в двусвязный список?	1	Да	за O(n)
			за O(1)
			за O(n^2)
За какое время происходит удаление элемента из двусвязного списка?	1	Да	за O(n)
			за O(1)
			за O(n^2)
За какое время происходит добавление/удаление в начале списка?	1	Да	за O(1)
			за O(n)
			за O(n^2)
Какие из данных утверждений верны?	2	Да	Каждый элемент списка содержит ключ, который идентифицирует этот элемент.
			В двусвязном списке элементы можно добавлять в любое место за константное время.
			В двусвязном списке каждый из элементов содержит информационную часть и указатель на следующий элемент.
		Да	В последнем и первом узле списка один из указателей нулевой
Любой список является связным?	1	Да	Да
			Нет

Продолжение таблицы П2.1

Вопрос	Стоимость	Верный	Вариант
Какие из данных утверждений верны?	2	Да	Любой список является упорядоченным
			Размер двусвязного списка должен быть известен заранее
			Указатель на голову в двусвязном списке не обязан указывать на начало списка
Над списком была произведена операция добавления в начало и операция добавления после пятого элемента, что будет выполнено быстрее?	2	Да	Операция добавления в начало будет быстрее
			Операция добавления после пятого элемента будет быстрее
			Они обе будут выполнены за одно и тоже время
В программе список представлен указателем на начало и на его хвост, в нем 10 элементов, над ним была произведена операция добавлениявв начало, операция добавления в конец и операция удаления 4 элемента, какие из утверждений верные?	4	Да	Операции добавления в начало и в конец списка произойдут примерно за одно и тоже время
			Операция добавления элемента в начало списка произойдет быстрее чем в конец
		Да	Операция удаления 4 элемента будет выполняться дольше чем операция добавления в конец списка
			Операция удаления 3 элемента произошла бы за то же время, что и операция удаленяи 4 элемента
			Все три операции произойдут за одно и то же время

Продолжение таблицы П2.1

Вопрос	Стоимость	Верный	Вариант
Какие из данных утверждений верны?	2	Да	Обход двусвязного списка возможен как вперед, так и назад
			Двусвязный список является контейнером с возможностью произвольного доступа
			Обход списка вперед происходит быстрее чем назад

В таблице П2.2 представлено содержимое базы с теорией

Таблица П2.2 — База с теорией

№	Содержимое
1	Связный список. Структура данных, представляющая собой конечное множество упорядоченных элементов (узлов), связанных друг с другом посредством указателей, называется связным списком. Каждый элемент связного списка содержит поле с данными, а также указатель (ссылку) на следующий и предыдущий элемент. Эта структура позволяет эффективно выполнять операции добавления и удаления элементов для любой позиции в последовательности. Причем это не потребует реорганизации структуры, которая бы потребовалась в массиве. Минусом связного списка, как и других структур типа «список», в сравнении его с массивом, является отсутствие возможности работать с данными в режиме произвольного доступа, т. е. список – структура последовательно доступа, в то время как массив – произвольного. Последний недостаток снижает эффективность ряда операций.

	Та особенность двусвязного списка, что каждый элемент имеет две ссылки: на следующий и на предыдущий элемент, позволяет двигаться как в его конец, так и в начало. Операции добавления и удаления здесь наиболее эффективны, чем в односвязном списке, поскольку всегда известны адреса тех элементов списка, указатели которых направлены на изменяемый элемент. Но добавление и удаление элемента в двусвязном списке, требует изменения большого количества ссылок, чем этого потребовал бы односвязный список. Возможность двигаться как вперед, так и назад полезна для выполнения некоторых операций, но дополнительные указатели требуют задействования большего количества памяти, чем таковой необходимо в односвязном списке.

2	Реализация связного списка
	
	Общая форма описания узла двунаправленного связного списка выглядит так:
	
	struct имя_списка
	{
	информационное_поле_1;
	информационное_поле_2;
	…
	информационное_поле_n;
	
	указатель_на_следующий_элемент;
	указатель_на_предыдущий_элемент;
	};
	
	Например определение узла двусвязного списка может выглядеть так:
	
	struct node
	{
	int data;

	node* next;
	node* prev;
	};
	+++
	Операции над списком: Добавление элемента.
	
	bool insert(node *begin, size_t where, uint32_t data)
	{
	while (begin && where)
	{
	begin = begin->next;
	where--;
	}
	if (!begin) return false;
	node *temp = node_init(data);
	temp->next = begin->next;
	temp->previous = begin;
	if (begin->next) begin->next->previous = temp;
	begin->next = temp;
	return true;
	}
	
	Как видно, функция принимает на вход начало списка, место,
	где нужно добавить новый элемент и, собственно, сами данные.
	
	Внутри тела функции есть цикл,который последовательно проходит
	по всем узлам вплоть до узла под номером where. Это говорит о
	том, что добавление
	происходит за время O(N)
	
	Также в двусвязном списке нельзя добавить элемент в произвольном
	месте, как это можно сделать в массиве, что говорит о том, что список -
	это структура не произвольного доступа.

3	Операции над списком: Удаление элемента.

	bool remove(node *begin, size_t where)
	{
	  while (begin && where)
	  {
	  begin = begin->next;
	  where--;
	  }
	  if (!begin)
	  return false;
	  node *temp = begin;
	  if (begin->previous)
	  begin->previous->next = begin->next;
	  if (begin->next)
	  begin->next->previous = begin->previous;
	  delete begin;
	  return true;
	}
	
	Улаление элемента из списка происходит аналогично добавлению.
	Операции выделения памяти противопоставляется операция высвобождения,
	а узлы в месте удаления будут указывать друг на друга, а не на удаленный узел.

4	Операции над списком: Просмотр значения узла.
	
	bool at(node *begin, size_t where, uint32_t &result)
	{
	  while (begin && where)
	  {
	  begin = begin->next;
	  where--;
	  }
	  if (!begin)
	  return false;
	  result = begin->data;
	  return true;

	} Как видно, даже чтобы просто посомтреть значение необходимого узла нужно пройти по всем узлам до него. Отсюда можно сделать вывод что операции удаления, добавления и итерации по списку происходят за время O(N)

5	Операции над списком: Добавление/удаление элемента в начале списка.
	
	void push_back(node *begin, uint32_t data)
	{
	node* temp = node_init(data);
	temp->next = begin;
	begin = temp;
	}
	
	uint32_t pop_back(node *begin)
	{
	uint32_t data = begin->data;
	node* temp = begin;
	begin = begin->next;
	delete temp;
	return data;
	}
	
	Как видно, в список все же можно добавлять элементы за константное время, если вставка/удаление происходит в начало списка.

ПРИЛОЖЕНИЕ 3 Содержимое массива codes

Массив codes в части с визуализацией будет содержать исходный код операций над списком на языке C++.

Его содержимое представлено в таблице ниже.

Таблица П3.1 — Содержимое массива codes

№	Операция	Содеримое

1	Добавление в начало	void push_back(uint32_t data)
		{
		node* begin = list.begin;
		if(begin)
		{
		begin->previous = node_init(data);
		begin->previous->next = begin;
		list = begin->previous;
		}else
		list = node_init(data);
		}

2	Удаление из начала	uint32_t pop_back()
		{
		node* begin = list.begin;
		uint32_t data = begin->data;
		if(begin->next) begin->next->previous = nullptr;
		node* next = begin->next;
		delete begin;
		list = next;
		return data;
		}

3	Вставка в середину	bool insert(size_t where, uint32_t data)
		{
		node* begin = list.begin;
		while (begin != where && begin)
		{
		begin = begin->next;
		}
		if (begin!=where)
		return false;
		node *temp = node_init(data);
		temp->next = begin->next;
		temp->previous = begin;
		if (begin->next)
		begin->next->previous = temp;
		begin->next = temp;
		return true;
		}