Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машинных

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Психологические тесты»

БГУИР КП 1- 40 02 01 322 ПЗ

Студент: гр. 950503 Полховский А.Ф.

Руководитель: старший преподаватель

кафедры ЭВМ Ковальчук А.М.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Факультет компьютерных систем и сетей

                                                                                   УТВЕРЖДАЮ

                                                                                   Заведующий кафедрой ЭВМ

                                                                                            Б.В. Никульшин

                                                                                                      (подпись)

                                                                                    «     »             2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Полховскому Артему Федоровичу

1. Тема проекта *Психологическе тесты.*

2. Срок сдачи студентом законченного проекта *15 декабря 2020 г*.

3. Исходные данные к проекту Файлы Users (содержит логины пользователей), Questions (содержит сущности вопросов), Answers (содержит информацию об возможных ответах пользователя) Results(содержит возможные результаты тестирования) stats (содержит статистику тестирования).

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Перечень используемых сокращений. Введение. 1.Постановка задачи. 2.Обзор выбранных методов и алгоритмов. 3.Обоснование выбранных методов и алгоритмов. 4.Описание программы для программиста. 5.Разработка алгоритмов. 6.Руководство пользователя. 7.Заключение. Литература. Содержание.*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

-1 Диаграмма классов

-2 Схема алгоритма int runTest(List<Question>& qns)

-3 Схема алгоритма void push\_back(T data)

6. Консультант по проекту *Ковальчук А.М.*

7. Дата выдачи задания *12 сентября 2020 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1,2 к 1 октября 2020 г. – 20 %;*

*разделы 3,4 к 1 ноября 2020 г. – 30 %;*

*разделы 5,6,7 к 1 декабря 2020 г. – 30 %;*

*оформление пояснительной записки и графического материала к 14 декабря 2020 г. 20 %*

*Защита курсового проекта с 21 декабря 2020 г. по 28 декабря 2020 г.*

РУКОВОДИТЕЛЬ *Ковальчук А.М.*

(подпись)

 Задание принял к исполнению  *Полховский А.Ф.*

(дата и подпись студента)

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ 3](#_Toc59672320)

[ВВЕДЕНИЕ 4](#_Toc59672321)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc59672322)

[2 ОБЗОР ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ 6](#_Toc59672323)

[3 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ 8](#_Toc59672324)

[4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА 10](#_Toc59672325)

[4.1 Описание классов 10](#_Toc59672326)

[4.2 Описание структуры. 1](#_Toc59672327)5

[5 РАЗРАБОТКА АЛГОРИТМОВ 1](#_Toc59672328)6

[5.1 Схемы алгоритмов. 16](#_Toc59672329)

[5.2 Алгоритм по шагам метода T& operator[](int index). 16](#_Toc59672330)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 1](#_Toc59672332)7

[ЗАКЛЮЧЕНИЕ 2](#_Toc59672333)5

[ЛИТЕРАТУРА 2](#_Toc59672334)6

[ВЕДОМОСТЬ ДОКУМЕНТОВ 2](#_Toc59672335)7

[ПРИЛОЖЕНИЕ А 2](#_Toc59672336)8

[ПРИЛОЖЕНИЕ Б 2](#_Toc59672337)9

[ПРИЛОЖЕНИЕ В](#_Toc59672338) 30

[ПРИЛОЖЕНИЕ Г](#_Toc59672339) 31

**ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ**

ОС – операционная система.

ЭВМ – электронно-вычислительная машина.

VS – Visual Studio.

STL (Standard Template Library) – Библиотека стандартных шаблонов.

UML – унифицированный язык моделирования (Unified Modeling Language).

**ВВЕДЕНИЕ**

Объектно-ориентированное программирование представляет собой технологию программирования, которая базируется на классификации и абстракции объектов. Одним из наиболее популярных средств объектно-ориентированного программирования, позволяющим разрабатывать программы, эффективные по объёму кода и скорости выполнения является С++.

C++ — [компилируемый](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [статически типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения. Помимо объектно-ориентированного программирования язык поддерживает такие парадигмы программирования, как процедурное и обобщенное программирование. Язык оснащён богатой стандартной библиотекой, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные функции, возможность поддержки многопоточности и многие другие возможности. C++ сочетает свойства как [высокоуровневых](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), так и [низкоуровневых языков](https://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). В сравнении с его предшественником — языком [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) — наибольшее внимание уделено поддержке [объектно-ориентированного](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [обобщённого программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

С++ широко используется в современном мире. На нём разрабатываются различные прикладные программы, операционных системы, движки для компьютерных игр, приложения для встраиваемых систем, драйверов устройств, а также многие другие программы, требующие от себя эффективность скорости исполнения и объема кода. С помощью него реализованы такие популярные программы, как Google Chrome, Adobe Photoshop, Visual Studio, Microsoft Word, СCleaner, Telegram Messenger, Steam и многие другие.

Исходя из вышеупомянутого, можно сделать вывод, что С++ является неплохим универсальным языком программирования для написания данной курсовой работы.

1. **ПОСТАНОВКА ЗАДАЧИ**

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В программе должна быть предусмотрена возможность хранения данных в нескольких файлах, связанных определённым образом. Данные должны хранить следующую информацию:

- информацию о пользователях программы;

- имена и логины пользователей;

- заготовленные вопросы и ответы;

- ответы пользователей;

- статистику тестирования;

Программа должна выполнять:

- добавление и удаление пользователей;

- прохождение теста;

- подсчёт результата тестирования;

- редактирование исходных тестов;

- добавление новых тестов;

- выдача статистики прохождения тестов пользователями;

При реализации операции редактирования, добавления, удаления информации необходимо предусмотреть операцию отмены последних действий. Разработать иерархию классов с использованием наследования (не меньше 3-х уровней наследования). Разработать и использовать в программе классы контейнеров, итераторов и алгоритмов (свой и STL). Производить обработку исключительных ситуаций.

# **ОБЗОР ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ**

Перед решением поставленной задачи необходимо определить, как следует хранить информацию и как ее обрабатывать.

Одной из наиболее полезных особенностей языка C++ является возможность определять собственные типы данных, которые будут лучше соответствовать в решении конкретных проблем. Таким типом данных являются классы.

В языке C++ классы очень похожи на структуры, за исключением того, что они обеспечивают гораздо большую мощность и гибкость. Помимо хранения данных, классы могут содержать и функции. Функции, определенные внутри класса, называются **методами**.

Все данные и методы классов имеют права доступа. По умолчанию, все содержимое класса является доступным для чтения и записи только для него самого. Для того, чтобы разрешить доступ к данным класса извне, используют модификаторы доступа.

Стандарт C++ представляет собой универсальный механизм для создания библиотек структур данных. Данный язык имеет собственную встроенную библиотеку классов-контейнеров – STL (Стандартной библиотекой шаблонов), элементы которой могут использоваться для хранения и обработки данных в зависимости от структурной организации программы.

Контейнеры в С++ – способы организации хранения данных. Они могут как подключаются к программе с помощью разработанных вручную шаблонных классов, так и с помощью встроенных библиотек STL. Контейнеры удобны тем, что хранить и обрабатывать данные программисту становиться проще, за счет наличия возможности изменять тип хранимых в них данных.

В данной программе использован самодельный контейнер в виде линейного однонаправленного списка.

Список — это структура данных элементов одного типа, связанных между собой последовательно посредством указателей – элемент списка, или по-другому, узел списка, хранит в себе указатель на следующий элемент списка. При чем узел, находящийся на самом верхнем уровне, на который не ссылаются другие узлы, называется первым (головным) элементом, головой списка.

Программе необходимо сохранять данные на диск и считывать данные из него. В стандартах С++ предложена новая концепция, которая гораздо прозрачнее и проще в использовании, чем была у языка С. При форматированном вводе/выводе числа хранятся на диске в виде серии символов. Есть несколько потенциальных проблем с форматированным выводом в дисковые файлы. Во-первых, надо разделять числа нечисловыми константами. Поскольку они хранятся в виде последовательности символов, а не в виде полей фиксированной длины, это единственный шанс узнать при извлечении, где кончается одно и начинается другое число. Во-вторых, между строками должны быть разделители – по тем же причинам.

**3 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ**

Для создания подходящей структуры и логики программы будем использовать классы, которые будут отвечать за вопросы и результаты. Также необходимы некоторые классы, которые будут использоваться программой для оперативной обработки данных. Вся информация будет храниться как в текстовых, так и в бинарных файлах.

В оперативной памяти данные вопросов и ответов должны будут размещаться в виде динамической структуры данных. А так как информация о вопросах будет хранится в объектах, то элементами этой структуры будут являться экземпляры класса. Данные будут размещены в оперативной памяти в контейнере - список, потому что он обеспечивает лёгкую вставку информации в контейнер.

При добавлении, удалении и редактировании информации необходимо предусмотреть отмену последовательности действий. Пользователь должен иметь возможность отменить действия, которые он совершил при работе с программой. Чтобы это реализовать, необходимо сохранять информацию о проделанных пользователем действий. Т.е. о том, что пользователь изменил, добавил, удалил. После закрытия программы данную информацию необходимо удалить, поэтому сохранять ее в файле нет необходимости. Отмена последовательности действий должна производиться в обратном порядке. Первое совершенное действие должно отмениться последним, а последнее первым.

Для реализации этой операции целесообразно использовать стек. В стеке должен храниться код операции (удаление, добавление, редактирование) и сама информация. При совершении операции данные заносится в вершину стека, при ее отмене, информация извлекается из вершины.

При сохранении информации о забронированных парковочных местах, необходимо сохранять не весь экземпляр класса, а только его часть. При этом сохранять необходимо атрибуты класса, имеющие тип string и атрибут типа int. Поэтому для этой цели будем использовать форматированный файловый ввод/вывод, а не двоичный. Выбор данного метода организации файлового ввода/вывода обусловлено простым типом и небольшим объемом хранимых данных.

**4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА**

**4.1 Описание классов**

4.1.1 Класс «TextObject» - базовый класс, предназначенный для упорядоченного вывода данных наследуемых классов.

Таблица 4.1 – класс «TextObject»

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| text | String | | Поле, хранящее текст |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| TextObject | void | void | Конструктор по умолчанию |
| TextObject | string | void | Конструктор с параметрами |
| print | string | void | Метод для вывода текста |
| printIn | string | void | Метод для вывода строки |
| cls | void | void | Метод для очистки консоли |
| toString | void | void | Метод для вывода текста |

4.1.2 Класс «Question» - класс наследуемый от «TextObject», содержащий сущность вопроса.

 Таблица 4.2 – класс «Question»

|  |  |  |
| --- | --- | --- |
| Private поля класса | | |
| Имя | Тип | Описание |
| \_weight | List<int> | Значения очков, дающихся пользователю за определенный ответ |
| \_answer | List<string> | Список вариантов ответа на вопрос |
| \_question | string | Содержание вопроса |

Продолжение таблицы 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Question | void | void | Конструктор по умолчанию |
| Question | string question, List<string>& answer, List<int>&weight | void | Конструктор с параметрами |
| ToString | void | void | Метод для вывода вопроса |
| parseStringToQuestion | string question, string answer | void | Метод для назначения ответа вопросу и выбор очков за ответ |
| answersToString | void | string | Метод для создания строки с ответами |
| getQuestion | void | string | Метод для получения вопроса |
| getSize | void | int | Метод для получения значимости вопроса |
| getWeightAt | int | int | Метод для получения количества очков за ответ |
| edit | void | void | Метод для редактирования вопроса и ответов |
| runTest | List<Question>& | int | Метод для запуска теста |
| maxWeight | List<Question>& | int | Метод для получение максимального количества очков за ответ |
| copy | Question& | void | Метод для копирования вопроса |

4.1.3 Класс «Result» - класс, наследуемый от класса «Question», содержащий информацию о результате.

Таблица 4.3 – класс «Result»

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| result | List<string> | | Список результатов |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Result | void | void | Конструктор по умолчанию |
| Result | List<string> | void | Конструктор для инициализации |
| getSize | void | int | Метод для получения количества результатов |
| parse | string | void | Метод для добавления результата в список |
| get | int | string | Геттер для получения результата из списка |

4.1.4 Базовый класс «Exception», предназначенный для обработки исключений.

Таблица 4.4 – Класс «Exception»

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| error | int | | Поле с кодом ошибки |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Exception | void | void | Конструктор по умолчанию, создание ошибки с нулевым кодом |
| Exception | int | void | Конструктор создания ошибки с определенным кодом |

4.1.5 Класс «ListException», наследуемый от класса «Exception», содержащий список возможных ошибок связанных с работой списка.

Таблица 4.5 – Класс «ListException»

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| ListException | void | void | Конструктор по умолчанию |
| ListException | int | void | Конструктор с параметрами |
| error | void | void | Метод для обработки кода ошибки |

4.1.6 Класс «InputException», наследуемый от класса «Exception», содержащий список возможных ошибок ввода.

Таблица 4.6 – Класс «InputException»

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| InputException | void | void | Конструктор по умолчанию |
| InputException | int | void | Конструктор с параметром |
| error | void | void | Метод для обработки кода ошибки |

4.1.7 Класс «FileException», наследуемый от класса «Exception», необходимый для обработки ошибок связанных с файлами.

Таблица 4.7 – Класс «FileException»

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| FileException | void | void | Конструктор по умолчанию |
| FileException | int | void | Конструктор с параметром |
| error | void | void | Метод для обработки кода ошибки |

4.1.8 Шаблонный класс «List», предназначенный для хранения объектов.

 Таблица 4.8 – Класс «List»

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| head | Node<T>\* | | Указатель на первый элемент списка |
| Size | int | | Размер списка |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| List | void | void | Конструктор по умолчанию |
| ~List | void | void | Деструктор |
| pop\_front | void | void | Метод для удаления первого элемента в списке |
| push\_back | T | void | Метод для добавления элемента в конец списка |
| output | List<T> | void | Метод для вывода списка в консоль |
| clear | void | void | Метод для очистки списка |
| getSize | void | int | Метод для получения количества элементов в списке |
| operator[] | int | T& | Перегрузка оператора [] |
| push\_front | T | void | Метод для добавления элемента в начало списка |
| copy | List<T>& | void | Метод для копирования списка |
| insert | T data, int index | void | Метод для добавления элемента в список по индексу |
| removeAt | int | void | Метод для удаления элемента из списка по индексу |
| pop\_back | void | void | Метод для удаления последнего элемента в списке |

4.1.9 Класс «File», предназначенный для работы с текстовыми и бинарными файлами.

 Таблица 4.9 – Класс «File»

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| filename | string | | Поле с именем файла |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| File | string | void | Конструктор с инициализацией имени файла |
| getData | List<string>& | void | Метод для получения списка строк из файла |
| setData | List<string>& | void | Метод для записи списка строк в файл |
| addData | string | void | Метод для добавления строки в файл |
| changeFilename | string | void | Сеттер поля filename |

**4.2 Описание структуры.**

В Приложении А приведена диаграмма классов к прецеденту «Психологические тесты». Данная диаграмма иллюстрирует основные классы проекта с атрибутами и методами.

1. **РАЗРАБОТКА АЛГОРИТМОВ**

В данном разделе рассмотрены описания и схемы алгоритмов, использованные в программе.

## 5.1 Схемы алгоритмов.

В приложении Б приведены схемы методов void push\_back(T data) и int runTest(List<Question>& qns).

## 5.2 Алгоритм по шагам перегрузки оператора T& operator[](const int index)

Данный оператор возвращает содержимое списка по указанному индексу.

Шаг 1 Начало.

Входные данные: глобальная переменная указатель this, указывающий на текущий объект – список. Head – голова списка. data – поле объекта cписка, pNext – указатель на следующий элемент списка. Индекс передаваемый в оператор int index. Счетчик для сверки с индексом int counter = 0;

Шаг 2 Проверяем current=this->head!=nullptr, если существует, то переходим в Шаг 3.

Шаг 3 Проверяем, если counter==index Шаг 5, если нет Шаг 4.

Шаг 4 Инкрементируем счетчик counter++, переопределяем current на следующий узел current=current->next Шаг 2.

Шаг 5 Вернуть return значение current->data Шаг 6.

Шаг 6 Конец.

В Приложении В представлен листинг программы со всеми комментариями.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Приложение позволяет работать с тестами в двух режимах. В режиме обычного пользователя и в режиме администратора.

Обычный пользователь может пройти тест, а администратор добавлять, редактировать, изменять, удалять содержимое тестов, а также добавлять пользователей и администраторов.

Обычный пользователь имеет возможность пройти тест, который подгружается из файлов после успешного входа в систему, после успешного прохождения теста пользователь получает результат своего тестирования, который вычисляется по формуле: количество набранных баллов / количество максимальных баллов \* количество ответов – 1, разделить по модулю на количество вариантов ответа. Данная формула позволяет легко и быстро строить большинство тестов и выдавать для них корректный результат.

Администратор имеет возможность редактировать тест, осуществляется это изменением вопроса на выбранной позиции напрямую, через консольный ввод. Удаление вопроса происходит обычным удалением элемента из списка. Добавление вопроса, здесь также вводится новый вопрос и добавляется в конец списка.

Администратор может исправить свою ошибку отменой последнего действия, после каждого редактирующего действия предыдущее состояние тела теста сохраняется в оперативную память и при необходимости используется для исправления ошибки. После работы с редактированием тела теста Администратор может подтвердить изменения, после чего они запишутся в файл.

Администратор может добавить пользователя, для этого вводится логин для входа и права на то, является ли пользователь администратором или нет. После этого данные вносятся в файл users.bin в виде: идентификатор и метка, которые потом будут выгружаться при входе в систему и сверяться с набранными данными.

Для запуска программы нужно запустить файл PsychoTest.exe. В папке с этим фалом должны находиться нижеперечисленные файлы:

Файл «users.bin», который содержит логины пользователей.

Файлы с заготовленными вопросами, ответами, результатами «Question.bin», «Answer.bin», «Result.bin» соответственно,

Файл «tests.txt», который содержит имена файлов, содержащие сами вопросы, ответы к ним, возможные результаты.

Файл «stats.txt», который содержит статистику прохождения тестов пользователями, создастся сам, в случае отсутствия в папке с .exe.

При запуске программы (рис.6.1) отображается главное меню, где пользователю предлагается выбрать прежде всего выбрать тест из существующих, а затем ввести имя-login, для входа в тест.

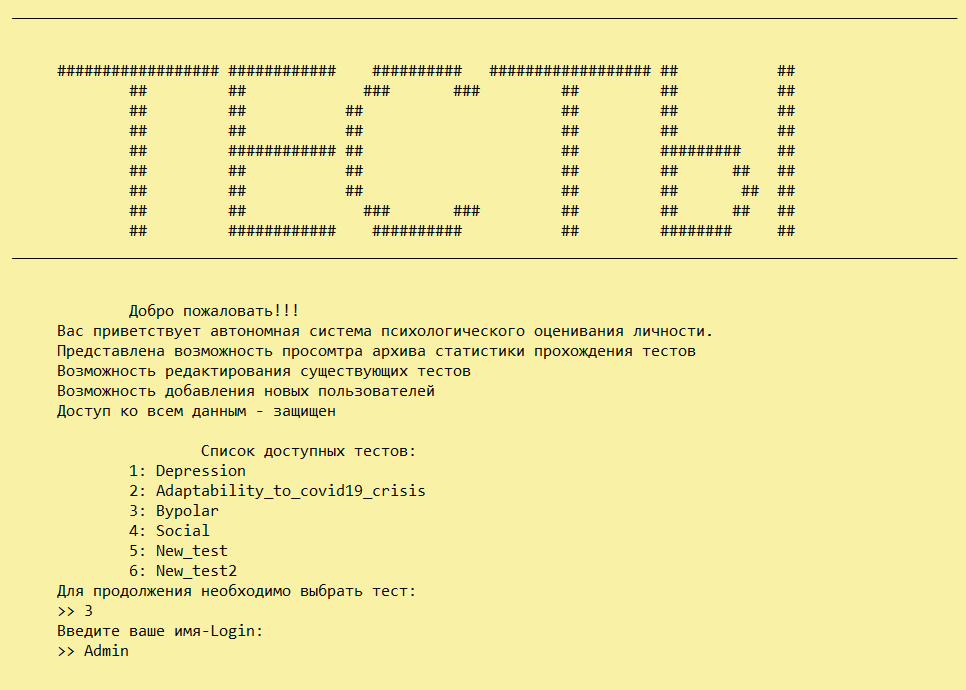


Рисунок 6.1

Если войти в систему под логином “Admin”, то откроется меню администратора, где ему нужно будет выбрать тип действия, который он хочет произвести.

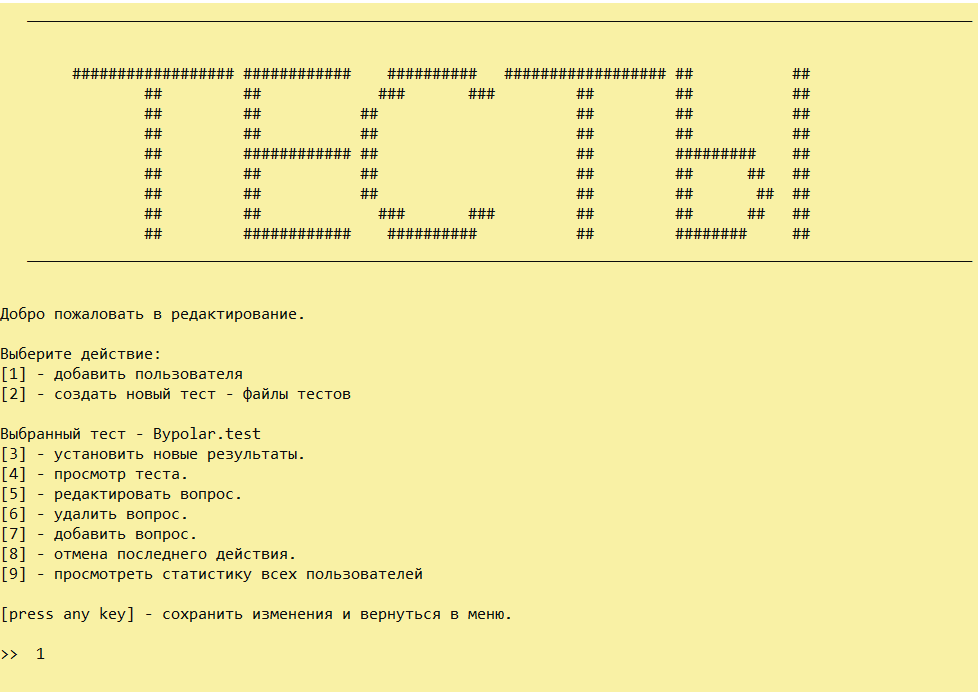


Рисунок 6.2

Пользователь с правами администратора может выполнить следующие дейcтвия:

* Добавлять новых пользователей. C правами администратора и без.
* Создавать новые файлы тестов.
* Устанавливать результаты в выбранный тест.
* Просматривать все данные выбранного теста (вопросы, возможные ответы, результаты).
* Удалять, редактировать вопрос из текущего теста.
* Добавлять вопрос с ответами.
* Отменять последние действия
* Просматривать/очищать статистику прохождения тестов всеми пользователями

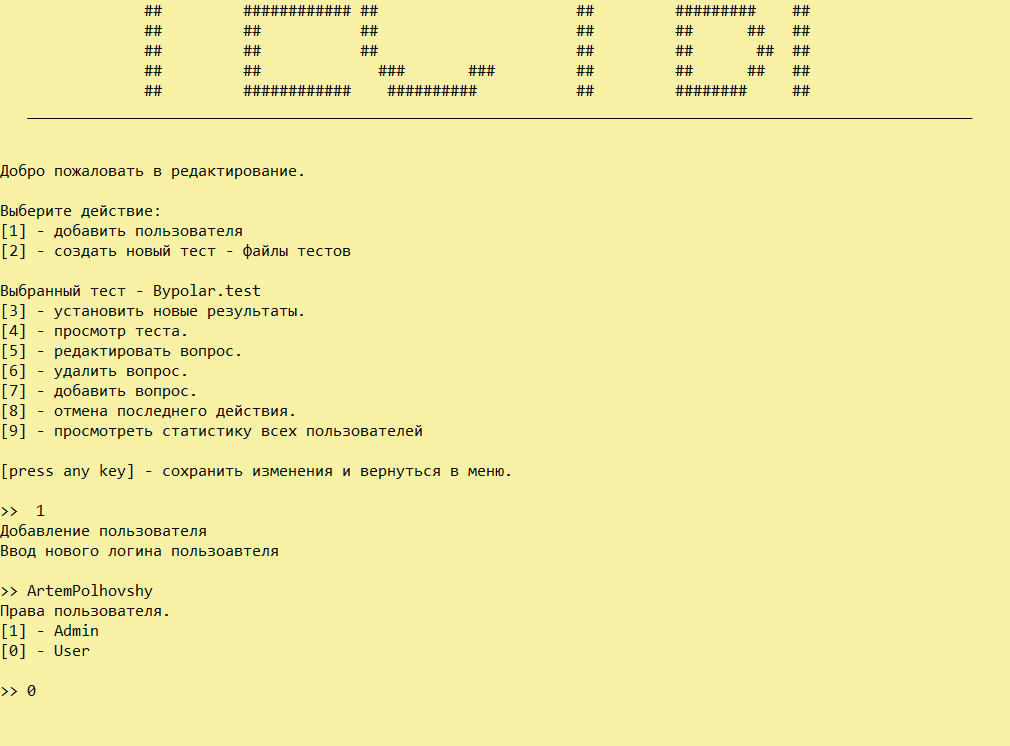


Рисунок 6.4

На рисунке 6.4 представлено поведение программы при добавлении пользователя. В дальнейшем, данное имя можно использовать для входа в систему с правами user (рис. 6.5).

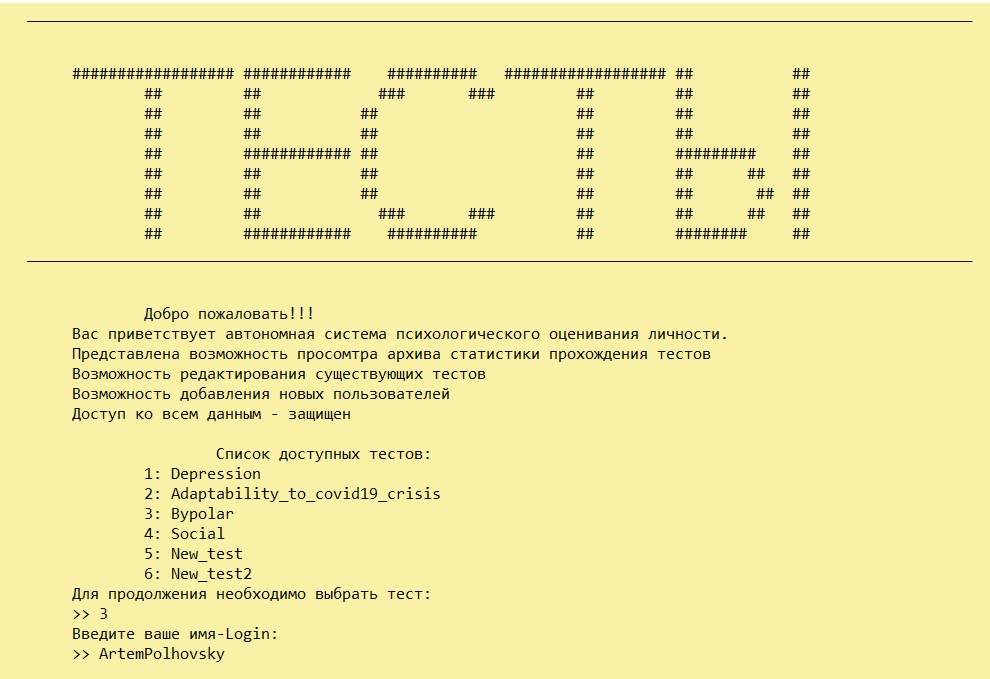


Рисунок 6.5

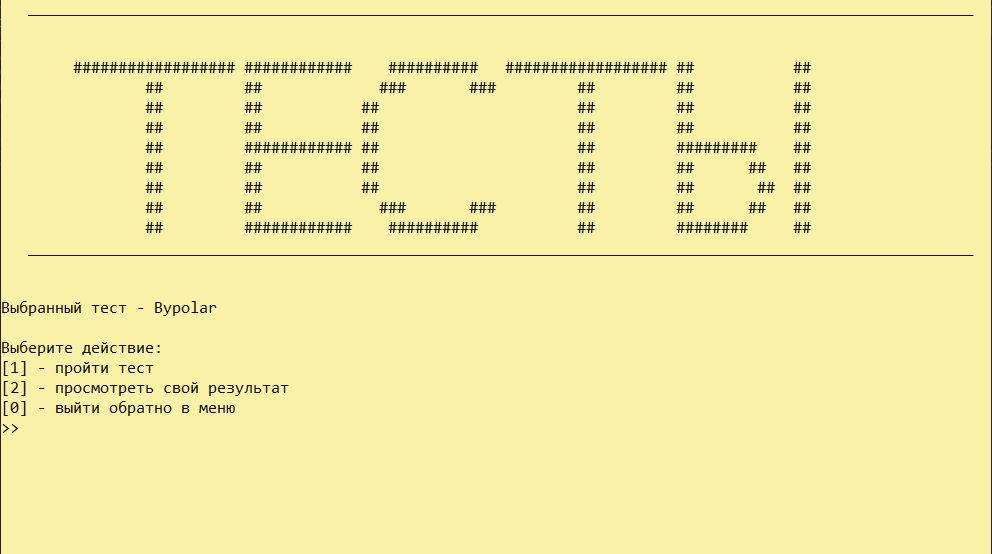


Рисунок 6.6

Меню пользователя с правами user выглядит следующим образом (рис.6.6).

Пользователю предлагается:

* Пройти выбранный тест.
* Просмотреть результат прохождения теста.

При попытке просмотреть результаты прохождения без прохождения теста пользователю будет сообщено об ошибке (рис6.7).

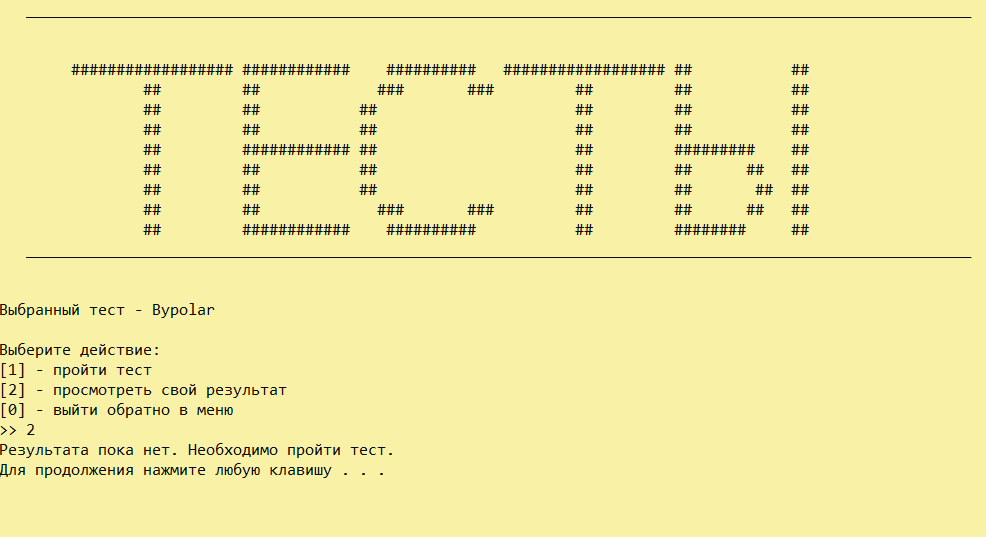


Рисунок 6.7

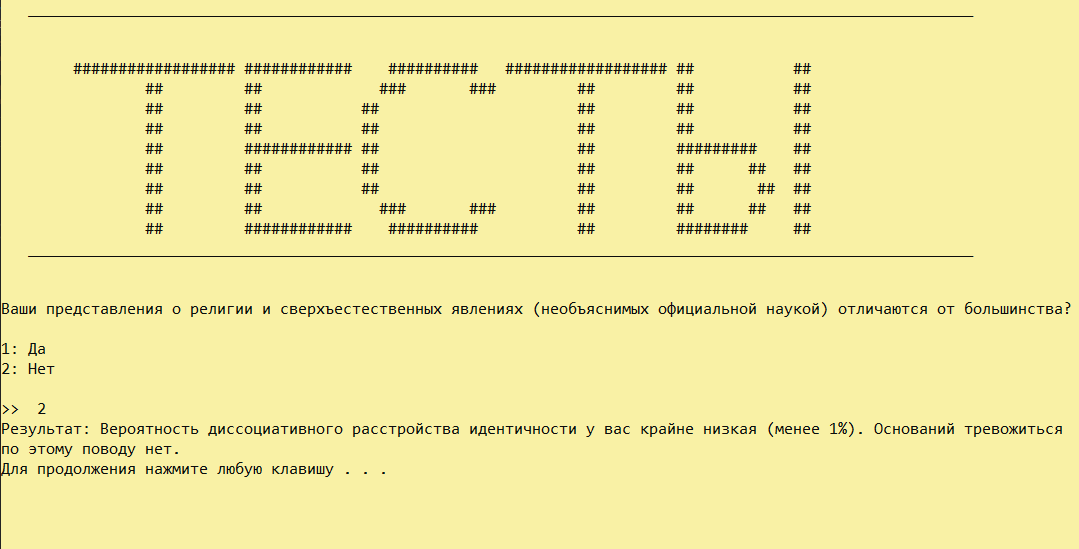


Рисунок 6.8

Вопросы и возможные ответы во время прохождения выводятся в консоль по одному, после ввода ответа пользователя консоль очищается, выводится следующий вопрос, и так до тех пор, пока не закончатся вопросы. После прохождения теста пользователю будут представлен его результат (рис.6.8).

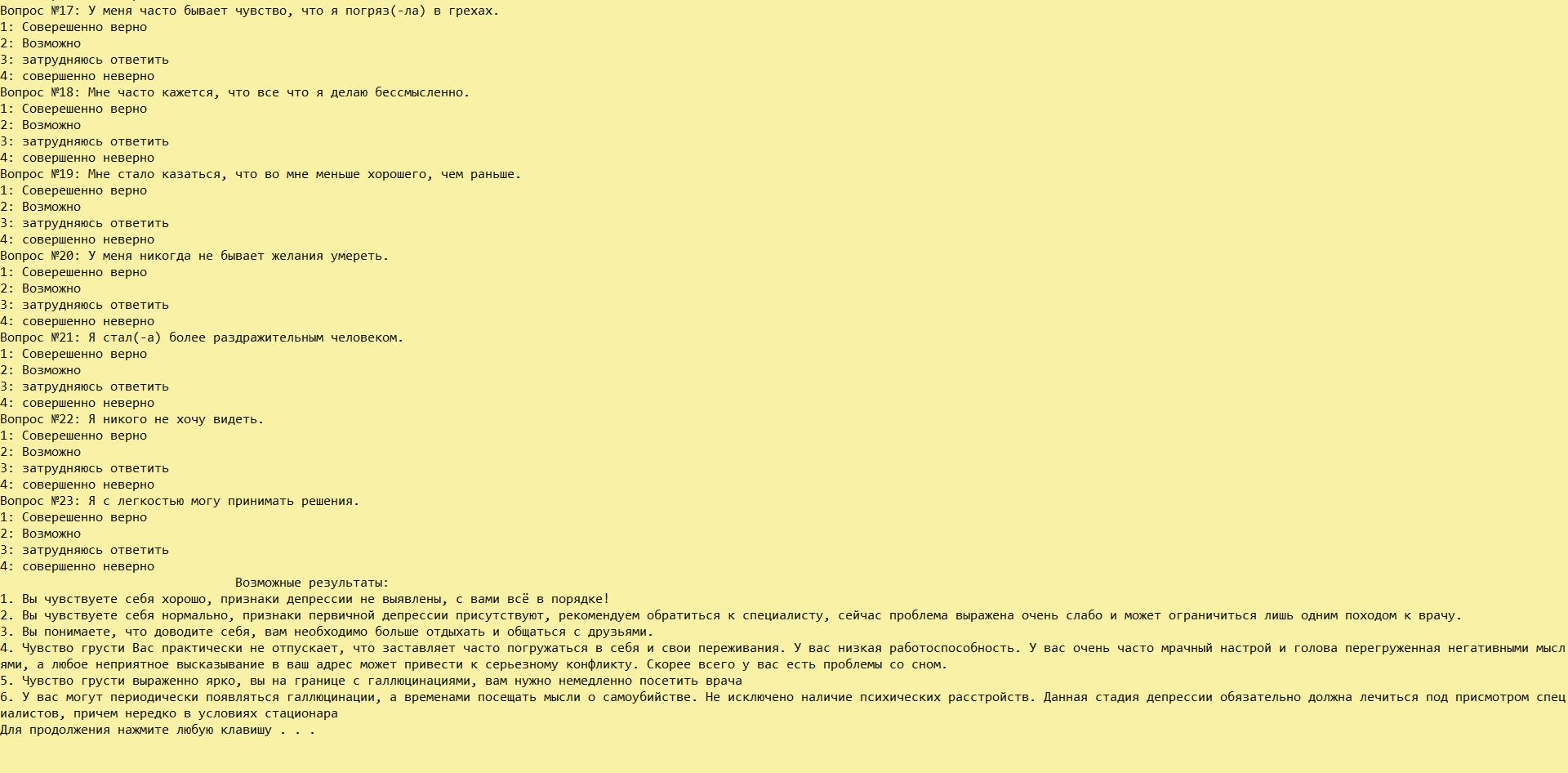


Рисунок 6.9

На рисунке 6.9 представлен результат исполнения программы при выборе администратором “[4] – Просмотр теста” раздела меню.

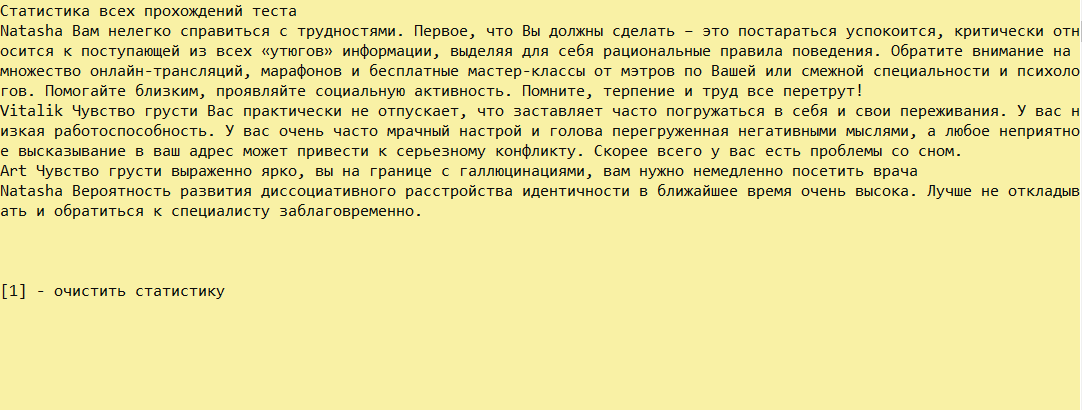


Рисунок 6.10

Вывод статистики всех пользователей, доступный пользователю с правами администратора, с возможностью очистки (рис. 6.10).

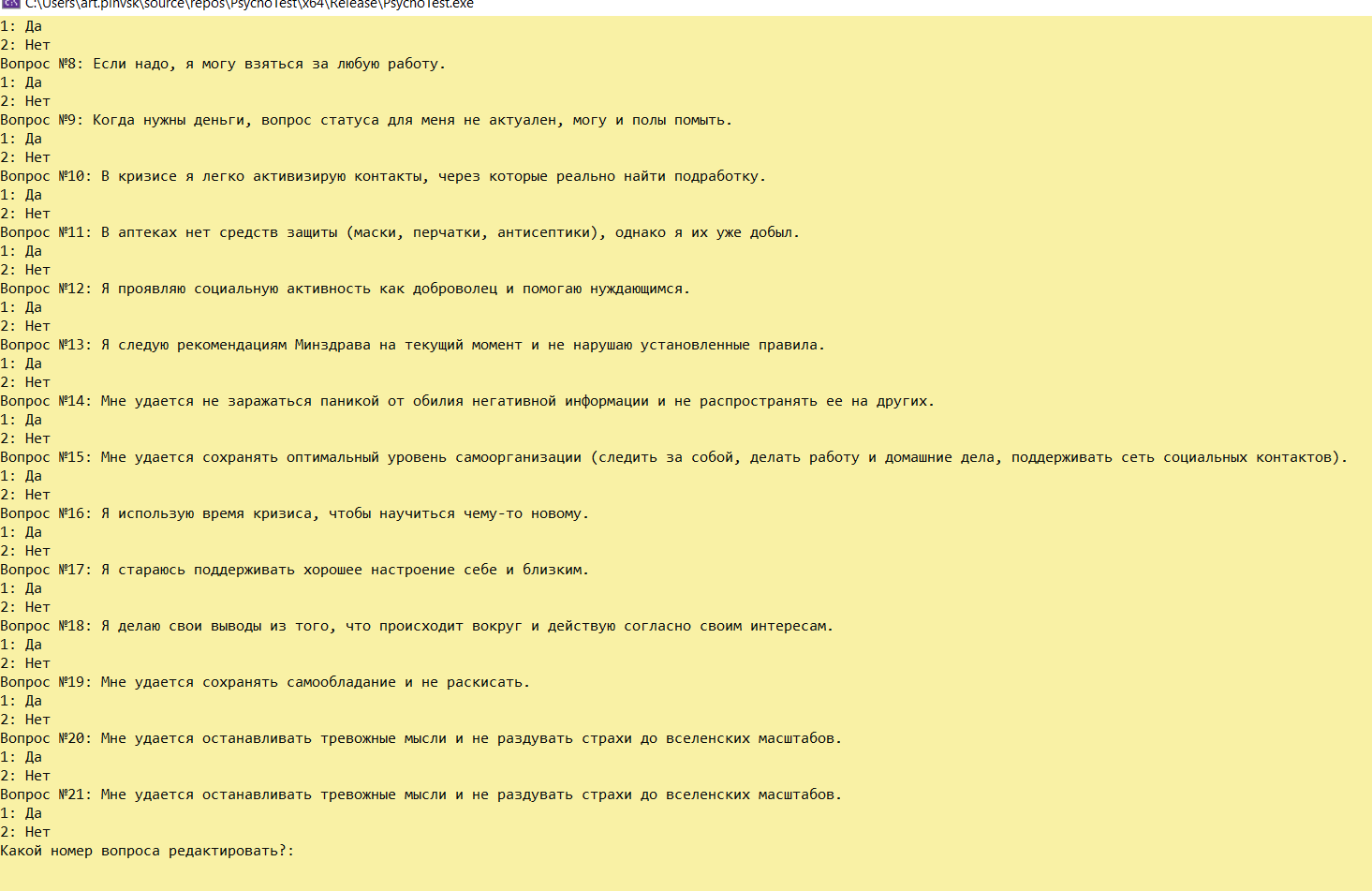


Рисунок 6.11

После выбора меню редактирование вопроса представлено на рисунке 6.11. После выбора номера вопроса, пользователь вводит новый вопрос и ответы к нему.

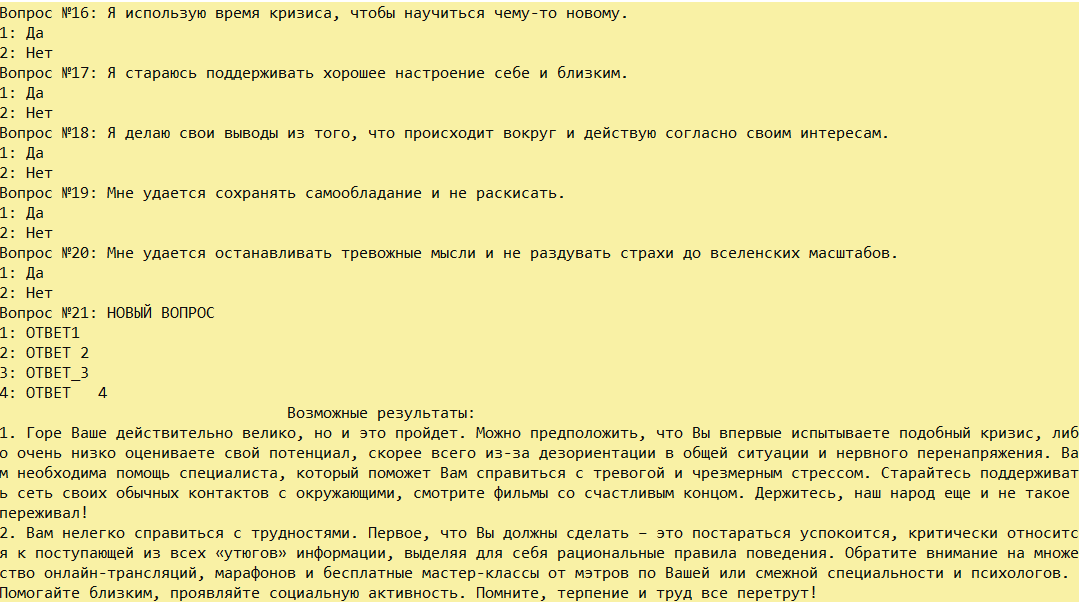


Рисунок 6.12

На рисунке 6.12 представлен результат исполнения программы при выборе администратором “[4] – Просмотр теста” раздела меню после редактирования вопроса.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы были закреплены теоретические и практические навыки по дисциплине «Конструирование программ и языки программирования». Усвоены современные концепции и приемы разработки объектно-ориентированного программного обеспечения при помощи языка С++. Результатом является программа, которая позволяет проходить психологические тесты с получением результата тестирования, а также позволяет администратору добавлять новые тесты и редактировать существующие.

При разработке программы использовался модульный принцип программирования. Такая структура программы позволяет в дальнейшем модифицировать отдельные ее части без нарушения работоспособности и потери функциональности приложения.

Программа имеет достаточно ясный и понятный интерфейс, обеспечивающий удобство в работе и доступ к необходимой пользователю информации с наименьшими затратами времени.

В ходе тестирования разработанного программного модуля все функции программа выполняет без ошибок и задержек. Программа работает корректно и без сбоев.

Таким образом, разработанное в ходе выполнения курсовой работы приложение является законченным программным продуктом.

**ЛИТЕРАТУРА**

1 Дейтел, Х. М. Как программировать на С++ /Х. М. Дейтел, П. Д. Дейтел; пер. с англ. – М. : Бином, 2007

2 Шилдт, Г. Программирование на C++/Г. Шилд. – Минск : ООО «Попури», 1998.

3 Страуструп, Б. Язык программирования С++/ Б.Страуструп; специ-

альное издание; пер. с англ. – Спб.: BHV, 2008. – 1098 с.

4. Луцик, Ю. А. Объектно-ориентированное программирование на языке C++: учебное пособие/Ю. А. Луцик, В. Н. Комличенко. - Минск: БГУИР, 2008. - 266 с.

5. Стефенс, Д. Р. С++. Сборник рецептов/Д. Р. Стефанс - КУДИЦ-ПРЕСС, 2007. - 624 с.

# **ВЕДОМОСТЬ ДОКУМЕНТОВ**

Ведомость документов представлена в Приложении Г.

**ПРИЛОЖЕНИЕ А**

(обязательное)

Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

(обязательное)

Схемы алгоритмов

**ПРИЛОЖЕНИЕ В**

(обязательное)

Листинг программы

**ПРИЛОЖЕНИЕ Г**

(обязательное)

Ведомость документов