**СОДЕРЖАНИЕ**

**ВВЕДЕНИE……………………………………………………………………….2**

**1 ПОСТАНОВКА ЗАДАЧИ…………………………………………………….3**

**2 ОБЗОР ЛИТЕРАТУРЫ……………………………………………………….4**

**2.1 Обзор методов и алгоритмов решения поставленной задачи………….4**

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ…………………………….5**

**3.1 Структура входных и выходных данных………………………………...5**

**3.2 Разработка диаграммы классов………...…………………………………6**

**3.3 Описание классов……………………………………………………………6**

**3.3.1 Классы дневника…………………………………………………………..6**

**3.3.2 Классы исключений……………………………………………………….9**

**3.3.3 Контейнерные классы…………………………………………………...12**

**3.3.4 Другие классы…………………………………………………………….16**

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ…………………………..20**

**4.1 Разработка схем алгоритмов……………………………………………...20**

**4.2 Разработка алгоритмов……………………………………………………22**

**РЕЗУЛЬТАТЫ РАБОТЫ……………………………………………………...23**

**ЗАКЛЮЧЕНИЕ………………………………………………………………...25**

**СПИСОК ЛИТЕРАТУРЫ…………………………………………………….26**

**ПРИЛОЖЕНИЕ А………………………………………………………………...**

**ПРИЛОЖЕНИЕ Б………………………………………………………………...**

**ПРИЛОЖЕНИЕ В………………………………………………………………...**

**ПРИЛОЖЕНИЕ Г………………………………………………………………...**

**ВВЕДЕНИЕ**

Язык C++ является довольно мощным инструментом для умелого разработчика. Сам язык имеет 3 парадигмы, засчёт этого можно по-разному выстраивать структуру программы. Он сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

Если сравнивать данный язык с C, то можно выделить разные парадигмы: C++ в первую очередь был создан для обретения других парадигм, речь идёт про ООП и обобщённое программирование.

Данный язык широко используется для разработки ПО, написания игр, создание операционных систем, написания драйверов и решения множества других задач.

Язык имеет богатую стандартную библиотеку, включающую в себя множество алгоритмов, контейнеров, ввод-вывод, строковые, языковую поддержку, регулярные выражения, поддержку многопоточности и многое другое.

Один из фреймворков под названием Qt позволяет реализовать красивый и удобный графический интерфейс для разработки ПО.

Таким образом для разработки простого приложения “*Ежедневник*” язык C++ подходит идеально.

**1 ПОСТАНОВКА ЗАДАЧИ**

Программа должна иметь удобный пользовательский интерфейс с

необходимыми пунктами меню с использованием фреймворка Qt. В программе должна быть предусмотрена возможность создания информации в виде файлов, связанных определенным образом.

Реализовать следующие возможности:

- календарь;

- заметки;

- другие методы в зависимости от специфики задачи.

Должна быть возможность добавления, удаления, просмотра информации, поиск по разным параметрам. Другие методы, в зависимости от специфики задачи. Необходимо предусмотреть операцию отмены последних действий. В программе необходимо разработать и использовать классы контейнеров, итераторов и алгоритмов, производить обработку исключений.

**2 ОБЗОР ЛИТЕРАТУРЫ**

**2.1 Обзор методов и алгоритмов решения поставленной задачи**

Для хранения и обработки данных в дневнике используется самостоятельно разработанный контейнер - бинарное дерево. Этот контейнер имеет методы добавления, удаления, поиска, вывода и метод “перестройки” бинарного дерева по другому параметру.

Используются алгоритмы поиска, сортировки по определенному полю, записи и считывания из файла. Для поиска и сортировки используются шаблонные методы классов Compare(T& obj, int Mode), где obj – объект класса и Mode – поле для сравнения объектов класса notes.

Для отмены предыдущих действий пользователя используется шаблонный контейнер QStack<T>, где T **–** класс.

Также используются контейнеры QMap<T>, QStringList для добавления в них данных.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

**3.1 Структура входных и выходных данных**

Таблица 3.1 - Файл phones.txt(информация о телефонах):

|  |  |
| --- | --- |
| ФИО | Телефон |
| Львов Иван Васильевич | 375445112215 |

Таблица 3.2 - Файл task.txt(информация о задачах):

|  |  |
| --- | --- |
| Задача | Выполненные задачи |
| Выполнить домашнее задание | Решить пример |

Таблица 3.3 - Файл finances.txt(информация о финансах и заметках):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Заголовок заметки | Заметка | Прибыль | Траты | Дата  заметки | Даты траты и прибыли |
| Планы | Прогуляться  Поужинать | 20 | 79.5 | 21.03.2022 | 29.01.2022 |

Таблица 3.4 - Файл notes.txt(информация о заметках):

|  |  |  |
| --- | --- | --- |
| Заголовок заметки | Заметка | Дата заметки |
| Планы | Прогуляться  Поужинать | 21.03.2022 |

Таблица 3.5 - Файл wishes.txt(информация о желаниях):

|  |  |  |
| --- | --- | --- |
| Желание | Путь достижения | Дата желания |
| Устроиться на работу | Подготовиться к собеседованию | 10.01.2022 |

**3.2 Разработка диаграммы классов**

Диаграмма классов для курсового проекта приведена в **Приложении А**.

**3.3 Описание классов**

**3.3.1 Класс дневника**

Класс paragraph – класс текста и описания

Поля:

String description – описание использования

Методы и операторы:

paragraph () - конструктор по умолчанию

~ paragraph () - деструктор

setDescription() – изменение описания

getDescription() – получить описание

Класс phoneBook – класс телефонной книги

Поля:

QMap<QString, QString> phone – имя и телефон человека

Методы и операторы:

phoneBook() - конструктор по умолчанию

~ phoneBook () - деструктор

void operator=( phoneBook & object) - оператор присваивания

friend ifstream& operator>>(ifstream& in, phoneBook & object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, phoneBook & object) - оператор записи в файл

Класс note – класс заметки

Поля:

QMap<QString, QStringList> page – дата заметки и заметка

Методы и операторы:

note() - конструктор по умолчанию

~ note() - деструктор

QStringList getPage(QString date); - возвращение заметки

void appendPage(QString date,QString note); - добавление заметки

void operator=( note & object) - оператор присваивания

friend ifstream& operator>>(ifstream& in, note & object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, note & object) - оператор записи в файл

Класс task – класс задачи

Поля:

QList<QString> tasks; - задачи

QList<QString> endedTasks; - выполненные задачи

Методы и операторы:

Nonmetal() - конструктор по умолчанию

~Nonmetal() – деструктор

QList<QString> getTasks() - возвращение поля task

void appendTask(QString task) - добавление в task

QList<QString> getEndedTasks() - получение выполненных задач

void appendEndedTasks(QString task) - добавление выполненных задач

bool containTask(QString task) - проверка на наличие в контейнере задачи

bool containEndedTask(QString task) - проверка на наличие в контейнере выполненной задачи

void operator=( task & object) - оператор присваивания

friend ifstream& operator>>(ifstream& in, task& object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, task& object) - оператор записи в файл

Класс wishNote – класс желаний

Поля:

string wish – заголовок

string way – путь достижения

Методы и операторы:

wishNote() - конструктор по умолчанию

~ wishNote () - деструктор

void operator=( wishes& object) - оператор присваивания

string getWish() – получение желания

void setWish(string wish) – изменение желания

string getWay() – получение пути достижения желания

void setWay(string way) – изменить путь достижения желания

int GetSearchMode() - получение поля для сравнения

void SetSearchParametres(wishes& obj1, wishes& obj2, int Mode) - инициализация поля для сравнения

bool Compare(wishes& obj, int Mode) - сравнение по полю

bool IsBetween(wishes& obj1, wishes& obj2, int Mode) - сравнение по полю

friend ifstream& operator>>(ifstream& in, wishes& object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, wishes& object)- оператор записи в файл

Класс financeNote – класс заметок о финансах

Поля:

QMap<QDate, float> money - дата и деньги

Методы и операторы:

financesNote() - конструктор по умолчанию

~ financesNote () - деструктор

void operator=(financesNote& object) - оператор присваивания

friend ifstream& operator>>(ifstream& in, financesNote & object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, financesNote & object)- оператор записи в файл

**3.3.2 Классы исключений**

Класс Exception – класс ошибки

Поля:

int Code - код ошибки

char Message[MESSAGE\_LEN] – сообщение

Методы и операторы:

Exception(int Code, const char\* Messge = "Exception") - конструктор с параметрами

friend ostream& operator<<(ostream& out , Exception& exc) - оператор вывода

Класс AlhoritmException – класс ошибки алгоритма

Поля:

char alhoritm[MESSAGE\_LEN] - название алгоритма

Методы и операторы:

AlhoritmException(const char\* alhoritm, int Code, const char\* Messge ="Exception") - конструктор с параметрами

friend ostream& operator<<(ostream& out, AlhoritmException& exc) - оператор вывода

Класс ArrayBoundsException – класс ошибки выхода за границы массива

Поля:

int index - индекс элемента за границей массива

Методы и операторы:

ArrayBoundsException(int index , const char\* alhoritm, int Code = 1, const char\* Messge = "Exception") - контруктор с параметрами

friend ostream& operator<<(ostream& out,ArrayBoundsException& exc) - оператор вывода

Класс NodeNotFoundException – класс ошибки ненайденного узла дерева

Поля:

char SearchedValue[MESSAGE\_LEN] - искомое значение

Методы и операторы:

NodeNotFoundException(const char\* SearchedValue, const char\* alhoritm ,int Code = 6, const char\* Message = "NodeNotFoundExcepton") - конструктор с параметрами

friend ostream& operator<<(ostream& out, NodeNotFoundException& exc) - оператор вывода

Класс NullValueException – класс ошибки NULL указателя

Поля:

char NullValue[MESSAGE\_LEN] - имя NULL указателя

Методы и операторы:

NullValueException(const char\* NullValue, const char\* alhoritm,int Code = 7, const char\* Message = "NullValueExcepton") - конструктор с параметрами

friend ostream& operator<<(ostream& out, NullValueException& exc) - оператор вывода

Класс InputException – класс ошибки ввода

Методы и операторы:

InputException(int Code, const char\* Message) - констрктор с параметрами

friend ostream& operator<<(ostream& out, InputException& exc) - оператор вывода

Класс FormatException – класс ошибки ввода в некорректном формате

Методы и операторы:

FormatException(int Code = 4, const char\* Message = "Некорректный формат ввода") - конструктор с параметрами

friend ostream& operator<<(ostream& out, FormatException& exc) - оператор вывода

Класс LanguageException – класс ошибки языка ввода

Поля:

int index - индекс некоректного символа

char str[MESSAGE\_LEN] - строка с ошибкой

Методы и операторы:

LanguageException(char\* str, int position, int Code = 5,const char\* Message = "Language exception") - конструктор с параметрами

friend ostream& operator<<(ostream& out, LanguageException& exc) - оператор вывода

Класс FileException – класс ошибки работы с файлом

Поля:

char FileName[MESSAGE\_LEN] - имя файла

Методы и операторы:

FileException(int Code, const char\* Message = "FileException", const charFileName = "") - конструктор с параметрами

friend ostream& operator<<(ostream& out , FileException& exc) - оператор вывода

Класс FileIncorrectModeException – класс ошибки открытия файла с некорректным режимом

Поля:

int FileMode - режим файла

Методы и операторы:

FileIncorrectModeException(int FileMode, const char\* FileName ,int Code = 2,const char\* Message = "FileIncorrectModeException") - конструктор с параметрами

friend ostream& operator<<(ostream& out, FileIncorrectModeException& exc) - оператор вывода

Класс FileNotOpenException – класс ошибки неоткрытого файла

Операторы:

FileNotOpenException(const char\* FileName ,int Code = 3,const char\* Message = "FileNotOpenException") - конструктор с параметрами

friend ostream& operator<<(ostream& out, FileNotOpenException& exc) - оператор вывода

Класс ExceptionInterface(функции для исключений) – класс интерфейса функций работы с ошибками ввода

Методы:

void input\_str(char\* str , char Mode) - проверка на язык ввода

void str\_check\_language\_english(char\* str) - проверка языка ввода

void str\_check\_language\_russian(char\* str) - проверка языка вводa

template<typename Value> void input\_checking\_format(Value& value, istream& in) - шаблонный метод проверки формата ввода

**3.3.3 Контейнерные классы**

Класс Tree<T> - шаблонный класс бинарного дерева

Поля:

Node<T>\* root - корень дерева

int vertexCount - количество узлов

int InsertionMode - параметр построения дерева

Методы:

Node<T>\* getSuccessor(Node<T>\* deleted) - поиск наследника для удаления

Tree() - конструктор

~Tree() - деструктор

Node<T>\* GetRoot() - получение вершины

int GetCount() - получение количества узлов

int GetInsertionMode() //получение поля построения дерева

void Insert(Node<T>\* newNode) - вставка узла в дерево

Mas<T>\* Search() - поиск узлов в дереве

void Pop(Node<T>\* deleted) - удаление узла из дерева

void Display() - вывод дерева

void Display(Node<T>\* current) - симметричный обход дерева

void WriteToFile(ofstream& stream) - запись дерева в файла

void WriteToFile(Node<T>\* current, ofstream& stream) - запись дерева в файл прямым обходом

void Reconfiguration(int Mode = -1) - перестройка дерева по новому полю

void ToMas(Node<T>\* current, Mas<T>\* mas)- перевод дерева в массив

void AllNull(Node<T>\* current) - разрушение дерева

void DeleteTree(Node<T>\* current) - удаление дерева

Класс TreeIterator<T> - шаблонный класс итератора бинарного дерева

Поля:

Tree<T>\* tree - дерево

Node<T>\* Current - текущий узел

int Index - индекс текущего узла

Методы:

TreeIterator(Tree<T>\* tree) - конструктор с параметром

~TreeIterator() - деструктор

T Value() – получение объекта узла

void operator=(Node<T>\* node) - оператор присваивания

bool operator==(Node<T>\* node) - оператор сравнения

bool operator!=(Node<T>\* node) - оператор сравнения

Node<T>\* operator\*() - оператор разадрессации

void operator++(int) - инкремент

void operator--(int) - декремент

Node<T>\* Begin() - установка итератора на начало дерева

Node<T>\* End() - установка итератора на конец дерева

Node<T>\* Last() - конец дерева

Node<T>\* Step(int& NewIndex) - переход к узлу с индексом

void Step(Node<T>\* current, Node<T>\*\* successor, int& NewIndex) - поиск узла с индексом

Класс Mas<T> - шаблонный класс массива узлов бинарного дерева

Поля:

Node<T>\*\* mas - массив узлов

int size - размер массива

int putloc - позиция для вставки

Методы:

Mas(int size) - конструктор с параметром

~Mas() - деструктор

Mas<T>\* Menu() - выбор действий с элементами массива

void Insert(Node<T>\* newNode) - вставка узла в массив

void Pop(int position) - удаление узла по позиции

Node<T>\* Pop() - удаление последнего узла

void Show() - вывод массива

void Filter() - фильтрация массива по полям узлов

bool IsEmpty() - проверка массива на пустоту

void ToTree(Tree<T>\* tree) - перевод массива в дерево

void ToTree(Tree<T>\* tree, int left, int right) - перевод массива в дерево

void Sort(int Mode = 0) - сортировка массива по полю узла

void QuickSort(int left, int right, int Mode) - быстрая сортировка

void Swap(int i, int j) - обмен узлов

Класс Node<T> - шаблонный класс узла бинарного дерева

Поля:

T value - объект

Node<T>\* left - левый сын

Node<T>\* right - правый сын

Методы и операторы:

Node() - конструктор по умолчанию

Node(T value) - конструктор с параметром

~Node() - деструктор

bool Compare(Node<T>\* obj, int Mode) - сравнение по полю

bool IsBetween(Node<T>\* obj1, Node<T>\* obj2, int Mode) - сравнение по полю

friend istream& operator>>(istream& in, Node<T>& object) - оператор ввода

friend ostream& operator<<(ostream& out, Node<T>& object) - оператор вывода

friend ifstream& operator>>(ifstream& in, Node<T>& object) - оператор считывания из файла

friend ofstream& operator<<(ofstream& out, Node<T>& object) - оператор записи в файл

Структура StackElement – шаблонная структура элемента стека

Поля:

Node<T>\* node - сораненный узел

char Operation - операция

Tree<T>\* tree - сохраненное дерево

int InsertionMode - параметр построения дерева

StackElement<T>\* Next - следующий элемент

Методы:

StackElement(Node<T>\* n, char o) - конструктор с параметрами

StackElement(Tree<T>\* t, int Mode) - конструктор с параметрами

~StackElement() - деструктор

Класс Stack<T> – шаблонный класс стека для отмены действий

Поля:

StackElement<T>\* Top - вершина стека

Методы:

Stack() - конструктор

~Stack() - деструктор

void Push(StackElement<T>\* NewElement) - занесение в стек

StackElement<T>\* Pop() - удаление из стека

void DeleteStack(StackElement<T>\* current) - удаление стека

**3.3.4 Другие классы**

Класс Alhoritm<T> – шаблонный класс алгоритмов

Методы:

static void ReadFromFile(Tree<T>\* tree, const char\* filename) - чтение из файла в дерево

static void WriteToFile(Tree<T>\* tree, const char\* filename) - запись дерева в фай

static void Insert(Tree<T>\* tree, Node<T>\* NewNode , Stack<T>& stack) - вставка нового узла в дерево

static void Search(Tree<T>\* tree , Stack<T>& stack) - поиск узлов в дереве

static void Display(Tree<T>\* tree) - вывод дерева

static void Sort(Tree<T>\* tree, Stack<T>&stack) - сортировка дерева по другому параметру

static void Cancellation(Tree<T>\* tree, Stack<T>& stack) - отмена действия

Класс FileNotOpenException - класс файлового потока

Поля:

char CurrentFile[MAX\_LEN\_OF\_STRING] - название файла

ifstream infile - поток считывания из файла

ofstream outfile - поток записив файл

Методы:

FileStream() - конструктор по умолчанию

~FileStream() - деструктор

bool EndOfStream() - достигнут ли конец файла

void Open(const char\* file, int Mode) - открытие файла

int IsOpen() - открыт ли файл

void Close() закрытие файла

template<typename T> void Write(Node<T>& node) - запись узла в файл

template<typename T>void Read(Node<T>& node) - считывание в узел

Класс QMainWindow - класс главного окна

Поля:

Ui::MainWindow \*ui – главное окно

nextWindowSecond \*secondWindow – второе окно

nextWindowThird \*thirdWindow – третье окно

nextWindowFourth \*fourthWindow – четвёртое окно

nextWindowFifth \*fifthWindow – пятое окно

nextWindowSixth \*sixthWindow – шестое окно

Методы:

void fifthWindow() – сигнал перехода на главное окно

void on\_pushButton\_clicked() - переход на окно с заметками

void on\_pushButton\_2\_clicked() - переход на окно с финансами

void on\_pushButton\_3\_clicked() - переход на окно с задачами

void on\_pushButton\_6\_clicked() - переход на окно с телефонной книгой

void on\_pushButton\_8\_clicked() - переход на окно с желаниями

Класс nextWindowSecond - класс второго окна

Методы:

void secondWindow() – сигнал второго окна

void on\_pushButton\_clicked() – переход в главное меню

void on\_pushButton\_2\_clicked() – добавление

void on\_pushButton\_3\_clicked() – отмена предыдущего действия

void on\_pushButton\_4\_clicked() – записать данные в файл

void on\_pushButton\_5\_clicked() – чтение данных из файла

void on\_pushButton\_6\_clicked() – установка названия

void on\_pushButton\_7\_clicked() – найти по ФИО

void on\_pushButton\_8\_clicked() – найти по номеру телефона

void on\_pushButton\_9\_clicked() – отсортировать по заданному параметру

Ui::nextWindowSecond \*ui – второе окно

Класс nextWindowThird - класс пятого окна

Методы:

void thirdWindow() – сигнал третьего окна

void on\_pushButton\_clicked() – переход в главное меню

void on\_pushButton\_2\_clicked() – добавление

void on\_pushButton\_3\_clicked() – отмена предыдущего действия

void on\_pushButton\_4\_clicked() – записать данные в файл

void on\_pushButton\_5\_clicked() – чтение данных из файла

void on\_pushButton\_6\_clicked() – установка названия

void on\_pushButton\_7\_clicked() – найти по прибыли

void on\_pushButton\_8\_clicked() – найти по тратам

void on\_pushButton\_9\_clicked() – найти заголовку заметки

void on\_pushButton\_10\_clicked() – найти описанию заметки

void on\_pushButton\_11\_clicked() – найти дату заметки

void on\_pushButton\_12\_clicked() – найти дату прибыли и / или траты

void on\_pushButton\_\_clicked() – отсортировать по заданному параметру

void money\_change() – изменение графика

void on\_calendar\_selectionChanged() – использование даты календаря

Ui::nextWindowThird \*ui – третье окно

Класс nextWindowFourth - класс пятого окна

Методы:

void fourthWindow() – сигнал перехода на главное окно

void on\_pushButton\_clicked() – переход в главное меню

void on\_pushButton\_2\_clicked() – добавление

void on\_pushButton\_3\_clicked() – отмена предыдущего действия

void on\_pushButton\_4\_clicked() – записать данные в файл

void on\_pushButton\_5\_clicked() – чтение данных из файла

void on\_pushButton\_6\_clicked() – установка названия

void on\_pushButton\_7\_clicked() – найти по заметке

void on\_pushButton\_8\_clicked() – найти по описанию заметки

void on\_pushButton\_9\_clicked() – отсортировать по заданному параметру

Ui::nextWindowFourth \*ui - четвёртое окно

QGraphicsLineItem \* lineItem\_ – для рисования

QGraphicsScene \* scene\_ – сцена

void initUI() – инициализация сцены

Класс nextWindowFifth - класс пятого окна

Методы:

void fifthWindow() – сигнал перехода на главное окно

void on\_pushButton\_clicked() – переход в главное меню

void on\_pushButton\_2\_clicked() – добавить задачу

void on\_pushButton\_3\_clicked() – добавить выполненную задачу

void on\_pushButton\_4\_clicked() – отмена предыдущего действия

void on\_pushButton\_5\_clicked() – записать данные в файл

void on\_pushButton\_6\_clicked() – чтение данных из файла

void on\_pushButton\_7\_clicked() – установка названия

void on\_pushButton\_8\_clicked() – найти по задаче

void on\_pushButton\_9\_clicked() – найти по выполненной задаче

void on\_pushButton\_10\_clicked() – отсортировать по заданному параметру

Ui::nextWindowFifth \*ui – пятое окно

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

**4.1 Разработка схем алгоритмов**

**Алгоритм по шагам функции void on\_pushButton\_2\_clicked()**

void nextWindowFifth::on\_pushButton\_2\_clicked() – объявление функции

Функция предназначена для добавления задачи в контейнер **QList<QString>**.

Алгоритм по шагам:

1.Начало.

2. Входные данные:

phones – контейнер с телефонами и именами

takenActions3 – стек для отмены действий

ui->TableWidget – таблица с телефонами

3. Проверка на содержание уже существующего номера телефона и ФИО:

phones->phoneContain(ui->lineEdit->text()) && phones->phoneContain(ui->lineEdit\_2->text(), если уже существует такой номер телефона и ФИО – переход к 3 шагу, иначе к 4

4. Вывод ошибки ввода QMessageBox:

QMessageBox\* msg = new QMessageBox();

msg->setWindowTitle("Неправильный ввод");

msg->setText("Вы записали уже существующий номер телефона и ФИО");

msg->exec(), переход к шагу 12.

5. Очистка таблицы, для занесения новых данных:

QTableWidget: ui->tableWidget->clear()

6. Добавление в QMap<QString, QString> номера телефона и ФИО при помощи метода addPhone:

phones->addPhone(ui->lineEdit\_2->text(), ui->lineEdit->text());

7. Создание и инициализация переменной для записи строк в таблице: int i = 0

8.Цикл foreach от QString temp до phones->getPhoneKeys()

9.Создание двух объектов виджетов для инициализации строки:

QTableWidgetItem \*itm1= new QTableWidgetItem();

QTableWidgetItem \*itm2 = new QTableWidgetItem();

10. Заполнение строки в TableWidget:

itm1->setText(temp);

ui->tableWidget->setItem(i,0,itm1);

itm2->setText(phones->getPhone(temp));

ui->tableWidget->setItem(i,1,itm2);

i++;

11.Запись в QStack для отмены действий:

takenActions3.push(phones);

12. Конец.

**Алгоритм по шагам шаблонного метода void Tree<T>:: void Insert(Node<T>\* newNode)**

template<typename T> void Tree<T>::Insert(Node<T>\* newNode) – объявление метода

Метод предназначен для добавления узлов в бинарное дерево .

Параметры метода: **Tree<T>\* tree –** указатель на дерево , к которому были применены действия пользователя, **Stack<T>& stack –** стек , в котором сохранены предыдущие действия пользователя.

Алгоритм по шагам:

1.Начало.

2. Входные данные:

Node<T>\* current – текущий указатель на элемент дерева

root – указатель на корень дерева

Node<T>\* newNode – новый элемент дерева

3.Добавить количество узлов: vertexCount++

4.Если указатель на корень равен NULL – создаём корень: root = newNode, иначе переходим к шагу 5.

5.Объявляем локальную переменную ***Node<T>\* current = root*** – указатель на текущий элемент: корень root

6.Цикл while, пока current !=NULL

7.Вызываем current->Compare(newNode , InsertionMode), если он возвращает false, переходим к шагу 8, иначе переходим к шагу 10.

8.Если current->left == NULL – создаём новый узел current->left = newNode и к шагу 12, иначе переходим к шагу 9.

9.Переходим на следующий узел: current = current->left и на шаг 7.

10.Если current->right == NULL, то создаём новый узел current->right = newNode и переходим к шагу 12, иначе к шагу 11.

11.Переходим на следующий узел current = current->right, переходим к шагу 10.

12.Конец

**4.2 Разработка алгоритмов**

Шаблонный метод **Mas<T>\* Tree<T>::Reconfiguration(int Mode)** – метод “балансировки” дерева.

Блок-схема приведена в **Приложении Б**.

Шаблонный метод **void Mas<T>::Sort()** – метод сортировки массива по разным полям.

Блок-схема приведена в **Приложении В**.

**РЕЗУЛЬТАТЫ РАБОТЫ**

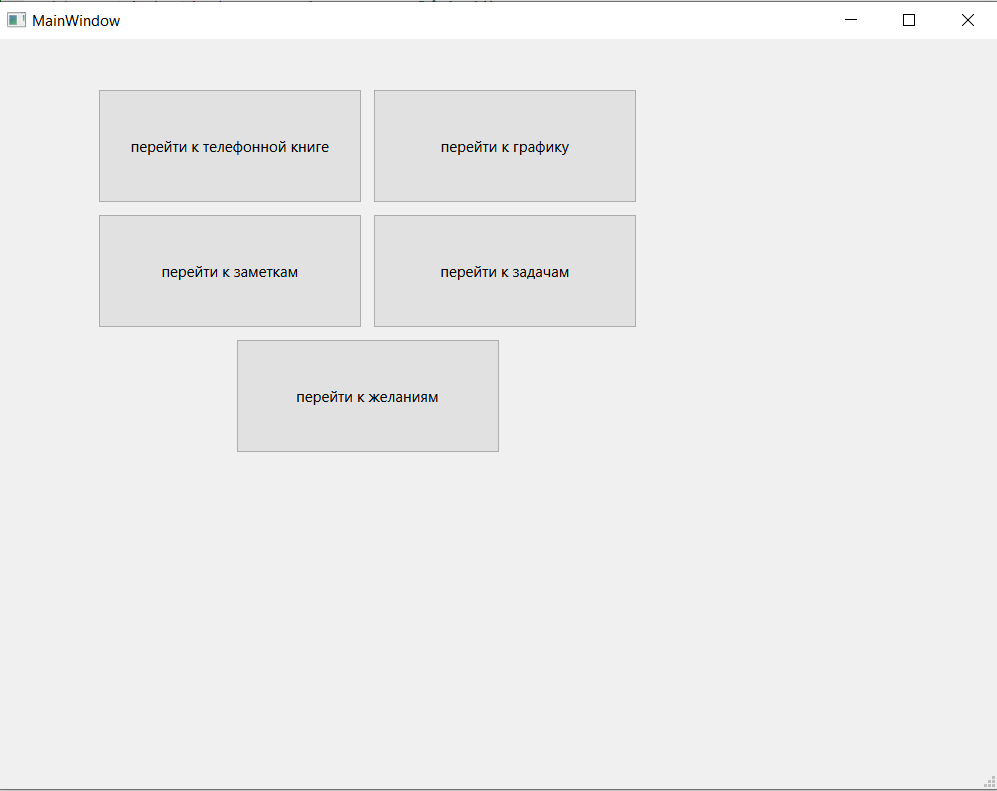
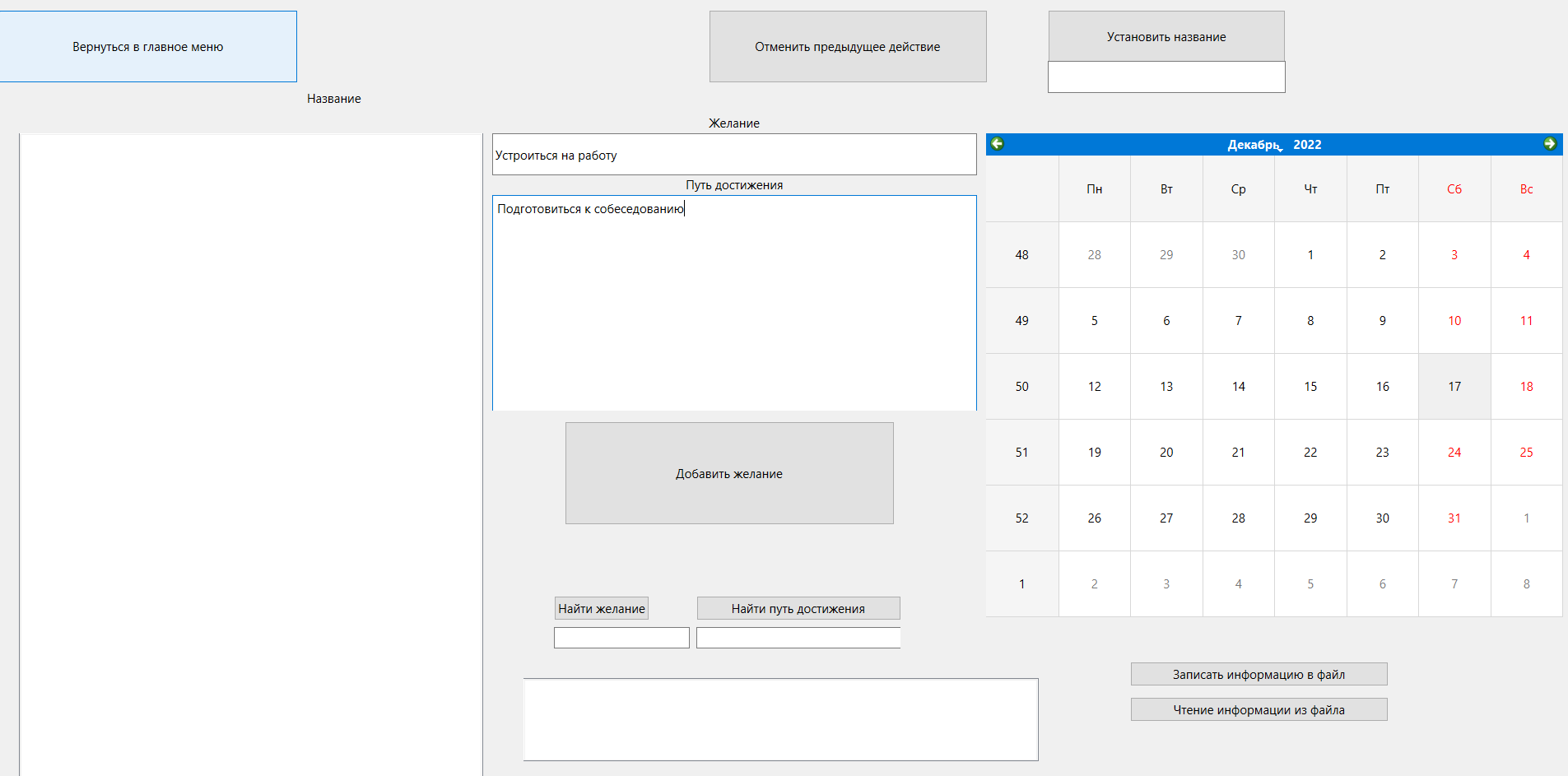
****

Рисунок 1 – меню

Рисунок 2 – добавление элементов в бинарное дерево

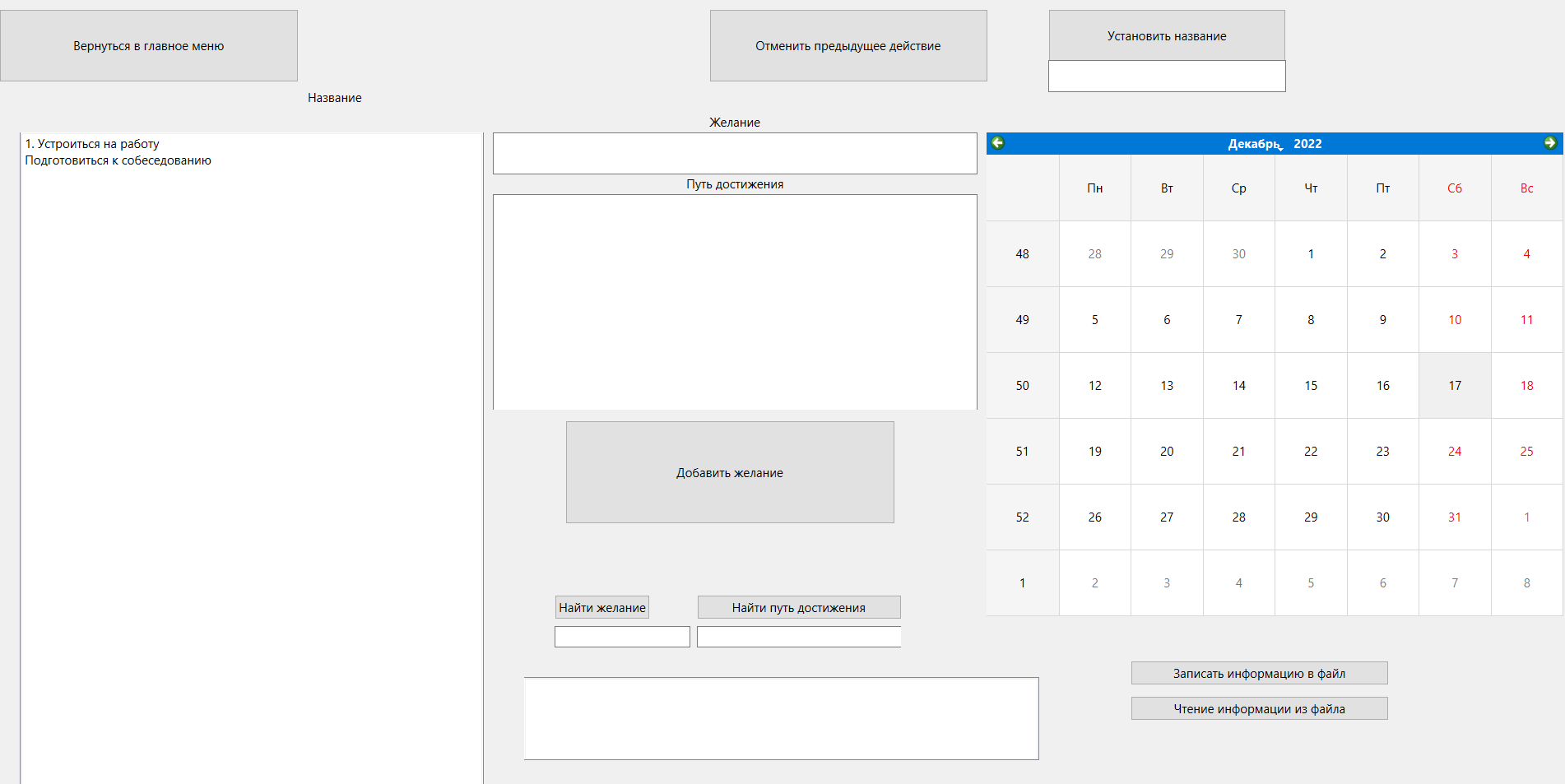
****

Рисунок 3 - вывод элементов бинарного дерева

**ЗАКЛЮЧЕНИЕ**

Ограничения на программу:

* Данная программа запущена в среде разработки Qt Creator 8.0.1
* Суммарный размер программы со всеми файлами – 176 KB

**СПИСОК ЛИТЕРАТУРЫ**

1. Шилдт Г. С++ базовый курс / Г. Шилдт - 3-е изд. М. : Вильямс , 2007 – 624с.

2. Страуструп Б. Язык программирования С++ : специальное издание / Б. Страуструп : пер. с англ. – Москва : Бином , 2006 – 1104с.

3. Конструирование программ и языки программирования: метод. Указания по К65 курсовому проектированию для студ. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич , А. М. Ковальчук , И. В. Лукьянова. - Минск: БГУИР , 2009.

4. Объектно-ориентированное программирование на языке С++:учеб. Пособие /Ю. А. Луцик , В. Н. Комличенко. – Минск: БГУИР , 2008.

**ПРИЛОЖЕНИЕ А**

(обязательное)

(Диаграмма классов)

**ПРИЛОЖЕНИЕ Б**

(обязательное)

(блок-схемашаблонного методаReconfiguration())

**ПРИЛОЖЕНИЕ В**

(обязательное)

(блок-схема шаблонного метода Sort())

**ПРИЛОЖЕНИЕ Г**

(обязательное)

(Полный код программы)