**Содержание.**

[**Введение. 4**](#_Toc477129285)

[**1.Теоритическая часть 4**](#_Toc477129286)

[**2. Выбор системы программирования 5**](#_Toc477129287)

[**Глава 3. Описание процесса проектирования приложения 6**](#_Toc477129288)

[**3.1.Взаимосвязь программных модулей 6**](#_Toc477129289)

[**3.1.1.Функция подборки рейса. 6**](#_Toc477129290)

[**3.1.2.Функция добавления рейса. 7**](#_Toc477129291)

[**3.1.3.Функция покупки билета. 8**](#_Toc477129292)

[**3.1.4. Функция очистки полей на вкладке “Настройка расписания” 9**](#_Toc477129293)

[**3.1.5.Функции удаление рейса из lvRaspisanie. 9**](#_Toc477129294)

[**3.1.5.1 Удаление первой строки. 9**](#_Toc477129295)

[**3.1.5.2 Удаление строки под номером. 10**](#_Toc477129296)

[**3.1.5.3 Удаление всех строк. 10**](#_Toc477129297)

[**3.1.6. Функция очистки полей информации о пассажире. 11**](#_Toc477129298)

[**3.2. Схема работы программы. 11**](#_Toc477129299)

[**3.3. Проектирование интерфейса. 12**](#_Toc477129300)

[**Глава 4. Инструкция для пользователя по работе с приложением 15**](#_Toc477129301)

[**Заключение. 16**](#_Toc477129302)

[**Список использованных информационных источников: 17**](#_Toc477129303)

[**Приложение 1 18**](#_Toc477129304)

**Аннотация**

В данном курсовом проекте рассматриваются различные алгоритмы сортировки данных. Подробно рассматривается принцип работы алгоритмов и способ их программной реализации.

In this course project, different algorithms for sorting data are used. The principle of operation of algorithms and their implementation is considered in detail.

**Введение.**

В наше время новые информационные технологии занимают очень важное место не только в специализированных, но и в повседневных сферах жизни. Компьютеры применяются в бизнесе, менеджменте, торговле, учебе и многих других сферах деятельности человека.

Компьютерные технологии очень удобны для выполнения разнообразных операций, но в разных сферах применения эти операции разные. Потому, каждая отдельная отрасль, которая использует специфические технические средства, нуждается в своих собственных программах, которые обеспечивают работу компьютеров.

Разработкой программного обеспечения занимается такая отрасль науки, как программирование. Она приобретает все большее и большее значение в последнее время, ведь с каждым днем компьютер становится все более необходимым, все более повседневным явлением нашей жизни. Ведь вычислительная техника прошлых лет уже почти полностью исчерпала себя и не удовлетворяет тем потребностям, которые появляются перед человечеством.

Таким образом, новые информационные технологии очень актуальны в наше время и нуждаются в большем внимании для последующей разработки и совершенствования. Рядом с этим, большое значение имеет также и программирование, которое является одним из фундаментальных разделов информатики и потому не может оставаться в стороне.

Программирование содержит целый ряд важных внутренних задач. Одной из наиболее важных задач для программирования является задача сортировки. Под сортировкой обычно понимают перестановки элементов любой последовательности в определенном порядке. Эта задача является одной из важнейших потому, что ее целью является облегчение последующей обработки определенных данных и, в первую очередь, задачи поиска. Одним из эффективных алгоритмов поиска является бинарный поиск. Он работает быстрее, чем, например, линейный поиск, но его возможно применять лишь при условии, что последовательность уже упорядочена, то есть отсортирована.

Вообще, известно, что в любой сфере деятельности, которая использует компьютер для записи, обработки и сохранения информации, все данные сохраняются в базах данных, которые также нуждаются в сортировке. Определенная упорядоченность для них очень важна, ведь пользователю намного легче работать с данными, которые имеют определенный порядок.

Задача сортировки в программировании не решена полностью. Ведь, хотя и существует большое количество алгоритмов сортировки, все же целью программирования является не только разработка алгоритмов сортировки элементов, но и разработка именно эффективных алгоритмов сортировки. Мы знаем, что одну и ту же задачу можно решить с помощью разных алгоритмов, и каждый раз изменение алгоритма приводит к новым, более или менее эффективным решениям задачи. Основными требованиями к эффективности алгоритмов сортировки является, прежде всего, эффективность по времени и экономное использование памяти.

# **Теоретическая часть**

* 1. **Поиск, анализ и выбор основных вопросов, подлежащих реализации**

В соответствии с целями данного курсового проекта, возможно начать проектирование и анализ требуемых для проекта программ.  
В начале необходимо отметить основные типы и виды сортировок, их основные характеристики. Это необходимо для выделения основных отличий, рассматриваемых в курсовом проекте сортировок от других существующих методов сортировок.

Далее следует описать, разработать и запрограммировать алгоритмы сортировки, рассматриваемые в курсовом проекте. Протестировать программы (массивы должны сортироваться) и отобразить это в отчете.

* 1. **Выбор программного обеспечения по реализации ИТ**

При написании программ для реализации сортировок массивов был использован язык программирования С++. Это один из широко используемых языков программирования, который можно использовать для написания программ, работающих в операционной среде Windows. Среда MS Visual Studio - это сложный механизм, обеспечивающий высокоэффективную работу программиста.

* 1. **Словарь предметной области**

**Введение**

Словарь предметной области представляет собой набор терминов, использованных при написании программной системы «Анализ и реализация алгоритмов сортировки».

**Словарь**

**Пользователь –** Человек, который использует программу, для достижения результата.

**Алгоритм -** Система последовательных операций (в соответствии с определёнными правилами) для решения какой-н. задачи.

**Алгоритм сортировки -** это алгоритм для упорядочивания элементов в списке.

**Массив -** [тип](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) или [структура данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в виде набора компонентов (элементов массива), расположенных в памяти непосредственно друг за другом. При этом доступ к отдельным элементам массива осуществляется с помощью [индексации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%B0%D1%86%D0%B8%D1%8F_(%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)), то есть через ссылку на массив с указанием номера (индекса) нужного элемента.

**Строка** -  [тип данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), значениями которого является произвольная последовательность (строка) символов [алфавита](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)).

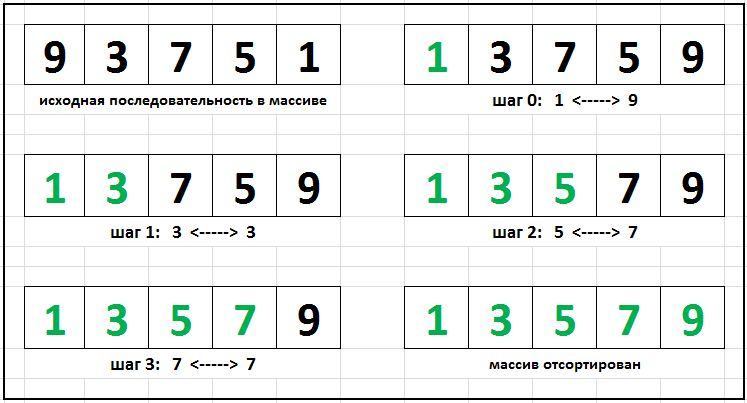
**Метод –** способ практического осуществления алгоритма.

# **1.4.Обзор алгоритмов сортировки**

Сортировка - это процесс перестановки объектов данного множества в определённом порядке. Цель сортировки - облегчить последующий поиск элементов в отсортированном множестве. Поэтому элементы сортировки присутствуют во многих задачах прикладного программирования.

* + 1. **Сортировка чисел**
       1. **Метод сортировки выбором**

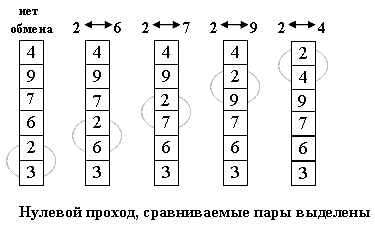
Метод основан на следующем правиле:

Выбирается элемент с наибольшим значением ключа он меняется местами с последним элементом arr [s-1]. Затем эти операции повторяются с оставшимися первыми s-1 элементами, затем - с s-2 первыми элементами и т.д. до тех пор, пока не останется только первый элемент - наименьший. Процесс сортировки методом простого выбора показан на рисунке 1:

*Рисунок 1. Сортировка выбором*

* + - 1. **Метод сортировки пузырьком**

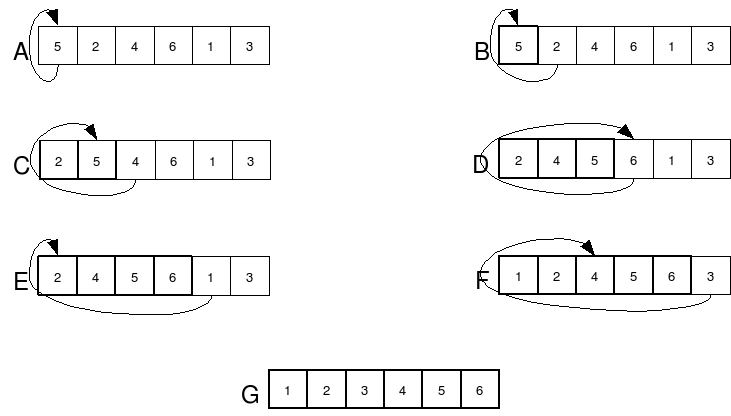
Метод основан на следующем правиле:

При пузырьковой сортировке сравниваются соседние элементы и меняются местами, если следующий элемент меньше предыдущего. Требуется несколько проходов по данным. Во время первого прохода сравниваются первые два элемента в массиве. Если они не в порядке, они меняются местами и затем сравнивается элементы в следующей паре. При том же условии они так же меняются местами. Таким образом сортировка происходит в каждом цикле пока не будет достигнут конец массива. Процесс сортировки методом пузырька показан на рисунке 2:

*Рисунок 2. Сортировка методом пузырька*

* + - 1. **Метод сортировки вставками**

Метод основан на следующем правиле:

На каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем. Стоит отметить что массив из 1-го элемента считается отсортированным. Процесс сортировки методом вставок показан на рисунке 3:

*Рисунок 3. Сортировка методом вставок*

* + - 1. **Метод сортировки слиянием**

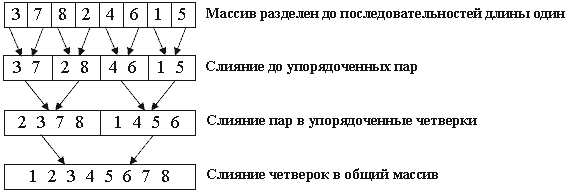
Конкретно процедуру сортировки слиянием можно описать следующим образом:

Если в рассматриваемом массиве один элемент, то он уже отсортирован — алгоритм завершает работу.

Иначе массив разбивается на две части, которые сортируются рекурсивно.

После сортировки двух частей массива к ним применяется процедура слияния, которая по двум отсортированным частям получает исходный отсортированный массив.

Процесс сортировки методом слияния показан на рисунке 4:



*Рисунок 4. Сортировка методом слияния*

* + 1. **Сортировка символов и строк**
       1. **Сортировка символов методом присваивания чисел символу**

Метод основан на следующем правиле:

Строка разбивается на символы, каждому символу присваивается число, согласно его положению в алфавите. Далее символы сортируются, так же, как и числа, например – методом слияния. Присваивание символу числа осуществляется, через специальную библиотеку.

* + - 1. **Сортировка строк в алфавитном порядке**

Метод сортировки основан на методе пузырька. Строки записываются в одну, при этом, между ними указывается разделитель. Программа разбивает строку, от и до разделителя, на символы, далее анализирует порядок символов и сравнивает с другими строками, после этого выстраивает их в алфавитном порядке.

* 1. **Описание пользователей.** **Обзор ключевых потребностей пользователей:**

|  |  |  |
| --- | --- | --- |
| Имя | Описание | Обязанности |
| Пользователь | Человек использующий программу | Выбирает объект сортировки и метод. Вводит входные данные. Получает результат работы ПО. |

* 1. **Обзор ключевых потребностей пользователей:**

|  |  |  |  |
| --- | --- | --- | --- |
| Необходимость | Приоритет | Требования | Предложенное решение |
| Выбор объекта сортировки | высокий | Выбрать объект сортировки из предложенных | Пользователь при входе в систему выбирает объект сортировки пользуясь интерфейсом |
| Выбор метода сортировки | высокий | Выбрать метод сортировки из предложенных | Пользователь выбирает метод сортировки пользуясь интерфейсом |
| Ввод данных | высокий | Ввести входные значения | Пользователь вводит данные, описываемые интерфейсом |
| Анализ выходных данных | высокий | Получение результата | Пользователь получает на выходе, отсортированные входные данные и использует их. |

* 1. **Характеристики системы**

Система обучения и контроля знаний студентов предположительно будет работать только в операционной системе Windows. Также можно отметить, что данный программный продукт не нуждается в связи с другими приложениями.

**Операционная среда**

Приложение может работать на машинах, где установлена ОС Windows 95/98/NT/2000/XP/2003Server.

**Функциональное Требование**

Система может быть запущена несколькими пользователями одновременно, при этом пользователи могут работать с системой по своему желанию, но изменять данные в одно время может только один человек.

**Применимость**

Система может работать на платформе Windows.

**Надежность**

Специальных требований по надежности не предъявлялось.

**Выполнение**

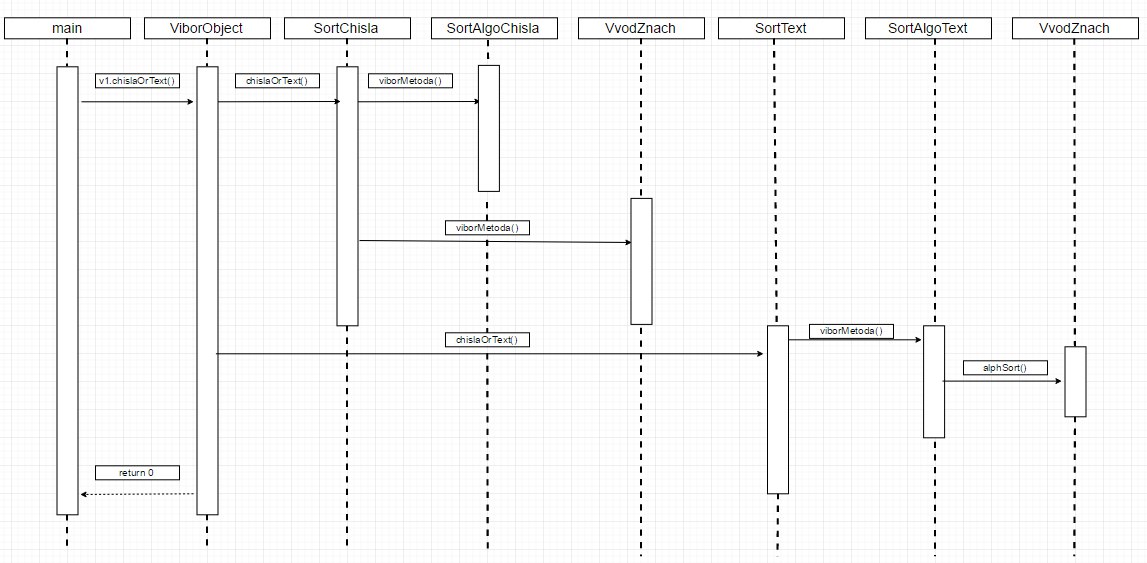
Использование ресурса: на диске занимает 838 КБ

**Ремонтопригодность**

Система не требует настройки.

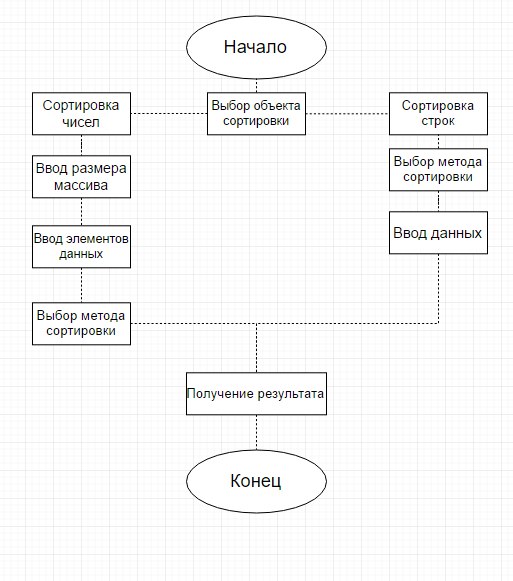
**Лицензирование Требований**

Ограничений использования нет.

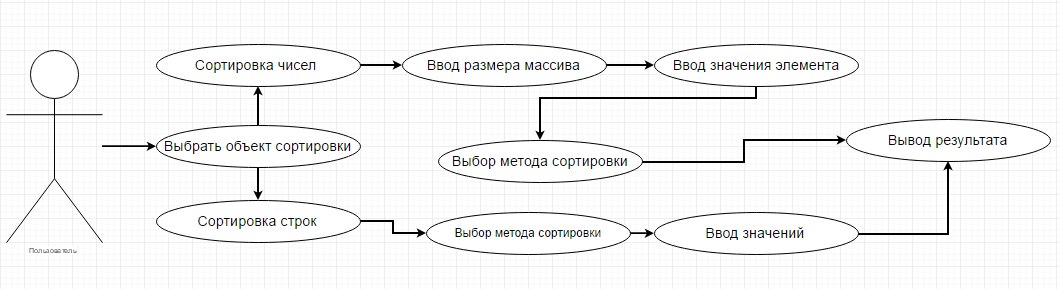
* 1. **Диаграмма последовательности**

*Рисунок 5. Диаграмма последовательности*

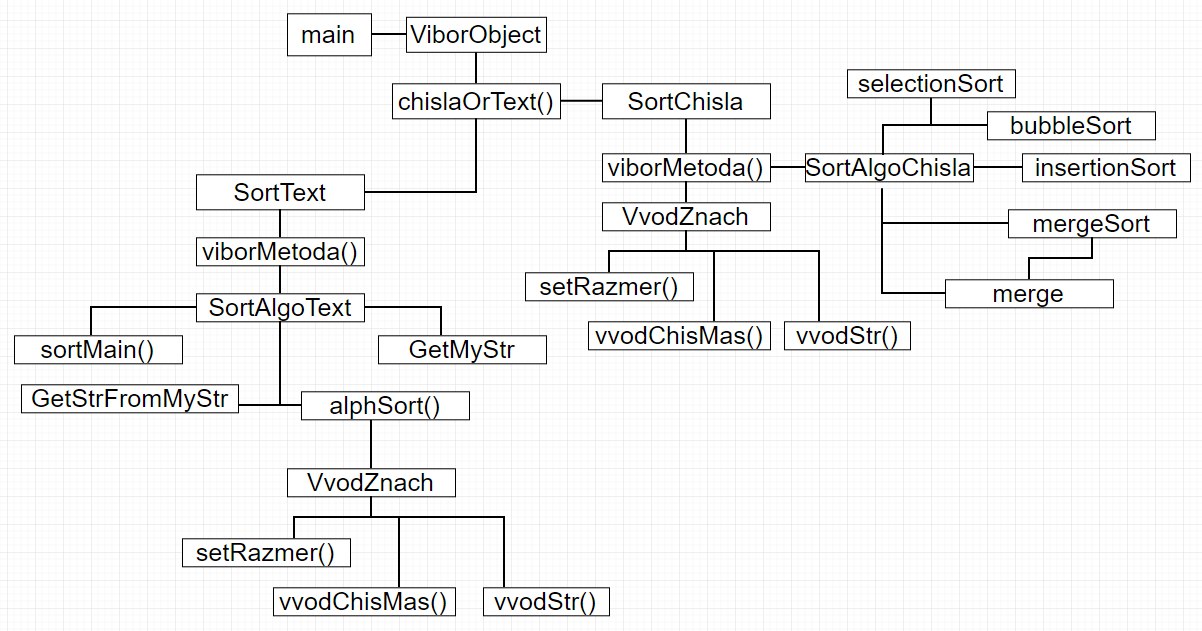
* 1. **Диаграмма деятельности**



*Рисунок 6. Диаграмма деятельности(интерфейса)*

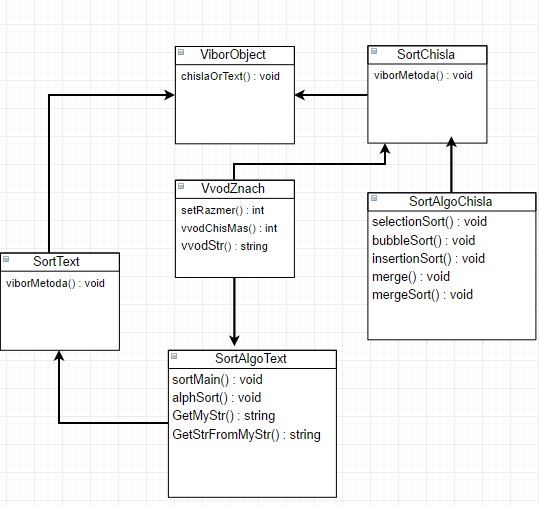
* 1. **Диаграмма Use Case**

*Рисунок 7. Диаграмма Use Case*

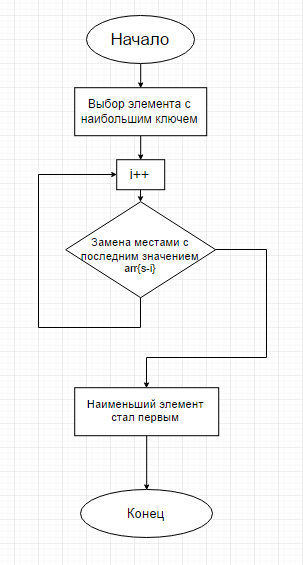
* 1. **Диаграмма объектов**

*Рисунок 8. Диаграмма объектов*

* 1. **Диаграмма классов**



*Рисунок 9. Диаграмма классов*



*Рисунок 10. Диаграмма действия алгоритма сортировки методом*

# **Практическая часть**

**Цель:** Реализовать алгоритмы сортировки в среде разработки MS Visual Studio на языке программирования C++.

* 1. **Взаимосвязь программных модулей**

Итак, первым делом необходимо определиться с функционалом, которым будет обладать модуль. Пользователь выбирает объект сортировки, далее в зависимости от выбора:

1. Если, пользователь выбрал сортировку чисел, он вводит размер массива и заполняет его. После пользователь выбирает метод сортировки.
2. Если, пользователь выбрал сортировку слов и строк, далее выбирается метод сортировки. При выборе сортировки букв в строке, пользователь вводит набор букв и далее они сортируются. При выборе сортировки строк в алфавитном порядке, пользователь вводит строки до тех пор, пока не введет пустую строку, либо не нажмет сочетание клавиш CTRL+Z. Далее строки сортируются и выводятся в консоль в алфавитном порядке.
   1. **Функция Main()**

В теле данной функции создается объект v1 класса ViborObject, после вызывается функция этого класса chislaOrText().Код реализации описанной функции представлен ниже:

int main(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "RUSSIAN");

ViborObject v1;

v1.chislaOrText();

system("PAUSE");

return 0;

}

* 1. **Классы**
     1. **Класс ViborObject**

Данный класс содержит функцию chislaOrText() являющееся реализацией пользовательского интерфейса. Служит для выбора пользователем объекта сортировки. В зависимости от выбранного объекта вызывает функцию класса соответствующего данному объекту. Код реализации данного класса представлен ниже:

class ViborObject {

public:

void chislaOrText() {

SortChisla v1;

SortText v2;

int k;

cout << "Выберите объект сортировки сортировки:\n";

cout << "Если вы хотите отсортировать числа, введите 1, если слова или строки, введите 2:";

cin >> k;

if (k == 1)

v1.viborMetoda();

if (k == 2)

v2.viborMetoda2();

}

};

C:\Users\htcvive\AppData\Local\Microsoft\Windows\INetCache\Content.Word\класс vviborObject.png

*Рисунок 11. Класс ViborObject*

* + 1. **Класс SortChisla**

Данный класс содержит функцию viborMetoda() являющееся реализацией пользовательского интерфейса. Служит для выбора пользователем метода сортировки. В зависимости от выбранного метода вызывает функцию класса соответствующего данному методу. Код реализации данного класса представлен ниже:

class SortChisla {

public:

void viborMetoda() {

SortAlgoChisla v1;

VvodZnach v2;

int v, n = 0;

int \*l = v2.vvodChisMas();

int t = v2.setRazmer();

cout << t+"\n";

cout << "Выберете метод сортировки:\n Введите 1 - Сортировка выбором (Selection sort)\n Введите 2 - Пузырьковая сортировка (Bubble sort)\n Введите 3 - Сортировка вствками (Insertion sort)\n Введите 4 - Сортировка слиянием (Merge sort)\n : ";

cin >> v;

while (n != 1) {

switch (v) {

case 1:

v1.selectionSort(l,t); n = 1;

break;

case 2:

v1.bubbleSort(l, t); n = 1;

break;

case 3:

v1.insertionSort(l, t); n = 1;

break;

case 4:

v1.mergeSort(l, t); n = 1;

break;

default:

cout << "Введите значение еще раз - ";

cin >> v;

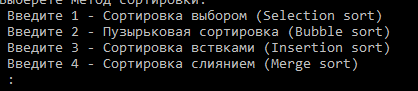
break;

}

}

}

};

****

*Рисунок 12.. Класс SortChisla*

* + 1. **Класс SortText**

Данный класс содержит функцию viborMetoda2() являющееся реализацией пользовательского интерфейса. Служит для выбора пользователем метода сортировки. В зависимости от выбранного метода вызывает функцию класса соответствующего данному методу. Код реализации данного класса представлен ниже:

class SortText {

public:

void viborMetoda2() {

SortAlgoText v1;

int v;

cout << "Выберете метод сортировки:\n Введите 1 - Сортировка букв в строке по Алфавиту\n Введите 2 - Сортировка нескольких строк\n : ";

cin >> v;

if (v == 1)

v1.alphSort();

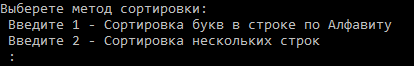
if (v == 2) {

v1.sortMain();

}

}

};

****

*Рисунок 13. Класс SortText*

* + 1. **Класс VvodZnach**

Данный класс содержит постоянные переменные k = 0; и n, а так же функции:

1. **setRazmer()** – Данная функция позволяет ввести размер массива. При вызове возвращает размер массива.

**Код:** int setRazmer() {

if (k == 0) {

cout << "Введите размер массива: ";

cin >> n;

k = 1;

return n;

}

**C:\Users\htcvive\AppData\Local\Microsoft\Windows\INetCache\Content.Word\функция ввода размера массива.png** if (k == 1) { return n; }

}

*Рисунок 14. Класс VvodZnach. Ввод размера массива*

1. *\****vvodChisMas()**– Данная функция позволяет пользователю заполнить целочисленный массив с клавиатуры. При вызове возвращает массив.

**Код:** int \* vvodChisMas() {

int n = setRazmer();

int \*mas = new int[n];

for (int i = 0; i < n; i++) {

cout << "Введите значение " << i + 1 << " элемента: ";

cin >> mas[i];

}

return mas; }

C:\Users\htcvive\AppData\Local\Microsoft\Windows\INetCache\Content.Word\функция ввода значения элемента.png

*Рисунок 15. Класс VvodZnach. Ввод значения элемента массива*

1. **vvodStr() –** Данная функция позволяет пользователь ввести набор букв для отсортировки. При вызове возвращает строку.

**Код:** string vvodStr() {

string str;

cout << "Введите набор букв для отсортировки: ";

cin >> str;

return str;}

* + 1. **Класс SortAlgoChisla**

Данный класс содержит функции:

1. **selectionSort(int \*data1, int n)** – Данная функция принимает на вход массив и его размер, далее реализует алгоритм сортировки выбором.

**Код:** void SortAlgoChisla::selectionSort(int \*data1, int n)

{

int j = 0;

int tmp = 0;

for (int i = 0; i < n; i++) {

j = i;

for (int k = i; k < n; k++) {

if (data1[j] > data1[k]) {

j = k;

}

}

tmp = data1[i];

data1[i] = data1[j];

data1[j] = tmp;

}

for (int i = 0; i < n; i++) {

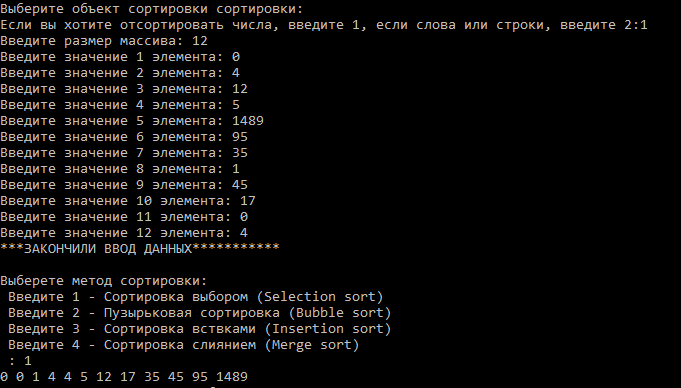
cout << data1[i];

cout << " ";

}

cout << endl;

}

****

*Рисунок 16. Класс SortAlgoChisla. Алгоритм сортировки выбором*

1. **bleSort(int \*data1, int n)** – Данная функция принимает на вход массив и его размер, далее реализует алгоритм сортировки пузырьком.

**Код:** void SortAlgoChisla::bubbleSort(int \*data1, int n)

{

int tmp = 0;

for (int i = 0; i < n; i++) {

for (int j = (n - 1); j >= (i + 1); j--) {

if (data1[j] < data1[j - 1]) {

tmp = data1[j];

data1[j] = data1[j - 1];

data1[j - 1] = tmp;

}

}

}

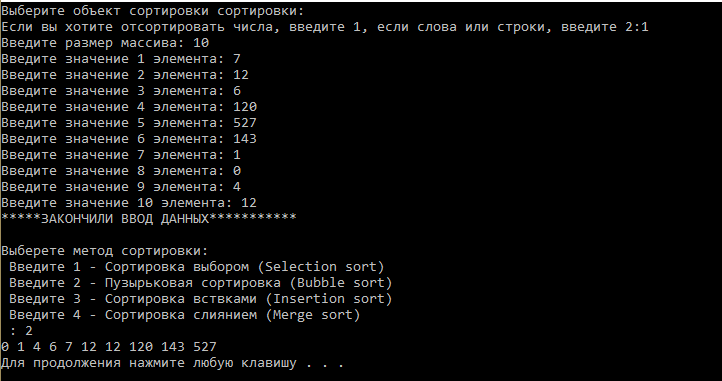
for (int i = 0; i < n; i++) {

cout << data1[i];

cout << " ";

}

cout << endl;

**** }

*Рисунок 17. Класс SortAlgoChisla. Алгоритм сортировки пузырьком*

1. **insertionSort(int \*data1, int n)** – Данная функция принимает на вход массив и его размер, далее реализует алгоритм сортировки вставками.

**Код:** void SortAlgoChisla::insertionSort(int \*data1, int n)

{

int key = 0;

int i = 0;

for (int j = 1; j < n; j++) {

key = data1[j];

i = j - 1;

while (i >= 0 && data1[i] > key) {

data1[i + 1] = data1[i];

i = i - 1;

data1[i + 1] = key;

}

}

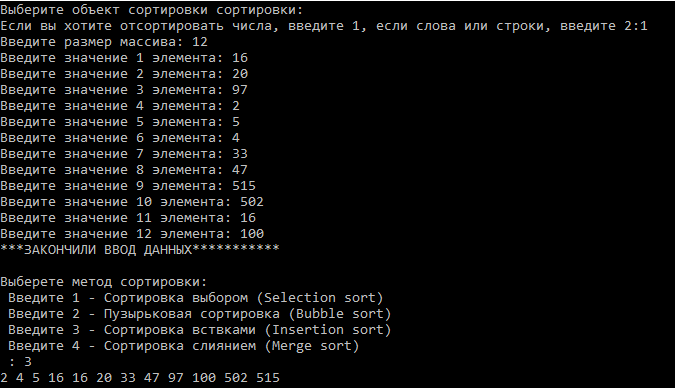
for (int i = 0; i < n; i++) {

cout << data1[i];

cout << " ";

}

cout << endl;

**** }

*Рисунок 18. Класс SortAlgoChisla. Алгоритм сортировки вставками*

1. **mergeSort(int \*data, int lenD)** - Данная функция принимает на вход массив и его размер, далее реализует алгоритм сортировки слиянием.

Код: void SortAlgoChisla::mergeSort(int \*data, int lenD)

{

if (lenD > 1) {

int middle = lenD / 2;

int rem = lenD - middle;

int\* L = new int[middle];

int\* R = new int[rem];

for (int i = 0; i < lenD; i++) {

if (i < middle) {

L[i] = data[i];

}

else {

R[i - middle] = data[i];

}

}

mergeSort(L, middle);

mergeSort(R, rem);

merge(data, lenD, L, middle, R, rem);

if (lenD == lenD) {

for (int i = 0; i < lenD; i++) {

cout << data[i];

cout << " ";

}

cout << endl;

}

}

}

void SortAlgoChisla::merge(int merged[], int lenD, int L[], int lenL, int R[], int lenR) {

int i = 0;

int j = 0;

while (i < lenL || j < lenR) {

if ((i < lenL) & (j < lenR)) {

if (L[i] <= R[j]) {

merged[i + j] = L[i];

i++;

}

else {

merged[i + j] = R[j];

j++;

}

}

else if (i < lenL) {

merged[i + j] = L[i];

i++;

}

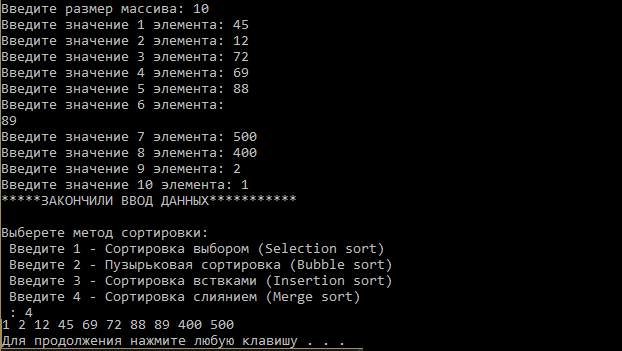
else if (j < lenR) {

merged[i + j] = R[j];

j++;

}

}

**** }};

*Рисунок 19. Класс SortAlgoChisla. Алгоритм сортировки слиянием*

* + 1. **Класс SortAlgoText**

Данный класс содержит функции:

1. **alphSort()** - Данная функция реализует алгоритм сортировки букв в строке по алфавиту.

**Код:** void alphSort()//сортировка букв в строке по алфавиту

{

VvodZnach v1;

string st = v1.vvodStr();

for (size\_t i = 0; i < st.length(); i++)

for (size\_t j = 0; j < st.length(); j++)

{

int a = int(st[j]);

int b = int(st[j + 1]);

if (a > b)

{

char buf = st[j];

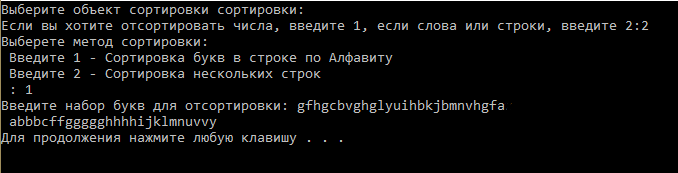
st[j] = st[j + 1];

st[j + 1] = buf;

}

}

cout << st << "\n";

**** }

*Рисунок 20 . Класс SortAlgoText. Алгоритм сортировки букв в строке по алфавиту*

1. **sortMain()** - Данная функция реализует алгоритм строк в алфавитном порядке и вызывает побочные функции getMyStr, GetStrFromMyStr и Sort.

**Код:** void sortMain() {

int count\_word = 0;

string \*S = new string[count\_word]; //Массив строк

string MyStr = GetMyStr(count\_word);

cout << "\*\*\*\*\*\*\*\*\*\*\*ЗАКОНЧИЛИ ВВОД ДАННЫХ\*\*\*\*\*\*\*\*\*\*\*\n\n";

//cout<<MyStr; //MyStr == многострочная строка

//Пишем функцию для взятия любой строчки из оной GetStrFromMyStr (обратимся к ней позднее)

//заполняем массив строками

for (int i = 0; i<count\_word; i++) {

S[i] = GetStrFromMyStr(MyStr, i + 1);

}

sort(S, count\_word); //Сортруем массив

//ВЫВОДИМ МАССИВ НА ЭКРАН

for (int i = 0; i<count\_word; i++) {

cout << S[i] << "\n";

}

delete[]S;

}

//сортировка нескольких строк

string GetMyStr(int &count\_word) {

string S, temp; //S - это буферная строка, в нее читаем информацию с клавиатуры

//temp - это итоговая строка, скомпилированная из всех полученных строк в S

cout << "Для прекращения ввода нажми CTRL+Z или введи пустую строку (НАЖМИ Enter)\n";

while ((getline(cin, S)) && (S.length())>2) { //Считываем символы с клавиатуры

temp = temp + S + '\n'; //Запоминаем строку в temp

count\_word++; //Увеличиваем счетчик строк

}

return temp; //Возвращаем скомпилированную строку

}

/\*ФУНКЦИЯ ДЛЯ ПОЛУЧЕНИЯ СТРОКИ ПО НОМЕРУ СТРОКИ\*/

string GetStrFromMyStr(string S, int N) {

int i = 0; //Количество встреченных разделителей

string buf; //Вытаскиваемая строка

const char \*temp = (S.c\_str()); //Указатель на начало строки S

while (i != N) { //Пока мы не встретим разделитель указанное число раз, будет цикл

while (\*temp != '\n') { //Ищем разделитель

if (i == N - 1) buf = buf + \*temp; //Если следующая итерация последняя, то начинаем собирать строчку (Это та самая, взятая по номеру)

\*temp++; //Переход к следующему символу

}

\*temp++; //Переход к след символу

i++; //Увеличиваем счетчик разделителей

}

return buf; //Возвращаем выбранную строчку

}

/\*ФУНКЦИЯ СОРТИРОВКИ СТРОК\*/

void sort(string \*S, int N)

{

for (int i = 0; i < N - 1; i++)

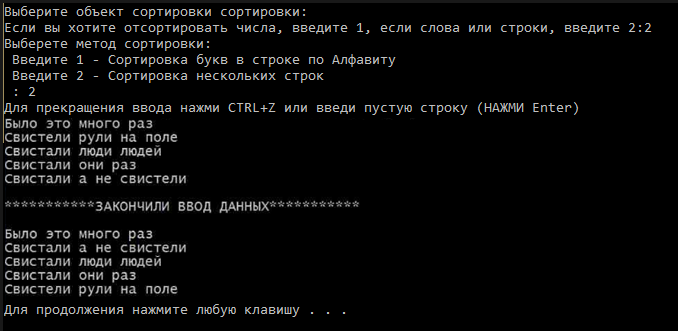
for (int j = i + 1; j < N; j++) {

if (strcmp(S[i].c\_str(), S[j].c\_str())>0) swap(S[i], S[j]);

}

}

};

****

*Рисунок 21. Класс SortAlgoText. Алгоритм сортировки строк в алфавитном порядке.*

* 1. **Характеристика программы**

Листинг – 322 строки.

Проект – 51,6 МБ

EXE-файл – 1,2 МБ

Время выполнения программы около 1 секунды

1. **Руководство пользователя**

**Заключение**

В данном проекте были реализованы алгоритмы сортировки в среде разработки MS Visual Studio на языке программирования C++. Так же закреплены знания по программированию.

**Список используемой литературы:**

**Программирование на языке высокого уровня Ефимушкина, С.П. Орлов, В.М. Чухонцев; Самар. гос. техн. ун-т. - Самара, 2002. 182с.**

**Экономическая информатика : В.П. Косарев/Л.В. Еремина, Москва, 2001.**

**Технология разработки програмного обеспечения, Учебник для вузов, Орлов С.А., Цикохер Б.Я., 4-е издание, СПб, Питер, 2012, 608с.**

**Технология разработки програмного обеспечения, Учебник, Л. Г. Гагарина, Е.В. Кокорева, Б.Д. Виснаул. М.:ИД ФОРУМ, НИЦ Инфра-М, 2013, 400с.**

**ПРИЛОЖЕНИЕ А**

#include "stdafx.h"

#include "iostream"

#include "cstdio"

#include "iomanip"

#include "string"

#include "locale"

#include <windows.h>

using namespace std;

class VvodZnach {

int k = 0, n;

public:

int setRazmer() {

if (k == 0) {

cout << "Введите размер массива: ";

cin >> n;

k = 1;

return n;

}

if (k == 1) { return n; }

}

int \* vvodChisMas() {

int n = setRazmer();

int \*mas = new int[n];

for (int i = 0; i < n; i++) {

cout << "Введите значение " << i + 1 << " элемента: ";

cin >> mas[i];

}

return mas;

}

string vvodStr() {

string str;

cout << "Введите набор букв для отсортировки: ";

cin >> str;

return str;

}

};

class SortAlgoChisla

{

public:

void SortAlgoChisla::selectionSort(int \*data1, int n)

{

int j = 0;

int tmp = 0;

for (int i = 0; i < n; i++) {

j = i;

for (int k = i; k < n; k++) {

if (data1[j] > data1[k]) {

j = k;

}

}

tmp = data1[i];

data1[i] = data1[j];

data1[j] = tmp;

}

for (int i = 0; i < n; i++) {

cout << data1[i];

cout << " ";

}

cout << endl;

}

void SortAlgoChisla::bubbleSort(int \*data1, int n)

{

int tmp = 0;

for (int i = 0; i < n; i++) {

for (int j = (n - 1); j >= (i + 1); j--) {

if (data1[j] < data1[j - 1]) {

tmp = data1[j];

data1[j] = data1[j - 1];

data1[j - 1] = tmp;

}

}

}

for (int i = 0; i < n; i++) {

cout << data1[i];

cout << " ";

}

cout << endl;

}

void SortAlgoChisla::insertionSort(int \*data1, int n)

{

int key = 0;

int i = 0;

for (int j = 1; j < n; j++) {

key = data1[j];

i = j - 1;

while (i >= 0 && data1[i] > key) {

data1[i + 1] = data1[i];

i = i - 1;

data1[i + 1] = key;

}

}

for (int i = 0; i < n; i++) {

cout << data1[i];

cout << " ";

}

cout << endl;

}

void SortAlgoChisla::mergeSort(int \*data, int lenD)

{

if (lenD > 1) {

int middle = lenD / 2;

int rem = lenD - middle;

int\* L = new int[middle];

int\* R = new int[rem];

for (int i = 0; i < lenD; i++) {

if (i < middle) {

L[i] = data[i];

}

else {

R[i - middle] = data[i];

}

}

mergeSort(L, middle);

mergeSort(R, rem);

merge(data, lenD, L, middle, R, rem);

if (lenD == lenD) {

for (int i = 0; i < lenD; i++) {

cout << data[i];

cout << " ";

}

cout << endl;

}

}

}

void SortAlgoChisla::merge(int merged[], int lenD, int L[], int lenL, int R[], int lenR) {

int i = 0;

int j = 0;

while (i < lenL || j < lenR) {

if ((i < lenL) & (j < lenR)) {

if (L[i] <= R[j]) {

merged[i + j] = L[i];

i++;

}

else {

merged[i + j] = R[j];

j++;

}

}

else if (i < lenL) {

merged[i + j] = L[i];

i++;

}

else if (j < lenR) {

merged[i + j] = R[j];

j++;

}

}

}

};

class SortAlgoText

{

public:

void alphSort()//сортировка букв в строке по алфавиту

{

VvodZnach v1;

string st = v1.vvodStr();

for (size\_t i = 0; i < st.length(); i++)

for (size\_t j = 0; j < st.length(); j++)

{

int a = int(st[j]);

int b = int(st[j + 1]);

if (a > b)

{

char buf = st[j];

st[j] = st[j + 1];

st[j + 1] = buf;

}

}

cout << st << "\n";

}

void sortMain() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int count\_word = 0;

string \*S = new string[count\_word]; //Массив строк

string MyStr = GetMyStr(count\_word);

cout << "\*\*\*\*\*\*\*\*\*\*\*ЗАКОНЧИЛИ ВВОД ДАННЫХ\*\*\*\*\*\*\*\*\*\*\*\n\n";

//cout<<MyStr; //MyStr == многострочная строка

//Пишем функцию для взятия любой строчки из оной GetStrFromMyStr (обратимся к ней позднее)

//заполняем массив строками

for (int i = 0; i<count\_word; i++) {

S[i] = GetStrFromMyStr(MyStr, i + 1);

}

sort(S, count\_word); //Сортруем массив

//ВЫВОДИМ МАССИВ НА ЭКРАН

for (int i = 0; i<count\_word; i++) {

cout << S[i] << "\n";

}

delete[]S;

}

//сортировка нескольких строк

string GetMyStr(int &count\_word) {

string S, temp; //S - это буферная строка, в нее читаем информацию с клавиатуры

//temp - это итоговая строка, скомпилированная из всех полученных строк в S

cout << "Для прекращения ввода нажми CTRL+Z или введи пустую строку (НАЖМИ Enter)\n";

while ((getline(cin, S)) && (S.length())>2) { //Считываем символы с клавиатуры

temp = temp + S + '\n'; //Запоминаем строку в temp

count\_word++; //Увеличиваем счетчик строк

}

return temp; //Возвращаем скомпилированную строку

}

/\*ФУНКЦИЯ ДЛЯ ПОЛУЧЕНИЯ СТРОКИ ПО НОМЕРУ СТРОКИ\*/

string GetStrFromMyStr(string S, int N) {

int i = 0; //Количество встреченных разделителей

string buf; //Вытаскиваемая строка

const char \*temp = (S.c\_str()); //Указатель на начало строки S

while (i != N) { //Пока мы не встретим разделитель указанное число раз, будет цикл

while (\*temp != '\n') { //Ищем разделитель

if (i == N - 1) buf = buf + \*temp;

//Если следующая итерация последняя, то начинаем собирать

\*temp++; //Переход к следующему символу

}

\*temp++; //Переход к след символу

i++; //Увеличиваем счетчик разделителей

}

return buf;//Возвращаем выбранную строчку

}

/\*ФУНКЦИЯ СОРТИРОВКИ СТРОК\*/

void sort(string \*S, int N)

{

for (int i = 0; i < N - 1; i++)

for (int j = i + 1; j < N; j++) {

if (strcmp(S[i].c\_str(), S[j].c\_str())>0) swap(S[i], S[j]);

}

}

};

class SortChisla {

public:

void viborMetoda() {

SortAlgoChisla v1;

VvodZnach v2;

int v, n = 0;

int \*l = v2.vvodChisMas();

int t = v2.setRazmer();

cout << t+"\n";

cout << "Выберете метод сортировки:\n Введите 1 - Сортировка выбором (Selection sort)\n Введите 2 - Пузырьковая сортировка (Bubble sort)\n Введите 3 - Сортировка вставками (Insertion sort)\n Введите 4 - Сортировка слиянием (Merge sort)\n : ";

cin >> v;

while (n != 1) {

switch (v) {

case 1:

v1.selectionSort(l,t); n = 1;

break;

case 2:

v1.bubbleSort(l, t); n = 1;

break;

case 3:

v1.insertionSort(l, t); n = 1;

break;

case 4:

v1.mergeSort(l, t); n = 1;

break;

default:

cout << "Введите значение еще раз - ";

cin >> v;

break;

}

}

}

};

class SortText {

public:

void viborMetoda() {

SortAlgoText v1;

int v;

cout << "Выберете метод сортировки:\n Введите 1 - Сортировка букв в строке по Алфавиту\n Введите 2 - Сортировка нескольких строк\n : ";

cin >> v;

if (v == 1)

v1.alphSort();

if (v == 2) {

v1.sortMain();

}

}

};

class ViborObject {

public:

void chislaOrText() {

SortChisla v1;

SortText v2;

int k;

cout << "Выберите объект сортировки сортировки:\n";

cout << "Если вы хотите отсортировать числа, введите 1, если слова или строки, введите 2:";

cin >> k;

if (k == 1)

v1.viborMetoda();

if (k == 2)

v2.viborMetoda();

}

};

int main(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "RUSSIAN");

ViborObject v1;

v1.chislaOrText();

system("PAUSE");

return 0;

}